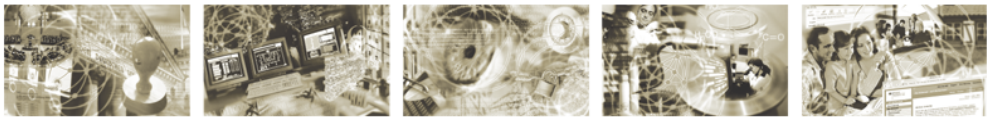
Bundesamt
für Sicherheit in der
Informationstechnik



BSI-Standards zur Internet-Sicherheit (ISi-Reihe)

Sicheres Bereitstellen von Web-Angeboten (ISi-Webserver)

Version 1.1 vom 23. Oktober 2017

Vervielfältigung und Verbreitung:

Bitte beachten Sie, dass das Werk einschließlich aller Teile urheberrechtlich geschützt ist.

Erlaubt sind Vervielfältigung und Verbreitung zu nicht-kommerziellen Zwecken, insbesondere zu Zwecken der Ausbildung, Schulung, Information oder hausinternen Bekanntmachung, sofern sie unter Hinweis auf die ISi-Reihe des BSI als Quelle erfolgen.

Dies ist ein Werk der ISi-Reihe. Ein vollständiges Verzeichnis der erschienenen Bände finden Sie auf den Internet-Seiten des BSI.
http://www.bsi.bund.de oder http://www.isi-reihe.de

Bundesamt für Sicherheit in der Informationstechnik
ISi-Projektgruppe
Postfach 20 03 63
53133 Bonn
Tel. +49 (0) 3018 9582-0
E-Mail: isi-redaktion@bsi.bund.de
Internet: http://www.bsi.bund.de

Inhaltsverzeichnis

1 Einleitung....7

2 Grundlagen....8

2.1 Protokolle und Kommunikationsstandards....8
2.1.1 HTTP....8
2.1.2 WebDAV....9
2.1.3 XML-RPC....9
2.1.4 SOAP....10
2.1.5 Datenbankkonnektoren....10
2.1.6 SQL....10
2.1.7 CORBA....11

2.2 Serverseitige Komponenten....12
2.2.1 Webserver....12
2.2.2 Web-Anwendungs-Server....13
2.2.3 Datenbank-Server....13
2.2.4 Verzeichnis-Server....14
2.2.5 Alt-Systeme....14
2.2.6 Reverse Proxy....15
2.2.7 Mail Transfer Agent (MTA)....16

2.3 Serverseitige Technologien....16
2.3.1 CGI....16
2.3.2 SSI....17
2.3.3 PHP....17
2.3.4 Weitere Skriptsprachen....17
2.3.5 JSP....18
2.3.6 JEE....18
2.3.7 ASP.NET / Mono, ASP.NET Core....18
2.3.8 Web-Service....18
2.3.9 AJAX/Atlas....19
2.3.10 Streaming Services....20

2.4 Aktive Inhalte....20

2.5 Werkzeuge zur Erstellung von Web-Angeboten....20
2.5.1 HTML/XML/CSS-Editoren....21
2.5.2 Redaktionssysteme....21

2.6 Technologien zur Authentisierung und Zugriffssteuerung....22
2.6.1 Formular-basierte Authentisierung....22
2.6.2 Basic Access Authentication....22
2.6.3 Digest Access Authentication....23
2.6.4 Host-basierte Authentisierung....23
2.6.5 Zertifikate....23
2.6.6 Cookies....24
2.6.7 Zugriffssteuerung anhand von Session-Tokens....24

2.7 Rechtliche Grundlagen....25

3 Sichere Grundarchitektur für normalen Schutzbedarf....28

3.1 Grundarchitektur der Infrastruktur....28
3.1.1 Variante der Grundarchitektur für einfache Web-Angebote....28
3.1.2 Variante der Grundarchitektur für komplexere Web-Angebote....29
3.1.3 Variante der Grundarchitektur mit CMS....30
3.1.4 Management....31

3.2 Komponenten der Grundarchitektur....32

3.2.1 Externer Webserver....32
3.2.2 Interner Webserver....33
3.2.3 Web-Anwendungs-Server....33
3.2.4 Hintergrund-Komponenten....34
3.2.5 Logging-Server....34
3.3 Grundarchitektur der Web-Anwendung....35
4 Komponenten sicher auswählen, konfigurieren und betreiben (normaler Schutzbedarf)....41
4.1 Grundanforderungen an ein sicheres Produkt....41
4.1.1 Betriebssystem....41
4.1.2 Webserver....42
4.1.3 Content Management System....42
4.1.4 Logging-Server....43
4.1.5 Hintergrundsysteme....43
4.1.6 Web-Anwendung....43
4.2 Konfiguration der Komponenten....44
4.2.1 Betriebssystem....45
4.2.2 Webserver....45
4.2.3 Content Management System....46
4.2.4 Logging-Server....46
4.2.5 Hintergrundsysteme....47
4.2.6 Web-Anwendung....47
4.3 Grundvorgaben für einen sicheren Betrieb....55
4.3.1 Betriebssystem....55
4.3.2 Webserver....55
4.3.3 Content Management System....55
4.3.4 Logging-Server....56
4.3.5 Hintergrundsysteme....56
4.3.6 Web-Anwendung....56
5 Gefährdungen und Empfehlungen mit Varianten für normalen und hohen Schutzbedarf....57
5.1 Gefährdungen durch Eindringen....57
5.1.1 Eindringen über bekannte Schwachstellen in Standardsoftware....57
5.1.2 Einschleusen von Betriebssystem-Befehlen über die Web-Anwendung (Command Execution)....60
5.1.3 Eindringen über Schwachstellen im CMS....61
5.1.4 Eindringen über Speicher-Management-Schwachstellen....62
5.1.5 Eskalation der Rechte mittels lokaler Speicher-Management- Schwachstellen....62
5.1.6 Eindringen durch Ausnutzen von Berechtigungen....62
5.1.7 Fehler in Datei-Uploads zur Ausführung von Betriebssystem-Befehlen....63
5.1.8 Ausführen von Befehlen beim Einbinden von Dateien mit Schadcode....64
5.2 Gefährdungen durch Verändern, Täuschen, Fälschen und Betrügen....65
5.2.1 Verschleierung von Angriffen....65
5.2.2 Umgehen von Logging-Mechanismen....65
5.2.3 Verändern der Webseite durch Ausnutzen von Berechtigungen....66
5.2.4 HTML Injection und HTTP Response Splitting....66
5.2.5 Session Riding....67
5.2.6 Replay-Angriffe....68
5.3 Gefährdungen durch Entwenden und Ausspähen....69
5.3.1 Entwenden der Identität eines legitimen Benutzers....69
5.3.2 Ausspähen von Konten bzw. gültigen Sitzungen im Datenstrom....70
5.3.3 Ausspähen von Konten und sensiblen Inhalten....70
5.3.4 Ausspähen und Manipulieren von Daten in Hintergrundsystemen nach Kompromittierung der Web-Anwendung....71
5.3.5 Auslesen von Daten aus den Hintergrundsystemen über Injection-Schwachstellen....71
5.3.6 Ausspähen von Informationen über Path Traversal....72
5.3.7 Erraten von Zugangsdaten....73

5.3.8 Entwenden von Zugangsdaten über die „Passwort zurücksetzen“-Funktion....74
5.3.9 Auf Cross-Site Scripting basierende Trojanische Pferde....75
5.3.10 Cross-Site-Scripting-Angriffe über Log-Dateien....75
5.3.11 Ausspähen von Systeminformationen über System- und Fehlermeldungen....76
5.3.12 Ausspähen von sensiblen Informationen aufgrund unzureichenden Zugriffsschutzes....77
5.3.13 Ausspähen durch die Verletzung des Minimalprinzips....77
5.3.14 Umgehen der Login-Abfrage....78
5.4 Gefährdungen durch Verhindern und Zerstören....78
5.4.1 Verhinderung von Benutzer-Zugriffen durch Auslösen der Konto-Sperre....78
5.4.2 Verhinderung des Web-Angebotes durch manipulierte Anfragen....79
5.4.3 Verhinderung durch Log-Überlauf....80
5.4.4 Verhinderung des Web-Angebotes durch Speicher-Management-Schwachstellen....81
5.4.5 Verhinderung des Web-Angebotes durch verteilte Dienstblockaden auf Anwendungsebene....81
6 Fazit....83
7 Anhang....84
7.1 Varianten der Grundarchitektur....84
7.1.1 Kleines Unternehmen....85
7.1.2 Mittelgroßes Unternehmen....89
7.1.3 Großes Unternehmen....92
7.2 Literaturverzeichnis....94

1 Einleitung

Die ISi-Reihe bietet allen Personen in Behörden und Unternehmen, die sich mit der IT-Sicherheit befassen, umfassende und aktuelle Informationen zu weiten Bereichen der IT-Sicherheit. Durch ihren Aufbau aus mehreren Teilen wendet sie sich zielgruppenspezifisch an Entscheidungsträger (mit dem Leitfaden ISi-L) sowie an diejenigen, die für die Umsetzung der IT-Sicherheit betraut sind (Studien ISi-S). Die unterschiedlichen Module der ISi-Reihe befassen sich jeweils mit einem sicherheitsrelevanten Teilbereich von IT-Systemen.

Die meisten Behörden und Unternehmen verfügen über einen eigenen Internetauftritt. Über diesen können Informationen bereitgestellt oder Dienstleistungen einer großen Benutzergruppe angeboten werden. Allerdings können Angreifer Schwachstellen in den Web-Angeboten nutzen, um Daten auszulesen, zu verändern oder zu infizieren. Auch Angriffe auf die Verfügbarkeit der Dienste sind häufig zu beobachten. Der Schutz der Web-Angebote vor solchen Angriffen, die oftmals einen Imageverlust mit sich bringen, hat für die Betreiber eine hohe Priorität.

Die vorliegende Studie zeigt, wie sich Behörden und Unternehmen gegen diese Gefahren schützen können. Sie zeigt in Abschnitt 3 wie das Web-Angebot und die zugrundeliegende Web-Anwendung aufgebaut werden sollten, um dem Schutz der Vertraulichkeit, Verfügbarkeit, Integrität und Authentizität Rechnung zu tragen. Empfehlungen zur Auswahl, Konfiguration und zum Betrieb werden in Abschnitt 4 gegeben. Die Anforderungen an die IT-Sicherheit variieren von Organisation zu Organisation. Um hier individuelle Anpassungen vorzunehmen, beispielsweise für den hohen Schutzbedarf oder Anpassungen an die Größe der Organisation, werden Varianten vorgestellt (Abschnitt 5) und bewertet.

Zusammen mit der Studie zur Sicheren Nutzung von Web-Angeboten [ISi-Web-Client] können bestehende Systeme abgesichert bzw. neue Systeme konzipiert, umgesetzt und betrieben werden. Hierzu sind die Checklisten, die ergänzend zu jeder Studie veröffentlicht werden, eine praktische Hilfe.

2 Grundlagen

In diesem Abschnitt werden die grundlegenden Protokolle, Werkzeuge und Technologien beschrieben, die bei der Bereitstellung von Web-Angeboten eine Rolle spielen. Das Verständnis der zugrundeliegenden Techniken dient auch der besseren Nachvollziehbarkeit der im Abschnitt 3 beschriebenen Grundarchitektur.

2.1 Protokolle und Kommunikationsstandards

Zunächst werden die Protokolle und Kommunikationsstandards erläutert, die für das sichere Bereitstellen von Web-Angeboten relevant sind. Tabelle 1 zeigt eine Übersicht der einzelnen Standards, die im Anschluss näher erklärt werden.

Protokoll	*Eigenschaften*
HTTP	Kommunikationsprotokoll zum Datenaustausch zwischen Webserver und Browser. Wird beispielsweise genutzt, um Webseiten zu Clients zu übertragen.
WebDAV	HTTP-Erweiterung zur Verwaltung von Dateien auf einem Webserver. Wird beispielsweise verwendet, um auf Online-Speichersysteme zuzugreifen.
XML-RPC	Ruft Funktionen auf entfernten Systemen über ein Netz auf. Die Kommunikation erfolgt über in HTTP eingebettete XML-Kommandos.
SOAP	Protokoll zum Aufruf von Web-Services. Die Repräsentation der übertragenen Daten erfolgt mit Hilfe von XML.
Datenbankkonnektoren	Standardisierte Datenbankschnittstellen (z. B. ODBC und JDBC) zum Abruf oder zur Änderung von Datenbank-Inhalten.
SQL	Sprache für Definition, Bearbeitung und Abfrage von Datenbank-Inhalten.
CORBA	Programmiersprachenunabhängige Spezifikation für die Kommunikation in verteilten Systemen.

Tabelle 1: Protokolle und Kommunikationsstandards und deren Eigenschaften

2.1.1 HTTP

Das Basis-Protokoll im Web ist HTTP, ein Protokoll der Anwendungsschicht des TCP/IP-Referenzmodells. Es ist in der aktuellen Version (HTTP/2) in [RFC 7540] definiert und basiert ursprünglich auf dem Client-Server-Prinzip, bei dem vom Client eine Anfrage gestellt wird (Request), welche vom Webserver beantwortet wird (Response). HTTP/2 kennt im Vergleich zur Version 1.1 mit Server-Push eine weitere Interaktionsart, bei der vom Server Daten an den Client geschickt werden können, ohne dass sie vom Client angefragt worden sind. Bis zur Version 1.1 arbeitet das HTTP-Protokoll zustandslos, was zur Konsequenz hat, dass nach erfolgter Datenübertragung (z. B. einer Webseite) die Verbindung zum Server wieder geschlossen und nicht aufrechterhalten wird. Ab Version 2 versteht HTTP auch Pipelining und Multiplexing, bei denen mehrere Anfragen und Antworten in einer über diesen Zeitraum offen gehaltenen Verbindung zusammengefasst werden können.

Bei HTTP handelt es sich ursprünglich um ein Klartextprotokoll, bei dem alle übertragenen Daten von einem möglichen Angreifer mitgelesen werden können. Deshalb wurde beim Entwurf von HTTP/2 zunächst erwogen grundsätzlich nur noch sichere Verbindungen über HTTPS (HTTP Secure) zuzulassen, welches eine verschlüsselte Verbindung auf Basis von TLS (Transport Layer Security) zwischen Webserver und Browser ermöglicht. Auch wenn sich diese Vorgehensweise nicht durchgesetzt hat, werden von großen Browserherstellern in ihren Clients ausschließlich verschlüsselte HTTP/2-Verbindungen unterstützt.

Für das Bereitstellen von Web-Angeboten über HTTP/2 ist es daher empfehlenswert HTTPS zu nutzen. Alte TLS-Versionen und der veraltete Vorgänger SSL sollten dabei nicht mehr eingesetzt werden (siehe [BSI-TR-02102]). Bei der Verwendung von HTTPS sollten auch HSTS (HTTP Strict Transport Security) und HPKP (HTTP Public Key Pinning) angeboten werden, die bei Verwendung kompatibler Clients einen zusätzlichen Schutz gegen Man-in-the-Middle-Angriffe ermöglichen.

Details zur genauen Arbeitsweise von HTTP und HTTPS können in der Studie „Sichere Nutzung von Web-Angeboten“ [ISi-Web-Client] nachgelesen werden.

2.1.2 WebDAV

WebDAV steht für Web-based Distributed Authoring and Versioning und ist ein offener Standard zur Bereitstellung und Verwaltung von Dateien auf einem Webserver. WebDAV erweitert die Version 1.1 des HTTP-Standards um zusätzliche Funktionen. Die WebDAV-Kommunikation zwischen Client und Server erfolgt ausschließlich über den HTTP-Port 80. Dies stellt einen wesentlichen Vorteil gegenüber anderen Protokollen mit vergleichbaren Funktionen, wie etwa FTP, dar. Diese verwenden für die Steuerung und den Datenaustausch unterschiedliche Verbindungen, wodurch es zu Konfigurationsproblemen bei Paketfiltern kommen kann.

Neben einfachen Datei-Operationen wie dem Hochladen, Umbenennen und Löschen von Dateien bietet WebDAV auch eine Versionskontrolle, die es mehreren Benutzern erlaubt, Dateien zu bearbeiten. Um all diese Funktionen nutzen zu können, ist jedoch eine vorherige Authentisierung der einzelnen Benutzer über die in HTTP zur Verfügung stehenden Authentisierungsverfahren (siehe Abschnitt 2.6) erforderlich. Für jeden Benutzer können unterschiedliche Berechtigungen vergeben werden. Dabei können beispielsweise auch nur bestimmte Dateitypen erlaubt werden. So können die Dateiendungen von ausführbaren Dateien (wie .exe) blockiert werden, um einer Verbreitung von Schadprogrammen vorzubeugen.

2.1.3 XML-RPC

XML-RPC (Extensible Markup Language Remote Procedure Call) ist ein Protokoll zum Aufruf von Funktionen auf entfernten Systemen, wobei die übertragenen Daten in einer XML-Struktur abgebildet werden. Die eigentliche Übertragung der XML-RPC-Nachrichten erfolgt über HTTP. Remote Procedure Calls (RPCs), wie sie beispielsweise durch XML-RPC umgesetzt werden, stellen die wesentliche Grundlage für verteilte Systeme dar. Sie ermöglichen es, Funktionen auf entfernten Systemen über ein Netz aufzurufen.

Da Funktionsaufrufe und Rückgabewerte in XML abgebildet werden, erfolgt eine Abstraktion von den zugrunde liegenden Programmiersprachen und Betriebssystemen. Dies bedeutet, dass Aufrufe unabhängig von der konkret verwendeten Programmiersprache bzw. des Betriebssystems erfolgen können. Funktionsaufrufe bestehen dabei aus dem Namen der Funktion, die aufgerufen werden soll, und den dazugehörigen Parametern. Funktionsrückgabe-Werte werden in einer ähnlichen Struktur wieder vom Server an den Client zurückgeliefert.

In XML-RPC sind keine Sicherheitsmaßnahmen vorgesehen. Es ist daher erforderlich, diese in der Programmlogik jenes Systems zu implementieren, welches XML-RPC für entfernte Funktionsaufrufe verwendet.

2.1.4 SOAP

SOAP stand ursprünglich für Simple Object Access Protocol. Dieses Akronym wird jedoch seit der Version 1.2 nicht mehr verwendet, da es die Funktionalität des Protokolls falsch beschreibt. SOAP ist heute ein eigenständiger Name des Protokolls. Es handelt sich um ein Rahmenwerk, das den Austausch von Daten zwischen Systemen über ein Netz regelt. Mit SOAP werden Aufrufe von Funktionen auf entfernten Systemen ermöglicht. Es kann somit als Nachfolger von XML-RPC verstanden werden. SOAP-Nachrichten basieren ebenfalls auf einer XML-Struktur und können mit Hilfe unterschiedlicher Protokolle übertragen werden.

Beispiele dafür sind sowohl das aus der E-Mail-Übertragung bekannte SMTP als auch das bereits genannte HTTP. Um eine verschlüsselte Verbindung einsetzen zu können, ist hierbei (wie auch bei XML-RPC) die Verwendung von HTTPS möglich. Eines der wichtigsten Anwendungsgebiete von SOAP ist die Bereitstellung und Nutzung von Web-Services.

2.1.5 Datenbankkonnektoren

Datenbankkonnektoren stellen eine standardisierte Schnittstelle zu Datenbanken und den zugehörigen Datenbank-Management-Systemen (DBMS) zur Verfügung. Sie ermöglichen es, unabhängig vom verwendeten DBMS, auf die in der Datenbank abgelegten Daten zuzugreifen oder diese zu verändern. Zusätzlich übernimmt ein Datenbankkonnektor den Auf- und Abbau sowie die Verwaltung einer Datenbankverbindung. Dadurch wird die Implementierung von Datenbankzugriffen im Zuge der Softwareentwicklung wesentlich vereinfacht. Datenbankkonnektoren erlauben einen sehr einfachen Zugriff auf Tabellen von Datenbanken oder Funktionen eines DBMS. Für die Verwaltung der Datenbanken und deren Inhalten verwenden Datenbankkonnektoren bekannte Datenbanksprachen wie etwa SQL (siehe Abschnitt 2.1.6).

Bekannte Datenbankkonnektoren sind unter anderem ODBC (Open Database Connectivity) und JDBC (Java Database Connectivity).

2.1.6 SQL

Die Structured Query Language (SQL) ist eine Datenbanksprache. Mit Hilfe von SQL-Anweisungen können Datenbestände definiert, abgefragt und verändert werden. Somit werden in SQL die Elemente von Datenverarbeitungssprache, Datenbeschreibungssprache und Datenaufsichtssprache vereint. Obwohl auch andere Datenbanksprachen existieren, hat sich SQL bei allen gängigen Datenbanken etabliert und ist auch von ANSI und ISO standardisiert.

Für Anwendungen und Web-Angebote, die auf SQL-Anweisungen für Datenbank-Abfragen zurückgreifen, müssen einige Sicherheitsmaßnahmen umgesetzt werden. Beispielsweise muss eine Anfälligkeit für SQL-Injections vermieden werden. Bei SQL-Injections handelt es sich um Schwachstellen bei Web-Angeboten, welche ungewollten Zugriff auf Datenbankinhalte erlauben (siehe Abschnitt 5.3.5). Diese Schwachstellen entstehen dann, wenn Web-Angebote Benutzereingaben nicht ausreichend filtern und Angreifer die SQL-Abfrage beeinflussen können. Können SQL-Abfragen manipuliert werden, besteht die Möglichkeit, Datenbestände vollständig auszulesen, zu verändern oder zu löschen.

Eine gebräuchliche Maßnahme zum Schutz vor SQL-Injections ist die Verwendung von Stored Procedures und Prepared Statements. In beiden Fällen werden die Datenbankanweisungen von den Benutzereingaben getrennt, so dass ein Angreifer keine in seinen Eingaben versteckten Kommandos mehr ausführen lassen kann. Bei Stored Procedures handelt es sich um eine Funktion des DBMS, zusammenhängende Anweisungen und Abläufe als fertige Prozeduren zu speichern. Dadurch hat ein Angreifer nicht mehr die Möglichkeit, die SQL-Abfrage durch eine SQL-Injection zu verändern. Bei Prepared Statements übernimmt das Datenbanksystem die Überprüfung der Parameter, die es durch die Trennung von Anweisungen und Parametern eindeutig identifizieren kann. Ein ähnlicher Schutz gegen SQL-Injections kann mit Hilfe von Frameworks erreicht werden. Frameworks, wie z. B. Hibernate, bilden eine zusätzliche Abstraktionsschicht zwischen Datenbank und der Implementierung des Datenbankzugriffs. Das Framework bietet dabei eine Schnittstelle für Programmierer, welche von der tatsächlich verwendeten Datenbank unabhängig ist und welche zur Ablage beliebiger Objekte geeignet ist.

Werden Stored Procedures und Prepared Statements oder Frameworks nicht eingesetzt, so müssen alle Daten, die von Benutzern stammen und als Datenbankeingabe verwendet, einer Eingabevalidierung unterzogen werden. Dabei werden die Zeichen gefiltert, mit denen die SQL-Anweisung und somit die Durchführung der Datenbankabfrage beeinflusst werden kann. Anwendungen, die SQL-Anweisungen für Datenbank-Abfragen verwenden, sind nach wie vor immer wieder anfällig für SQL-Injection-Angriffe, weshalb auf die genannten Schutzmaßnahmen nicht verzichtet werden sollte.

2.1.7 CORBA

Eine bekannte und oftmals eingesetzte Technik zum Aufruf entfernter Prozeduren ist CORBA. Dabei handelt es sich um einen Standard für verteilte Systeme, welcher für Common Object Request Broker Architecture steht. Mit Hilfe dieses Standards ist es möglich, Kommunikationsverbindungen zwischen Prozessen auf unterschiedlichen Systemen aufzubauen und Daten auszutauschen. Durch eine von Programmiersprachen unabhängige Schnittstellen-Definition (Interface Definition Language, IDL) können Programme, die in unterschiedlichen Programmiersprachen implementiert wurden, miteinander kommunizieren. Um eine Kommunikationsverbindung zwischen zwei Prozessen aufbauen zu können, werden auf beiden Seiten sogenannte Object Request Broker (ORB) benötigt. Diese werden in der von der jeweiligen Seite verwendeten Programmiersprache implementiert und haben die Aufgabe, Daten zu empfangen beziehungsweise zu versenden. Die Kommunikation zwischen einzelnen ORBs kann entweder über herstellerspezifische Protokolle oder über das herstellerunabhängige Internet Inter-ORB Protocol (IIOP) erfolgen.

Ähnliche Konzepte wie CORBA stehen auch in verschiedenen Programmiersprachen zur Verfügung. Zwei der gebräuchlichsten Möglichkeiten in diesem Zusammenhang sind RMI (Remote Method Invocation) und DCOM (Distributed Component Object Model). Das aus dem Java-Umfeld stammende RMI erlaubt den Aufruf von entfernten Java-Methoden. DCOM stellt ein objektorientiertes RPC-System auf Grundlage des DCE-Standards zur Verfügung. Bei DCE (Distributed Computing Environment) handelt es sich um einen Industriestandard für verteilte Anwendungen, welcher auf dem Client-Server-Modell basiert.

Da offene Architekturen mit Funktionsaufrufen über Systemgrenzen hinweg eine Vielzahl an Schnittstellen bieten, muss besonderes Augenmerk darauf gelegt werden, wer auf diese Schnittstellen zugreifen darf und welche Daten übertragen werden. Sicherheitsaspekte, insbesondere die Authentisierung von Teilnehmern, werden in der CORBA Security Specification abgedeckt.

2.2 Serverseitige Komponenten

Um ein Web-Angebot zur Verfügung stellen zu können, werden sowohl Hardware- als auch Software-Komponenten benötigt. Abhängig von der Funktionalität der Web-Anwendung werden dabei unterschiedliche Server-Typen benötigt. Die Basis-Komponenten stellen der Webserver und der Web-Anwendungs-Server dar. Für eine persistente Datenhaltung kommen meist Datenbank-Server zum Einsatz. Daneben kommen für einfache Operationen häufig auch Verzeichnis-Server zum Einsatz, auf deren Daten Clients meist nur lesend zugreifen können. Derartige Verzeichnisse werden beispielsweise zur Ablage von Benutzerdaten verwendet.

2.2.1 Webserver

Ein Webserver ist eine Software-Komponente, mit der Web-Angebote über HTTP und HTTPS bereitgestellt werden können. Damit wird ein Framework zur Verfügung gestellt, dessen Funktionen von der Web-Anwendung genutzt werden können. Häufig wird auch die Hardware, auf der eine Webserver-Software installiert ist, als Webserver bezeichnet.

Funktion eines Webservers

Der Webserver ist die Kern-Komponente jedes Web-Angebotes. Er nimmt die Anfragen der Benutzer entgegen und liefert, sofern möglich, selbst die entsprechende Antwort zurück. Dies ist beispielsweise bei statischen Web-Anwendungen der Fall. Angefragte Inhalte werden dabei sofort ohne Aufruf dynamischer Funktionen durch den Webserver zurückgeliefert. Bei dynamischen Web-Anwendungen leitet der Webserver die Anfrage meist an einen Web-Anwendungs-Server weiter. Dort werden dynamische Funktionen durchgeführt, wie beispielsweise der Aufbau einer Webseite auf Grund von Datenbank-Inhalten, und das Ergebnis wieder an den Webserver gesandt. Manche Webserver haben den Web-Anwendungs-Server für einige Programmiersprachen integriert (z. B. unterstützt der Apache Webserver die Skriptsprache PHP (siehe Abschnitt 2.3.3)). In diesem Fall wird die Anwendung auf dem Server lokal ausgeführt und muss nicht an einen Web-Anwendungs-Server weitergeleitet werden.

Da der Webserver direkt vom Benutzer angesprochen wird, bietet er eine exponierte Angriffsfläche. Der Webserver muss sicherstellen, dass nur legitime Anfragen an Hintergrundsysteme weitergeleitet werden, dass ein Benutzer nur auf die Inhalte Zugriff bekommt, für die er autorisiert ist, und dass der Webserver nicht selbst durch die Ausnutzung von Software-Schwachstellen kompromittiert werden kann.

Bereitstellen von dynamischen Web-Angeboten

Ein Web-Angebot stellt Benutzern Informationen und Funktionen zur Verfügung. Ein Benutzer greift mittels eines Browsers darauf zu. Die sichere Nutzung von Web-Angeboten wird in der Studie „Sichere Nutzung von Web-Angeboten“ [ISi-Web-Client] beschrieben. Um Web-Angebote betreiben zu können, welche dynamische Funktionen zur Verfügung stellen, bieten sich die folgenden Architekturen an:

- Realisierung der dynamischen Funktion über externe Programme, die über Schnittstellen aufgerufen werden. Beispiele hierfür sind CGI und SSI. Beim Aufruf eines Web-Angebotes werden Programme direkt auf dem Webserver aufgerufen. Die Ergebnisse des Programmaufrufes werden in jenes Ergebnis eingebettet, das an den Browser des Benutzers zurückgeliefert wird.
- Realisierung der dynamischen Funktion in Form von Funktionen oder Modulen, die im Webserver integriert sind. Ein Beispiel dazu ist PHP als Modul in Apache. Der wesentliche Unter-

schied zum ersten Punkt besteht darin, dass keine externen Programme aufgerufen werden. Die dynamische Funktion wird direkt in die anzuzeigenden Webseiten, beispielsweise in Form von Skriptcode, eingebettet. Vor Auslieferung der Webseite an den Browser wird der Skriptcode interpretiert und das Ergebnis in die Antwort des Servers eingeschlossen.

- Realisierung der dynamischen Funktion auf einem eigenständigen Web-Anwendungs-Server wie JBoss, Weblogic oder WebSphere. Bei dieser Architekturform werden Anfragen zunächst vom Webserver entgegengenommen und verarbeitet. Jene Teile der Anfrage, welche den Aufruf von dynamischen Funktionen erfordern, werden an einen Web-Anwendungs-Server weitergeleitet. Dieser führt die benötigten Funktionen und die eventuelle Kommunikation mit Hintergrundsystemen durch und liefert das Ergebnis an den Webserver zurück. Der Webserver bettet das Ergebnis des Web-Anwendungs-Servers anschließend in jene Antwort ein, die an den Browser des Clients gesandt wird.

2.2.2 Web-Anwendungs-Server

Zahlreiche Web-Angebote benötigen zusätzlich zum Webserver weitere Systeme. Beispielsweise werden eigene Web-Anwendungsserver benötigt, wenn Web-Angebote mit Hilfe von Java oder .NET bereitgestellt werden. Programmiersprachen wie PHP, ASP oder Perl funktionieren größtenteils ohne zusätzlichen Web-Anwendungs-Server, da die benötigten Funktionen meist direkt im Webserver integriert sind.

Ein Web-Anwendungs-Server wird verwendet, um dynamische Web-Angebote zur Verfügung stellen zu können. Dabei werden Anfragen vom Webserver mit entsprechenden Parametern an den Web-Anwendungs-Server weitergereicht. Dieser ruft in weiterer Folge die für die Verarbeitung der Anfrage erforderlichen Skripte, Methoden oder Funktionen auf. Je nach Art des Aufrufs führt der Anwendungsserver auch Anfragen an Hintergrundsysteme, beispielsweise an Datenbank- oder Verzeichnis-Server, durch. Das Ergebnis des Web-Anwendungs-Servers wird dann wieder an den Webserver zurückgegeben.

In einem Web-Anwendungs-Server sind einige wichtige Funktionen und Frameworks, welche häufig zum Betrieb von Web-Angeboten benötigt werden, vereinigt. Durch die Kapselung und Abstraktion von Schnittstellen zu anderen Systemen (z. B. Hintergrundsysteme) kann eine saubere Trennung zwischen Anwendung und Datenhaltung erfolgen. Frameworks, die auf Web-Anwendungs-Servern bereitgestellt werden können, umfassen eine Vielzahl an Funktionen zur Kommunikation mit Hintergrundsystemen. Beispielsweise werden abstrakte Funktionen zum Auslesen und zur Manipulation von Datenbankinhalten geboten, welche nur geringe Kenntnisse über die tatsächlich eingesetzte Datenbank erfordern. Darüber hinaus sind in Frameworks verschiedenste Sicherheitsfunktionen bereits implementiert (z. B. Prepared Statements für SQL-Anfragen, welche die Ausnutzung von SQL-Injection-Schwachstellen verhindern). Durch Hinzufügen weiterer Web-Anwendungs-Server zu einem Cluster ist eine Skalierung möglich, ohne dass hierfür die Anwendung modifiziert werden muss.

2.2.3 Datenbank-Server

Um Daten dauerhaft zu speichern, kommen im Zusammenhang mit Web-Angeboten meist Datenbanken zum Einsatz. Diese werden gemeinsam mit den zugehörigen Datenbank-Management-Systemen meist auf dedizierten Datenbank-Servern betrieben. Grundsätzlich werden folgende Formen von Datenbanken unterschieden:

- Hierarchische Datenbanken stellen Datenobjekte in einer Eltern-Kind-Beziehung dar.

- Netz-basierte Datenbanken können Datenobjekte über Netze miteinander verknüpfen.
- Relationale Datenbanken bilden Datenobjekte in Tabellen ab. Diese Tabellen können untereinander in Beziehungen stehen.
- Objektorientierte Datenbanken legen Datenobjekte als Objekte im Sinne der objektorientierten Programmierung ab. Dies bedeutet, dass Objekte in einer objektorientierten Datenbank die gleichen Eigenschaften aufweisen wie Objekte in der Programmierung.

Datenbank-Server müssen speziell geschützt werden und der Zugriff auf die darauf befindlichen DBMS darf nur durch autorisierte Ressourcen erfolgen. Darüber hinaus muss im Sinne des Minimalprinzips auch der Zugriff auf Datenobjekt-Ebene klar definiert und gewartet werden. Wie jede andere Software können auch DBMS Schwachstellen aufweisen, die von Angreifern ausgenutzt werden können, um Zugriff zu vertraulichen Daten zu erhalten. Neben der Einschränkung des Zugriffs müssen daher Maßnahmen umgesetzt werden, um bekannte Schwachstellen zu beseitigen. Speziell für Datenbank-Systeme müssen im Web-Angebot Maßnahmen zum Schutz vor SQL-Injection getroffen werden, da Angreifer sonst möglicherweise Inhalte der Datenbank unberechtigterweise auslesen oder verändern können.

Besonders vertrauliche Inhalte einer Datenbank (z. B. Passwörter) sollten durch Anwendung von kryptografischen Verfahren vor unbefugtem Zugriff geschützt werden. Dies erfordert ein geeignetes Schlüsselmanagement und verhindert das Auslesen vertraulicher Daten im Klartext.

2.2.4 Verzeichnis-Server

Mit Hilfe eines Verzeichnis-Servers kann eine zentrale Verwaltung der Benutzerdaten sowie von Rechten erfolgen. Diese Daten werden meist in einer hierarchischen Datenbank abgelegt. Der Zugriff auf einen Verzeichnisdienst erfolgt in der Regel über LDAP (Lightweight Directory Access Protocol). Dieses setzt auf TCP/IP auf und erlaubt die Abfrage und Modifikation von Informationen auf dem Verzeichnis-Server.

Da auf Verzeichnis-Servern häufig sensible Daten gespeichert werden, sind auch sicherheitsrelevante Faktoren zu berücksichtigen. Zum einen sollten sensible Daten (z. B. Passwörter) durch Anwendung geeigneter kryptografischer Verfahren vor unbefugtem Zugriff geschützt werden. Zum anderen besteht bei LDAP, ähnlich wie bei SQL, die Möglichkeit, Anfragen zu manipulieren. Es sind daher geeignete Maßnahmen umzusetzen, um sogenannte LDAP-Injections zu verhindern.

2.2.5 Alt-Systeme

Dort, wo Informationssysteme in Organisationen über mehrere Jahrzehnte gewachsen sind, existiert oft eine Vielzahl an Alt-Systemen, die zum Teil geschäftskritische Anwendungen beziehungsweise Daten beherbergen. Bei diesen Alt-Systemen kann es sich auch noch um Großrechner handeln, die über ein einfaches Terminal angesprochen werden können. Diese Systeme sind jedoch auch über entsprechende Schnittstellen ansprechbar, die einen Datenaustausch ermöglichen. Somit besteht die Möglichkeit, mit Hilfe von Web-Anwendungen neue Zugangsmöglichkeiten zu solchen Alt-Systemen zu schaffen. Web-Anwendungen sind in diesem Sinne ein neues Frontend für Alt-Systeme.

Diese Art des Zugangs birgt jedoch auch sicherheitstechnische Probleme in sich. Bisher bildeten diese Alt-Systeme ein abgeschottetes System, weshalb Schwachstellen kaum relevant waren. Durch die Anbindung mit Hilfe von Web-Anwendungen werden sie jedoch auch für externe Angreifer erreichbar, falls die Web-Anwendung entsprechende Sicherheitslücken aufweist. Es muss daher

zum einen für einen geeigneten Schutz der Alt-Systeme gesorgt und zum anderen auf ein entsprechendes Sicherheitslevel der angebundenen Web-Anwendungen geachtet werden.

2.2.6 Reverse Proxy

Proxys kommen meist auf der Client-Seite zur Nutzung von Web-Angeboten zum Einsatz. Daneben besteht jedoch auch die Möglichkeit, diese auf der Server-Seite einzusetzen, um Zugriffe zu optimieren (Caching) bzw. eine Filterung durchzuführen. Befindet sich der Proxy auf der Server-Seite, spricht man von einem Reverse Proxy. Alle Anfragen, die an den zugehörigen Webserver gerichtet sind, werden zuerst vom Proxy angenommen. Dieser entscheidet anhand eines konfigurierbaren Regelwerks, ob er die Anfrage selbst beantworten kann (Caching), ob er die Anfrage an den Webserver bzw. einen der Webserver im Cluster weiterleitet oder ob er die Abfrage aus Sicherheitsgründen abweist. Im Folgenden werden die wichtigsten Funktionen eines Reverse Proxys kurz erläutert.

Erfolgt eine Terminierung der Verbindung und Filterung der Daten auf einem Proxy, sind jedoch auch entsprechende datenschutzrechtliche Aspekte zu beachten. Beispielsweise könnten IP-Adressen jedes Clients, Zeitpunkte von Anmeldungen und die aufgerufenen Seiten mitprotokolliert werden.

Caching

Statische Inhalte wie Bilder oder statischer HTML-Text können auf einem Reverse Proxy zwischengespeichert werden. Werden diese Inhalte angefragt, können diese direkt vom Reverse Proxy beantwortet werden. Durch dieses Caching können Antwortzeiten verkürzt und die Auslastung der Webserver reduziert werden.

Caching kann jedoch auch zu sicherheitskritischen Problemen führen. Werden beispielsweise Inhalte auf dem Reverse Proxy zwischengespeichert, für die im Normalfall eine Autorisierung auf dem Webserver erforderlich ist, muss sichergestellt werden, dass diese Inhalte auch nur an autorisierte Benutzer ausgeliefert werden.

Lastverteilung

Kann ein Reverse Proxy eine Anfrage selbst beantworten, weil die erforderlichen Daten im Cache verfügbar sind, ist es nicht erforderlich, die Anfrage an den dahinter liegenden Webserver weiterzuleiten. Auf diese Weise kann die Last auf den Webservern verringert werden. Da alle Anfragen zunächst an den Reverse Proxy gerichtet werden, kann dieser auch Anfragen auf mehrere Server verteilen. Somit kann eine Load-Balancer-Funktion realisiert werden.

Authentisierung

Ein Reverse Proxy erlaubt es, die Authentisierung aus dem Webserver auszulagern. Dadurch wird eine Art Einmal-Anmeldung (in der Fachliteratur auch als Single Sign On bezeichnet) möglich, wenn der Reverse Proxy die Authentisierung von Benutzern für mehrere Webserver übernimmt. Hat sich der Benutzer einmal am Reverse Proxy angemeldet, steht ihm die Benutzung mehrerer Server zur Verfügung, ohne sich erneut anmelden zu müssen. Zusätzlich kann die Weiterleitung von Anfragen von einer Authentisierung am Reverse Proxy abhängig gemacht werden.

Verschlüsselung

Die Terminierung einer Ende-zu-Ende-Verschlüsselung (z. B. bei der Verwendung von HTTPS) kann am Reverse Proxy erfolgen. Nur wenn die Entschlüsselung bereits am Reverse Proxy durchgeführt wird, ist es möglich, diesen für die Filterung von Anfragen zu verwenden. Zudem entlastet

eine Terminierung der Verschlüsselung am Reverse Proxy den dahinter liegenden Webserver, da dieser keine zusätzlichen Ressourcen für die Entschlüsselung aufwenden muss. Ein weiterer Vorteil dieser Variante ist die Unabhängigkeit des Verschlüsselungskanals von dem tatsächlich verwendeten Webserver. Auf diese Weise können aufeinanderfolgende Anfragen auch von unterschiedlichen Servern bearbeitet werden, ohne dass der Verschlüsselungskanal geändert werden muss.

2.2.7 Mail Transfer Agent (MTA)

Viele Web-Anwendungen sind in der Lage, zu bestimmten Zwecken E-Mails zu generieren (bspw. beim Passwort zurücksetzen). Zum Versand von E-Mails wird ein Mail Transfer Agent (MTA) verwendet. Das kann entweder ein lokal installiertes Programm (bspw. /usr/bin/sendmail) oder ein über das Netzwerk erreichbarer SMTP-Server sein.

Details zum sicheren Betrieb von E-Mail-Servern können in der Studie [ISi-Mail-Server] nachgelesen werden.

2.3 Serverseitige Technologien

Um dynamische Web-Angebote zur Verfügung stellen zu können, ist eine Programmlogik auf Server-Seite notwendig. In vielen Fällen kann diese Funktion vom Webserver übernommen werden (z. B. über einfache Skripte oder Server Side Includes (SSI)). Bei komplexen Web-Angeboten wird häufig ein Web-Anwendungs-Server mit einem entsprechenden Framework eingesetzt. Der Web-Anwendungs-Server muss jedoch nicht notwendigerweise vom Webserver getrennt sein, da viele gängige Webserver bereits einen Web-Anwendungs-Server integriert haben (z. B. Tomcat bei Apache). Die Web-Anwendungs-Server beziehungsweise die zugehörigen Frameworks erlauben die Realisierung von umfassenden Web-Angeboten. Darüber hinaus wird durch diese der Zugriff auf Hintergrund- oder Alt-Systeme wesentlich vereinfacht. Häufig eingesetzte Programmiersprachen und Frameworks für dynamische Web-Angebote werden in den folgenden Abschnitten beschrieben.

2.3.1 CGI

CGI ist das Akronym für Common Gateway Interface und ist eine Methode, um Webseiten dynamisch und interaktiv zu gestalten. Die Generierung von dynamischen Inhalten wird dabei über externe Anwendungen, die vom Webserver aufgerufen werden, realisiert. Für diese Aufrufe stellt CGI eine Schnittstelle zwischen Webserver und System-Anwendung zur Verfügung. Es handelt sich daher bei CGI um keine Programmiersprache, sondern lediglich um eine Funktionalität, Programme vom Webserver aus auszuführen. CGI-Programme können daher mit beliebigen Programmiersprachen erstellt werden, sofern sich diese vom Webserver aus aufrufen lassen. Abhängig von der verwendeten Technik liegt ein CGI-Programm entweder als Binärdatei oder als Skript vor. Typische Beispiele für CGI-Sprachen sind C, Perl, TCL, Unix-Shell und viele andere mehr.

Beim traditionellen CGI wird für jede Anfrage bei Bedarf vom Webserver ein Prozess gestartet, welcher das Skript ausführt. Bei der moderneren Alternative FastCGI wird einmalig ein Prozess gestartet, der die Anfragen vom Webserver über eine Netzwerkverbindung (UNIX Domain Sockets oder TCP Port 9000) entgegen nimmt. Moderne FastCGI Handler können mehrere Prozesse mit unterschiedlichen Einstellungen verwalten. Dadurch kann jeder Prozess für eine andere Webseite zuständig sein und mit anderen Zugriffsrechten laufen. Eine Isolation einzelner Webseiten voneinander ist so wesentlich einfacher möglich.

Da die dynamischen Funktionen von CGI-Programmen realisiert werden, obliegt es auch diesen, ein entsprechendes Sicherheitsniveau zu gewähren. Dies bedeutet beispielsweise, dass die Überprüfung von Parametern von jedem CGI-Programm selbst durchgeführt werden muss.

2.3.2 SSI

Ähnlich wie CGI sind auch Server Side Includes (SSI) eine Methode, um dynamische Seiten zu erzeugen. SSI erlauben es, beliebige Dateien oder die Rückgabewerte von Systembefehlen in ein Web-Angebot einzubinden. Die Möglichkeiten, dynamische Web-Angebote zu gestalten, sind jedoch mit SSI eher begrenzt, weshalb SSI heute nur noch wenig in Verwendung ist.

Analog zu CGI ist es auch bei SSI möglich, Schwachstellen über die eingebunden Systembefehle auszunutzen. Kann beispielsweise der Pfad einer eingebundenen Datei beeinflusst werden, so können bestimmte Dateien auf dem Server ausgelesen beziehungsweise Befehle auf dem System ausgeführt werden.

2.3.3 PHP

PHP ist eine Skript-Sprache, die es ermöglicht, dynamische Web-Angebote zu realisieren. Wesentliche Eigenschaften von PHP sind die leichte Erlernbarkeit und die breite Unterstützung für Datenbankanbindungen.

Die Besonderheit bei PHP liegt u. a. darin, dass statische Inhalte (bspw. Bilder oder Stylesheets) weiterhin vom Webserver ausgeliefert werden. Dadurch muss der Laufzeitbenutzer des Webservers die Berechtigungen zum Lesen der statischen Inhalte des CMS haben. PHP bietet besondere Möglichkeiten zur Härtung des Systems an. Beispielsweise können die zur Verfügung stehenden Systemfunktionen eingeschränkt werden. Der Handler PHP-FPM bietet eine eigene chroot-Funktionalität an, die es Administratoren erspart, selbst eine chroot-Umgebung anzulegen. chroot steht für change root und ist eine Funktion unter Unix-basierten Betriebssystemen, um das Rootverzeichnis eines Prozesses und seiner Kindprozesse zu ändern. Dies bewirkt, dass ein Programm, das auf ein definiertes Verzeichnis eingeschränkt wurde, nicht mehr auf Dateien außerhalb dieses Verzeichnisses zugreifen kann.

Wenn mehrere PHP-Anwendungen auf einem Server betrieben werden oder eine Anwendung PHP-Dateien von außerhalb des Webserver-Dokumentenverzeichnisses lädt, kann der Einsatz der PHP-Funktion „open-basedir“ zur Gesamtsicherheit des Systems beitragen. Die „open-basedir“ Direktive erlaubt es, den Dateisystem-Bereich, auf welchen eine PHP-Anwendung zugreifen darf, beliebig einzuschränken. Dies geschieht unabhängig von den Einstellungen des Webservers. Somit können die Auswirkungen einer Kompromittierung der Anwendung auf die angegebenen Pfade reduziert und ein Befall des gesamten Systems verhindert werden.

In der Vergangenheit sind in PHP eine Vielzahl von Schwachstellen aufgetreten.

2.3.4 Weitere Skriptsprachen

Neben PHP konnten sich noch weitere Skriptsprachen für die Realisierung von dynamischen Web-Angeboten etablieren. Die bekanntesten Beispiele sind Ruby, Python und Perl. Im Wesentlichen bieten all diese Skriptsprachen ähnliche Funktionen und sind daher auch ähnlichen Sicherheitsproblemen ausgesetzt. Es hat sich gezeigt, dass ein Großteil der bekannten Schwachstellen in Web-Anwendungen unabhängig von der verwendeten Programmiersprache ist.

2.3.5 JSP

Java Server Pages werden in erster Linie für die Präsentationsschicht von Java-Web-Anwendungen verwendet. Die darzustellenden Daten werden meist in sogenannten JavaBeans (Container zur Datenübertragung) abgelegt, welche einen einfachen Zugriff erlauben. Es ist jedoch auch möglich, Geschäfts-Logik in JSP-Seiten zu implementieren. Dies führt allerdings zu einer unsauberen Trennung zwischen Funktion und Darstellung der Daten und steht damit im Gegensatz zum Model-View-Controller-Ansatz, der eine klare Trennung zwischen Daten, Funktion und Präsentation vorsieht.

2.3.6 JEE

Die Java Enterprise Edition, abgekürzt JEE, spezifiziert eine Softwarearchitektur für transaktionsbasierte Java-Anwendungen. Damit ist es möglich, dynamische Inhalte durch das Einbetten von Java-Code in HTML- und XML-Dokumenten zu erstellen.

Java bietet eine Reihe von Sicherheitsfunktionen. So ist Java beispielsweise typensicher, was eine Überprüfung des Datentyps von Variablen und Parametern bei deren Verwendung impliziert. Java verhindert per Design Speicher-Management-Schwachstellen wie Buffer Overflows und Heap Overflows in der Anwendung. Damit ist ein Angreifer nicht mehr in der Lage, die Kontrolle über ein Programm zu erlangen, indem er den Speicherbereich des Programms mit manipulierten Eingaben befüllt. Andere Schwachstellen-Klassen stellen jedoch auch in Java eine Gefährdung dar. Mit Hilfe einer sogenannten Sandbox bietet Java jedoch die Möglichkeit, Code in einer sicheren und abgeschotteten Umgebung auszuführen, ohne dass dabei das Betriebssystem gefährdet wird. Mit Hilfe von JEE Security ist zudem eine restriktive Verwaltung von Systemressourcen möglich.

2.3.7 ASP.NET / Mono, ASP.NET Core

ASP.NET (Active Server Pages.NET) ist der Nachfolger von ASP und wird wie dieser vor allem in Microsoft-Umgebungen verwendet, da diese Technologie in erster Linie auf dem Microsoft Internet Information Server lauffähig ist. Bei ASP.NET handelt es sich allerdings nicht um eine eigene Programmiersprache, sondern um ein Framework, welches das Verfassen der Programmlogik in unterschiedlichen Programmiersprachen ermöglicht. Mit Mono steht neben der Implementierung von Microsoft auch zusätzlich eine unter Linux und macOS lauffähige Variante zur Verfügung.

Für ASP.NET existieren eine Reihe von Sicherheitsfunktionen. Ein Beispiel dafür ist der Gatekeeper-Mechanismus, der aus unterschiedlichen Modulen besteht und verschiedene Sicherheitsfunktionen anbietet (z. B. Filter und Authentisierung). Zudem steht ein eigenes Anti-Cross-Site-Scripting-Framework zur Verfügung. Mit Hilfe eines weiteren Frameworks kann eine rollenbasierte Zugriffskontrolle umgesetzt werden.

ASP.NET Core ist der Nachfolger von ASP.NET. Es handelt sich hierbei um eine vollständige Neuentwicklung von ASP.NET, welche nun allerdings plattformunabhängig und unter einer Open-Source-Lizenz verfügbar ist.

2.3.8 Web-Service

Ein Web-Service ist mit einer Web-Anwendung vergleichbar. Der Unterschied besteht darin, dass die Ausgabe von Ergebnissen nicht für einen Browser aufbereitet wird, sondern in anders strukturierter Form (z. B. SOAP) zur Verfügung gestellt wird. Durch die Vernetzung von Web-Services

kann eine Service-orientierte Architektur (SOA) aufgebaut werden. Dabei werden einzelne Teile einer Anwendung als Web-Service implementiert. Diese können dann fortlaufend von mehreren Anwendungen genutzt werden. Auf diese Weise wird die Wiederverwendbarkeit von Funktionen erhöht und die Wartung der einzelnen Anwendungsteile erleichtert.

Da Web-Services die gleichen Protokolle wie Web-Anwendungen verwenden, sind sie hinsichtlich der Sicherheitsanforderungen mit Web-Anwendungen gleichzusetzen. Auf Grund der Offenheit einer Service-orientierten Architektur muss im Vergleich zu geschlossenen Architekturen in besonderer Weise auf Zugriffsschutz geachtet werden. Für die Kommunikation mit Web-Services sind folglich die Anforderungen an Authentizität, Integrität und Vertraulichkeit besonders hoch. Das Abhören von Anfragen und den zugehörigen Antworten im Klartext sowie die Fälschung oder Veränderung von Nachrichten muss verhindert werden.

Die notwendigen Sicherheitsanforderungen können durch entsprechenden Einsatz von kryptografischen Verfahren umgesetzt werden. Im Sinne einer Service-orientierten Architektur ist es auch möglich, einzelne Sicherheitsmaßnahmen als eigene Services zu implementieren.

Ein wichtiger Begriff im Zusammenhang mit Web-Services ist WSDL (Web Service Description Language). Mit Hilfe von WSDL werden funktionale Angaben zu einem Web-Service gemacht, die erforderlich sind, um einen solchen Service nutzen zu können. Eine WSDL-Datei stellt die Beschreibung der Schnittstelle eines Web-Services dar. Es legt dar, welche Funktionen das Web-Service zur Verfügung stellt, wie diese aufgerufen werden und welche Parameter zum Aufruf benötigt werden. Eine WSDL-Datei enthält folglich die wesentlichen Informationen (z. B. Zugangspunkt und -protokoll), um die Nutzung von Web-Services zu ermöglichen.

Sicherheitsaspekte in Zusammenhang mit WSDL betreffen vor allem die erforderlichen XML-Parser. Diese sind notwendig, um die an den Web-Service übertragenen Daten zu verarbeiten. Da als Parser oft Eigenentwicklungen zum Einsatz kommen, sind diese für eine Vielzahl von Angriffen anfällig. Dabei werden meist absichtlich falsch gestaltete XML-Nachrichten verwendet, die zum Absturz des Parsers oder des gesamten Web-Services führen können. Ein Beispiel dafür ist eine XML-Bombe. Dabei handelt es sich um ein XML-Dokument, dessen Teilelemente mehrfach auf sich selbst referenzieren, wodurch Probleme beim Einlesen des Dokuments durch den Parser entstehen können.

Eine weitere Bedrohung für Web-Services ist das Ausspähen und Wiedereinspielen von unzureichend geschützten SOAP-Nachrichten (ein sogenannter Replay-Angriff, siehe Abschnitt 5.2.6).

2.3.9 AJAX/Atlas

Atlas ist ein Framework für AJAX (sieh Abschnitt 2.4), welches als Web-Service läuft. Mit Hilfe dieser Technologie wird versucht, Programme, die bisher nur auf PCs eingesetzt wurden, als Web-Anwendung nachzubilden. Dies bedeutet, dass Teile von Seiten nachgeladen werden können, ohne dass der Rest der Seite erneut aufgebaut werden muss. Damit wird im Gegensatz zu herkömmlichen Anwendungen eine viel bessere Performance erzielt.

Der Einsatz von AJAX wird allerdings nicht empfohlen, da hierfür Benutzer auf den Clients Aktive Inhalte zulassen müssen. Bei der Verwendung Aktiver Inhalte bestehen vor allem Gefahren durch Session Riding und Cross-Site-Scripting (siehe Abschnitt 4.2.6).

2.3.10 Streaming Services

Mit Hilfe von Streaming Services werden in erster Linie Audio- und Videodaten zu Clients übertragen. Dies erfordert auf der Client-Seite jedoch die Verwendung spezieller Programme oder Plug-Ins für den Browser. Um Berechtigungskonzepte für die dargestellten Inhalte durchsetzen zu können, werden oft sogenannte Digital-Rights-Management-Systeme (DRMS) verwendet. Diese stellen beispielsweise sicher, dass ein Client Daten nur dann anzeigen oder kopieren kann, wenn er die entsprechenden Rechte in Form einer Lizenz besitzt.

Streaming Services verwenden eine Reihe von Protokollen, um Daten an die Clients zu senden. Am gebräuchlichsten sind dabei das Real Time Streaming Protocol (RTSP) und das Resource Reservation Protocol (RSVP).

2.4 Aktive Inhalte

Interaktive Funktionen in Web-Angeboten können auch durch aktive Inhalte umgesetzt werden, die auf dem Client-System ausgeführt werden. Es ist allerdings fast immer möglich, diese Funktionalitäten auch mit dynamischen oder sogar statischen Inhalten ebenfalls bereitzustellen.

Die verschiedenen Techniken, die unter dem Oberbegriff *aktive Inhalte* zusammengefasst werden, unterscheiden sich unter anderem durch die Art, wie sie in eine HTML-Seite integriert werden. Skriptsprachen wie JavaScript, JScript oder VBScript werden im Textformat direkt in den HTML-Code eingefügt oder in einer aus dem HTML-Code aufgerufenen Datei abgelegt. Eine weitere Möglichkeit, ausführbaren Code in eine HTML-Seite zu integrieren, ist, im HTML-Code referenzierte und vorkompilierte Programm-Module wie Java-Applets oder ActiveX-Controls aufzurufen.

Eine weitere Technik im Bereich der aktiven Inhalte ist AJAX (Asynchronous JavaScript and XML), das auf JavaScript und XML-basiert. Häufig wird auch Flash zu den aktiven Inhalten gezählt, da der Funktionsumfang über die Darstellung reiner Animationen hinausgeht und es damit nicht mehr nur als Plug-In betrachtet werden kann.

Aktive Inhalte bringen dadurch, dass auf dem Client-System „fremder“ Code ausgeführt wird, eine Reihe von Sicherheitsproblemen mit sich. Browser-Hersteller und Anbieter von Plug-Ins versuchen deshalb, diese Probleme durch Maßnahmen wie eingeschränkte Rechte für Java-Applets, Active-X Controls oder andere Plug-Ins zu begrenzen. Insbesondere Bedrohungen im Zusammenhang mit Plugins (z. B. Flash) zählen derzeit zu den häufigsten und gravierendsten.

2.5 Werkzeuge zur Erstellung von Web-Angeboten

Es stehen unterschiedliche Werkzeuge zur Verfügung, die eine Abstraktionsschicht bei der Erstellung von Web-Angeboten schaffen. Zur Erstellung von Web-Inhalten werden Editoren und Redaktionssysteme (sogenannte Content Management Systeme, kurz CMS) angeboten, die ein Gestalten von Webseiten ermöglichen, ohne mit den Details der zugrunde liegenden Auszeichnungssprache vertraut zu sein.

Dies kann auf der einen Seite Sicherheitsprobleme aufwerfen, indem zum einen die Transparenz eingeschränkt wird, und zum anderen nicht ausreichend geschultes Personal dazu verleitet wird, eigenständige Web-Angebote zu entwickeln. Auf der anderen Seite können gute Content-Management-Systeme sicherstellen, dass auch weniger qualifiziertes Personal Inhalte publizieren kann, ohne dabei die Sicherheit des Gesamtsystems zu gefährden.

2.5.1 HTML/XML/CSS-Editoren

Die Erstellung von Webseiten ist zwar prinzipiell mit jedem beliebigen Editor möglich, für einen Laien ist jedoch die Formatierung von Texten mittels Auszeichnungselementen bzw. das Konzept der Trennung von Inhalt und Design nur schwer verständlich. Um auch solchen Personen die Möglichkeit zu geben, Inhalte professionell anzuordnen, verbergen spezielle Editoren den wahren Aufbau der Dateien und stellen sie nach dem WYSIWYG-Prinzip (What You See Is What You Get) dar.

Umfangreichere Editoren stellen auch Funktionen zur Verfügung, um die geänderten Dateien automatisiert in das Web-Angebot einzupflegen.

2.5.2 Redaktionssysteme

Redaktionssysteme werden eingesetzt, um Inhalte aufzubereiten und in geeigneter Weise zu organisieren. Gängige Vertreter von Redaktionssystemen sind Content-Management-Systeme (CMS) und Document-Management-Systeme (DMS).

CMS (Content Management System)

Content-Management-Systeme stellen meist eine Web-basierte Benutzeroberfläche zur Verfügung, die – ähnlich den zuvor behandelten Editoren – den zugrunde liegenden Aufbau der Darstellung wie z. B. HTML-Code und JavaScript abstrahiert. Bei Einsatz eines CMS sollte ein eigener Server verwendet werden, welcher für Benutzer des damit gestalteten Web-Angebotes nicht zugänglich ist.

Oftmals werden zahlreiche Vorlagen für Design und Funktionalität (z. B. Newsletter-Bestellung, Umfragen etc.) angeboten, die in das Web-Angebot sehr einfach und effizient eingebunden werden können. Auf diese Weise kann eine Art Bausatz für Webseiten realisiert werden, mit dem sehr einfach Webseiten gestaltet und mit Inhalten gefüllt werden können.

Durch die Verwendung von Vorlagen kann zudem eine einheitliche Darstellung der einzelnen Webseiten eines Web-Angebotes sichergestellt werden. Darüber hinaus bieten CMS meist auch eine Versionsverwaltung, die es ermöglichen, dass mehrere Personen am gleichen Web-Angebot arbeiten.

Wichtiger Bestandteil eines CMS ist die Rechteverwaltung. Ein CMS sollte mindestens die Rollen „Administrator“ und „Redakteur“ zur Verfügung stellen. Gute CMS stellen ein Rechtekonzept zur Verfügung, das unterschiedliche Berechtigungsstufen für jedes Element der Webseite für jeden Benutzer ermöglicht. So kann klar definiert werden, wer welche Änderungen an welchen Inhalten vornehmen darf.

Da diese Systeme das Fundament von Web-Angeboten bilden, sind Sicherheitsaspekte von besonderer Bedeutung. Einige der Funktionen eines CMS sind jedoch im Hinblick auf die Sicherheit besonders kritisch, da sie möglicherweise auch von Angreifern missbraucht werden können. Beispiele hierfür sind das Hochladen von Dateien, das Generieren von Inhalten und das Ausführen von Programmen.

CMS können auch über eine Vielzahl an Erweiterungen in ihrer Funktionalität ausgebaut werden. Diese stellen ein zusätzliches Sicherheitsrisiko dar, da das Sicherheitsniveau der Erweiterungen oft unzureichend ist und diese in der Regel nicht von den Herstellern des CMS unterstützt werden. Zudem sind die Bezugsquellen von solchen Erweiterungen nicht immer als vertrauenswürdig einzustufen.

DMS (Document Management System)

Dokumenten-Management-Systeme dienen der Verwaltung von Bildern, Archiven und anderen elektronischen Dokumenten. In diesem Sinne weist ein DMS ähnliche Eigenschaften und somit auch ähnliche Sicherheitsprobleme wie ein CMS auf.

Die Archivierung von elektronischen Dokumenten zählt zu den wichtigsten Eigenschaften von Dokumenten-Management-Systemen. Sie sind meist mit einer umfangreichen Suchfunktion ausgestattet, um auf abgelegte Dokumente effektiv zugreifen zu können. Darüber hinaus erlauben gängige DMS das Hinzufügen von Meta-Daten zu Dokumenten. Dies vereinfacht nicht nur deren Kategorisierung, sondern auch die Suche nach speziellen Inhalten.

2.6 Technologien zur Authentisierung und Zugriffssteuerung

Um die Identität des Benutzers feststellen und ihm entsprechende Rechte zuteilen zu können, gibt es unterschiedliche Mechanismen, die in den folgenden Abschnitten vorgestellt werden. Hat sich ein Benutzer erfolgreich authentisiert (beispielsweise durch Angabe von Benutzername und Passwort), so wird ihm eine sogenannte Session zugewiesen. Bei einer Session handelt es sich um eine Sitzung, die einem Benutzer zugeordnet ist und die eine zum Server hergestellte aktive Verbindung bezeichnet. Um bei Web-Angeboten dennoch von Benutzern abhängige Zustände abbilden zu können (z. B. den Anmeldestatus eines Benutzers oder den Inhalt eines Warenkorbs), werden Sessions verwendet.

Eine Session wird durch eine eindeutige Session-ID identifiziert. Diese wird nach einer erfolgreichen Anmeldung zum Client übertragen und bei jeder weiteren Anfrage an den Server wieder mitgesandt. Dadurch erkennt der Webserver, dass die Anfrage in einem bestimmten Kontext steht und kann sie einem Benutzer zuordnen.

2.6.1 Formular-basierte Authentisierung

Die Formular-basierte Authentisierung ist eine weit verbreitete Authentisierungs-Methode und wird bei den meisten Web-Anwendungen eingesetzt. Dabei werden die Anmeldedaten über ein Formular an die Web-Anwendung übergeben. Die Authentisierung des Benutzers erfolgt dann durch die Web-Anwendung, welche überprüft, ob die übergebenen Anmeldedaten für den angegebenen Benutzer korrekt sind. Der Vorteil dieser Art der Authentisierung ist die nahtlose Einbindung der Anmelde-Funktion in eine Web-Anwendung, da die Angabe der Anmeldedaten einfach über Eingabefelder in der Web-Anwendung erfolgt. Da die Authentisierung durch die Web-Anwendung vorgenommen wird, ist zudem eine sehr flexible Handhabung von Anmeldeversuchen (z. B. Handhabung von fehlgeschlagenen Anmeldeversuchen, Fehlermeldungen) möglich. Eine erhöhte Flexibilität bringt allerdings auch das Problem von Schwächen in der Implementierung mit sich. Wichtig ist es auch dafür Sorge zu tragen, dass die Anmeldedaten über eine gesicherte Verbindung übertragen werden.

2.6.2 Basic Access Authentication

HTTP Basic Access Authentication wurde im Rahmen von HTTP/1.0 [RFC 1945] spezifiziert und stellt einen einfachen Authentisierungsmechanismus zur Verfügung. Der Anmeldevorgang wird dabei jedoch nicht über die Web-Anwendung, sondern durch den Webserver selbst durchgeführt. Die Anmeldedaten werden dabei nur Base64-codiert und nicht verschlüsselt versendet, weshalb diese Art der Authentisierung nur über eine gesicherte Kommunikation benutzt werden darf. Ist der

Angreifer in der Lage, die Kommunikation abzuhören, so kann dieser die Kodierung umkehren und Benutzername und Passwort im Klartext auslesen. Basic Access Authentication bietet einen Schutz auf Verzeichnis-Ebene, da definiert werden kann, welcher Benutzer welche Zugriffsfunktionen auf welche Verzeichnisse anwenden darf. Diese Art der Authentisierung wird von allen gängigen Webservern und Browsern unterstützt, gilt jedoch als veraltet und wird kaum noch verwendet.

2.6.3 Digest Access Authentication

Digest Access Authentication wurde zusammen mit HTTP/1.1 eingeführt. Es basiert auf der Basic Access Authentication, wurde jedoch um Sicherheitsfunktionen erweitert, so dass es eine Abhilfe für einige – aber nicht alle – Schwächen von Basic Access Authentication bereitstellt. Gegen einige Angriffsszenarien bietet auch Digest Access Authentication keinen guten Schutz. So gibt es für einen Man-in-the-Middle-Angriff mehrere vielversprechende Ansätze. Der einfachste ist es dem Client vorzuenthalten, dass der Server überhaupt Digest Access Authentication anbietet. In diesem Fall wird der Client Basic Access Authentication zur Anmeldung nutzen, so dass der Angreifer im Klartext die benötigten Zugangsdaten erhält, mit denen er sich über Digest Access Authentication beim Server anmelden kann.

Der Sicherheitsgewinn durch Verwendung von Digest Access Authentication hängt zum Teil auch von der Implementation ab. Um Replay-Attacken wirkungsvoll verhindern zu können, dürfen die von Digest Access Authentication verwendeten NONCEs jeweils nur einmalig verwendet werden. Wegen des damit verbundenen hohen Ressourcen-Verbrauchs lässt der Standard aber auch Implementierungen zu, bei denen NONCEs wiederverwendet werden. Spezifiziert ist die Digest Access Authentication in [RFC 2617].

2.6.4 Host-basierte Authentisierung

Bei der Host-basierten Authentisierung werden die Zugriffsrechte auf Basis der IP-Adresse bestimmt. Diese Art der Authentisierung ist jedoch anfällig für IP-Spoofing. Dabei fälscht ein Angreifer die IP-Adresse der von ihm gesandten Netzpakete, wodurch die darin enthaltenen Anfragen unter anderen Zugriffsrechten ausgeführt werden.

2.6.5 Zertifikate

Zertifikat-basierte Authentisierung basiert auf einer Public-Key-Infrastruktur. Der Nachweis der Identität erfolgt dabei über ein Zertifikat, welches den öffentlichen Schlüssel einer Entität (z. B. Benutzer) beinhaltet und von einer Zertifizierungsstelle (CA) signiert sein muss. Die Zertifizierungsstelle ist somit auch für die Verifizierung der Identität vor Signieren des Zertifikats verantwortlich. In der Regel wird dies von einer eigenen Registrierungsstelle (RA) erledigt.

Meist kommt bei der Zertifikat-basierter Authentisierung ein sogenanntes Zwei-Faktor-Verfahren zum Einsatz. Dies bedeutet, dass zum Nachweis der Identität nicht nur der Besitz des Zertifikats, sondern auch ein Passwort erforderlich ist. Je nach Anwendungsgebiet kann die Speicherung des Zertifikats auf unterschiedlichen Medien erfolgen. Beispiele hierfür sind Token, Chipkarten oder Software-Zertifikatsspeicher.

2.6.6 Cookies

Cookies sind in der Regel kleine Text-Dateien, die Informationen über HTTP-Sitzungen lokal auf Client-Seite speichern. Diese werden bei erneuten Anfragen an einen Webserver in der HTTP-Kopfzeile mitgesendet und erlauben somit dem Server ein Wiedererkennen des Clients. Der Server kann mit Hilfe des Cookies beispielsweise erkennen, welcher Benutzer eine Anfrage sendet.

Im Cookie können außer einer eindeutigen ID noch andere Informationen gespeichert werden. Ein Beispiel hierfür ist die Angabe, für welche Domain und welchen Pfad ein Cookie gültig ist. Da das Mitsenden von Cookies vom Browser geregelt wird, ist dieser auch dafür verantwortlich, dass sie nur von der zugehörigen Domain, wie vom Server definiert, ausgelesen werden können. Cookies können zusätzlich Flags enthalten. Wird das `httponly`-Flag gesetzt, so kann das Cookie von JavaScript nicht mehr gelesen oder verändert werden.

Mit Hilfe von Cross-Site Scripting ist es möglich, Cookies von anfälligen Domains zu stehlen und sich damit beispielsweise gegenüber einem Webserver als ein anderer Benutzer auszugeben. Ein Angreifer schleust dabei Code in die Web-Anwendung ein, welcher auf dem Client des Benutzers ausgeführt wird und dem Angreifer das gewünschte Cookie im Hintergrund übermittelt.

2.6.7 Zugriffssteuerung anhand von Session-Tokens

Ein großer Teil aktueller Web-Anwendungen verwendet zur Authentifizierung der Nutzer einen Session-Token (auch Session-Identifier genannt). Dieser Token wird vom Webserver generiert, validiert und anschließend als Cookie an den Client gesendet sowie für die Dauer der Gültigkeit der Session auf dem Server persistent gespeichert. Der Client speichert den Cookie und sendet ihn bei jeder neuen Anfrage an den Server mit. Der Server kann anschließend anhand des Session-Tokens des Client-Cookies einen Abgleich mit dem lokal gespeicherten Wert vornehmen und so den Benutzer authentifizieren.

Die Abbildung 2.1 stellt diesen Vorgang als Sequenzdiagramm dar.

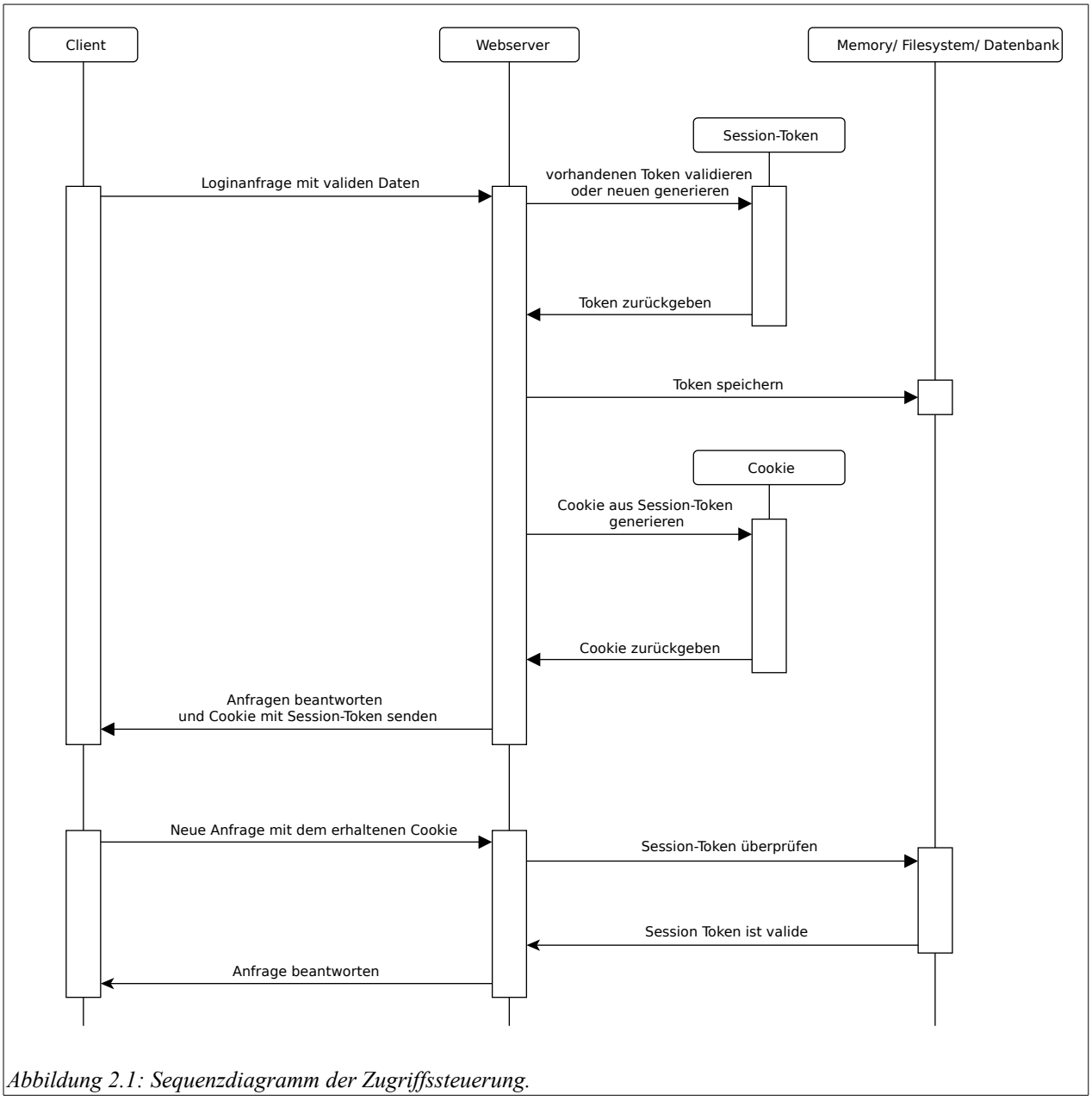


Abbildung 2.1: Sequenzdiagramm der Zugriffssteuerung.

Die Implementierung der Cookies und Session-Tokens ist von System zu System unterschiedlich. So werden teilweise die Session-Tokens erst nach der erfolgten Authentifizierung erstellt und an den Client in Form eines Cookies gesendet oder aber die Session-Tokens werden dem Client bereits beim ersten Besuch der Webseite übermittelt, aber erst nach erfolgreichem Log-In mit erweiterten Rechten versehen. Auch die Anzahl der Session-Tokens kann sich unterscheiden.

2.7 Rechtliche Grundlagen

Dieser Abschnitt ist ein allgemeiner Hinweis auf relevante rechtliche Aspekte ohne Anspruch auf Vollständigkeit. Die nachfolgenden Hinweise sind lediglich ein Auszug der rechtlich relevanten Maßnahmen, da eine vollständige Abhandlung den Rahmen dieses Dokuments sprengen würde. Für

einen konkreten Einsatz der Web-Anwendungen und für die dafür benötigten Systeme sollten die Maßnahmen mit Juristen abgestimmt werden.

Generell kann festgehalten werden, dass die Anforderungen des Datenschutzes, der Netzanbindung und ähnliche Regelungen stets eingehalten werden müssen. Daneben sind auch Vorgaben der Europäischen Union (z. B. Anforderung an die Verarbeitung personenbezogener Daten und den Schutz der Privatsphäre in der elektronischen Kommunikation) einzuhalten.

Personal

Sämtliche Mitarbeiter, die bei nicht-öffentlichen Stellen beschäftigt sind und die Zugang zu personenbezogenen Daten der Web-Anwendung haben bzw. diese administrieren oder Zugriff auf Logging-Daten haben, sind nach § 5 BDSG auf das Datengeheimnis zu verpflichten.

Formale Anforderung „Impressum“

Aus § 5 TMG ergibt sich eine Impressumspflicht für geschäftsmäßige Anbieter von Web-Angeboten. Der Begriff „geschäftsmäßig“ ist weit zu verstehen und setzt keine Verfolgung kommerzieller Interessen voraus. Die in der Vorschrift normierten Voraussetzungen sind stets einzuhalten, d. h. bei Betrieb eines Web-Angebotes sind u. a. folgende Angaben zu veröffentlichen:

- Name und Anschrift des Anbieters,
- bei juristischen Personen Rechtsform und Vertretungsberechtigter,
- Informationen zur schnellen Kontaktaufnahme (u. a. E-Mail-Adresse),
- ggf. Angabe der zuständigen Aufsichtsbehörde,
- Handels-, Vereins-, Partnerschafts- oder Genossenschaftsregister sowie die entsprechende Registernummer,
- Umsatzsteuer-Identifikationsnummer,
- zusätzliche Pflichten für besondere Berufsgruppen.

Absicherung nach Stand der Technik

Aus § 13 Abs. 7 TMG ergibt sich die Pflicht, dass geschäftsmäßig angebotene Telemediendienste (darunter zählen beispielsweise gewerbliche Webseiten) durch Maßnahmen nach Stand der Technik abgesichert werden müssen. Empfehlungen können in der BSI-Publikation „Absicherung von Telemediendiensten nach Stand der Technik“ [TMG-Papier] nachgelesen werden.

Logging

Logging von Daten sollte in dem Umfang erfolgen, wie es für die Erkennung von Angriffen, Angriffsversuchen und Sicherheitsverletzungen und ggf. deren Rückverfolgung erforderlich ist. Personenbezogene Daten dürfen nur in dem Umfang mitprotokolliert werden, in dem dies gesetzlich zulässig ist.

Vom Logging-System aufgezeichnete Daten, die der Zuordnung erkannter Ereignisse (etwa Angriffe oder Sicherheitsverletzungen) zu deren Verursachern dienen, dürfen von den Verantwortlichen nur in bestimmten Fällen an befugte Dritte (diese sind im Vorfeld zu definieren) weitergegeben werden. Dies ist zum Beispiel notwendig, wenn aufgrund des Ereignisses, dessen Auswirkungen oder sich ergebenden Gefährdungen entschieden wurde, den Verursacher des Ereignisses zu ermitteln oder die Weitergabe der Daten gemäß der entsprechenden Richtlinie vorgesehen ist. (Im Rahmen einer

Richtlinie kann festgelegt werden, ab welchem Gefährdungs- bzw. Schadensausmaß in jedem Fall versucht werden muss, den Verursacher zu ermitteln).

Mitgeloggte Daten werden spätestens nach dem gesetzlich vorgeschriebenen Zeitraum gelöscht.

3 Sichere Grundarchitektur für normalen Schutzbedarf

Die vorangegangenen Abschnitte haben die grundlegenden Protokolle, Dienste, Komponenten und Techniken zur Nutzung von Web-Angeboten eingeführt. Auf der Basis dieser Grundlagen beschreibt der folgende Abschnitt eine Grundarchitektur für das sichere Bereitstellen eines Web-Angebotes. Zu dieser Grundarchitektur gehört zum einem die Platzierung der verschiedenen Komponenten, aufbauend auf der Grundarchitektur aus [ISi-LANA], und zum anderen die Architektur der Web-Anwendung.

Web-Angebote werden für eine Vielzahl unterschiedlicher Zwecke eingesetzt. Je nach Umfang und Komplexität eines Web-Angebotes bietet die Grundarchitektur Spielräume für Realisierungsvarianten, aus denen auf Basis der Anforderungen und unter den Gesichtspunkten der Sicherheit und der Wirtschaftlichkeit die geeignetste Lösung ausgewählt werden kann.

3.1 Grundarchitektur der Infrastruktur

Die sichere Grundarchitektur der Infrastruktur basiert auf den Empfehlungen aus [ISi-LANA] und wurde um für Web-Angebote relevante Komponenten erweitert. Es werden zwei Varianten vorgestellt, mit denen unterschiedliche Web-Angebote realisiert werden können.

Zusätzlich wird in einer dritten Variante gezeigt, wie ein CMS (Content Management System) eingebunden werden kann. Ein CMS ist ein Redaktionssystem, mit dem Inhalte von Webseiten ohne tiefgehende technische Kenntnisse gestaltet und verändert werden können. Wird ein solches System eingesetzt, so müssen spezielle Sicherheitsvorkehrungen getroffen werden, da bei einer Kompromittierung des CMS das gesamte Web-Angebot unter der Kontrolle des Angreifers steht.

3.1.1 Variante der Grundarchitektur für einfache Web-Angebote

Für einfache Web-Angebote, die dynamische Funktionen bereitstellen sollen, jedoch keinen Zugriff auf umfangreiche Daten benötigen, kann die in Abbildung 3.1 gezeigt Grundarchitektur eingesetzt werden. Das Web-Angebot wird dabei durch den externen Webserver (WWW) und den Web-Anwendungs-Server (WWW AS) realisiert. Je nach Einsatzszenario können Webserver und Web-Anwendungs-Server auch physisch zusammengelegt werden. Zur Pflege des Web-Angebotes und für die Benutzer aus dem internen Netz kann ein interner Webserver WWW int. eingesetzt werden. Das Logging erfolgt über einen separaten Logging-Server, der über einen Paketfilter (PF3) mit dem Webserver und dem Web-Anwendungs-Server verbunden ist. Je nach Last der Systeme kann anstatt des separaten Logging-Servers auch das Logging in der Management-Zone (siehe Abschnitt 3.1.4) genutzt werden.

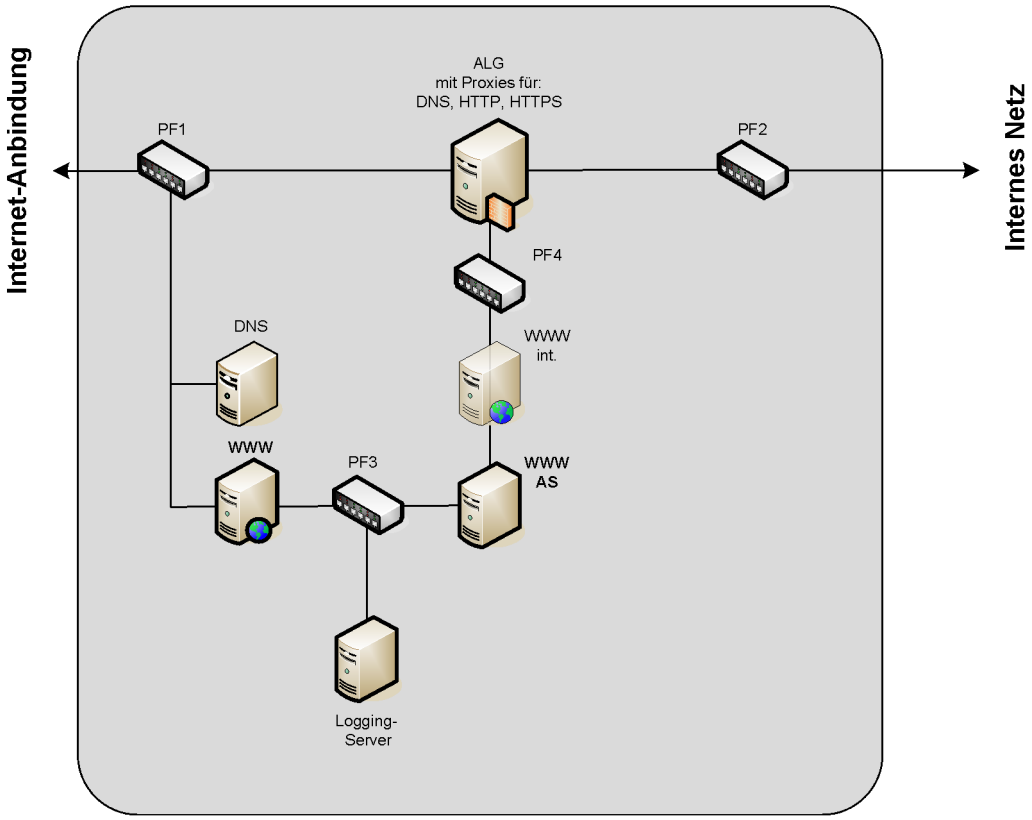


Abbildung 3.1: Grundarchitektur für einfache Web-Angebote ohne Hintergrundsysteme

3.1.2 Variante der Grundarchitektur für komplexere Web-Angebote

Komplexere Web-Angebote benötigen häufig Hintergrundsysteme wie Datenbanken oder Verzeichnis-Server um ihre Dienste anbieten zu können. Meist bieten solche Web-Angebote primär dynamische Funktionen und werden selten mit statischen Inhalten aktualisiert. Klassische Beispiele hierfür sind Suchmaschinen, Auktionsplattformen, Online-Banking-Systeme und Social-Networking-Plattformen. Erweiterungen zu diesen Portalen werden in der Regel von Entwicklern vorgenommen, die zusätzliche Funktionen hinzufügen, jedoch nicht von Redakteuren mit Hilfe von Redaktionstools verfasst.

Im Fokus der vorliegenden Studie sind die beiden Webserver WWW und WWW int. und der Web-Anwendungs-Server WWW AS (Abbildung 3.2). Dabei bezeichnet WWW den externen und WWW int den internen Webserver. Des Weiteren werden der Logging-Server und exemplarische Hintergrund-Komponenten, die sich in der Abbildung hinter Paketfilter 5 (PF5) befinden, behandelt. Als Hintergrundsysteme werden alle Systeme bezeichnet, die Funktionen oder Daten für den Web-Anwendungs-Server zur Verfügung stellen. Hintergrundsysteme werden in der Fachliteratur auch als Backend-Systeme bezeichnet. Die in dieser Studie angeführten Hintergrundsysteme sind als mögliche Varianten aufgeführt, in der Praxis jedoch sind selten alle vorhanden.

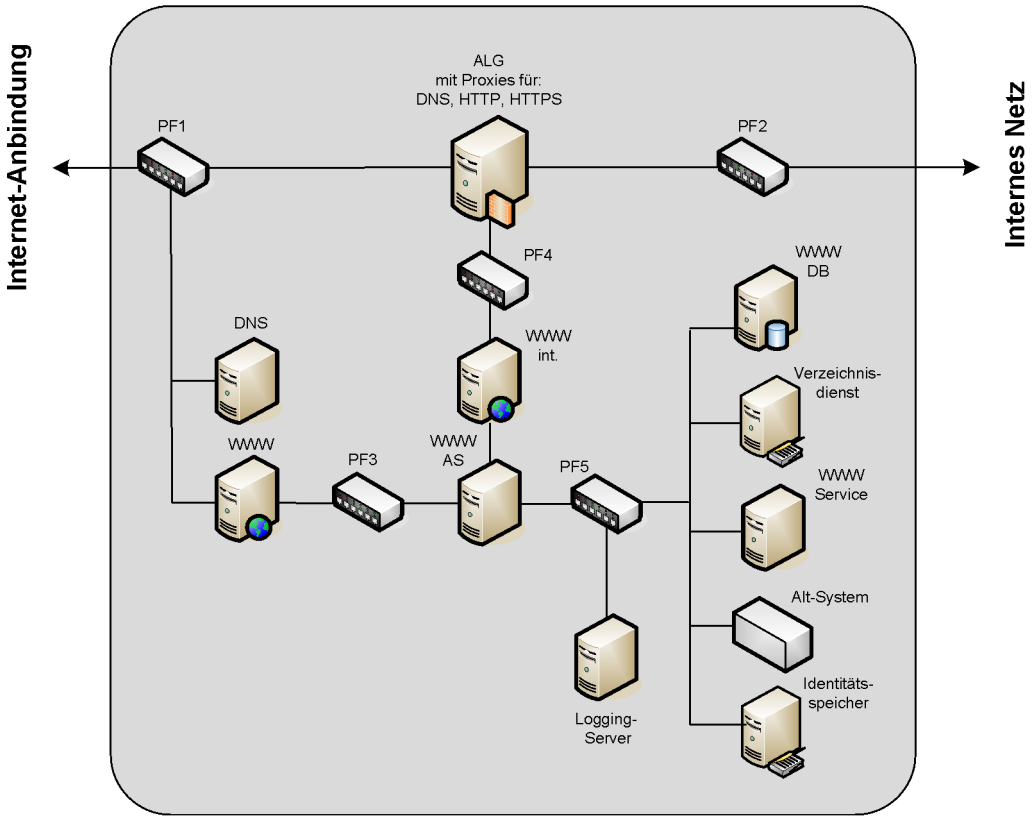


Abbildung 3.2: Sichere Grundarchitektur für komplexe Web-Angebote

3.1.3 Variante der Grundarchitektur mit CMS

Web-Angebote, die vor allem aktuelle Inhalte transportieren sollen, werden in der Regel auf CMS gestützten Architekturen implementiert. Klassische Beispiele hierfür sind Nachrichtenportale, E-Government-Web-Angebote, Weblogs und Informationsportale von Unternehmen. In diesen Fällen besteht die Notwendigkeit, dass auch Personen ohne technisches Hintergrundwissen mit Hilfe eines einfachen Redaktionstools (CMS) Inhalte des Web-Angebotes ändern können.

Zusätzlich zum Server WWW int. wird bei dieser Variante (Abbildung 3.3) ein eigenes Redaktionssystem eingesetzt. Alle Inhalte, die auf dem Redaktionssystem eingepflegt werden, werden auf den Web-Anwendungs-Server oder auf den Datenbank-Server übertragen, damit diese auch extern erreichbar sind. Dies stellt sicher, dass ein externer Angreifer keine Möglichkeit hat, auf Funktionen des CMS zuzugreifen, die potenziell Schwachstellen aufweisen können.

Es ist prinzipiell auch möglich, dynamische Inhalte über das CMS zu verwalten. Auch hier muss sichergestellt werden, dass lediglich die benötigten Funktionen vom Web-Anwendungs-Server bereitgestellt werden, um eine geringe Angriffsfläche zu bieten.

Viele CMS sind darauf ausgelegt, auf demselben System wie das Web-Angebot zu laufen. Derartige CMS sind ungeeignet für die in der Grundarchitektur geforderte Infrastruktur.

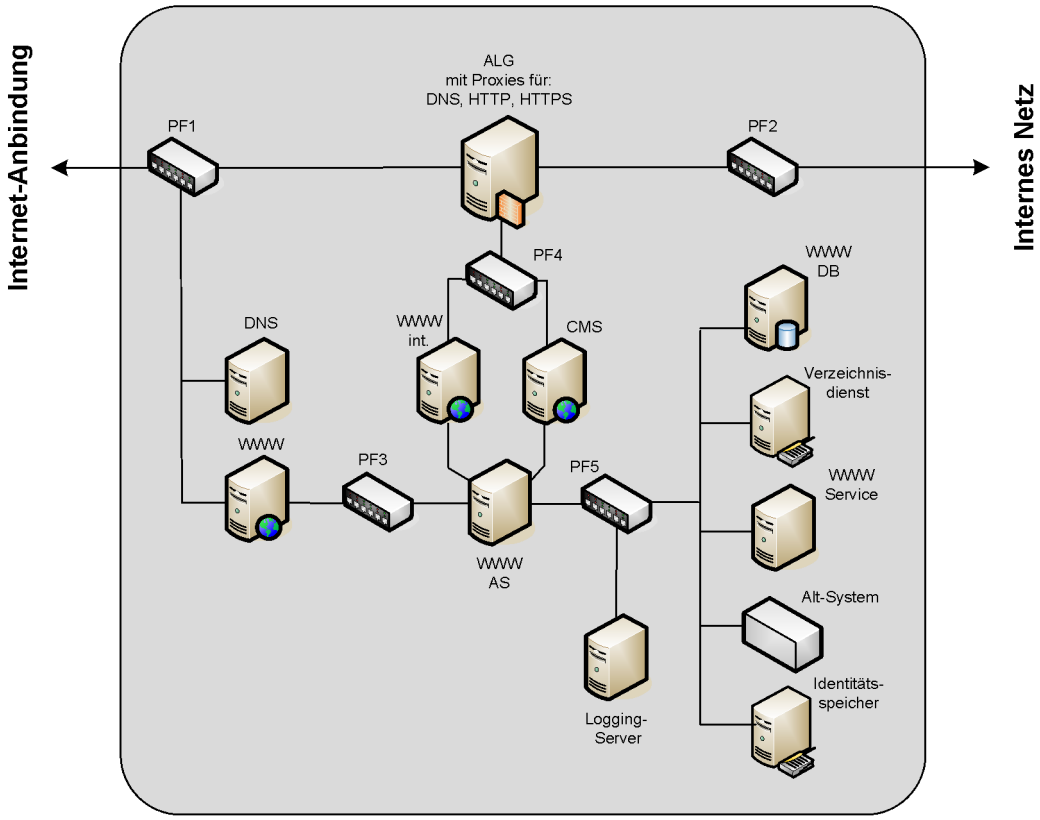


Abbildung 3.3: Sichere Grundarchitektur mit einem Content Management System

3.1.4 Management

Grundsätzlich werden die Komponenten aus der Management-Zone gemäß [ISi-LANA] administriert. Die Wartung des Web-Angebotes kann über die Administrationszugänge aus der Management-Zone heraus oder über ein CMS erfolgen.

Bei der Management-Zone handelt es sich um ein durch Paketfilter abgesichertes Netz, welches ausschließlich zur Administration und Wartung der eingesetzten Systeme verwendet wird. Abbildung 3.4 zeigt den Aufbau der Grundarchitektur für normalen Schutzbedarf gemeinsam mit der zugehörigen Management-Zone.

Durch Verwendung einer Management-Zone erfolgt eine vollständige Trennung der Nutzdaten-Schnittstellen von den Management-Schnittstellen der einzelnen Systeme, wodurch die Sicherheit erhöht wird. Fernadministration darf nur möglich sein, wenn sich der Administrator zuvor über eine gesicherte Verbindung von einem sicheren Client in die Management-Zone eingewählt hat. Von dort aus ermöglicht eine zentrale Management-Station die Administration der Systeme über eine gesicherte Verbindung. Die Uhrzeit aller Systeme wird mit Hilfe des Network Time Protocol (NTP) synchronisiert, wobei als Referenzzeitquelle ein DCF77-Empfangsmodul verwendet werden kann.

Aus Sicherheitsgründen sind die Teilsegmente der Management-Zone durch Paketfilter getrennt. Um den Hardware-Aufwand zu reduzieren, besteht bei erhöhtem Risiko die Möglichkeit, einzelne Paketfilter zusammenzulegen. Diese Möglichkeiten werden ausführlich in [ISi-LANA] beschrieben.

Für das Logging des Web-Angebotes wird aus Performancegründen ein separater Logging-Server vorgeschlagen. Wenn nur eine geringe Last vorliegt, kann jedoch auch das Logging in die Management-Zone verlagert werden.

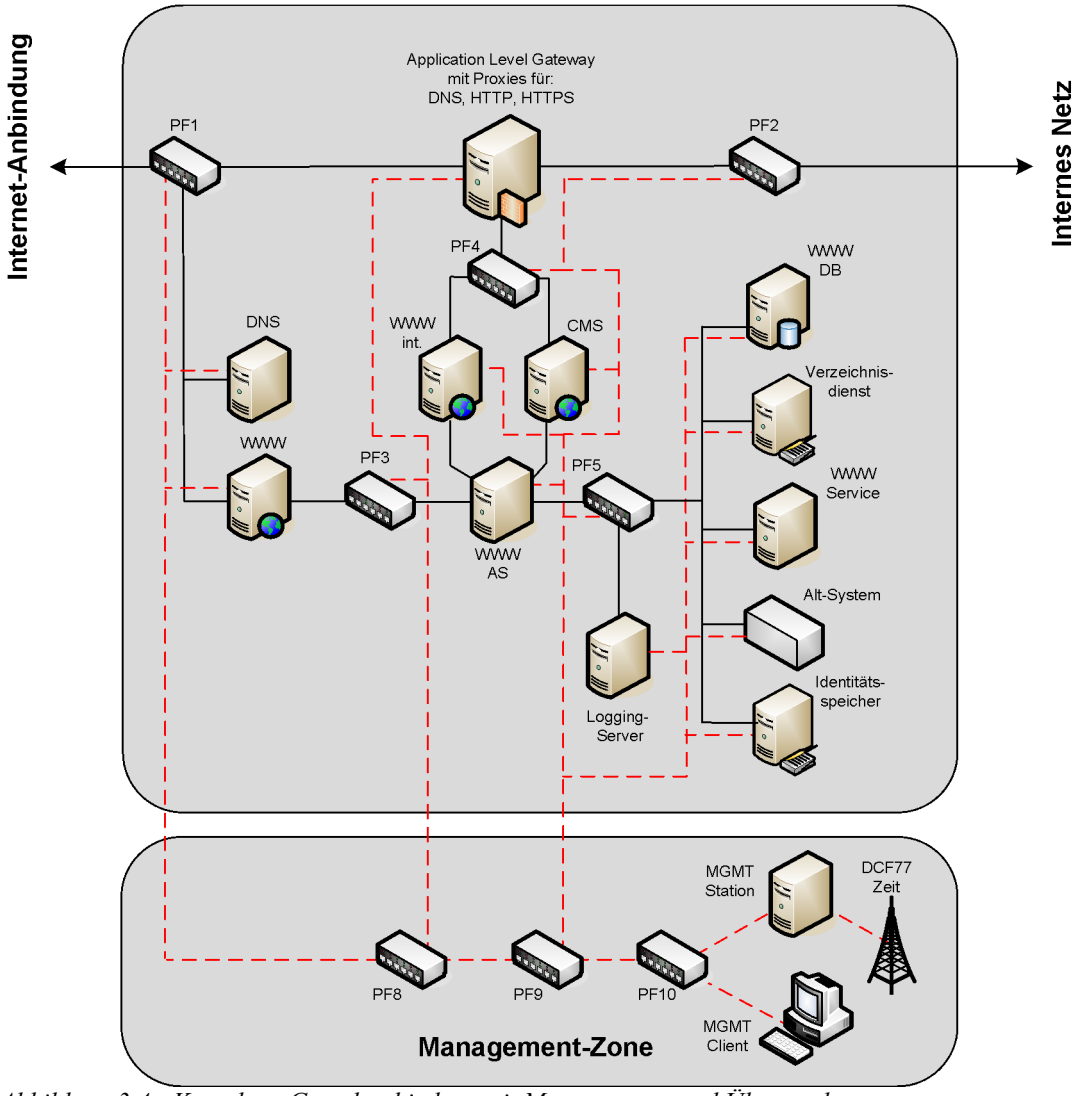


Abbildung 3.4: Komplette Grundarchitektur mit Management- und Überwachungsnetz

3.2 Komponenten der Grundarchitektur

Nachfolgende Abschnitte beschreiben die in den Abbildungen im Abschnitt 3.1 gezeigten Komponenten der Grundarchitektur näher. Detaillierte Informationen zu Auswahl, Konfiguration und Betrieb der Komponenten können in Abschnitt 4 nachgelesen werden.

3.2.1 Externer Webserver

Externe Benutzer des Web-Angebotes kommunizieren ausschließlich mit dem Webserver mit der Bezeichnung WWW in der ersten DMZ (hinter Paketfilter 1). Dieser wird nachfolgend externer Webserver genannt. Eine DMZ ist eine Demilitarisierte Zone, also ein Netz, in welchem kontrollierter Zugang zu einzelnen Diensten (z. B. zu einem Web-Angebot) erlaubt wird.

3.2.2 Interner Webserver

Für den Zugriff auf das Web-Angebot aus dem internen Netz steht ein eigener Webserver, nachfolgend interner Webserver (WWW int.) genannt zur Verfügung. Dieser befindet sich in derselben DMZ wie der Web-Anwendungs-Server. Um aus dem internen Netz auf diesen Webserver zugreifen zu können, muss die Anfrage über das ALG laufen. Das ALG (Application-Level Gateway) arbeitet auf der Anwendungsschicht des TCP/IP-Referenzmodells und kann aktiv in die Kommunikation zwischen Client und Server eingreifen. Der interne Webserver kann zusätzliche interne Informationen bereitstellen.

Über den internen Webserver wird auch das Web-Angebot gepflegt. Neue Inhalte können darüber zur Datenbank oder zum externen Webserver übertragen werden. Für diesen Zweck kann auch das in Abschnitt 3.1.3 beschriebene CMS zusätzlich oder alternativ Verwendung finden.

3.2.3 Web-Anwendungs-Server

Die Web-Anwendung selbst läuft auf dem Web-Anwendungs-Server. Der Zugriff auf diesen erfolgt jedoch nicht direkt, sondern ausschließlich über einen Webserver. Die Anforderungen an die Architektur der Web-Anwendung werden in Abschnitt 3.3 näher erläutert, die Themen „Sichere Beschaffung“ bzw. „Entwicklung“ in Abschnitt 4.

Da die Verarbeitung einer Anfrage im Web-Anwendungs-Server ein sehr komplexer Vorgang sein kann, sind eine Reihe von Sicherheitsaspekten zu berücksichtigen:

- Die Kommunikation zwischen Web-Anwendungs-Server und Hintergrundsystemen sollte nur dann unverschlüsselt erfolgen, wenn weitere, dem Schutzbedarf der übertragenen Daten angemessene Maßnahmen getroffen wurden, um zu verhindern, dass ein fremder Zugriff auf das verwendete Netz durchgeführt werden kann. Erfolgt die Kommunikation mit den Hintergrundsystemen über nicht vertrauenswürdige Netze (z. B. über das Internet), so darf die Datenübertragung ausschließlich verschlüsselt erfolgen.
- Benutzer und Webserver müssen authentisiert und autorisiert werden. Darunter wird zum einen die Verifikation der Identität eines Benutzers verstanden, welche beispielsweise durch Eingabe eines Benutzernamens und des zugehörigen Kennwortes erreicht wird. Zum anderen fällt die Zuweisung von Zugriffsrechten, welche mit der Identität verbunden sind, darunter. Authentisierung und Autorisierung werden deshalb getrennt behandelt, da sie aus sicherheitstechnischer Sicht grundlegend verschiedene Vorgänge beschreiben. Ein Angreifer kann beispielsweise nach erfolgreicher Authentisierung mit einem minderprivilegierten Benutzerkonto durch Ausnutzen einer Schwachstelle Aktionen durchführen, zu welchen er eigentlich nicht autorisiert ist.
- Es muss ein sicheres Session-Management implementiert sein, wenn Nutzer authentisiert werden sollen. Das Session-Management muss auf eine sichere Art und Weise umgesetzt werden, um Identitätsdiebstahl verhindern zu können. Typische Angriffe aus diesem Bereich sind Session Riding (Ausführen von Befehlen unter der Benutzung fremder Sessions) und Session Hijacking (Übernehmen fremder Sessions).

- Der Zugriff auf Dateien und das lokale Einbinden von Skripten am Webserver oder Web-Anwendungs-Server muss abgesichert werden. Dies geschieht durch Festlegung einer Liste an Dateien, welche explizit zur Einbindung erlaubt sind (sogenanntes Whitelisting). Alle Ressourcen, die nicht in der Liste aufgeführt sind, dürfen durch die Web-Anwendung nicht eingebunden werden. Ein Angreifer hat dadurch auch nicht die Möglichkeit auf die von ihm auf dem Server abgelegten Dateien oder Skripte zuzugreifen.
- Um die Ausfallsicherheit zu erhöhen, sollte ein Cluster realisiert werden. Durch Bildung eines Clusters werden mehrere Web-Anwendungs-Server so verbunden, dass bei Ausfall eines Systems eingehende Anfragen von den jeweils anderen Systemen übernommen werden. Der Ausfall eines Systems soll für einen Benutzer möglichst transparent sein und ohne Unterbrechung oder Beeinträchtigung des Dienstes erfolgen. Cluster werden ebenfalls dazu verwendet, um die Last eines einzelnen Servers zu verringern, indem eingehende Anfragen auf alle Systeme des Clusters gleichmäßig aufgeteilt werden. Beim Betrieb von Clustern muss jedoch auf eine geeignete Konfiguration geachtet werden, um Wettlaufsituationen (Race Conditions) zu vermeiden. Das bedeutet, dass zwei Web-Anwendungs-Server nicht gleichzeitig Sitzungsdaten eines Benutzers verändern dürfen, da sonst eine der beiden Änderungen durch die jeweils andere überschrieben wird. Eine solche Lösung ist in Umgebungen mit erhöhten Anforderungen an die Verfügbarkeit sinnvoll.

3.2.4 Hintergrund-Komponenten

Alle Hintergrund-Komponenten befinden sich in einer DMZ, die durch Paketfilter PF5 vom Web-Anwendungs-Server getrennt ist. Bei Bedarf können die Hintergrundkomponenten noch weiter voneinander separiert in unterschiedlichen DMZ untergebracht werden. Folgende Komponenten sind Beispiele für Hintergrundsysteme:

- Datenbanken
- Verzeichnis-Server
- Web-Services
- Alt-Systeme
- Identitätsspeicher

Die Systeme müssen jedoch nicht notwendigerweise auf getrennten Hardware-Systemen untergebracht werden. Eine Trennung auf einer Hardware unter Verwendung Virtueller Maschinen ist auch denkbar. Die in den Abbildungen gewählte getrennte Darstellung dient nur der Übersichtlichkeit.

3.2.5 Logging-Server

Das Logging-System befindet sich in einer eigenen DMZ hinter Paketfilter PF5 und muss besonders gut geschützt sein. Sowohl der Webserver als auch der Web-Anwendungs-Server und ggf. die Hintergrundsysteme schreiben ihre Log-Einträge auf diesen Server. Es muss durch geeignete Logging-Frameworks sichergestellt werden, dass die Systeme auf dem Logging-Server nur neue Nachrichten hinzufügen, jedoch keine alten Einträge lesen, verändern, überschreiben oder löschen können. Im Falle einer Kompromittierung des Webservers bzw. des Web-Anwendungs-Servers darf es einem Angreifer weder möglich sein, Lesezugriff auf den Logging-Server zu erlangen noch dessen Logging-Inhalte zu verändern. Andernfalls könnte ein Angreifer seine Spuren verwischen oder Informationen über andere Systeme erlangen.

3.3 Grundarchitektur der Web-Anwendung

Als Sammelbegriff für die Sicherheitsaspekte, die im Zusammenhang mit der Architektur und der Implementierung von Web-Anwendungen berücksichtigt werden müssen, hat sich der Begriff Web-Application-Security durchgesetzt.

Web-Application-Security

Da Web-Anwendungen in der Regel, wie viele andere Netzdienste, von überall aus dem Internet zugänglich sind, sind diese besonderen Bedrohungen ausgesetzt. Aus diesem Grund haben sich in der Vergangenheit verschiedene Praktiken etabliert, um auf die Vielzahl von Bedrohungen reagieren zu können. Die folgende Liste deckt die wesentlichen Punkte ab, welche zum sicheren Betrieb von Web-Angeboten beachtet werden müssen:

- Verwendung einer sicheren Architektur – Die Architektur einer Web-Anwendung sollte so gewählt werden, dass sie bei Bedarf eine verschlüsselte Speicherung und den sicheren Austausch von vertraulichen Daten erlaubt. Zudem müssen Möglichkeiten für die Implementierung von Filtern von eingehenden und ausgehenden Daten zur Verfügung gestellt werden. Filter werden dazu verwendet, um die Durchführung von Angriffen und die Ausnutzung von Schwachstellen zu erschweren.
- Implementierung eines Konfigurationsmanagements – Konfigurationsdateien müssen geschützt werden und dürfen nicht über die Web-Anwendung ausgelesen werden können, da sie einem Angreifer wertvolle Informationen über die Interna der Web-Anwendung liefern können.
- Absicherung der Einbindung von Dateien – Bindet die Web-Anwendung externe Dateien ein, welche zur Ausführung benötigt werden, müssen besondere Sicherheitsanforderungen berücksichtigt werden. So darf es beispielsweise einem Angreifer nicht möglich sein, eigene Dateien und somit eigene Funktionen in das Web-Angebot einzubinden, da sonst die Kontrolle über das Web-Angebot übernommen werden kann.
- Einschränkung der Ausführung externer Programme – Dem Benutzer darf es nicht möglich sein, mit Hilfe der Web-Anwendung externe Programme aufzurufen und somit deren Ausführung zu beeinflussen. Besonders anfällig sind Web-Angebote, welche externe Programme aufrufen und dabei Benutzereingaben als Parameter verwenden. Werden diese Parameter nicht einer umfassenden Filterung unterzogen, so kann ein Angreifer beliebige Befehle ausführen.
- Einschränkung von Datei-Uploads – Wird den Benutzern eine Funktion zum Hochladen von Dateien geboten, ist darauf zu achten, dass deren Endungen und Inhalte den erwarteten Datei-Typen entsprechen. Andernfalls ist es Angreifern möglich, eigene Skripte hochzuladen und somit die Kontrolle über den Webserver zu erlangen. Außerdem muss eine maximal erlaubte Dateigröße definiert werden. Zusätzlich ist es notwendig, die hochgeladenen Dateien einer Prüfung auf Schadprogramme zu unterziehen.
- Filterung von System- und Fehlermeldungen – Da System- und Fehlermeldungen häufig Systeminformationen und Details über die Interna einer Anwendung enthalten, dürfen diese nicht direkt an den Benutzer ausgegeben werden. Dem Benutzer dürfen Fehlermeldungen nur in einer Form angezeigt werden, die keine für einen Angreifer nützlichen Informationen enthalten.
- Verwendung von Kryptografie – Wenn Verschlüsselung eingesetzt wird, so müssen die verwendeten kryptografische Algorithmen und Prozeduren dem aktuellen Stand der Technik entsprechen (siehe [BSI-TR-02102]) und für den Einsatz auf der Zielplattform getestet wurden sein. Es dürfen keine kryptografischen Funktionen oder Algorithmen verwendet werden, für welche bekannte Schwachstellen im jeweiligen Anwendungsfall existieren.

- Absicherung von Administrationszugängen – Administrationszugänge müssen in ausreichender Weise vom Benutzer-Bereich getrennt werden.
- Prüfung der Validität von Eingaben – Jede Form von Eingaben, die vom Benutzer an die Web-Anwendung übermittelt wird, muss einer umfassenden Validierung unterzogen werden, bevor sie weiterverarbeitet wird. Es dürfen nur jene Eingaben akzeptiert werden, welche das Verhalten der Web-Anwendung nicht in unvorhergesehener Weise beeinflussen können. Unabhängig von einer Client-seitigen Validierung müssen Benutzereingaben immer auch auf der Server-Seite überprüft werden. Der Grund dafür ist, dass Client-seitige Validierungsmaßnahmen (wie sie beispielsweise mit Hilfe von JavaScript durchgeführt werden können) sehr leicht von Angreifern umgangen werden können.
- Umsetzung von Authentisierung und Integritätsprüfung von Anfragen – Wenn es notwendig ist, dass zwischen den einzelnen Benutzern einer Web-Anwendung unterschieden wird, so müssen diese authentisiert werden. Durch eine geeignete Form der Authentisierung wird ferner sichergestellt, dass jedem Benutzer nur jene Rechte zugewiesen werden, die er für die Ausführung einer bestimmten Funktion tatsächlich benötigt. Durch diese Vorgehensweise (Minimalprinzip bzgl. der Rechtevergabe) wird sichergestellt, dass die Angriffsfläche möglichst klein gehalten wird.
- Schutz des Quelltextes und anderer sensitiver Systeminformationen – Der Quelltext von Anwendungen sowie andere sensitive Systeminformationen müssen vor Zugriff durch nicht autorisierte Personen geschützt werden. Beispielsweise dürfen auf einem Server im Produktiveinsatz keine Backups einzelner Komponenten verfügbar sein.
- Sichere Einbindung von Hintergrundsystemen – Der Zugriff auf Hintergrundsysteme muss über ein eigenes Netz durchgeführt werden, zu welchem Benutzer keinen direkten Zugriff haben. Kann dies nicht sichergestellt oder umgesetzt werden, so darf die Kommunikation ausschließlich verschlüsselt und mit entsprechender Authentisierung erfolgen.

Schema der Grundarchitektur

Um ein Web-Angebot sicher bereitstellen zu können, muss vor allem die Architektur der Web-Anwendung so beschaffen sein, dass sie die Umsetzung der Anforderungen der Web-Application-Security unterstützt.

Die nachfolgende Abbildung 3.5 illustriert, welche Komponenten wie zusammenspielen müssen, um sowohl die Web-Anwendung selbst als auch die angebundenen Hintergrund-Komponenten zu schützen.

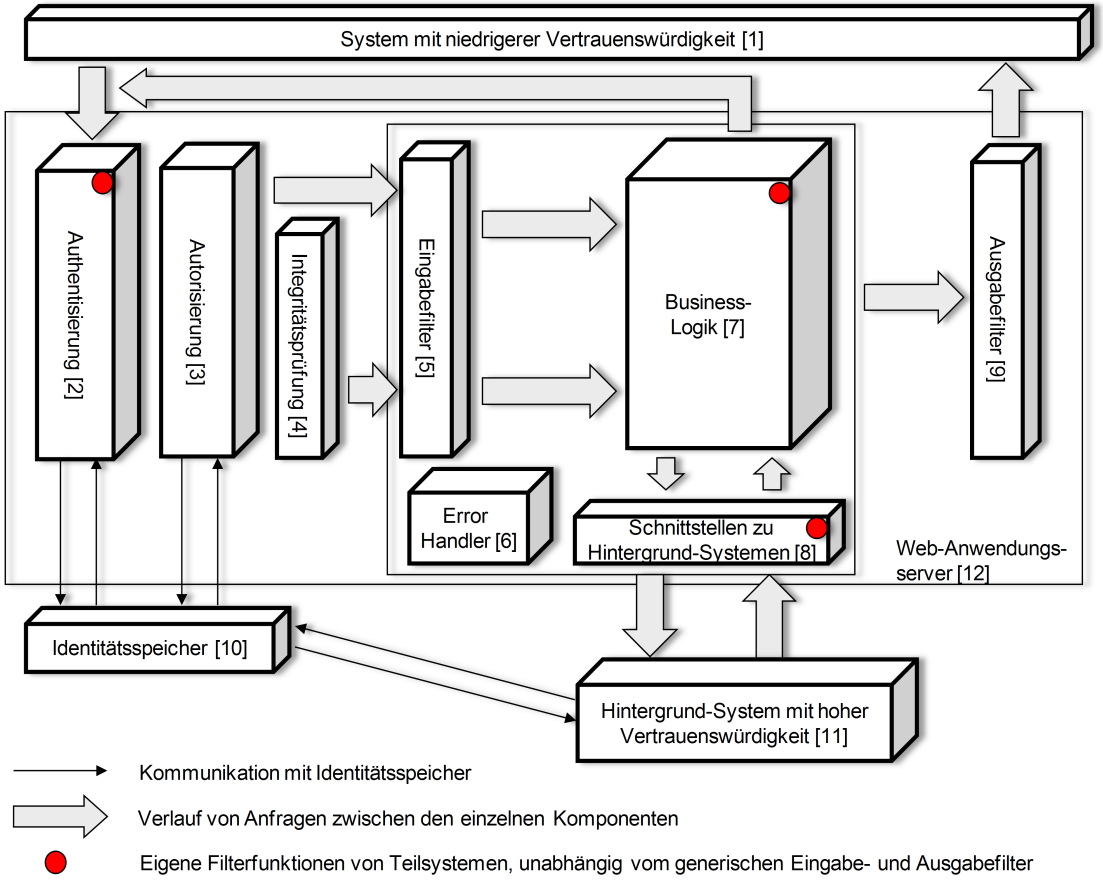


Abbildung 3.5: Architektur einer sicheren Web-Anwendung

Grenzen des Vertrauens

Um übergebene Parameter richtig behandeln zu können, müssen Grenzen des Vertrauens definiert werden. Als solche gelten alle Übergänge zwischen zwei Bereichen unterschiedlicher Vertrauenswürdigkeit. Diese Übergänge können sich innerhalb der Anwendung befinden oder aber auch an Schnittstellen zu anderen Systemen auftreten. Eine klassische Grenze des Vertrauens besteht bei der Übergabe von Benutzereingaben an die Anwendung, da die Benutzereingaben als nicht vertrauenswürdig eingestuft werden müssen. Eine solche Grenze besteht deshalb, da ein Angreifer mit Hilfe manipulierter Eingabedaten möglicherweise das Verhalten des Web-Angebotes beeinflussen kann.

Jegliche Daten, die eine Vertrauensgrenze überqueren, müssen vom Empfängersystem gefiltert werden, um potenzielle Angriffe abzuwehren.

Kommunikation zwischen den Subsystemen

Die Kommunikation zwischen Subsystemen muss gesichert erfolgen. Dies bedeutet, dass bei unsicheren Kanälen zusätzliche Schutzmaßnahmen durch geeignete Anwendung kryptografischer Verfahren (z. B. Verschlüsselung) zu treffen sind. Gelingt einem Angreifer die Kompromittierung eines Systems, ist er dadurch nicht in der Lage, die Kommunikation weiterer Systeme abzuhören und zusätzliche Angriffspunkte herauszufinden.

System mit niedrigerer Vertrauenswürdigkeit [1]

Als System mit niedriger Vertrauenswürdigkeit werden alle Systeme bezeichnet, die sich nicht authentisieren bzw. die zwar authentisiert aber dem Web-Anwendungs-Server nicht als vertrauens-

würdig bekannt sind. Gängiges Beispiel hierfür ist der Client bzw. der Zugriff des Clients über den vorangestellten Webserver. Da vom Betreiber des Web-Angebotes kein Einfluss auf die Clients ausgeübt werden kann, gelten diese als Systeme niedrigerer Vertrauenswürdigkeit.

Authentisierung [2]

Mit Hilfe der Authentisierung wird sichergestellt, dass Benutzer, beispielsweise durch die Eingabe von Benutzername und Kennwort, gegenüber dem Web-Angebot identifiziert werden. Hierfür ist eine Kommunikation mit dem Identitätsspeicher [10] erforderlich. Da eine Authentisierung für jede Anfrage eines Clients an die Web-Anwendung erfolgt, wird für anonyme Zugriffe ein eigener Benutzer verwendet. Zudem wird der anonyme Benutzer automatisch verwendet, falls der Authentisierungsversuch fehlschlägt. Eine Authentisierung muss für jede Interaktion zwischen Benutzer und IT-System erfolgen und eine Weiterbearbeitung einer Anfrage darf nur nach erfolgreicher Authentisierung erfolgen. Hierfür ist jedoch die Authentisierung als anonymer Benutzer ausreichend.

Es wird zwischen zwei Zugriffstypen unterschieden:

- Anonyme Zugriffe. Diese Zugriffe besitzen die niedrigste Berechtigungsstufe und es ist keine Prüfung der Integrität der Anfragen notwendig.
- Zugriff durch bekannte Benutzer oder Systeme. Diese Zugriffe verfügen über die jeweils zugeordneten Berechtigungen und müssen daher auch auf ihre Integrität hin überprüft werden.

Da sich die Authentisierung vor dem Eingabefilter befindet, muss die Authentisierung selbst eigene Filter implementieren, um sich vor Angriffen zu schützen.

Autorisierung [3]

Mit Hilfe der Autorisierungdaten wird überprüft, ob ein Benutzer über die erforderlichen Berechtigungen verfügt, um auf die angeforderte Geschäfts-Logik bzw. Hintergrundsysteme zugreifen zu dürfen. Wie bei der Authentisierung ist für diese Überprüfung eine Kommunikation mit dem Identitätsspeicher [10] erforderlich. Für die Autorisierung werden keine Benutzereingaben verwendet, da diese direkt von der Authentisierung angesprochen wird.

Integritätsprüfung [4]

Eine Integritätsprüfung findet nur dann statt, wenn ein Zugriff durch einen authentisierten Benutzer oder ein authentisiertes System erfolgt. Die Prüfung der Integrität der Anfrage dient der Sicherstellung, dass die Anfrage auf dem Weg zum Server nicht verändert wurde.

Eingabefilter [5]

Der Eingabefilter dient dem Schutz der Geschäfts-Logik vor schädlichen oder fehlerhaften Benutzereingaben. Der Zugriff auf externen Input erfolgt ausschließlich über diesen restriktiven Filter.

Der Eingabefilter wird zentral implementiert, um sicherzustellen, dass alle Daten eine Überprüfung hinsichtlich des Inputs durchlaufen. Zudem wird der Eingabefilter benutzt, um gesichert auf Variablen zuzugreifen. Eine Filterfunktion kann auch in abstrahierten Schnittstellen zu Hintergrundsystemen (z. B. Datenbanken) implementiert werden.

Error-Handler [6]

Mit Hilfe eines Error-Handlers wird sichergestellt, dass die Anwendung bei Auftreten unerwarteter Fehler in einen sicheren Zustand übergeht. Dabei muss auch gewährleistet werden, dass offene Sessions ordnungsgemäß terminiert werden. Der Error-Handler muss so implementiert werden, dass dieser keine sensiblen Informationen in Fehlermeldungen ausgibt.

Geschäfts-Logik [7]

Die Geschäfts-Logik ist eine zentrale Komponente einer Web-Anwendung, da sie die eingehenden Anfragen bearbeitet und die Ergebnisse an den Ausgabefilter weitergibt. Hierfür greift die Geschäfts-Logik auf Hintergrundsysteme oder andere Geschäfts-Logiken zu. Der Zugriff auf Hintergrundsysteme darf nur über definierte Schnittstellen erfolgen.

Bei der Implementierung der Geschäfts-Logik ist das Minimalprinzip einzuhalten. Dies bedeutet, dass für die Verarbeitung von Daten und die Ausführung von Operationen nur die mindestens erforderlichen Rechte verwendet werden dürfen.

Als Ergänzung zum Eingabefilter können in der Geschäfts-Logik zusätzliche kontextsensitive Filtermodule implementiert werden. Dies erlaubt eine feinere Filterung als ausschließlich mit dem Eingabefilter.

Schnittstellen zu Hintergrundsystemen [8]

Um einen sicheren Zugriff auf Hintergrundsysteme zu ermöglichen, erfolgt dieser ausschließlich über entsprechende Schnittstellen (z. B. über Stored Procedures oder Frameworks). Diese Schnittstellen können bei Bedarf zusätzliche Filter enthalten.

Erfolgt ein Zugriff auf ein System mit höherer Vertrauenswürdigkeit, wird zusätzlich eine Überprüfung der Authentizität und der Integrität der Anfragen und Antworten vorgenommen. Auf diese Weise wird sichergestellt, dass die Inhalte tatsächlich vom vertrauenswürdigen System kommen und nicht auf dem Weg zum Server manipuliert wurden.

Wie in Abbildung 3.2 gezeigt, befinden sich die angesprochenen Hintergrundsysteme in der Grundarchitektur der Infrastruktur hinter dem Paketfilter 5.

Ausgabefilter [9]

Ähnlich wie Benutzereingaben sind auch die Ausgaben der Web-Anwendung in geeigneter Weise zu filtern. Auf diese Weise wird eine Anfälligkeit gegenüber Cross-Site Scripting und anderen Angriffsmöglichkeiten auf den Browser verringert. Dadurch wird es einem Angreifer erschwert, Code auf den Clients der Benutzer des Web-Angebotes auszuführen.

Falls es sich bei dem zugreifenden Client nicht um einen Browser handelt, können auch andere alternative Filter für den jeweiligen Client eingesetzt werden.

Identitätsspeicher [10]

Der Identitätsspeicher dient der Ablage aller Benutzer- und Systemkonten. Diese Konten beinhalten Informationen über die Stufe der Vertraulichkeit und über die Berechtigungen auf einzelne Ressourcen (Geschäfts-Logik, Ressourcen von Hintergrundsystemen, ...).

Die Implementierung des Identitätsspeichers kann auf unterschiedliche Arten erfolgen, beispielsweise über eine Hintergrund-Datenbank oder einen Hintergrund-Verzeichnisdienst. Da den Kontodaten im Identitätsspeicher eine zentrale Rolle zukommt, muss für einen entsprechenden Schutz dieser Informationen gesorgt werden.

Die Grundarchitektur der Infrastruktur sieht eine Platzierung des Identitätsspeichers hinter Paketfilter PF5 vor.

Hintergrundsysteme mit hoher Vertrauenswürdigkeit [11]

Hintergrundsysteme mit hoher Vertrauenswürdigkeit vertrauen einer Web-Anwendung nicht aus Prinzip (kein Trusted-Client-Prinzip). Möchte eine Web-Anwendung Daten vom Hintergrundsystem lesen oder auf diese schreibend zugreifen, ist eine Authentisierung erforderlich. Hierfür werden die Authentisierungsmerkmale des Clients von der Geschäfts-Logik an das Hintergrundsystem weitergeleitet. Erst wenn die übermittelten Zugangsdaten verifiziert wurden, können Hintergrundsysteme der Web-Anwendung vertrauen und ihr die nach dem Minimalprinzip erforderlichen Rechte einräumen. Um zusätzliche Sicherheit zu erlangen, kann die Integrität der Kommunikation mit Hilfe von kryptografischen Signaturen sichergestellt werden.

Web-Anwendungs-Server [12]

All die zuvor beschriebenen Komponenten stellen den Web-Anwendungs-Server dar. In einer SOA-Umgebung (siehe Abschnitt 2.3.8) können einzelne Blöcke als eigene Web-Services implementiert werden, in einer herkömmlichen Architektur sind jedoch alle Teilkomponenten im Web-Anwendungs-Server integriert.

4 Komponenten sicher auswählen, konfigurieren und betreiben (normaler Schutzbedarf)

In dem vorangegangenen Abschnitt wurden verschiedene Architekturen zum sicheren Bereitstellen von Web-Angeboten vorgestellt. In den drei folgenden Unterabschnitten sollen nun umfassende Empfehlungen gegeben werden, wie die in der Grundarchitektur enthaltenen Komponenten sicher ausgewählt, konfiguriert und betrieben werden können.

4.1 Grundanforderungen an ein sicheres Produkt

Der folgende Abschnitt beschreibt grundlegende Anforderungen an Komponenten der Grundarchitektur, die bereits bei der Beschaffung des Produkts berücksichtigt werden müssen. Allgemeine Mindestanforderungen an die Ausstattung und Leistungsumfang von Netz-Komponenten befinden sich in [ISi-LANA].

4.1.1 Betriebssystem

Diese Studie gibt keine Empfehlungen für bestimmte Betriebssysteme. Vielmehr werden Kriterien aufgelistet, die bei der Wahl eines Betriebssystems berücksichtigt werden sollten.

Ein wesentlicher Punkt bei der Wahl des Betriebssystems ist eine angemessene fortlaufende Unterstützung durch den Hersteller, um in regelmäßigen Abständen Sicherheits-Updates für vorhandene Schwachstellen zu erhalten. Bei der Verfügbarkeit von Sicherheits-Updates sind zwei Aspekte zu beachten. Zum einen die Dauer, bis bekannte Schwachstellen gefixt werden, und zum anderen die Anzahl und Schwere der bekannten Schwachstellen. Zusätzlich sollte die Möglichkeit bestehen, Sicherheits-Updates auf vertrauenswürdigem Weg zu erhalten (z. B. versehen mit digitaler Signatur des Herstellers).

Manche Betriebssysteme bieten zusätzliche Sicherheitserweiterungen wie beispielsweise „non-executable stacks“. Dabei handelt es sich um eine Sicherheitsfunktion, welche die Ausnutzung von Speicher-Management-Schwachstellen erschwert. Diese und ähnliche Erweiterungen erlauben es, ein Betriebssystem vor etwaigen Angriffen besser zu schützen.

Unter Linux kann die Verwendung von SELinux die Möglichkeiten eines Angreifers stark einschränken. SELinux steht für Security-Enhanced Linux und ist eine Erweiterung des Linux-Kernels und verschiedener Systemprogramme um eine verbesserte Zugriffskontrolle. Die durch SELinux eingeführte Zugriffskontrolle setzt das Prinzip des „Mandatory Access Control“ um, welches Ressourcen und Prozesse mit Attributen versieht und Zugriffe anhand von diesen und definierten Regeln steuert. SELinux arbeitet komplett unabhängig von der traditionellen Linux-Zugriffskontrolle und ist in vielen Distributionen standardmäßig implementiert oder kann über den Paketmanager nachinstalliert werden.

Beim Einsatz mehrerer Systeme ist auch auf eine zentrale Management-Fähigkeit der Betriebssysteme zu achten. Auf diese Weise wird die Administration wesentlich vereinfacht und somit auch sicherer gestaltet. Zudem sollte Logging auf einen externen Server möglich sein, um Log-Daten zentral zu sammeln und auszuwerten.

4.1.2 Webserver

Der Webserver ist die zentrale Schnittstelle zum Benutzer und somit die erste Angriffsfläche. Aus diesem Grund ist es notwendig, bereits bei der Auswahl des Webservers sehr genau festzulegen, welche Sicherheitsmaßnahmen der Webserver unterstützen muss.

Für manche Webserver gibt es spezielle Sicherheits-Frameworks (z. B. mod_security für Apache), mit denen verschiedene Sicherheitsmaßnahmen, wie beispielsweise eine Eingabefilterung, implementiert werden können. Einzelne Sicherheitsfunktionen lassen sich zum Teil auch mit anderen Erweiterungen für Webserver realisieren. So kann beispielsweise mit dem Modul mod_rewrite eine Regel-basierte Weiterleitung und somit eine Filterung der Anfragen erfolgen.

Ein wichtiger Punkt bei der Wahl eines Webservers ist die Flexibilität und Qualität des Rechtesystems. Eine der gebräuchlichsten Varianten dabei ist die Anwendung von Benutzerkonten, Gruppen oder Rollen. Rechte können jedoch auch auf IP-Ebene zur Autorisierung vergeben werden.

Zu beachten sind auch die Rechte, die der Webserver selbst benötigt, um ordnungsgemäß laufen zu können. Gemäß dem Minimalprinzip sollte dies mit möglichst geringen Rechten erfolgen. Um zu verhindern, dass aus dem Verzeichnis eines kompromittierten Webservers ausgebrochen werden kann, sollte dieser eine sogenannte Chroot-Umgebung unterstützen. Dabei handelt es sich um einen abgeschlossenen Bereich, aus welchem ein Angreifer nicht ohne Weiteres heraus kann, sollte ihm die Kompromittierung des Webservers gelingen. Diese Maßnahme erfordert allerdings, dass alle vom Webserver benötigten Dateien ebenfalls im Verzeichnis des Webservers und somit im abgeschotteten Bereich abgelegt werden.

Um eine sichere Kommunikation mit dem Client zu ermöglichen, muss der Webserver gängige Zertifikat-basierte Verschlüsselungsmethoden wie beispielsweise TLS unterstützen.

Ähnlich wie beim Betriebssystem ist es auch beim Webserver wichtig, dass eine langfristige Unterstützung durch den Hersteller in Form von regelmäßigen Sicherheits-Updates und Erweiterungen gegeben ist. Außerdem sollte Logging auf einen externen Server unterstützt werden.

4.1.3 Content Management System

Wird für den Betrieb eines Web-Angebotes ein Content Management System benutzt, so sollten dabei folgende Aspekte berücksichtigt werden:

- Grundsätzlich sollte das CMS möglichst schlank implementiert werden. Dies bedeutet, dass nur Funktionen zur Verfügung gestellt werden dürfen, die für den reibungslosen Betrieb nötig sind. Ein darüber hinausgehender Funktionsumfang bietet unnötige Angriffsflächen und erschwert effiziente Sicherheitsüberprüfungen.
- Da ein CMS auch oft eine eigene Web-Anwendung ist, gelten hier die gleichen Anforderungen wie für Web-Anwendungen.
- Zwischen Administration und Darstellung sollte eine Trennung möglich sein. Das bedeutet, dass jener Server, welcher die Inhalte nach außen hin zum Abruf bereithält, aus Sicherheitsgründen nicht gleichzeitig über eine Administrationsoberfläche zur Änderung der Inhalte verfügen darf. Änderungen des Web-Angebotes dürfen nur über den internen Web-Server oder ein Content Management System vorgenommen werden.
- Das CMS muss eine Rechteverwaltung anbieten, welche mindestens die Rollen Administrator und Redakteur zur Verfügung stellt. Idealerweise sollte das Rechtekonzept ermöglichen, unterschiedliche Berechtigungsstufen für jedes Element der Webseite nutzerabhängig zu vergeben.

Eine Hilfestellung bei der Auswahl eines geeigneten CMS bietet die CMS-Studie des BSI, bei der die Sicherheit von fünf ausgewählten Content-Management-Systemen im Detail untersucht wird [CMS-Studie].

4.1.4 Logging-Server

Logging-Server dienen der zentralen Sammlung und Auswertung von Logging-Informationen. Falls kein Intrusion Detection System verwendet wird, sollte es der Logging-Server ermöglichen, typische Angriffsmuster zu erkennen. In diesem Fall ist es zudem erforderlich, Schwellenwerte definieren zu können, bei denen eine Benachrichtigung an den Administrator erfolgt. Logging-Daten müssen geschützt werden. Um die Log-Einträge beispielsweise vor unbefugter Manipulation zu schützen, sollte es möglich sein, diese mit kryptografischen Zeitstempeln zu versehen. Dadurch kann bei einer späteren Analyse festgestellt werden, zu welcher Zeit die Einträge erstellt und ob diese von Unbefugten verändert wurden.

4.1.5 Hintergrundsysteme

Hintergrundsysteme beinhalten häufig vertrauliche Daten und müssen daher eine geeignete Authentisierung und Autorisierung unterstützen. Zusätzlich sollte ein konfigurierbarer Zugriffsschutz auf Basis eines Rollen-Modells möglich sein. Auf diese Weise kann beispielsweise der Zugriff eines Benutzers auf Teile der Datenbank beschränkt werden. Auf ähnliche Weise lassen sich auch Funktionen, die ein Benutzer oder eine Rolle ausführen darf, einschränken. Dies kann beispielsweise mit ausgewählten Stored Procedures für Datenbanken realisiert werden. Dabei handelt es sich um zusammenhängende Abfolgen von Datenbank-Anweisungen, welche auch mit Bedingungen (z. B. Inhaber einer bestimmten Rolle) hinterlegt werden können.

4.1.6 Web-Anwendung

Grundprinzipien bei der Auswahl bzw. Entwicklung von Web-Anwendungen

Für den langfristig stabilen und sicheren Betrieb einer Web-Anwendung ist es wichtig zu wissen, wie lange die aktuelle Version eingesetzt werden kann, ob Updates ohne Beeinträchtigung von momentan genutzten Funktionen installiert werden können und wie lange Sicherheitsupdates verfügbar sind. Diese Fragen werden in der Release- Strategie durch die Hersteller der Web-Anwendungen beantwortet. Bei der Auswahl oder Entwicklung einer Web-Anwendung ist sicherzustellen, dass die Releasestrategie des Produkts dem beabsichtigten Verwendungszweck entspricht.

Bei der Auswahl und Entwicklung einer Web-Anwendung sollte auf Aktive Inhalte vollständig verzichtet werden. Zudem sollte eine barrierefreie Implementierung (siehe auch barrierefreies E-Government) angestrebt und auf die Einhaltung des Minimalprinzips geachtet werden. Benutzereingaben sind stets als potenziell gefährlich einzustufen. Sie müssen daher auf Server-Seite gefiltert werden. SQL-Abfragen sollten in Form von Prepared Statements implementiert werden, die SQL Injection per Design verhindern. Dabei handelt es sich um vorbereitete Datenbank-Abfragen, welche Platzhalter anstatt von Parametern enthalten. Die Platzhalter werden erst zu einem späteren Zeitpunkt durch die jeweiligen Benutzereingaben ersetzt. Durch diese Vorgehensweise ist ein Benutzer nicht mehr in der Lage, die Syntax der Datenbankbefehle zu verändern. Des Weiteren sollte eine verschlüsselte Kommunikation unterstützt werden.

Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen

Bei der Entwicklung von Web-Anwendungen sollte darauf geachtet werden, dass der Softwareentwicklungsprozess ausreichend dokumentiert ist. Zudem sollte eine klare Trennung zwischen Entwicklungs-, Test- und Produktionsumgebung erfolgen. Dies beinhaltet sowohl die Systeme als auch die personellen Ressourcen. Sicherheitsmaßnahmen dürfen nicht nur punktuell, sondern sollten integriert in den Gesamtentwicklungsprozess umgesetzt werden. Dies bedeutet, dass folgende Punkte beachtet werden müssen:

- Bei der Planung der Implementierung müssen auch die Anforderungen hinsichtlich der Sicherheit erhoben werden.
- Beim Design sollte eine Gefährdungsanalyse durchgeführt werden, um erforderliche Sicherheitsmaßnahmen frühzeitig zu identifizieren.
- Für die Implementierung sollten Sicherheitsframeworks verwendet werden. Zudem ist es empfehlenswert, begleitende Quellcode-Überprüfungen durchzuführen, um die Qualität des Codes sicherzustellen. Durch begleitende Quellcode-Audits bereits während der Entwicklung kann ein signifikanter Sicherheitsgewinn erzielt werden. Der entstandene Mehraufwand bei der Entwicklung kann durch einen geringeren Aufwand bei der Wartung der Systeme aufgewogen werden. Je nachdem, wie groß die Ressourceneinsparung beim Wartungsaufwand ausfällt, ist eine Senkung des Gesamtaufwands möglich.
- Während der Testvorgänge sollten sowohl automatisierte als auch manuelle Sicherheitstests durchgeführt werden.
- Bei der Installation der Software muss sichergestellt werden, dass alle Passwörter geändert werden müssen.
- Im laufenden Betrieb ist die Qualität der Software zu messen. Durch eine entsprechende Rückmeldung kann so eine Verbesserung der Qualität des Softwareentwicklungsprozesses erzielt werden.

Richtlinien für sichere Beschaffung von Web-Anwendungen

Bei der Beschaffung von Web-Anwendungen müssen Sicherheitskriterien in den Einkaufsbedingungen verankert werden. Des Weiteren ist zu berücksichtigen, dass die festgelegten Sicherheitskriterien nicht nur für einzelne Module oder Teil-Anwendungen, sondern für die gesamte Web-Anwendung gelten müssen. Ansonsten könnten für Schwachstellen anfällige Komponenten die Sicherheit des Gesamtsystems unterlaufen. Voraussetzung für diese Vorgehensweise ist, dass die Sicherheitsanforderungen auch auf alle Teil-Anwendungen anwendbar sind. Ist dies nicht der Fall, können vergleichbare Sicherheitsanforderungen auch speziell für einzelne Komponenten formuliert werden.

Bei der Durchführung von Abnahmetests sollten Sicherheitsaspekte berücksichtigt werden. Dies kann zum Beispiel in Form einer Quellcode-Überprüfung oder durch Penetrationstests stattfinden.

4.2 Konfiguration der Komponenten

Wurden alle für die Grundarchitektur benötigten Komponenten beschafft, müssen diese konfiguriert werden. Die Punkte, die es dabei zu beachten gilt, werden im folgenden Abschnitt erläutert.

4.2.1 Betriebssystem

Eine sichere Konfiguration des Betriebssystems ist eine Grundvoraussetzung, um Web-Anwendungen sicher bereitstellen zu können. Grundsätzlich sollten auf einem System nur jene Dienste und Programme installiert sein, die für den Betrieb erforderlich sind. Dies bedeutet, dass eine Vielzahl nicht benötigter Programme (wie beispielsweise Compiler oder Editoren) entfernt werden sollten. Dienste, die für den Betrieb erforderlich sind, sollten nur dann von außen sichtbar bzw. erreichbar sein, wenn dies für die Funktionalität notwendig ist.

Um das Betriebssystem vor Angriffen zusätzlich zu schützen, sollten entsprechende Sicherheitsprogramme zum Einsatz kommen. Beispiele dafür sind:

- Programme zur Prüfung der Integrität der Systemsicherheit
- Host-basierte Paketfilter
- Virenschutzprogramme
- Kryptografisch gesicherte Administrationszugänge

Um möglichst wenig Information über das Betriebssystem und die darauf befindlichen Dienste preiszugeben, sollte Banner-Spoofing verwendet werden. Dabei liefern die auf dem Server betriebenen Dienste nicht ihre korrekte Programmbezeichnung und Versionsnummer zurück, sondern ersetzen diese durch eine verallgemeinerte, bereinigte Versionsnummer (z. B. „Apache 2“ statt „Apache 2.4.25“). Dadurch ist es für einen Angreifer schwieriger, Rückschlüsse auf die tatsächlich vom System verwendeten Programme zu ziehen.

Auch müssen alle Standard-Benutzerkonten entfernt beziehungsweise umbenannt werden. Ein Beispiel dafür ist das unter Windows-Betriebssystemen oftmals vorhandene Gastkonto. So bietet sich auch an, privilegierte Benutzerkonten, wie Administrator oder root, zu deaktivieren und stattdessen eigene Benutzer mit den notwendigen administrativen Rechten anzulegen.

4.2.2 Webserver

Bei der Konfiguration des Webservers sollte nach Möglichkeit eine Chroot-Umgebung angelegt werden. Dabei handelt es sich um einen abgeschotteten Bereich innerhalb des Computersystems (eine sogenannte Sandbox), die es einem Angreifer erschwert nach der Kompromittierung das Webservers Zugriff auf das gesamte System zu erlangen. Zudem müssen alle mitgelieferten Demo-Anwendungen und Systemdokumentationen entfernt werden.

Der Webserver sollte ferner nur über jene Rechte verfügen, die für den Betrieb erforderlich sind. Normalerweise benötigt ein Webserver keine Schreibrechte im Betriebssystem und sollte deshalb diese auch nicht besitzen. Da Log-Dateien an einen dedizierten Logging-Server übermittelt werden, sind hierfür keine lokalen Schreibrechte notwendig. Alle Dateien, die nicht mehr verändert werden sollen, wie beispielsweise statische HTML-Seiten, müssen mit Schreibschutz versehen werden. Falls der Webserver dem Nutzer automatisch generierte Verzeichnislisten anzeigen kann, sollte diese Funktion deaktiviert werden.

Grundsätzlich müssen nicht benötigte HTTP-Methoden des Webservers deaktiviert werden. In den meisten Installationen genügen die Methoden GET und POST. Bei einzelnen Einsatzszenarien können darüber hinaus auch weitere Methoden notwendig sein.

Für die Verschlüsselung zwischen Webserver und Client mittels HTTPS muss das TLS-Protokoll mit der Version 1.2 verwendet werden. Zum zusätzlichen Schutz gegen Man-in-the-Middle-Angriffe sollte darüber hinaus der HSTS-Header gesetzt werden.

Standard-Fehlermeldungen bergen oft die Gefahr in sich, zu viel Information preiszugeben. Aus diesem Grund sollten eigene Fehlermeldungen konfiguriert werden, die den Benutzer über einen aufgetretenen Fehler informieren, aber keine Details dazu preisgeben. Gleichzeitig kann damit der Komfort für die Benutzer erhöht werden.

Wird die Funktionalität des Webservers mit Hilfe von Modulen erweitert, so müssen diese denselben Sicherheits-Anforderungen wie der Webservers selbst genügen. Zudem müssen Module nach dem Minimalprinzip ausgewählt werden.

4.2.3 Content Management System

Wird für die Bereitstellung eines Web-Angebotes ein CMS (Redaktionssystem zur Gestaltung der Inhalte eines Web-Auftritts) verwendet, so ist dieses gemäß den Vorgaben der Grundarchitektur zu implementieren. Dabei muss eine sichere Trennung zwischen dem für die Benutzer erreichbaren Webserver und dem nur für den Inhalteanbieter erreichbaren CMS erfolgen.

Die dem CMS vorgeschalteten Paketfilter und das ALG (siehe Abbildung 3.3 - Paketfilter 3, 4 und 5) müssen dazu so konfiguriert werden, dass nur von internen IP-Adressen auf das CMS zugegriffen werden kann. Der Zugang zur Administrationsoberfläche darf nur Administratoren möglich sein.

Wie bei anderen Komponenten sollten auch beim CMS alle Standard-Benutzerkonten entfernt und die Standard-Passwörter geändert werden.

Falls für den Betrieb des CMS Funktionen modular nachgeladen werden, sollte eine restriktive Auswahl der erforderlichen Module erfolgen. Da ein CMS auch eine eigene Web-Anwendung ist, gelten hier dieselben Anforderungen wie für Web-Anwendungen im Übrigen.

4.2.4 Logging-Server

Der Logging-Server sollte so konfiguriert werden, dass mindestens folgende Ereignisse protokolliert werden:

- Erfolgreiche und fehlgeschlagene Anmeldungen an der Web-Anwendung.
- Erfolgreiche Zugriffe auf Ressourcen.
- Fehlgeschlagene Zugriffe auf Ressourcen aufgrund von :
  - mangelnden Berechtigungen,
  - nicht vorhandenen Ressourcen,
  - Server-Fehlern.
- Allgemeine Fehlermeldungen, soweit sie eventuell später von Interesse sein könnten. Das schließt alle nicht explizit als uninteressant eingestuften Meldungen mit ein.

Um die Anzahl an Fehlalarmen zu reduzieren und dennoch Angriffe erkennen zu können, sollten Schwellenwerte für die Benachrichtigung von Administratoren gesetzt werden. Wird eine Anwendung in den Produktivbetrieb übergeführt, sind diese Schwellenwerte in geeigneter Weise anzupassen.

Es muss sichergestellt werden, dass die Logdateien nur für die Administratoren zugänglich sind. Sie dürfen nicht über eigne URLs oder mit Hilfe geeigneter Parameter von außen abrufbar sein.

Bei der Auswertung der Logdateien muss darauf geachtet werden, dass das beim Parsen des Textes keine Sicherheitslücken in den verwendeten Applikationen getriggert werden können.

4.2.5 Hintergrundsysteme

Bei Hintergrundsystemen sollte die Vergabe von Rechten für konfigurierte Rollen und Benutzerkonten auf Basis des Minimalprinzips erfolgen. So können beispielsweise vorhandene Stored Procedures für bestimmte Benutzer oder Rollen in der Datenbank gesperrt werden. Zudem sind alle Standard-Konten und Beispielinhalte zu entfernen.

4.2.6 Web-Anwendung

Filterung von Anfragen und Ausgaben

Alle an die Web-Anwendung herangetragenen Daten, unabhängig von Kodierung oder Form der Übermittlung, müssen als potenziell gefährlich behandelt und entsprechend gefiltert werden. Dabei ist ein generischer, sehr restriktiver Filter zu implementieren. Alle Daten, die an die Geschäfts-Logiken übermittelt werden, werden zuvor durch den Eingabe-Filter bearbeitet.

Ist es für einzelne Funktionen notwendig, den generischen Filter aufzulockern, muss dies explizit für den Zugriff auf die Daten definiert werden. Zusätzlich ist es möglich, kontextsensitive Filter in der Geschäfts-Logik oder in den Hintergrundsystemen zu implementieren. Auf diese Weise können auf die Funktionen der Geschäfts-Logik beziehungsweise der Hintergrundsysteme angepasste Filterungen durchgeführt werden.

Schutz vor Clickjacking

Beim Clickjacking verleitet ein Angreifer den Nutzer dazu, auf einen Link zu klicken, der zwar für den Nutzer harmlos erscheint, tatsächlich aber zu einer potentiell bösartigen Webseite führt. Dies wird beispielsweise dadurch erreicht, dass die gutartige Webseite mittels HTML-Frames in die Webseite eines Angreifers eingebunden wird.

Damit die Webseite vor dem beliebigen Einbinden in HTML-Frames geschützt ist, muss das Senden des Headers „X-Frame-Options“ bei der Auslieferung von Webseiten aktiviert sein. Die folgende Tabelle erläutert die verschiedenen Parameter des X-Frame-Options-Headers.

Wert	*Bezeichnung*
X-Frame-Options: deny	Die Webseite darf nicht in einem Frame eingebunden werden.
X-Frame-Options: sameorigin	Die Webseite darf nur in einem Frame derselben Basisadresse verwenden werden.
X-Frame-Options: allow-from: example.com	Die Webseite darf nur in einem Frame der Basisadresse example.com eingebunden werden.

Schutz vor Cross-Site-Scripting-Angriffen

Zum Schutz des Clients müssen die Eingaben einer Web-Anwendung auch auf Cross-Site-Scripting-Angriffe hin gefiltert werden. Mit Cross-Site-Scripting (XSS) wird das Einschleusen von bösartigem HTML- und JavaScript-Code in eine Webseite bezeichnet. Ein derartiger Angriff ist möglich, wenn eine Web-Anwendung nicht vertrauenswürdige Daten entgegen nimmt und ohne ausrei-

chende Validierung bzw. Codierung an einen Webbrowser sendet. XSS erlaubt es einem Angreifer somit, JavaScript Code im Browser eines Opfers auszuführen, um z. B. Sessions zu übernehmen, Seiteninhalte zu verändern oder den Benutzer auf eine bösartige Seite umzuleiten. HTML-Codierung ist der grundsätzliche Schutzmechanismus gegen diese Angriffe. Viele Frameworks bzw. Sprachen stellen hierfür entsprechende Funktionen zur Verfügung.

Beim persistenten Cross-Site-Scripting schleust ein Angreifer JavaScript Code in Eingabefelder, welche von einer Datenbank gespeichert und zu einem späteren Zeitpunkt von der Web-Anwendung wieder ausgelesen werden (Beispiele dafür sind Kommentarfunktionen oder Benutzernamen). Der gefährliche Code wird somit persistent in der Datenbank hinterlegt und bei jedem Benutzer, der die Seite aufruft, ausgeführt. Ein Angreifer hätte somit die Möglichkeit, großflächig Benutzerdaten abzugreifen.

Einen grundlegenden Schutz bietet das Filtern der Ausgaben der Web-Anwendung. Auf diese Weise können auch Lücken im Eingangsfilter entdeckt werden.

Darüber hinaus muss der in den Browsern integrierte XSS-Schutz aktiviert werden. Dies kann durch Senden des Headers „X-XSS-Protection“ bei der Auslieferung von Webseiten aktiviert werden. Die folgende Tabelle erläutert die verschiedenen Parameter dieses Headers.

Wert	***Bezeichnung***
X-XSS-Protection: 0	Der XSS-Schutz ist deaktiviert.
X-XSS-Protection: 1	Der XSS-Schutz ist aktiviert. Falls ein XSS-Angriff vom Browser erkannt wurde, wird die Webseite bereinigt.
X-XSS-Protection: 1; mode=block	Der XSS-Schutz ist aktiviert. Falls ein XSS-Angriff vom Browser erkannt wurde, wird die Webseite nicht angezeigt.
X-XSS-Protection: 1; report=http://example.com	Der XSS-Schutz ist aktiviert. Falls ein XSS-Angriff vom Browser erkannt wurde, wird die Webseite bereinigt. Darüber hinaus werden Details des Angriffs an die Adresse example.com gesendet.

Eine weitere Möglichkeit, XSS-Angriffen vorzubeugen, ist die Nutzung des „Content-Security-Policy“-Headers. Hierbei handelt es sich um eine standardisierte Maßnahme, die auch vor anderen Angriffen, wie beispielsweise Clickjacking, schützen kann. Eine grundlegende Idee ist hierbei, dass beim Ausliefern der Webseite im Header deklariert wird, von welchen Basisadressen bestimmte Ressourcen (Texte, Bilder, Videos etc.) geladen werden dürfen. Die folgende Tabelle erläutert beispielhaft verschiedene Parameter dieses Headers. Die vollständige Parameterliste kann bei der W3C nachgelesen werden.

Wert	***Bezeichnung***
Content-Security-Policy: img-src 'self'	Bilder dürfen nur von derselben Basisadresse geladen werden.
Content-Security-Policy: script-src 'self'	Skripte (z. B. JavaScript) dürfen nur von derselben Basisadresse geladen werden.

Wert	*Bezeichnung*
Content-Security-Policy: frame-ancestors 'self'	Nur Webseiten der Basisadresse dürfen diese Webseite in einen Frame einbinden.
Content-Security-Policy: form-action 'self'	Formulare dürfen nur an eine Webseite mit derselben Basisadresse gesendet werden.

Schutz vor ausführbaren Ressourcen

Bei einer Webseite werden normalerweise viele unterschiedliche Ressourcen ausgeliefert – z. B. Texte, Bilder und Videos. Damit der Browser versteht, wie er diese verschiedenen Ressourcen interpretieren soll, werden sogenannte MIME-Typen eingesetzt. Falls beim Ausliefern einer Ressource vom Webserver kein MIME-Typ vorhanden ist bzw. deklariert wurde, versucht der Browser selbstständig festzustellen, um welchen Dateityp es sich handeln könnte.

Falls es sich allerdings bei der Ressource um eine ausführbare Datei handelt, könnte dies zu einem Sicherheitsproblem werden. Deshalb muss bei der Auslieferung von Webseiten einerseits immer der korrekte MIME-Typ gesetzt werden. Darüber hinaus muss der Header „X-Content-Type-Options“ gesendet werden. Dieser Header weißt den Browser an, dass der MIME-Typ nicht selbstständig festgestellt werden soll, sondern als korrekt angesehen werden soll. Die folgende Tabelle erläutert den Parameter dieses Headers.

Wert	*Bezeichnung*
X-Content-Type-Options: nosniff	Verhindert, dass der Browser selbstständig den MIME-Typ feststellt.

Filtermethoden

Für die Filterung von Anfragen und Ausgaben stehen verschiedene Methoden mit unterschiedlichen Vor- und Nachteilen zur Verfügung:

Beim Blacklisting erfolgt eine Filterung auf bestimmte nicht erlaubte Zeichen beziehungsweise Zeichenketten. Der Nachteil dieser Variante ist jedoch, dass nur gegen bekannte Muster gefiltert wird. Zudem schützt diese Art der Filterung nicht vor allen Permutationen von Angriffsmustern und kann somit umgangen werden. Beispielsweise können Zeichen in unterschiedlichen Kodierungsformen dargestellt werden. Werden nicht alle davon gefiltert, besteht die Möglichkeit zur Umgehung des Filtermechanismus.

Diese Probleme werden beim Whitelisting vermieden, da nur explizit erlaubte Zeichen beziehungsweise Zeichenketten den Filter passieren können. Dabei muss sichergestellt sein, dass sich aus den erlaubten Zeichen keine Angriffsmuster bilden lassen. Allerdings muss bei dieser Variante eine ständige Anpassung der Filterregeln an Veränderungen in der Anwendung erfolgen, da ansonsten auch erlaubte Zeichen(-ketten) gefiltert werden.

Eine weitere Möglichkeit der Filterung ist das sogenannte „Maskieren“. Dabei werden Syntax-spezifische Zeichen durch Voranstellen eines Steuerzeichens besonders als solche gekennzeichnet. Sehr effektiv, vor allem beim Schutz vor Cross-Site-Scripting-Angriffen, ist das Kodieren von Syntax-spezifischen Zeichen. Beispielsweise können ausgewählte Zeichen HTML-codiert werden (z. B. kann das Zeichen '>' vor dessen Ausgabe im HTML-Code durch '>' ersetzt werden, ohne dass sich dies negativ auf die Ausgabe im Browser auswirkt. Script-Code kann jedoch nicht mehr ohne Weiteres ausgeführt werden, wenn dafür notwendige Zeichen HTML-kodiert werden.)

Die angeführten Varianten können kombiniert werden, um deren Vorteile zu vereinen.

Authentisierung und Benutzermanagement

Es gibt mehrere gängige Authentisierungsverfahren, die den Zugriff auf Web-Angebote steuern können. Die mit Abstand weiteste Verbreitung hat dabei das Ein-Faktor-Verfahren mit Benutzername und Passwort. Dieses ist für den normalen Schutzbedarf ausreichend, sofern die gesamte Kommunikation über verschlüsselte bzw. vertrauenswürdige Kanäle läuft. Weitere mögliche Verfahren sind Challenge-Response, Einmalpasswörter und zertifikatsbasierte Verfahren.

Wird ein Web-Angebot mittels Benutzername und Passwort geschützt, ist das Passwort das einzig geheime Element. Der Benutzername wird in der Regel nicht als geheim angesehen. Dies ist insofern problematisch, als dass ein Angreifer mit diesem Wissen einen Brute-force-Angriff durchführen kann. Dabei werden alle möglichen Passwörter zum bekannten Benutzernamen automatisiert durchprobiert. Hat der Benutzer ein einfaches Passwort gewählt, welches beispielsweise in einem Wörterbuch gefunden werden kann, besteht eine hohe Wahrscheinlichkeit, dass der Angriff Erfolg hat. Um einen solchen Angriff zu erschweren, müssen die Mindestanforderungen an Passwörter in einer Richtlinie festgelegt werden. Zusätzlich kann eine Teergrube implementiert werden. Diese beiden Maßnahmen werden in den folgenden Abschnitten beschrieben.

Passwortrichtlinie

Im Rahmen einer Passwortrichtlinie sollten Mindestanforderungen an Passwörter definiert werden. Dabei sollten folgende Kriterien berücksichtigt werden:

- Mindestanzahl der Zeichen
- Verbotene Zeichenketten
- Zeichentypen, die enthalten sein müssen, beispielsweise mindestens zwei Großbuchstaben oder mindestens zwei Ziffern
- Ob und unter welchen Bedingungen die Gültigkeitsdauer eines Passwortes eingeschränkt werden soll.
- Art der Vergabe eines Passwortes
- Art der Änderung eines Passwortes

Im Rahmen der Passwortrichtlinie sollte auch definiert werden, ob und welche Passwort-Verwaltungswerkzeuge verwendet werden dürfen. Die Vorgaben der Passwortrichtlinie sollten soweit möglich technisch erzwungen werden.

Teergrube

Eine sogenannte „Teergrube“ (engl. Tarpit) bezeichnet ein System, welches einen Angriff verlangsamt, bei dem alle möglichen Passwörter für ein Benutzerkonto durchprobiert werden (Brute-Force-Angriff), ohne das Benutzerkonto längerfristig zu sperren. Um Brute-force-Angriffe zu unterbinden, besteht grundsätzlich die Möglichkeit, ein Benutzerkonto nach einer definierten Anzahl an misslungenen Anmeldeversuchen zu sperren. Dies bietet allerdings die Möglichkeit eines Denial-of-Service-Angriffs, da ein Angreifer dann beliebige Benutzerkonten sperren kann, bei denen er den Anwendernamen kennt. Gegen Brute-Force-Angriffe eigenen sich daher Teergruben besser, bei denen ein Benutzerkonto nach Überschreiten der maximalen Anzahl an misslungenen Anmeldeversuchen nicht vollständig, sondern nur für ein begrenztes Zeitintervall gesperrt wird. Je nach Variante können bei länger andauernden Angriffen längere Zeitintervalle aktiviert werden. So lässt sich bei sinnvoll gewählten Zeitspannen die Gefahr von Denial-of-Service-Angriffen verringern.

„Passwort vergessen“-Funktion

Da es immer wieder erforderlich ist, Passwörter zurückzusetzen, sollte auf eine sichere Implementierung dieser Funktion geachtet werden.

Eine Möglichkeit ist, dass bei der Registrierung des Benutzers ein eigenes Master-Passwort zum Zurücksetzen des Benutzerkonto-Passworts generiert wird. Dieses Passwort sollte der Benutzer ausgedruckt an einem sicheren Ort aufbewahren. Zusätzlich wird die E-Mail-Adresse des Benutzers gespeichert. Vergisst ein Benutzer das Passwort seines Benutzerkontos, so fordert er ein neues Passwort an. Dazu authentisiert er sich mit dem Master-Passwort. Ist das Master-Passwort korrekt, wird an die E-Mail-Adresse des Benutzers ein nur für diesen Zweck einmalig gültiger Link gesendet unter dem er über eine verschlüsselte Verbindung ein neues Passwort setzen kann.

Session-Management

Bei der Implementierung von Session-Management müssen Session-Frameworks verwendet werden, welche eine sichere Handhabung der Sitzungsdaten gewährleisten. Session-IDs dürfen nicht als GET-Parameter übergeben werden, sondern nur als Cookie im Header. Cookies sollen nur als http-only-Cookies verwendet werden. Auf diese Weise wird verhindert, dass Session-IDs in Log-Dateien von Webservern erscheinen beziehungsweise lokal im Cache der besuchten Webseiten des Clients vorhanden bleiben. Zudem sollte der Austausch von Session-Daten nur über sichere Kanäle erfolgen.

Jede Form von Session-Management kann jedoch unter Ausnutzung von Cross-Site-Scripting-Schwachstellen kompromittiert werden. Es sollte daher sehr großer Wert darauf gelegt werden, dass derartige Schwachstellen nicht vorhanden sind.

Session-IDs müssen mit Hilfe kryptografischer Zufallszahlengeneratoren zufällig erzeugt werden. Nach jeder erfolgreichen Authentisierung müssen alte IDs verworfen und eine neue Session-ID vergeben werden. Um Session Riding bei sicherheitskritischen Anfragen wie z. B. Anstoßen einer Transaktion oder Löschen eines Benutzerkontos auszuschließen, müssen zusätzliche Authentisierungsschritte (z.B. mit Hilfe von Smartcards, TANs etc.) verwendet werden, die jeweils nur für diese eine Aktion gültig sind.

Sicherheitskritische Eigenschaften eines autorisierten Benutzers, die über eine Sitzung hinaus Gültigkeit haben, wie z. B. Berechtigungen, dürfen nur Server-seitig gespeichert werden. Cookies, welche auf dem Client abgelegt werden, sind dafür nicht geeignet. Die Zuordnung von Session-ID zu Berechtigungen darf nur Server-seitig erfolgen. Session-Daten sind zudem in geeigneter Weise zu schützen (beispielsweise durch Ablage in ein geschütztes Temporär-Verzeichnis oder durch Verschlüsselung).

Session Time-outs können helfen, Brute-Force-Angriffe zu verhindern. Dabei wird nach einer gewissen Zeit, in welcher der Benutzer keine Aktionen mehr an der Web-Anwendung durchgeführt hat, die Session beendet und der Benutzer abgemeldet. Einem Angreifer steht dadurch nur ein begrenzter Zeitraum zur Verfügung, in dem er alle möglichen Session-IDs durchprobieren und dadurch eine Sitzung übernehmen kann. Um die sichere Handhabung von Session-IDs zu gewährleisten, stehen mehrere Möglichkeiten zur Verfügung, die auch parallel verwendet werden können:

- Ablauf der Session nach Inaktivität
- Wechsel der Session-ID im Hintergrund

Darüber hinaus muss jedoch auch das manuelle Abmelden von einer Session möglich sein, wobei die Session-Daten zerstört werden müssen und die Session von diesem Zeitpunkt an als ungültig betrachtet wird.

Bei Verwendung einer persistenten Authentisierung (Wiedererkennung des Benutzers über dessen Session-ID) muss zusätzlich eine flüchtige Authentisierung eingesetzt werden, damit Session Riding erschwert wird. Schutz gegen Session Riding kann durch einen zusätzlichen Parameter realisiert werden, der zur Laufzeit in jede Antwort des Servers eingeschlossen wird. Dieser Parameter muss für jede Antwort neu mit Hilfe eines Zufallszahlen-Generators erstellt und in alle sicherheitskritischen Links und Formulare eingebettet werden. Ruft der legitime Benutzer einen dieser Links auf bzw. sendet ein Formular an den Server, prüft dieser die Gültigkeit des eingebetteten Parameters. Da nur der Browser eines legitimen Benutzers diesen flüchtigen Parameter kennen kann, wird sichergestellt, dass der Aufruf nicht von einem anderen Client oder durch Klick auf einen externen Link ausgelöst wurde. Man-in-the-Middle-Angriffe, durch welche auch ein Auslesen des zusätzlichen Parameters möglich ist, müssen durch eine gesicherte Übertragung (z. B. mittels HTTPS) verhindert werden. Es ist jedoch anzumerken, dass die Schutzmaßnahme des zusätzlichen Parameters durch Cross-Site-Scripting-Angriffe ausgehebelt werden kann.

Sicherer Umgang mit Dateien

Um die Sicherheit des Web-Angebotes nicht zu gefährden, sollte der Upload und das Einbinden von Dateien bzw. das über Benutzereingaben gesteuerte Erzeugen von Dateien eingeschränkt werden. Fehler in diesem Bereich können sehr schnell zu einer vollständigen Kompromittierung des Systems führen, da ein Angreifer die Möglichkeit bekommen kann, beliebige Kommandos mit den Rechten des Webservers auszuführen.

Einbinden von Dateien

Dateien, die von der Anwendung eingebunden und ausgeführt werden, dürfen nicht von Benutzern manipulierbar sein. Darüber hinaus müssen auch die Pfade von eingebunden Dateien vor Manipulation geschützt werden. Werden Dateien über ein Netz eingebunden (remote include), muss eine Whitelist sicherstellen, dass diese nur von vertrauenswürdigen Servern kommen können. Andernfalls kann ein Angreifer möglicherweise beliebigen Code in das Web-Angebot einschleusen. Wird kein Whitelisting unterstützt, so sollte diese „remote include“-Funktion auf dem Server deaktiviert werden.

Datei-Upload und -Erzeugung

Beim Hochladen von Dateien, die vom Benutzer angegeben werden, müssen folgende Regeln beachtet werden:

- Hochgeladene Dateien dürfen nicht im Wurzelverzeichnis des Webservers gespeichert werden. Stattdessen müssen zufällige Ordner- und Dateinamen gewählt werden.
- Der Pfad, unter dem Dateien gespeichert werden, darf nicht von Benutzern beeinflusst werden können.
- Der Inhalt der Dateien muss einer Plausibilitätsüberprüfung unterzogen werden.
- Dateien müssen von einem Virenschutzprogramm untersucht werden.
- Es sollte eine maximale Dateigröße spezifiziert werden.
- Es dürfen nur erlaubte Dateiformate hochgeladen werden.

Für unter Benutzereinfluss erzeugte Dateien muss zusätzlich sichergestellt werden, dass diese am Webserver nicht ausgeführt werden können.

Nicht benötigte Dateien im Wurzelverzeichnis des Web-Angebotes

Dateien, die zur Bereitstellung des Web-Angebotes notwendig sind und Programmcode enthalten, werden bei Aufruf der entsprechenden URL vom Webserver interpretiert bzw. ausgeführt. In keinem Fall darf jedoch deren Inhalt (d. h. der Quelltext) an den anfragenden Client zurückgeliefert werden. Dateien, die Programmcode enthalten und vom Webserver eventuell nicht als solche erkannt werden, können beispielsweise durch Editoren entstehen (z. B. .php~). Derartige Sicherungsdateien sind oft mit Dateiendungen versehen (z. B. .bak), die dazu führen, dass diese vor der Ausgabe nicht interpretiert werden. Ruft ein Angreifer diese Dateien ab, so erhält er Zugriff auf den Quellcode des Web-Angebotes. Auch bei Konfigurationsdateien ist darauf zu achten, dass diese keine Endungen (z. B. .inc) aufweisen, welche Informationen preisgegeben können.

Auch temporäre Dateien können sicherheitskritische Informationen beinhalten und müssen daher vor unberechtigtem Zugriff geschützt werden.

Konfigurationsdateien

Grundsätzlich müssen Konfigurationsdaten außerhalb des Quelltextes in entsprechenden Konfigurationsdateien gespeichert werden. Diese Dateien müssen außerhalb des Wurzelverzeichnisses des Webservers abgelegt werden und dürfen von außen (Internet) nicht zugänglich sein. Bei Konfigurationseinstellungen, die vertrauliche Daten beinhalten, muss zudem für eine entsprechende Verschlüsselung gesorgt werden.

System- und Fehlermeldungen

Bei der Ausgabe von System- und Fehlermeldungen muss sichergestellt werden, dass keine sicherheitskritischen Informationen an den Benutzer weitergegeben werden. Fehlermeldungen an den Benutzer dürfen folglich nur Informationen beinhalten, welche keine Rückschlüsse auf eingesetzte Frameworks oder Hintergrundsysteme zulassen. Beim Auftreten von unerwarteten Fehlern muss dafür gesorgt werden, dass die Anwendung in einen definierten sicheren Grundzustand übergeht. Dies kann mit geeigneten Error-Handlern realisiert werden. Darüber hinaus müssen alle betroffenen Sessions terminiert werden. Bei Produktivsystemen dürfen keine Debugging-Informationen, also detaillierte Fehlermeldungen für Entwickler, ausgegeben werden.

Einsatz von Kryptografie

Kryptografie wird bei der Bereitstellung von Web-Angeboten häufig eingesetzt. So werden kryptografische Methoden für die Generierung von Session-IDs, die Berechnung von Passwort-Hashes, den Austausch von Client- und Server-Zertifikaten und die verschlüsselte Kommunikation zwischen zwei Endpunkten verwendet.

Beim Einsatz von Kryptografie in sicherheitskritischen Bereichen, also an allen Stellen, wo vertrauliche Daten über nicht vertrauenswürdige Netze übertragen werden, ist darauf zu achten, dass nur etablierte Algorithmen zur Anwendung kommen. Diese müssen dem Stand der Technik entsprechen (siehe [BSI-TR-02102]).

Eine besondere Bedeutung bei der Kryptografie kommt den verwendeten Schlüsseln zu. Diese müssen je nach Einsatzgebiet über eine gewisse Mindestlänge verfügen und verschiedenen mathematischen Anforderungen (z. B. Komplexität) genügen. Zudem muss für einen entsprechend sicheren Transport beziehungsweise Austausch von Schlüsseln gesorgt werden. Gleiches gilt auch für deren Speicherung.

Aufruf von externen Programmen

Beim Aufruf von externen Programmen auf dem Server sollten nach Möglichkeit keine externen Benutzereingaben verwendet werden. Sollte dies erforderlich sein, so müssen diese Eingaben sehr restriktiv gefiltert werden, um Command-Injection-Angriffe zu verhindern.

Schnittstellen zu Hintergrundsystemen

Ein direkter Zugriff auf Hintergrundsysteme aus der Geschäfts-Logik heraus darf vom Design her nicht möglich sein. Stattdessen sollten für Zugriffe vordefinierte Prozeduren und Wrapper-Funktionen verwendet werden. Hintergrundsysteme dürfen Web-Anwendungen nicht vertrauen und erfordern daher eine eigene Authentisierung. Dies bedeutet, dass Benutzer nicht nur an der Web-Anwendung, sondern auch am Hintergrundsystem authentisiert werden müssen.

Sichere Anbindung von Hintergrundsystemen an die Web-Anwendung

Eine Anbindung der Web-Anwendung an Hintergrundsysteme sollte in erster Linie über vertrauenswürdige Kanäle erfolgen. Jedoch ist auch bei der Kommunikation über vertrauenswürdige Kanäle eine Authentisierung erforderlich.

Erfolgt die Anbindung über unsichere Kanäle, so muss ein kryptografischer Tunnel mit entsprechender Verschlüsselung und Authentisierung verwendet werden.

4.3 Grundvorgaben für einen sicheren Betrieb

Technische Grundvorgaben für einen sicheren Betrieb der Netz-Infrastruktur sind in [ISi-LANA] festgehalten. Für den sicheren Betrieb sind insbesondere auch organisatorische Tätigkeiten wichtig.

4.3.1 Betriebssystem

Um einen sicheren Betrieb zu gewährleisten, dürfen nur Server in die Produktivumgebung eingebunden werden, die vollständig konfiguriert wurden. Zudem muss eine entsprechende Integration des Systems in das Patch-Management erfolgen. Auf diese Weise wird auch sichergestellt, dass Änderungen an der Konfiguration nur nach ausführlichen Tests in einer Testumgebung und nach dem Vier-Augen-Prinzip auf Produktivsysteme übernommen werden. Es dürfen daher auch keine Programme auf dem Produktivsystem kompiliert werden.

Darüber hinaus ist es wichtig, regelmäßig verfügbare Sicherheits-Updates einzuspielen. Bei Linux- und Unix-Systemen besteht zusätzlich die Möglichkeit, das Betriebssystem durch entsprechende Kernel-Patches besonders abzusichern.

Um Kompromittierungen des Webauftritts und die ungewollte Auslieferung von Schadprogrammen zu unterbinden bzw. zu erschweren, sollte im Hintergrund eine Echtzeitprüfung durch ein Virenschutzprogramm erfolgen. Idealerweise wird dieses durch eine periodische Prüfung auf mögliche Rootkit-Infektionen ergänzt.

Im laufenden Betrieb sollte für ein entsprechendes Monitoring gesorgt werden. Dies bedeutet, dass zumindest in regelmäßigen Abständen geplante Überprüfungen der Systemsicherheit durchgeführt werden müssen. Zudem müssen anfallende Log-Dateien (z. B. die Ergebnisse einer Integritätsprüfung) regelmäßig, z. B. täglich, ausgewertet werden.

4.3.2 Webserver

Der Betrieb eines Webservers sollte mit möglichst stark eingeschränkten Rechten gemäß dem Minimalprinzip erfolgen. Dies bedeutet unter anderem, dass der Webserver mit den Rechten eines möglichst niedrig privilegierten Benutzers laufen sollte. Wie beim Betriebssystem muss auch beim Webserver eine regelmäßige Überprüfung der Server-Sicherheit und eine entsprechende Auswertung von Log-Dateien erfolgen.

4.3.3 Content Management System

Wird für die Bereitstellung des Web-Angebotes ein Content Management System benutzt, so sind bei dessen Betrieb folgende Punkte zu beachten:

Zugriffsversuche auf das CMS von nicht explizit erlaubten internen Systemen sollten mitprotokolliert und in regelmäßigen Abständen überprüft werden. Zugriffe durch externe Systeme dürfen nicht möglich sein.

Da es sich bei einem CMS auch um eine Web-Anwendung handelt, gelten hier dieselben Anforderungen wie für Web-Anwendungen (siehe Abschnitt 4.3.6).

4.3.4 Logging-Server

Logging-Server müssen so konfiguriert werden, dass beim Schreiben von Log-Dateien nur neue Einträge hinzugefügt und keine alten Einträge überschrieben beziehungsweise gelöscht werden können. Zudem müssen Log-Dateien in regelmäßigen Abständen gesichert werden, wobei auch für die Sicherungskopien dieselben Sicherheitsanforderungen wie für die ursprünglichen Log-Dateien einzuhalten sind.

Auf Grund des Umfangs von Log-Dateien sollten diese konsolidiert werden, um eine bessere Auswertung zu ermöglichen. Über die mitprotokollierten Ereignisse sollten ein Revisor und ein Administrator täglich in Form einer Zusammenfassung informiert werden. Beim Überschreiten von definierten Schwellenwerten ist jedoch eine sofortige Benachrichtigung des Revisors beziehungsweise des Administrators sicherzustellen.

4.3.5 Hintergrundsysteme

Sind auf Hintergrundsystemen Daten mit hohen Vertraulichkeitsanforderungen gespeichert, so müssen diese durch Verschlüsselung geschützt werden. Zudem sollte regelmäßig eine Sicherungskopie erstellt werden. Als Richtwert dazu kann ein tägliches Backup gelten. Abhängig von der Änderungshäufigkeit der Daten können auch kürzere Intervalle für Datensicherungen notwendig sein. Neben der Erstellung der Sicherungskopien müssen diese in regelmäßigen Abständen überprüft werden, um die fehlerfreie Wiederherstellung im Ernstfall gewährleisten zu können.

Die Administration von Hintergrundsystemen darf nur über das Management-Netz erfolgen. Zudem gilt es, die Server-Sicherheit in regelmäßigen Abständen zu überprüfen.

4.3.6 Web-Anwendung

Web-Anwendungen müssen vor Inbetriebnahme einer geeigneten Sicherheitsüberprüfung unterzogen werden. Für Anwendungen mit hohem Schutzbedarf wird auch eine Sicherheitsüberprüfung des Quellcodes empfohlen. Dabei kann auf Sicherheitszertifizierungen einzelner Anwendungen und auf die Einhaltung von anerkannten Programmierrichtlinien zurückgegriffen werden.

Eine automatisierte Überprüfung ist jedoch weder für hohen noch für normalen Schutzbedarf ausreichend, da damit nur ein kleiner Bruchteil an Schwachstellen in der Anwendung gefunden werden kann.

Auch während des laufenden Betriebs sollten regelmäßig, z. B. halbjährlich, Sicherheitsüberprüfungen im Zuge einer Revision durchgeführt werden. Dies ist vor allem nach größeren Umstellungen (z. B. Änderungen an der Konfiguration) empfehlenswert. Für den Fall, dass während dieser Überprüfungen Sicherheitslücken aufgedeckt werden, sollten unbedingt entsprechende Maßnahmen eingeleitet werden, die die Sicherheitslücken beseitigen.

5 Gefährdungen und Empfehlungen mit Varianten für normalen und hohen Schutzbedarf

Dieser Abschnitt analysiert die Gefährdungen, denen die Grundarchitektur ausgesetzt ist. Er stellt dar, wie die aus den Architektur- und Konfigurationsvorgaben der Abschnitte 3 und 4 resultierenden Maßnahmen diese Sicherheitsgefahren bei normalem Schutzbedarf abwehren. Darüber hinaus werden auch Varianten der Grundarchitektur vorgeschlagen, die mit höherem Aufwand höheren Schutzbedarf abdecken oder die bei geringerem technischen Realisierungsaufwand ein höheres, in speziellen Szenarien jedoch noch akzeptables Risiko mit sich bringen.

Lösungen, die sich mit geringerem Hardware-Aufwand realisieren lassen, sind in der Beschaffung oft kostengünstiger. Mit der Vereinfachung können jedoch die Sicherheitsrisiken steigen. Zudem erhöht sich unter Umständen der Aufwand für die Konfiguration und den Betrieb der Komponenten. Der Benutzer muss daher von Fall zu Fall entscheiden, ob die mit technischen Vereinfachungen einhergehenden Nachteile und Risiken auf Dauer nicht unwirtschaftlicher sind als die Grundarchitektur in der Grundkonfiguration. Vereinfachungsmaßnahmen sollten daher stets aufs Neue abgewogen werden. In jedem Fall eignen sie sich nur für Anwendungen mit höchstens normalem Schutzbedarf.

Das Erreichen eines höheren Sicherheitsniveaus ist in der Regel mit Mehrkosten verbunden: entweder aufgrund der Beschaffungskosten für zusätzliche Sicherheitskomponenten oder in Folge eines erhöhten Aufwands für die Einrichtung und den Betrieb einer anspruchsvolleren Web-Anwendung. Empfehlungen für Schutzmaßnahmen über die Grundarchitektur hinaus richten sich daher an Benutzer mit überdurchschnittlichem Schutzbedarf oder an Betreiber exponierter Web-Angebote, die ein besonders attraktives Angriffsziel bieten.

Durch Ausnutzen von Schwachstellen ergeben sich zahlreiche Gefährdungen in den Kategorien Verfügbarkeit, Vertraulichkeit und Integrität/Authentizität. Im folgenden Abschnitt werden die Angriffsmöglichkeiten beschrieben, die im Web genutzt werden können, um in geschützte Systeme einzudringen. Es handelt sich dabei um Gefährdungen auf der Anwendungsschicht des TCP/IP-Referenzmodells, welche speziell für Web-Angebote und die Komponenten zu deren Bereitstellung zutreffen.

5.1 Gefährdungen durch Eindringen

Im schlimmsten Fall kann ein Angreifer bestehende Schwachstellen ausnutzen, um das Web-Angebot bzw. die Hintergrundsysteme unter seine Kontrolle zu bringen. Durch das Eindringen in Systeme ergeben sich zahlreiche Folgegefährdungen. Sobald ein System kompromittiert wurde, kann ein Angreifer dieses beispielsweise nutzen, um weitere Systeme anzugreifen. Des Weiteren ist eine Veränderung des Web-Angebotes durch den Angreifer möglich, wodurch die Gefahr der Verbreitung von falschen Informationen oder die Verhinderung von Diensten besteht.

5.1.1 Eindringen über bekannte Schwachstellen in Standardsoftware

Bedrohung: Ausführen beliebiger Befehle auf dem System

Schwachstelle: Unzureichendes Patch-Management

Bekannte Schwachstellen in Standardsoftware können mit Hilfe öffentlich zugänglicher Tools oft sehr leicht ausgenutzt werden. Diese Tools sind in vielen Fällen auch von unerfahrenen Angreifern anwendbar und ermöglichen das Eindringen in die betroffenen Systeme.

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Das Patch-Management gemäß [ISi-E] kann sicherstellen, dass bekannte Schwachstellen beseitigt werden, indem die entsprechenden Sicherheits-Patches auf den Systemen eingespielt werden bzw. eine neue Version der betroffenen Software verwendet wird.

Restrisiko: Sogenannte Zero-Day-Schwachstellen – also Software-Fehler, die noch nicht veröffentlicht wurden und die nur dem Angreifer bekannt sind oder für die noch kein Patch vom Hersteller existiert – stellen ein Sicherheitsrisiko dar, das nur sehr schwer durch Maßnahmen reduziert werden kann.

Als grundsätzliche Maßnahme muss definiert werden, wie schnell ein Patch nach Veröffentlichung eingespielt werden soll. Je kürzer dieser Zeitraum ist, desto weniger Zeit verbleibt, um die Auswirkungen des Patches zu testen. Je länger dieser Zeitraum ist, desto länger sind die Systeme angreifbar. Diese Entscheidung muss bei der Ausarbeitung des Patch-Management-Prozesses getroffen werden.

Variante 5.1.1 A für hohen Schutzbedarf: Virtuelles Patching

Anwendungsbereich: hoher Schutzbedarf bezüglich Verfügbarkeit, Vertraulichkeit oder Integrität

Phase im Ablaufplan: Konzeption

Die Zeit zwischen dem Bekanntwerden einer Schwachstelle und Veröffentlichung eines Patches kann mittels „virtueller Patches“ überbrückt werden. Dabei handelt es sich um Systeme, welche bekannte Angriffsmuster im Datenstrom entdecken und die entsprechenden Anfragen nicht an die anfälligen Systeme weiterleiten. Dazu muss ein Dienst beispielsweise eines Anti-Viren-Herstellers in Anspruch genommen werden, der entsprechende Signaturen für neue Software-Schwachstellen bereitstellt. Alternativ oder zusätzlich dazu kann auch eine Filterung auf bestimmte Datei- bzw. MIME-Typen implementiert werden, wenn Sicherheitslücken für bestimmte Formate veröffentlicht werden. Neben den beschriebenen Einsatzszenarien kann virtuelles Patching auch in Umgebungen eingesetzt werden, in denen ein Service-Anbieter nicht die Möglichkeit hat, Schwachstellen direkt in den Anwendungen bzw. am System zu beheben. Dies ist beispielsweise bei einem Service-Anbieter der Fall, dessen Kunden ihre Web-Angebote selbst verwalten. Diese Variante kann auch gewählt werden, um dem Lieferanten bzw. den Softwareentwicklern einen größeren Zeitraum zum Beheben von Sicherheitsproblemen zur Verfügung zu stellen.

Verbreitete Arten für virtuelles Patching sind Web Application Firewalls (auch Application Level Firewalls genannt) und Intrusion-Prevention-Systeme. In der Praxis sollte der Behebung der Schwachstellen direkt in den Systemen grundsätzlich der Vorzug vor dem virtuellen Patching gegeben werden.

Beim Einsatz von Web Application Firewalls für virtuelles Patching muss beachtet werden, dass die von einem Drittanbieter zur Verfügung gestellten Signaturen in der Regel keine proprietären Anwendungen schützen können. Wird eine Schwachstelle in einer proprietären Anwendung gefunden, gibt es kein entsprechendes Update für die Web Application Firewall. Solche Updates können nur für Standardsoftware mit bekannten Fehlern in bekannten Konfigurationen angeboten werden. Es muss beachtet werden, dass durch virtuelles Patching nur vordefinierte „Angriffsmuster“ gefiltert werden können. Angriffe können daher nur verhindert werden, wenn diese Muster bekannt sind und umfassend implementiert werden.

Neben dem notwendigen Aufwand zum Betrieb von Systemen für virtuelles Patching muss auch der finanzielle Aufwand für deren Beschaffung und Lizenzen beachtet werden.

Abbildung 5.1 zeigt die Positionierung eines virtuellen Patching-Systems innerhalb der Grundarchitektur.

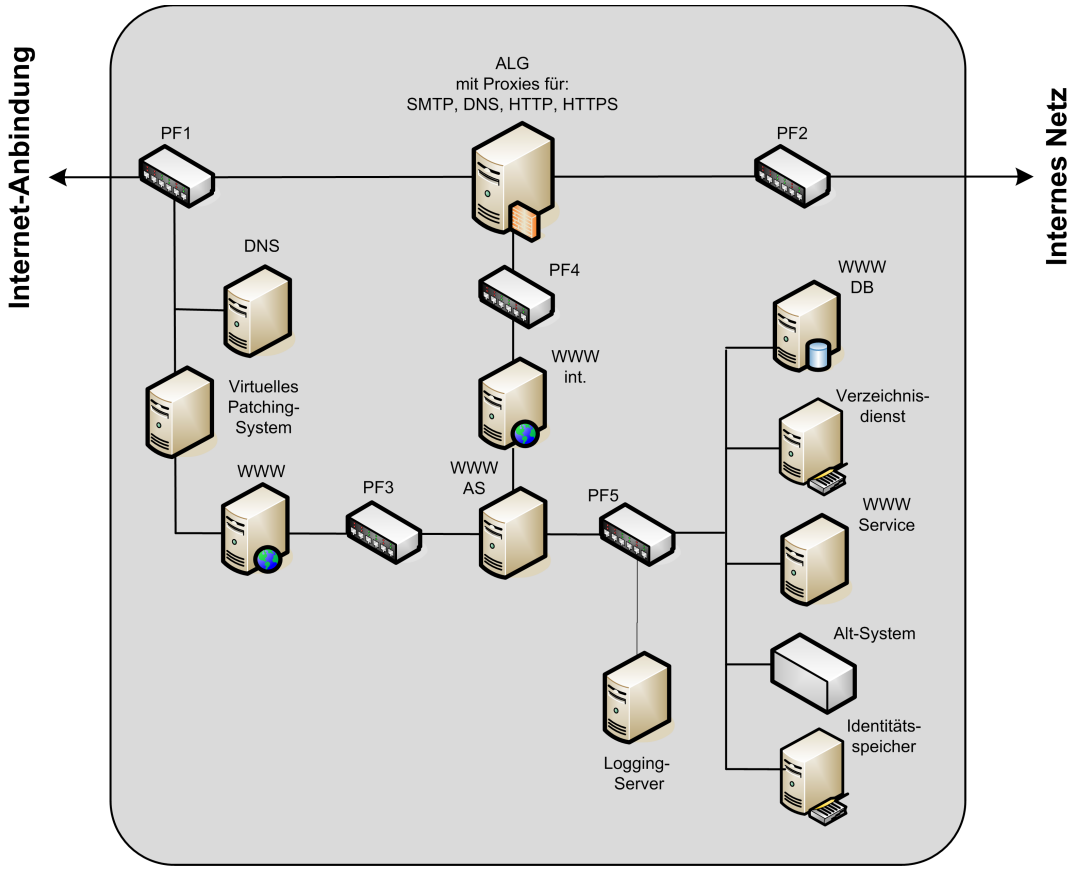


Abbildung 5.1: Sichere Grundarchitektur mit virtuellem Patching

Restrisiko: Auch bei Einsatz von virtuellem Patching besteht kein Schutz vor unveröffentlichten Schwachstellen, die zwar Angreifern, nicht jedoch dem Software-Hersteller bekannt sind.

Umsetzungsaufwand: mittlerer bis hoher Beschaffungs- und Realisierungsaufwand

Variante 5.1.1 B:Variante für hohen Schutzbedarf: Software zur Erkennung von Systemanomalien

Anwendungsbereich: hoher Schutzbedarf bezüglich Vertraulichkeit oder Integrität

Phase im Ablaufplan: Konzeption

Softwarekomponenten können erkennen, ob Anomalien in unterschiedlichen Bereichen des Systems (z. B. Call-Tree-Analyse zur Laufzeit) auftreten. Dabei kann eine große Anzahl von unterschiedlichen Parametern bei der Ausführung von Programmen überwacht werden. Überschreitet ein Messwert die gesetzten Grenzen, können entsprechende Maßnahmen wie z. B. das Herunterfahren des Systems eingeleitet werden. Host-basierte Intrusion-Detection-Systeme können dabei eingesetzt werden.

Restrisiko: Systemanomalien können zwar erkannt, nicht jedoch verhindert werden. Werden Gegenmaßnahmen wie z. B. das Herunterfahren des Systems eingeleitet, so besteht für einen Angreifer die Möglichkeit, bewusst die Verfügbarkeit des Dienstes zu stören. Zudem besteht die Gefahr, dass erfahrene Angreifer einen Weg finden, um vom System nicht erkannt zu werden.

Umsetzungsaufwand: mittlerer bis hoher Beschaffungs- und Realisierungsaufwand

5.1.2 Einschleusen von Betriebssystem-Befehlen über die Web-Anwendung (Command Execution)

Bedrohung: Ausführen beliebiger Befehle auf dem System

Schwachstelle: Unzureichende Filterung von Anfragen, Schwachstellen in der Web-Anwendung

Werden nicht ausreichend validierte Benutzereingaben in der Web-Anwendung verwendet und weiter verarbeitet, kann ein Angreifer dort Schwachstellen ausnutzen, die zur Ausführung von Befehlen mit den Rechten des Web-Anwendungs-Servers führen.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Kontextsensitive Filterung von Anfragen
- Einschränkung hinsichtlich der Aufrufe von externen Programmen

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Trotz getroffener Sicherheitsmaßnahmen können Software-Fehler nie ausgeschlossen werden, welche es einem Angreifer ermöglichen, Befehle über die Web-Anwendung auszuführen. Werden alle angeführten Maßnahmen getroffen, wird jedoch bereits ein hohes Sicherheitsniveau erreicht.

5.1.3 Eindringen über Schwachstellen im CMS

Bedrohung: Ausführen beliebiger Befehle auf dem System

Schwachstelle: Unzureichende Filterung von Anfragen, Schwachstellen im CMS

Publiziert das CMS keine statischen Inhalte, sondern verwendet dynamische Funktionen, kann das CMS auf dieselben Arten angegriffen werden wie die Web-Anwendung. Dadurch kann möglicherweise die Kontrolle über das CMS oder den Webserver übernommen werden.

Gegenmaßnahmen in der Grundarchitektur (siehe Abschnitt 3):

- CMS ist für externe Nutzer nicht erreichbar

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Anforderungen an das CMS
- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Trotz getroffener Sicherheitsmaßnahmen können Software-Fehler nie ausgeschlossen werden. Es besteht die Gefahr von Zero-Day-Schwachstellen (siehe Abschnitt 5.1.1).

Variante 5.1.3 A für hohen Schutzbedarf: Publizierung ausschließlich statischer Inhalte

Anwendungsbereich: hoher Schutzbedarf bezüglich Verfügbarkeit, Vertraulichkeit oder Integrität

Phase im Ablaufplan: Konzeption

Um den maximalen Schutz des Systems zu gewährleisten, kann z.B. ein CMS eingesetzt werden, das alle Inhalte statisch publiziert und keine dynamischen Funktionen auf dem Web-Anwendungs-Server zur Verfügung stellt. Dies schränkt die Möglichkeiten des Web-Angebotes zwar ein, ist aber in vielen Fällen umsetzbar und deutlich sicherer. Alternativ kann beim Publizieren ausschließlich statischer Inhalte auch auf den Einsatz eines CMS vollständig verzichtet werden.

Restrisiko Die Maßnahme schützt wirkungsvoll gegen Angriffe auf das CMS und das Web-Angebot. Sie schränkt jedoch die Möglichkeiten des Web-Angebotes ein.

Umsetzungsaufwand: geringer Beschaffungs- und Realisierungsaufwand, sofern die Web-Anwendung ohne dynamische Funktionen auskommt.

5.1.4 Eindringen über Speicher-Management-Schwachstellen

Bedrohung: Ausführen beliebiger Befehle am System

Schwachstelle: Speicher-Management-Schwachstelle

Haben Softwarekomponenten Speicher-Management-Schwachstellen wie z. B. Puffer-Überläufe oder Format-String-Schwachstellen, kann ein Angreifer diese ausnutzen, um eigene Instruktionen in den Speicher des betroffenen Systems zu schreiben und sie zur Ausführung zu bringen. Auf solchem Weg ist es möglich, beliebige Befehle mit den Rechten der betroffenen Komponente auszuführen und somit Kontrolle über diese zu erlangen.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Patch-Management gemäß [ISi-E] für Standardsoftware

Restrisiko: Es besteht weiterhin das Risiko von Zero-Day-Schwachstellen (siehe Abschnitt 5.1.1).

5.1.5 Eskalation der Rechte mittels lokaler Speicher-Management-Schwachstellen

Bedrohung: Erhöhung von Privilegien

Schwachstelle: Speicher-Management-Schwachstelle

Konnte ein Angreifer bereits Zugriff auf das System mit eingeschränkten Rechten erlangen, kann er mit Hilfe einer lokalen Speicher-Management-Schwachstelle wie z. B. einem Puffer-Überlauf seine Rechte erhöhen, um das System vollständig zu kompromittieren.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Chroot-Umgebung zur Einschränkung der Angriffsfläche

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Konfiguration nach dem Minimalprinzip

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Patch-Management gemäß [ISi-E] für Standardsoftware

Restrisiko: Es besteht weiterhin die Möglichkeit von Zero-Day-Schwachstellen (siehe Abschnitt 5.1.1).

5.1.6 Eindringen durch Ausnutzen von Berechtigungen

Bedrohung: Inhalte der Webseite werden dauerhaft geändert

Schwachstelle: Schwachstellen in der Konfiguration des Webservers bzw. Web-Anwendungs-Servers

Sind die Berechtigungen auf dem Webserver bzw. dem Web-Anwendungs-Server nicht ausreichend restriktiv gesetzt, kann ein Angreifer Dateien hochladen, ändern oder löschen. Kritische Funktionen, die eine Manipulation von Dateien auf dem Server erlauben (z. B. PUT und COPY des WebDAV-Protokolls), dürfen nur authentisierten Administratoren bzw. explizit berechtigten Personen zugänglich sein. Dürfen auch nicht berechtigte Benutzer Dateien auf den Webserver hochladen, ermöglicht dies einem Angreifer, eine Shell in das System einzuschleusen. Dabei handelt es sich um eine Datei, welche Befehle des Angreifers am Webserver entgegennimmt und zur Ausführung bringt.

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Minimalprinzip bei der Rechtevergabe
- Input-Validation

Restrisiko: Nicht dokumentierte Funktionen des Webservers können bei der Konfiguration übersehen werden. Wird eine solche Funktion einem Angreifer bekannt, so kann dieser die Schwachstelle möglicherweise dazu nutzen, um das System zu kompromittieren.

5.1.7 Fehler in Datei-Uploads zur Ausführung von Betriebssystem-Befehlen

Bedrohung: Ausführen beliebiger Befehle auf dem System

Schwachstelle: Unzureichende Prüfung von Dateien, die hochgeladen werden

Nutzt ein Angreifer eine Schwachstelle, die es erlaubt, Dateien auf dem Server zu erstellen und zur Ausführung zu bringen, so kann er auch eine Shell auf dem betroffenen System platzieren. Dadurch können beliebige Befehle mit den Rechten des Web-Anwendungs-Servers ausgeführt werden.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Einschränkungen hinsichtlich Datei-Upload und -Erzeugung

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Das Restrisiko für Fehler in Datei-Uploads ist sehr gering. Ein Restrisiko besteht weiterhin durch Zero-Day-Schwachstellen (siehe Abschnitt 5.1.1).

Variante 5.1.7 A für hohen Schutzbedarf: Schreibgeschütztes System

Anwendungsbereich: hoher Schutzbedarf bezüglich Integrität

Phase im Ablaufplan: Konzeption

Um den maximalen Schutz des Systems zu gewährleisten, kann das gesamte System auf einem schreibgeschützten Medium wie z. B. einer DVD in einem DVD-ROM Laufwerk betrieben werden.

Restrisiko Wird das System auf einem schreibgeschützten Medium betrieben, so stehen unter Umständen flüchtige Dateisysteme zur Verfügung, welche im Speicher des Systems betrieben werden. Unter gewissen Umständen kann ein Angreifer in diesen Speicher des Systems schreiben. Das Auftreten einer derartigen Schwachstelle und die Ausnutzung durch einen Angreifer sind jedoch unwahrscheinlich und selbst in diesem Fall steht nach einem Neustart des Systems der originale Inhalt wieder zur Verfügung.

Umsetzungsaufwand: geringer Beschaffungs- und Realisierungsaufwand, sofern die Web-Anwendung ohne Adaption auf einem schreibgeschützten Medium lauffähig ist

5.1.8 Ausführen von Befehlen beim Einbinden von Dateien mit Schadcode

Bedrohung: Ausführen beliebiger Befehle auf dem System

Schwachstelle: Dateinamen eingebundener Dateien können manipuliert werden

Fließen nicht ausreichend validierte Benutzereingaben in Dateinamen von Dateien ein, die vom Web-Anwendungs-Server eingebunden und ausgeführt werden, können kritische Dateien lokal oder – abhängig von der Konfiguration – auch aus der Ferne eingebunden werden. Damit besteht die Möglichkeit, dass ein Angreifer Programmcode für eine Shell einbindet und so beliebige Befehle mit den Rechten des Web-Anwendungs-Servers ausgeführt werden können.

Gegenmaßnahmen in der Grundarchitektur (siehe Abschnitt 3):

- Kontextsensitive Filterung von Anfragen

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Restriktionen beim Einbinden von Dateien

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Werden alle angeführten Maßnahmen umgesetzt, besteht nur noch eine geringe Wahrscheinlichkeit, dass derartige Schwachstellen im Web-Angebot auftreten. Insbesondere bei von Drittanbietern entwickelten Systemen besteht die Gefahr von Zero-Day-Schwachstellen (siehe Abschnitt 5.1.1).

Variante 5.1.8 A für hohen Schutzbedarf: Deaktivierung der Einbindungsfunktion

Anwendungsbereich: hoher Schutzbedarf bezüglich Verfügbarkeit, Vertraulichkeit oder Integrität

Phase im Ablaufplan: Konzeption, Konfiguration

Um das Einbinden von externen Dateien vollständig in der Konfiguration sperren zu können, müssen die dynamisch eingebundenen Inhalte vor dem Ausrollen auf den Webserver bereits fest eingebunden werden. Dies ist abhängig von der Web-Anwendung mehr oder weniger aufwendig, stellt jedoch sicher, dass keine Dateien dynamisch zur Laufzeit eingebunden werden können.

Restrisiko: Wird die Funktion zum Einbinden von Dateien vollständig deaktiviert, besteht für einen Angreifer keine Möglichkeit mehr, fremden Code einzubinden. Mit der Deaktivierung kann jedoch auch eine Einschränkung der Funktionalität bzw. Mehraufwand bei Konfiguration und Wartung einhergehen.

Umsetzungsaufwand: geringer bis hoher Realisierungsaufwand; geringer Konfigurationsaufwand

5.2 Gefährdungen durch Verändern, Täuschen, Fälschen und Betrügen

Tricks von Angreifern wie das Manipulieren von Web-Angeboten und daraus resultierender Betrug sind eine große Gefahrenquelle im Web. Phishing ist ein häufiger Angriff in dieser Gefährdungskategorie. Darunter wird der Versuch verstanden, mit Hilfe von gefälschten Nachrichten oder Web-Auftritten den Benutzer zur Herausgabe von vertraulichen Daten (z. B. Anmeldeinformationen zu Online-Banking-Systemen) zu bewegen. Es gibt jedoch auch zahlreiche weitere Angriffe, die sich das Fälschen und Täuschen zunutze machen.

5.2.1 Verschleierung von Angriffen

Bedrohung: Manipulation von Log-Dateien, Verschleiern von Angriffen

Schwachstelle: Log-Einträge werden nicht ausreichend geschützt abgelegt

Werden Log-Dateien nicht ausreichend geschützt, kann ein Angreifer seine Spuren verwischen, indem er diese Informationen löscht bzw. manipuliert. Dies erschwert nicht nur das Erkennen von erfolgreichen Angriffen, sondern macht die Informationen auch wertlos für eine strafrechtliche Verfolgung der Täter.

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Absicherung des Logging-Servers (kryptografischer Zeitstempel)

Restrisiko: Die angeführten Maßnahmen schützen ausschließlich die bereits abgelegten Daten. Gelingt es einem Angreifer, die Daten vor deren Speicherung auf dem Logging-Server zu manipulieren, kann er seine Angriffe dennoch verschleiern.

5.2.2 Umgehen von Logging-Mechanismen

Bedrohung: Verschleiern von Angriffen

Schwachstelle: Manipulierbarkeit der vorgesehenen Seitenabfolge

Logging muss Server-seitig erfolgen und darf nicht durch den Benutzer beeinflusst werden können. Einige verbreitete Logging-Varianten vertrauen darauf, dass eingebundene Links

automatisch vom Browser aufgerufen werden und die zu protokollierende Information an den Logging-Server senden. Dies kann in der Applikation z. B. über ein eingebundenes Bild erfolgen. Da ein Angreifer diese Abfragen sehr einfach z. B. über einen konfigurierbaren Proxy verwerfen kann, verhindert er die Protokollierung dieser Anfragen, da diese nicht bis zum Server gelangen.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Logging auf Server-Ebene und nicht über eine eigene Schnittstelle in der Web-Anwendung

Restrisiko: Schwachstellen im Logging-Mechanismus selbst können den Schutz aushebeln, den das sichere Design gewährleisten soll. Wird der Logging-Mechanismus nicht in sicherer Art und Weise implementiert, kann ein Angreifer das ordnungsgemäße Logging stören oder dieses außer Kraft setzen.

5.2.3 Verändern der Webseite durch Ausnutzen von Berechtigungen

Bedrohung: Inhalte der Webseite werden dauerhaft geändert

Schwachstelle: Schwachstellen in der Konfiguration des Webservers bzw. Web-Anwendungs-Servers

Sind die Berechtigungen auf dem Webserver bzw. dem Web-Anwendungs-Server nicht ausreichend restriktiv gesetzt, kann ein Angreifer Dateien hochladen, ändern oder löschen und somit die Kontrolle über das System erlangen (siehe Abschnitt 5.1.6).

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Minimalprinzip bei der Rechtevergabe

Restrisiko: Nicht dokumentierte Funktionen des Webservers können bei der Konfiguration übersehen werden. Erlangt ein Angreifer Kenntnis von einer solchen Funktion, so kann er diese möglicherweise dazu benutzen, um das betroffene Web-Angebot zu kompromittieren. Ein weiteres Restrisiko geht von Zero-Day-Schwachstellen aus (siehe Abschnitt 5.1.1).

Variante 5.2.3 A für hohen Schutzbedarf: Schreibgeschütztes System

Details siehe Variante im Abschnitt 5.1.7.

5.2.4 HTML Injection und HTTP Response Splitting

Bedrohung: Inhalte der Webseite werden temporär verändert/erweitert dargestellt

Schwachstelle: Unzureichende Filterung von Anfragen, Spoofing-Schwachstellen

Werden Benutzereingaben ohne ausreichende Validierung in die Antwort des Webservers eingebaut, kann ein Angreifer Anfragen bzw. Links erstellen, die den Server dazu veranlassen, Inhalte des Angreifers wiederzugeben. Klickt ein Benutzer auf einen solchen Link bzw. wird er durch andere Angriffe dazu gebracht, eine manipulierte Anfrage an den Server zu senden, kann er nicht mehr unterscheiden, welche Elemente von der Web-Anwendung selbst erstellt und welche Elemente vom Angreifer hinzugefügt wurden.

Einige Seiten bieten beispielsweise Weiterleitungen zu anderen Web-Angeboten an. Oftmals wird die Adresse des Web-Angebots als Parameter in der URL übergeben und ist somit von einem Angreifer frei manipulierbar. Wird dieser Parameter nicht ausreichend geprüft (hier kann eine Whitelist mit zugelassenen Weiterleitungen sinnvoll sein), so kann der Angreifer eine Link erzeugen, der zu einer vertrauenswürdigen Seite gehört, den Nutzer jedoch auf eine andere Seite weiterleitet. Eine solcher verschleierter Link kann beispielsweise so aussehen:

www.vertrauenswürdigeseite.de/redirect?url=www.boeseseite.de

Werden die Benutzereingaben nicht ausreichend geprüft, so kann es einem Angreifer auch gelingen zusätzliche Zeichen im Header der Antwort zu platzieren. So ist es ihm möglich Cookies zu setzen oder Nutzdaten einzufügen.

Dies kann unter anderem dazu missbraucht werden, Falsch-Informationen zu verbreiteten bzw. hybride Phishing-Angriffe durchzuführen. Unter diesen versteht man das Ausnutzen einer Cross-Site-Scripting-Schwachstelle für Phishing-Angriffe.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Kontextsensitive Eingabefilterung
- Kontextsensitive Ausgabefilterung

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Trotz getroffener Sicherheitsmaßnahmen können Software-Fehler nie ausgeschlossen werden. Schafft es beispielsweise ein Angreifer, die Ausgabefilterung eines Web-Angebotes zu umgehen, so können dadurch beliebige Inhalte in die Seite eingebunden werden.

5.2.5 Session Riding

Bedrohung: Ausführen von Funktionen im Kontext eines privilegierten Benutzers

Schwachstelle: Unzureichendes Session-Management

Überprüft eine Web-Anwendung nicht, ob Anfragen, die den Zustand der Web-Anwendung beeinflussen, aus der Web-Anwendung selbst heraus gestellt wurden, kann ein Angreifer die aktive Sitzung eines Benutzers nutzen, um mit den Rechten des betroffenen Benutzers Anfragen an die Web-Anwendung abzusetzen.

Ein Angreifer kann so z.B. die Web-Anwendung dazu bringen, das Passwort eines Benutzers auf einen vom Angreifer definierten Wert zu ändern. Dazu muss der Angreifer die dafür notwendige Anfrage kennen und den angemeldeten Benutzer bzw. dessen Browser dazu bringen, diese Anfrage abzusetzen. Dies kann beispielsweise über einen manipulierten Link (www.beispiel.de/changepasswd?password=kennt_der_angreifer) erfolgen. Wird der Link aufgerufen, wird über die aktive Sitzung die Anfrage an der Web-Anwendung abgesetzt und dort ausgeführt. Die notwendigen Authentisierungsdaten werden von dem Browser, unbe-

merkt vom Benutzer, mitgeschickt, da sich der Benutzer in der aktiven Session befindet und sich bereits authentisiert hat.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Session-Management mit zusätzlicher flüchtiger Authentisierung

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Ein Angreifer kann durch Zufall die flüchtigen Authentisierungsdaten erraten. Werden zu deren Erzeugung kryptografische Zufallszahlengeneratoren verwendet, ist es jedoch sehr unwahrscheinlich, dass die Daten erraten werden können. Ein weiteres Restrisiko geht von Cross-Site-Scripting-Schwachstellen aus, durch welche sowohl die Session-ID als auch flüchtige Authentisierungsdaten ausgelesen werden können.

5.2.6 Replay-Angriffe

Bedrohung: Ausführen von Funktionen im Kontext eines privilegierten Benutzers

Schwachstelle: Absender von Anfragen wird nicht eindeutig identifiziert

Selbst wenn alle Anfragen an ein Hintergrundsystem gültig signiert sein müssen, um akzeptiert zu werden, kann ein Angreifer eine solche signierte Anfrage abfangen und erneut an das Zielsystem senden. Dadurch kann der Befehl beliebige weitere Male abgesetzt werden. Diese Art des Angriffs wäre z. B. bei der Durchführung einer Überweisung denkbar.

Gegenmaßnahmen in der Grundarchitektur (siehe Abschnitt 3):

- Verschlüsselung bei Kommunikation über unsichere Kanäle

Restrisiko: Ein Angreifer kann die Verschlüsselung brechen bzw. ein interner Mitarbeiter kann mit den entsprechenden Zugriffsrechten potenziell die Kommunikation mithören, bevor der kryptografische Tunnel aufgebaut wird.

Variante 5.2.6 A für hohen Schutzbedarf: Zusätzlicher Zeitstempel für Anfragen

Anwendungsbereich: hoher Schutzbedarf bezüglich Integrität

Phase im Ablaufplan: Konzeption

Jede Anfrage an ein Hintergrundsystem muss mit einem Zeitstempel versehen werden, der bei der Signatur der Anfrage ebenfalls mit signiert wird. Das Empfänger-System überprüft, ob es bereits ein Paket mit dem gesetzten Zeitstempel empfangen hat und verwirft das Paket bei positiver Prüfung. Ein Zeitstempel wird daher verwendet, da dessen Gültigkeitsdauer limitiert ist und somit die entsprechenden Tabellen mit empfangenen Zeitstempeln überschaubar gehalten werden können.

Restrisiko: Trotz getroffener Sicherheitsmaßnahmen können Software-Fehler nie ausgeschlossen werden. Bei Auftreten von Zero-Day-Schwachstellen (siehe Abschnitt 5.1.1) in kryptografischen oder Zeitstempelalgorithmen sind weiterhin Angriffe möglich.

Umsetzungsaufwand: mittlerer Realisierungsaufwand

5.3 Gefährdungen durch Entwenden und Ausspähen

Da das Web für immer vertraulichere Geschäftsangelegenheiten verwendet wird, ist die Sicherstellung der Vertraulichkeit besonders wichtig. Folgender Abschnitt beschreibt Angriffe, die das Entwenden oder Ausspähen von Informationen ermöglichen, und zeigt Gegenmaßnahmen auf.

5.3.1 Entwenden der Identität eines legitimen Benutzers

Bedrohung: Stehlen und Verwenden gültiger Sitzungen

Schwachstelle: Unzureichendes Session-Management

Genügt das Session-Management nicht den notwendigen Sicherheitsanforderungen (z. B. ist die Session an eine nicht ausreichend komplexe Session-ID gebunden), kann ein Angreifer eine aktiv bestehende Sitzung eines legitimen Benutzers übernehmen und dessen Rechte nutzen, als hätte er die Zugangsdaten des betroffenen Benutzers entwendet.

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Vorgaben an das Session-Management

Restrisiko: Schwachstellen wie z. B. Cross-Site Scripting oder mangelnde Verschlüsselung in der Kommunikation ermöglichen es einem Angreifer trotz sicheren Session-Managements, eine gültige Session und damit auch die Identität eines legitimen Benutzers zu übernehmen. Auf diese Weise lassen sich schädigende Aktionen unter Verwendung des erlangten Benutzerkontos vornehmen.

Variante 5.3.1 A für hohen Schutzbedarf: Session an IP-Adresse binden

Anwendungsbereich: hoher Schutzbedarf bezüglich Vertraulichkeit oder Integrität

Phase im Ablaufplan: Konzeption

Um das Verwenden einer gestohlenen Session zu verhindern, kann die Session bei hohem Schutzbedarf an die IP-Adresse des Benutzers gebunden werden, der sich an der Web-Anwendung anmeldet. Dies hat allerdings den Nachteil, dass Benutzern mit dynamisch wechselnder IP-Adresse bei einem solchen Wechsel die Session terminiert wird.

Restrisiko: Ein Angreifer, der sich hinter demselben NAT-System wie der betroffene Benutzer befindet, kann die Session dennoch verwenden; Trojanische Pferde, die auf Cross-Site Scripting basieren, können trotz Bindung der Session an die IP-Adresse die aktive Sitzung missbrauchen.

Umsetzungsaufwand: geringer Realisierungsaufwand

5.3.2 Ausspähen von Konten bzw. gültigen Sitzungen im Datenstrom

Bedrohung: Stehlen von Administrator- und Benutzerkonten, Stehlen bzw. Verwenden gültiger Sitzungen

Schwachstelle: Kommunikation über unsichere Kanäle erfolgt unverschlüsselt

Bei mangelnder Verschlüsselung der Kommunikation über unsichere Kanäle kann ein Angreifer sensible Daten mithören und verwenden. Gelangt er auf diesem Weg in Besitz von gültigen Zugangsdaten zu bestehenden Konten bzw. kann er aktive Session-IDs mitlesen, können die betroffenen Konten kompromittiert werden.

Gegenmaßnahmen in der Grundarchitektur (siehe Abschnitt 3):

- Verschlüsselung bei Kommunikation über unsichere Kanäle

Restrisiko: Die verwendeten Algorithmen können gebrochen und die Kommunikation auf diesem Weg entschlüsselt werden. Ein Angreifer hat dadurch die Möglichkeit, die übertragenen Daten im Klartext einzusehen und somit an Zugangs- oder Session-Daten zu gelangen.

5.3.3 Ausspähen von Konten und sensiblen Inhalten

Bedrohung: Stehlen von Administrator- und Benutzerkonten

Schwachstelle: Sensible Daten liegen unverschlüsselt vor

Werden sensible Daten unverschlüsselt auf einem der Systeme abgelegt, kann ein Angreifer nach erfolgreicher Kompromittierung des betroffenen Systems diese Daten auslesen.

Gegenmaßnahmen in der Grundarchitektur (siehe Abschnitt 3):

- Sensible Daten dürfen nur verschlüsselt abgelegt werden

Restrisiko: Die verwendeten Algorithmen können gebrochen und die Daten auf diesem Weg entschlüsselt bzw. der Schlüssel ermittelt werden. Wird der zur Verschlüsselung verwendete kryptografische Schlüssel nicht sicher aufbewahrt (z. B. im Klartext gemeinsam mit den verschlüsselten Daten), so kann ein Angreifer Zugriff auf den Schlüssel erlangen und damit die Daten entschlüsseln. Einem Angreifer kann es auch gelingen die Daten aus dem Hauptspeicher auszulesen.

Variante 5.3.3 A für hohen Schutzbedarf: Schutz des kryptografischen Schlüssels mit Passwort

Anwendungsbereich: hoher Schutzbedarf bezüglich Vertraulichkeit

Phase im Ablaufplan: Konzeption

Um zu verhindern, dass ein Angreifer den kryptografischen Schlüssel bzw. das Verschlüsselungszertifikat direkt auslesen kann, kann dieses mit einem Passwort geschützt werden. Dieses Passwort muss nach jedem Neustart des betroffenen Servers manuell eingegeben werden. Durch diese Variante wird jedoch ein automatischer Neustart unterbunden. Dies kann unter Umständen nicht erwünscht sein.

Restrisiko: Das Passwort kann z. B. durch Social Engineering in Erfahrung gebracht werden. Ferner besteht ein Restrisiko, dass der Besitzer des kryptografischen Schlüssels das Passwort nicht ausreichend sicher gewählt hat, wodurch dieses durch Brute-Force-Angriffe von einem Angreifer in Erfahrung gebracht werden kann. Einem Angreifer kann es auch gelingen den Schlüssel aus dem Hauptspeicher auszulesen.

Umsetzungsaufwand: Geringer Konfigurationsaufwand

5.3.4 Ausspähen und Manipulieren von Daten in Hintergrundsystemen nach Kompromittierung der Web-Anwendung

Bedrohung: Unautorisierter Zugriff auf Inhalte von Hintergrundsystemen

Schwachstelle: Trusted-Client-Prinzip zwischen Web-Anwendungs-Server und Hintergrundsystemen

Authentifiziert sich der Web-Anwendungs-Server mit einem technischen Benutzer an einem Hintergrundsystem, kann ein Angreifer diese Rechte nach Kompromittierung des Web-Anwendungs-Servers nutzen, um auf das Hintergrundsystem zuzugreifen. Beispielsweise kann ein Angreifer dadurch Zugang zu einer Datenbank mit sensiblen Inhalten erlangen.

Regelt der Web-Anwendungs-Server selbst die Berechtigungen, die der jeweilige Benutzer am Hintergrundsystem hat, kann dieses Berechtigungssystem nach Kompromittierung des Systems umgangen werden.

Gegenmaßnahmen in der Grundarchitektur (siehe Abschnitt 3):

- Kein Trusted-Client-Prinzip

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Der Web-Anwendungs-Server greift nicht mit einem technischen Benutzer auf die Hintergrundsysteme zu, sondern verwendet hierfür die Authentisierungsinformation des jeweiligen Benutzers.

Restrisiko: Ein Angreifer, der den Web-Anwendungs-Server erfolgreich kompromittiert hat, kann eventuell die notwendigen Authentisierungsinformationen von legitimen Benutzern abfangen und verwenden, um zumindest einen Teil der Daten aus dem Hintergrundsystem auslesen zu können. Des Weiteren besteht die Gefahr, dass ausgehend vom kompromittierten System Hintergrundsysteme angegriffen werden.

5.3.5 Auslesen von Daten aus den Hintergrundsystemen über Injection-Schwachstellen

Bedrohung: Unautorisierter Zugriff auf Inhalte von Hintergrundsystemen

Schwachstelle: Unzureichende Filterung von Anfragen

Fließen nicht ausreichend validierte Benutzereingaben in Abfragen an ein Hintergrundsystem ein, können diese Abfragen manipuliert werden. Während die Schwachstelle selbst unabhängig vom verwendeten Hintergrundsystem ist, sind die Auswirkungen davon stark abhängig.

Klassische Gefährdungen sind z. B. SQL Injection (bei Datenbank-Systemen), LDAP Injection (bei Verzeichnis-Servern) und XPATH Injection (bei Verarbeitung von XML-Dokumenten). Die Auswirkungen reichen vom unberechtigten Auslesen bzw. Manipulieren von Daten bis hin zur vollständigen Kompromittierung des Hintergrundsystems.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Kontextsensitive Eingabefilterung

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Es besteht die Gefahr, dass der verwendete Eingabefilter eine Schwachstelle aufweist und es einem Angreifer dadurch möglich ist, trotz Filterung eine Injection-Schwachstelle auszunutzen.

5.3.6 Ausspähen von Informationen über Path Traversal

Bedrohung: Unautorisierter Zugriff auf Inhalte des Web-Angebotes, Auslesen von Systeminformationen, Systemdateien und Quellcode

Schwachstelle: Unzureichende Filterung von Anfragen, Dateinamen eingebundener Dateien können manipuliert werden

Kann der Pfad von Dateien, auf die zugegriffen wird, durch Benutzereingaben manipuliert werden, kann ein Angreifer andere Dateien als vorgesehen ansprechen und auf diesem Weg Informationen wie z. B. Quellcode, Passwörter oder Konfigurationseinstellungen einsehen.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Kontextsensitive Eingabefilterung
- Rechtevergabe nach dem Minimalprinzip

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Auch wenn die Web-Anwendung selbst sicher implementiert ist, kann der Web-Anwendungs-Server oder der Webserver anfällig auf Path Traversal sein. Auch das falsche Zusammenspiel unterschiedlicher Systeme kann eine Path-Traversal-Schwachstelle ermöglichen. Lange Zeit gab es z. B. zwischen Apache und Tomcat ein derartiges Problem, das einem Angreifer Zugriff auf geschützte Bereiche ermöglichte.

Variante 5.3.6 A für hohen Schutzbedarf: Pfad-Normalisierung

Anwendungsbereich: hoher Schutzbedarf bezüglich Vertraulichkeit

Phase im Ablaufplan: Implementierung

Um Fehler bei der Implementierung der Filter zu vermeiden, kann der übergebene Pfad zuerst einer Pfad-Normalisierung unterzogen werden. Dabei werden Strings, die bei Path-Traversal-Angriffen häufig verwendet werden, beseitigt. Dabei müssen auch die unterschiedlichen Codierungen beachtet werden.

Restrisiko: Schwachstellen bei der Implementierung der Pfad-Normalisierung können dazu führen, dass die Filterfunktion von einem Angreifer umgangen werden kann.

Umsetzungsaufwand: Erleichtert die Implementierung des Eingabefilters hinsichtlich Path Traversal.

5.3.7 Erraten von Zugangsdaten

Bedrohung: Stehlen von Administrator- und Benutzerkonten

Schwachstelle: Verwundbarkeit gegen Brute-Force und Wörterbuch-Angriffe bzw. unzureichende Schutzmaßnahmen; Enumeration ist möglich; Default- und Test-Kontos werden nicht entfernt; Verwendung von unzureichenden Passwörtern

Zahlreiche Schwachstellen können dazu führen, dass ein Angreifer Zugangsdaten zu einem System errät. Werden bekannte Standard-Konten nicht entsprechend geschützt, kann ein Angreifer die bekannten Zugangsdaten nutzen, um die Rechte dieser Benutzer zu erlangen. Werden die Passwörter zwar geändert, jedoch nicht ausreichend sicher gewählt, kann ein Angreifer einen erfolgreichen Brute-Force-Angriff auf Basis eines Wörterbuchs bzw. einfacher Zeichenkombinationen durchführen. Wird dies nicht durch eine Teergrube erschwert, kann der Angreifer abhängig von der Komplexität der Passwörter unter Umständen sehr rasch an Zugangsdaten gelangen.

Bietet das System zusätzlich noch die Möglichkeit, gültige Benutzernamen in Erfahrung zu bringen, kann der Angreifer seine Angriffsfläche entsprechend erhöhen, indem er mehrere Benutzer angreift.

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Entfernen von Default-Konten
- Einsatz einer Teergrube
- Passwortrichtlinie

Restrisiko: Passwörter können durch Social Engineering ermittelt werden. Außerdem kann nicht verhindert werden, dass ein sicheres Passwort durch Zufall erraten wird, auch wenn dies sehr unwahrscheinlich ist.

Variante 5.3.7 A für hohen Schutzbedarf: Mehr-Faktoren-Authentisierung

Anwendungsbereich: hoher Schutzbedarf bezüglich Vertraulichkeit und Integrität

Phase im Ablaufplan: Konzeption

Für die Authentisierung können mehrere Mechanismen parallel eingesetzt werden. Z. B. können bei einer Zwei-Faktoren-Authentisierung sowohl der Besitz einer Chipkarte als auch die Kenntnis der zugehörigen PIN sicherstellen, dass der Benutzer der ist, für den er sich ausgibt. Abhängig von den Sicherheitsanforderungen können die Anzahl und die Art der Mechanismen gewählt werden.

Restrisiko: Es besteht die Möglichkeit, dass alle eingesetzten Systeme von einem Angreifer überlistet werden, auch wenn die Wahrscheinlichkeit dazu sehr gering ist.

Umsetzungsaufwand: Abhängig von den Mechanismen

5.3.8 Entwenden von Zugangsdaten über die „Passwort zurücksetzen“-Funktion

Bedrohung: Stehlen von Administrator- und Benutzerkonten

Schwachstelle: Verwundbarkeit der „Passwort zurücksetzen“-Funktion; unverschlüsselte Kommunikation über unsichere Kanäle

Wird eine „Passwort zurücksetzen“-Funktion angeboten, bietet diese bei schlechter Implementierung Angriffsmöglichkeiten. Werden Zugangsdaten über unsichere Kanäle wie z. B. nicht verschlüsselte E-Mails versendet, können diese Daten ausgespäht werden.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Vorgaben für die „Passwort zurücksetzen“-Funktion

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Der versendete Link kann von einem Angreifer abgefangen und selbst zum Zurücksetzen des Passworts verwendet werden.

Variante 5.3.8 A für hohen Schutzbedarf: Verzicht auf „Passwort zurücksetzen“-Funktion

Anwendungsbereich: hoher Schutzbedarf bezüglich Vertraulichkeit oder Integrität

Phase im Ablaufplan: Konzeption

Die Implementierung der „Passwort zurücksetzen“-Funktion in der Web-Anwendung kann auf ein anderes Medium wie z. B. ein Call-Center ausgelagert werden. Dies erhöht zwar den Aufwand im Betrieb deutlich, verhindert jedoch Angriffe, die auf Schwachstellen in Web-Anwendungen basieren. Ein entsprechender Prozess muss definiert werden.

Alternativ dazu kann man dem Benutzer bei der Anmeldung die Möglichkeit bieten, die „Passwort zurücksetzen“-Funktion zu deaktivieren, wenn man diese Entscheidung den Benutzern überlassen möchte.

Restrisiko: Ein Call-Center kann anfällig sein für Social Engineering. Dadurch besteht beispielsweise die Möglichkeit, dass ein Angreifer Mitarbeiter des Call-Cen-

ters davon überzeugt, das Passwort für einen gewünschten Benutzer zurückzusetzen.

Umsetzungsaufwand: Hoher Aufwand im Betrieb

5.3.9 Auf Cross-Site Scripting basierende Trojanische Pferde

Bedrohung: Stehlen von Administrator- und Benutzerkonten, Stehlen bzw. Verwenden gültiger Sitzungen

Schwachstelle: Unzureichende Filterung von Ausgaben

Ist der Ausgabefilter nicht ausreichend sicher, kann ein Angreifer (durch verschiedene Angriffsvektoren) Script-Code im Kontext der betroffenen Domain im Browser des Benutzers zur Ausführung bringen. Dies kann z. B. auch mit Hilfe von Social Engineering durchgeführt werden.

Als Schadcode für diesen Angriff kann ein auf Cross-Site Scripting basierendes Trojanisches Pferd gewählt werden, das es einem Angreifer ermöglicht, die gesamte Kommunikation zwischen Opfer und Webserver auszuspähen. Zusätzlich kann der Angreifer beliebige Anfragen mit den Rechten des Opfers an den Webserver vom kompromittierten Client aus stellen. Auf diesem Weg ist es möglich, Session Hijacking, also das Übernehmen von Sitzungen eines Benutzers, durchzuführen, selbst wenn die Session an die IP-Adresse gebunden ist.

Je nach Art des auf Cross-Site Scripting basierenden Trojanischen Pferds können weitere Funktionen, wie z. B. ein Keylogger, vom Angreifer verwendet werden. Damit ist es möglich, alle vom Benutzer durchgeführten Tastatureingaben aufzuzeichnen und an den Angreifer zu übermitteln.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Kontextsensitive Ausgabefilterung

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

5.3.10 Cross-Site-Scripting-Angriffe über Log-Dateien

Bedrohung: Stehlen und verwenden gültiger Sitzungen

Schwachstelle: Log-Einträge werden ungefiltert geschrieben

Werden die Log-Dateien vom Administrator über ein Web-Interface eingesehen, kann ein Angreifer bei mangelnder Ausgabefilterung auch über diesen Weg einen Cross-Site-Scripting-Angriff starten. Dies ist besonders gefährlich, da über die Log-Files die Zielgruppe der Administratoren erreicht werden kann, deren Benutzerkonten für Angreifer ein besonders lohnendes Ziel darstellen.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Kontextsensitive Ausgabefilterung bei Log-Dateien

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Immer wieder werden Browser-Schwachstellen entdeckt, die Cross-Site Scripting ermöglichen, ohne dass die Web-Anwendung selbst eine Schwachstelle aufweisen muss. Das Ausnutzen einer solchen Sicherheitslücke kann jedoch Server-seitig nicht verhindert werden.

Variante 5.3.10 A für hohen Schutzbedarf: Kein Zugriff auf Log-Dateien über Browser

Anwendungsbereich: Variante für hohen Schutzbedarf

Phase im Ablaufplan: Konzeption

Wenn die Log-Dateien nicht über einen Browser abgerufen werden können, ist dieser Angriffsvektor nicht möglich

Restrisiko: Abhängig vom verwendeten Programm, mit dem die Log-Dateien geöffnet werden, können andere Schwachstellen wie z. B. Speicher-Management-Schwachstellen zu Problemen führen. Dabei muss ein Angreifer die Inhalte der Log-Dateien so manipulieren können, dass spezifische Schwachstellen der zur Analyse verwendeten Werkzeuge ausgenutzt werden. Die Wahrscheinlichkeit dazu wird als gering eingeschätzt.

Umsetzungsaufwand: Geringer Realisierungsaufwand

5.3.11 Ausspähen von Systeminformationen über System- und Fehlermeldungen

Bedrohung: Auslesen von Systeminformationen, Systemdateien und Quellcode

Schwachstelle: Systeminformationen werden preisgegeben; detaillierte Fehlermeldungen werden ausgegeben

Gibt eine Web-Anwendung im Fehlerfall detaillierte Informationen über die Ursache des Fehlers aus, kann ein Angreifer diese Informationen für weiterführende Angriffe nutzen. Solche Informationen können z. B. Pfade, Versionsnummern oder Abfragen an Hintergrundsysteme enthalten.

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Anforderungen an die Web-Anwendung hinsichtlich System- und Fehlermeldungen

Restrisiko: Eine umfassende Prüfung aller möglichen Zustände des Gesamtsystems und dessen Reaktion darauf ist in der Regel nicht durchführbar, wodurch hier leicht Schwachstellen übersehen werden können.

Variante 5.3.11 A für hohen Schutzbedarf: Beenden einer Session im Fehlerfall

Anwendungsbereich: hoher Schutzbedarf bezüglich Vertraulichkeit oder Integrität

Phase im Ablaufplan: Konzeption

Treten Fehler in der Web-Anwendung auf, die nicht erwartet wurden, kann die betreffende Session beendet werden, um weitere Angriffe in derselben Session zu unterbinden. Dies schützt vor Schwachstellen, die mehrere Schritte zur erfolgreichen Ausnutzung benötigen, da die Anwendung zuerst in einen bestimmten „fehlerhaften“ Zustand versetzt werden muss.

Restrisiko: Ist die Abfolge des mehrstufigen Angriffs nicht von der Session abhängig, kann diese Maßnahme keinen Schutz bieten. Dies ist beispielsweise bei Web-Angeboten möglich, bei welchen ein Benutzer nicht angemeldet sein muss und auch keine Session-IDs für gerade auf dem System arbeitende Benutzer geführt werden. Die Verfügbarkeit und der Nutzerkomfort verringern sich bei dieser Variante.

Umsetzungsaufwand: Mittlerer Realisierungsaufwand

5.3.12 Ausspähen von sensiblen Informationen aufgrund unzureichenden Zugriffsschutzes

Bedrohung: Auslesen von Systeminformationen, Systemdateien und Quellcode, unautorisierter Zugriff auf Inhalte des Web-Angebotes

Schwachstelle: Schwachstellen in der Konfiguration des Webservers bzw. Web-Anwendungs-Servers

Bei Fehlkonfiguration des Webservers bzw. des Web-Anwendungs-Servers kann ein Angreifer Zugriff auf sensible Informationen wie z. B. Quellcode oder Inhalte der Web-Anwendung erhalten. Ein klassischer Fehler in diesem Bereich ist die Ausgabe von Include-Dateien, wenn die Datei über den Browser direkt angesprochen wird und der Webserver aufgrund der Datei-Endung eine falsche Funktion ausführt – in diesem Fall die Ausgabe des Quellcodes.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Konfiguration der Systeme mit minimalen Rechten

Restrisiko: Zero-Day-Schwachstellen können genutzt werden, um derartige Angriffe auch bei einem sicher konfigurierten System durchzuführen (siehe Abschnitt 5.1.1).

5.3.13 Ausspähen durch die Verletzung des Minimalprinzips

Bedrohung: Auslesen von Systeminformationen, Systemdateien und Quellcode

Schwachstelle: Nicht-Einhalten des Minimalprinzips

Werden nicht benötigte Dateien auf Systemen vergessen, kann ein Angreifer unter Umständen sensible Daten aus diesen auslesen. Darunter fallen z. B. Quellcode aus Dateien mit veränder-

ter Endung (z. B. „.old“ oder „.bak“), Sicherungsdateien oder nicht geschützte Administrationsoberflächen.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Minimalprinzip

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Keine Entwicklung auf dem Produktivsystem

Restrisiko: Bei der Umsetzung der vorgeschlagenen Maßnahmen ist kein Restrisiko bzgl. der Gefährdung bekannt.

5.3.14 Umgehen der Login-Abfrage

Bedrohung: Unautorisierter Zugriff auf Inhalte des Web-Angebotes

Schwachstelle: Manipulierbarkeit der vorgesehenen Seitenabfolge

Wird nicht bei jeder Anfrage die Authentisierung des Benutzers geprüft, kann ein Angreifer bekannte Seiten direkt aufrufen, ohne den Umweg über das Login zu gehen.

Gegenmaßnahmen in der Grundarchitektur (siehe Abschnitt 3):

- Vorgaben an Authentisierung und Autorisierung

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Trotz getroffener Sicherheitsmaßnahmen können Software-Fehler nie ausgeschlossen werden. Möglicherweise werden im eingesetzten Web-Angebot Schwachstellen entdeckt, welche die Umgehung der Login-Abfrage ermöglichen.

5.4 Gefährdungen durch Verhindern und Zerstören

Manche Angriffe können die Verfügbarkeit der Systeme gefährden. Grundsätzlich können alle Gefährdungen, mit denen in Systeme eingedrungen werden kann, auch zur Einschränkung der Verfügbarkeit missbraucht werden.

Angriffe, die speziell zur Verhinderung, Zerstörung oder Umgehung von Komponenten der Grundarchitektur verwendet werden, sind im folgenden Abschnitt beschrieben.

5.4.1 Verhinderung von Benutzer-Zugriffen durch Auslösen der Konto-Sperre

Bedrohung: Sperren von Benutzerkonten

Schwachstelle: Funktion zum automatischen Sperren von Benutzerkonten

Konto-Sperren können sicherstellen, dass ein Angreifer keine effizienten Brute-Force-Angriffe (Erraten von Benutzernamen oder Passwörtern) durchführen kann. Eine solche Konto-Sperre wird aktiv, wenn das Passwort für einen Benutzer mehrmals falsch eingegeben wurde. Die Anzahl ist dabei frei wählbar.

Dies schützt jedoch nicht nur vor Brute-Force-Angriffen, sondern erlaubt es auch einem Angreifer, Benutzer vom System auszusperren, indem er bewusst mehrmals ein falsches Passwort angibt.

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Verzicht auf Account-Sperren
- Einsatz von Teergruben als Ersatz

Restrisiko: Mit einem automatisierten Programm kann ein Angreifer auch eine Teergrube dazu verwenden, um einen Benutzer permanent vom System auszusperren. Im Gegensatz zu einer echten Account-Sperre kann die Verhinderung nur so lange aufrechterhalten werden, wie das Programm des Angreifers aktiv ist.

Variante 5.4.1 A für hohen Schutzbedarf: Verzicht auf Maßnahmen gegen Brute-Force-Angriffe

Anwendungsbereich: hoher Schutzbedarf bezüglich Verfügbarkeit

Phase im Ablaufplan: Konzeption

Ist die Verfügbarkeit der Accounts wichtiger als der Schutz der Zugangsdaten, kann auf Maßnahmen gegen Brute-Force-Angriffe verzichtet werden. Um den Schutz der Konten bestmöglich zu gewährleisten, sollte die Passwortrichtlinie besonders starke Passwörter vorschreiben.

Restrisiko: Ein Brute-Force-Angriff kann wesentlich leichter zum Erfolg führen, wenn die Schutzmaßnahmen durch die Passwortrichtlinie nicht sicher genug gewählt werden. Integrität und Vertraulichkeit der Daten sind stärker bedroht.

Umsetzungsaufwand: Weniger Aufwand als in der Basis-Variante.

5.4.2 Verhinderung des Web-Angebotes durch manipulierte Anfragen

Bedrohung: Verhinderung des Web-Angebotes, Verhinderung von Hintergrundsystemen

Schwachstelle: Unzureichende Filterung von Anfragen

Werden Anfragen unzureichend validiert, kann das Web-Angebot verhindert werden. Die zugrunde liegenden Schwachstellen können vielfältig sein. Eine vollständige Auflistung ist daher nicht möglich. Z. B. kann eine Anfrage eine Endlosschleife in der Web-Anwendung auslösen, die die Systemressourcen vollständig verbraucht. Ein solcher Angriff kann in Kombination mit einer Injection-Schwachstelle auch auf einem Hintergrundsystem durchgeführt werden.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen

Gegenmaßnahmen in der Grundkonfiguration (siehe Abschnitt 4.2):

- Kontextsensitive Eingabefilterung

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Überprüfung des Sicherheitsniveaus

Restrisiko: Die Logik-Fehler, die einer solchen Gefährdung zugrunde liegen, können sehr komplex sein und sind daher schwer zu entdecken. In diesem Bereich ist die Gefahr gegeben, dass einzelne Logik-Fehler übersehen und nicht bei der Filterung berücksichtigt werden. Erlangt ein Angreifer Kenntnis über eine solche Schwachstelle, kann der Filter umgangen und das Web-Angebot dadurch möglicherweise verhindert werden.

5.4.3 Verhinderung durch Log-Überlauf

Bedrohung: Verhinderung des Web-Angebotes

Schwachstelle: Log-Einträge werden ungefiltert geschrieben

Ist der Logging-Server nicht in der Lage, genügend Speicherplatz für das Logging zur Verfügung stellen zu können, kann es zu einem Überlauf der Log-Dateien kommen. Abhängig von der Implementierung kann dies neben der Verhinderung des Logging-Servers selbst auch zu einer Verhinderung der Web-Anwendung führen, wenn diese ohne Logging-Server nicht richtig funktioniert.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Definierte Anforderungen an den Logging-Server

Restrisiko: Ein Hardware- oder Software-Fehler im Logging-Server kann selbst bei einem ausreichend groß dimensionierten Speicherplatz eine Verhinderung ermöglichen. Des Weiteren besteht die Möglichkeit, dass durch eine hohe Anzahl an Log-Meldungen der Speicher aufgebraucht wird, wenn der Server nicht in regelmäßigen Abständen von einem Administrator gewartet wird.

Variante 5.4.3 A für normalen Schutzbedarf: Log Rotation

Anwendungsbereich: normaler Schutzbedarf bezüglich Verfügbarkeit

Phase im Ablaufplan: Konzeption

Ist ein Verlust älterer Log-Einträge weniger kritisch als ein Log-Überlauf, kann ein Log-Rotation-Mechanismus eingesetzt werden. Dieser überschreibt die ältesten Log-Einträge, sollte der Speicherplatz nicht mehr ausreichen.

Restrisiko: Alte Log-Einträge gehen verloren, wodurch ein Angreifer bei einer sehr hohen Anzahl an Log-Einträgen bei gleichzeitig geringem Speicher die Möglichkeit hat, Spuren eines Angriffs zu verwischen.

Umsetzungsaufwand: Geringer Realisierungsaufwand

5.4.4 Verhinderung des Web-Angebotes durch Speicher-Management-Schwachstellen

Bedrohung: Verhinderung des Web-Angebotes, Verhinderung von Hintergrundsystemen

Schwachstelle: Speicher-Management-Schwachstelle

Speicher-Management-Schwachstellen können nicht in allen Fällen dazu genutzt werden, um Befehle auf dem System auszuführen. Eine Möglichkeit den Betrieb von Systemen grundlegend zu stören, bieten sie jedoch praktisch immer. Findet ein Angreifer eine Speicher-Management-Schwachstelle in einem der Systeme, kann er dessen Betrieb damit verhindern.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen
- Sicherheitsfunktionen des Betriebssystems

Gegenmaßnahmen durch sicheren Betrieb von Web-Angeboten (siehe Abschnitt 4.2.6):

- Patch-Management gemäß [ISi-E] für Standardsoftware

Restrisiko: Sogenannte Zero-Day-Schwachstellen in Standardsoftware stellen weiterhin eine Bedrohung dar (siehe Abschnitt 5.1.1).

5.4.5 Verhinderung des Web-Angebotes durch verteilte Dienstblockaden auf Anwendungsebene

Bedrohung: Verhinderung des Web-Angebotes, Verhinderung von Hintergrundsystemen

Schwachstelle: Möglichkeit zur Überlastung des Web-Angebotes

Durch Aufruf von ressourcenintensiven Operationen eines Web-Angebotes, welche zu einer hohen Last für das Web-Angebot selbst oder Hintergrundsysteme führen, ist ein Angreifer in der Lage, Dienstblockaden durchzuführen. Dies geschieht durch gleichzeitigen Aufruf der ressourcenintensiven Operation von einer großen und verteilten Anzahl an Systemen (DDoS). Diese sind beispielsweise Teile eines Bot-Netzes, das unter der Kontrolle des Angreifers steht. Durch die stark ansteigende Last ist das Web-Angebot nicht mehr in der Lage, Anfragen in der üblichen Zeit zu bearbeiten bzw. neue Anfragen entgegenzunehmen.

Gegenmaßnahmen durch Grundanforderungen an ein sicheres Produkt (siehe Abschnitt 4.1):

- Richtlinien für sichere Softwareentwicklung bei Web-Anwendungen
- Richtlinien für sichere Beschaffung von Web-Anwendungen
- Anwendung des Minimalprinzips

Restrisiko: Selbst wenn alle Funktionen eines Web-Angebotes so realisiert wurden, dass sie ausschließlich die minimal benötigten Ressourcen in Anspruch nehmen, kann ein Angreifer durch Erhöhung der am Angriff beteiligten Rechner eine Dienstblockade auf Anwendungsebene hervorrufen.

Variante 5.4.5 A für hohen Schutzbedarf: Redundanz

Anwendungsbereich: hoher Schutzbedarf bezüglich Verfügbarkeit

Phase im Ablaufplan: Konzeption

> Alle wichtigen Komponenten werden redundant ausgelegt. Diese Maßnahme muss in Rahmen des Notfallvorsorgekonzept mit betrachtet werden. So kann es beispielsweise zusätzlich notwendig sind, wichtige Komponenten vorkonfiguriert bereit zu halten, um einen schnellen Austausch durchführen zu können.

Restrisiko: Redundanz beseitigt die Bedrohung des Ausfalls nicht, sie verringert lediglich die Eintrittswahrscheinlichkeit.

Umsetzungsaufwand: zusätzlicher Aufwand im Betrieb

6 Fazit

Web-Angebote bieten einem Angreifer eine Vielzahl unterschiedlicher Angriffsmöglichkeiten, die für eine erfolgreiche Kompromittierung genutzt werden können. Das größte Gefährdungspotential geht von der Web-Anwendung selbst aus, da diese zum Teil sehr komplexe Funktionen und Schnittstellen bietet. Gefahren die von unterstützenden Systemen ausgehen, wie Webservern oder Hintergrundsystemen, dürfen dabei aber nicht vernachlässigt werden.

Die vorliegende Studie stellt Maßnahmen vor, die es Anbietern von Web-Angeboten ermöglichen, diese sicher zu beschaffen, zu konfigurieren und zu betreiben. Dabei wird darauf geachtet, effizient vorzugehen. Dies wird durch eine Auswahl an Lösungen ermöglicht, die direkt in die Web-Angebote integriert werden können und nicht durch kostspielige Zusatz-Komponenten realisiert werden müssen.

Neben den rein technischen Punkten werden auch relevante Aspekte aus den Bereichen Recht und Sicherheitsbewusstsein behandelt, da sie eng mit der Thematik verzahnt sind und somit nicht isoliert betrachtet werden sollten.

Um einen sicheren Betrieb von Web-Angeboten zu ermöglichen, wird empfohlen, die in dieser Studie angeführten Bedrohungen zu beachten und die ebenfalls vorgestellten Gegenmaßnahmen umzusetzen. Werden alle im Rahmen dieser Studie genannten Empfehlungen korrekt umgesetzt, so kann dadurch ein hohes Sicherheitsniveau in Bezug auf Web-Angebote erreicht werden.

Die Basis für das sichere Bereitstellen von Web-Angeboten ist eine sichere Anbindung an das Internet. Zur Absicherung der unteren drei Schichten des TCP/IP-Referenzmodells baut die Studie auf [ISi-LANA] auf. Bei Verwendung anderer Internet-Dienste können die Maßnahmen zum sicheren Bereitstellen von Web-Angeboten um die Anforderungen der Studien zum jeweiligen Thema ergänzt werden.

7 Anhang

7.1 Varianten der Grundarchitektur

In Abschnitt 3 wurde eine sichere Grundarchitektur für eine große Organisation mit normalem Schutzbedarf vorgestellt, die das sichere Anbieten von Web-Diensten im Internet beschreibt.

Darüber hinaus wurden in Abschnitt 5 verschiedene Architektur- und Konfigurationsvarianten beschrieben. Hier wurde gezeigt wie sich einerseits kleine, unkritische IT-Infrastrukturen mit moderatem Aufwand realisieren lassen, andererseits aber auch wie großen Unternehmen mit hohem Schutzbedarf durch ergänzende Maßnahmen und Erweiterungen entsprochen werden können.

Eine Abweichung von der Grundarchitektur hat jedoch Auswirkungen auf die Gefährdungen, denen das Unternehmen ausgesetzt ist. Hier muss der Nutzer abwägen, ob das unter Umständen erhöhte Restrisiko zu tragen ist oder nicht.

Im folgenden Anhang werden einige Beispiele vorgestellt, wie die Architektur an eine veränderte Unternehmensgröße oder einen veränderten Schutzbedarf angepasst werden kann und welche Konsequenzen dies auf die Gefährdungslage hat.

Im Fokus stehen dabei die Änderungen, die sich auf das Sicherheits-Gateway beziehen. Änderungen, die das interne Netz, das Management-Systems und die Internetanbindung betreffen, sind in [ISi-LANA] beschrieben.

Für jedes vorgestellte Szenario werden Annahmen getroffen, die beispielhaft eine kleine, mittelgroße bzw. große Organisation kennzeichnen. Aufbauend auf diesen Annahmen werden Änderungsmöglichkeiten aufgegriffen, die bereits als Varianten in Abschnitt 5 vorgestellt worden sind. Die Änderung der Gefährdungslage des Unternehmens durch die Abweichung von der Grundarchitektur wird jeweils in einer Tabelle dargestellt.

In der Tabelle bedeutet „↑↑“ deutlich erhöhte Gefährdung, „↑“ erhöhte Gefährdung, „↔“ Gefährdung, wie bei Verwendung der Grundarchitektur, „↑↓“ teils erhöhte, teils erniedrigte Gefährdung, „↓“ verringerte Gefährdung und „↓↓“ deutlich verringerte Gefährdung.

7.1.1 Kleines Unternehmen

Annahmen

Für ein kleines Unternehmen werden folgende Annahmen getroffen:

- Es handelt sich um keine Behörde.
- Wenige (5-10) Clients und Server im internen Netz.
- Ein interner Webserver wird nicht benötigt.
- Ein Datenbank-Server wird nicht benötigt.

Szenario ohne eigenen Webserver

Für ein kleines Unternehmen ist es im Normalfall nicht empfehlenswert einen eigenen Webserver zu betreiben, da viele Ressourcen (sowohl monetär als auch personell) benötigt werden.

Änderungen der Grundarchitektur für den normalen Schutzbedarf

Bei Bedarf kann ein Webserver von einem externen Anbieter angemietet werden, um dort beispielsweise eine Internet-Präsens zu schaffen. Dabei sind die Sicherheitsrichtlinien des jeweiligen Anbieters zu prüfen.

Das Sicherheits-Gateway kann für ein kleines Unternehmen mit normalem Schutzbedarf gemäß [ISi-LANA] folgendermaßen aussehen:

- Einstufiges Sicherheits-Gateway: Die Paketfilter (PF2, PF3, PF4, PF5), das ALG und der Virenschutz entfallen.
- Sämtliche Webserver und der DNS-Server entfallen.
- Die Hintergrund-Systeme und der Logging-Server entfallen.

Konsequenzen für die Gefährdung des Unternehmens

Ohne eigenen Webserver bietet das Sicherheits-Gateway eine geringere Angriffsfläche. Das interne Netz ist jedoch wegen des Verzichts auf eine PAP-Struktur (Paketfilter-ALG-Paketfilter) nur unzureichend geschützt.

Die Konsequenzen für das Web-Angebot sind vom jeweiligen externen Anbieter abhängig und daher nicht allgemein zu beurteilen.

Szenario mit eigenem Webserver

In einigen Szenarien kann es auch für kleine Unternehmen unerlässlich sein, einen eigenen Webserver zu betreiben.

Änderungen der Grundarchitektur für den normalen Schutzbedarf

Von einer Vereinfachung der PAP-Struktur (Paketfilter-ALG-Paketfilter) ist in diesem Szenario dringend abzuraten. Es werden folgende Varianten für den normalen Schutzbedarf verwendet (vgl. Abbildung 7.1):

- Die Paketfilter (PF3, PF4, PF5) entfallen.
- Der interne Webserver entfällt.
- Die Hintergrund-Systeme entfallen.
- Die Server WWW, WWW AS und DNS werden zusammengelegt.
- Für das Protokollieren der Web-Anwendungen wird kein eigener Logging-Server verwendet (Beschränkung auf das Logging in der Management-Zone).

Konsequenzen für die Gefährdung des Unternehmens (mit eigenem Webserver)

Kleines Unternehmen – normaler Schutzbedarf		
Bedrohung	***Gefährdung***	***Bemerkung***
Eindringen	↑↑	Vergrößerte Angriffsfläche wegen Verstoß gegen das Minimalprinzip („Ein Dienst pro Server“) Der Anwendungs-Server ist nur durch einen Paketfilter vor Angriffen aus dem Internet geschützt
Verändern/Täuschen	↔	Kein Erhöhtes Risiko im Vergleich zur Grundarchitektur
Entwenden/Ausspähen	↓	Verringertes Risiko, da keine Hintergrund-Systeme verwendet werden
Verhindern/Zerstören	↑	Vergrößerte Angriffsfläche wegen Verstoß gegen das Minimalprinzip

Tabelle 2: Konsequenzen für die Gefährdung eines kleinen Unternehmens mit normalem Schutzbedarf (mit eigenem Webserver)

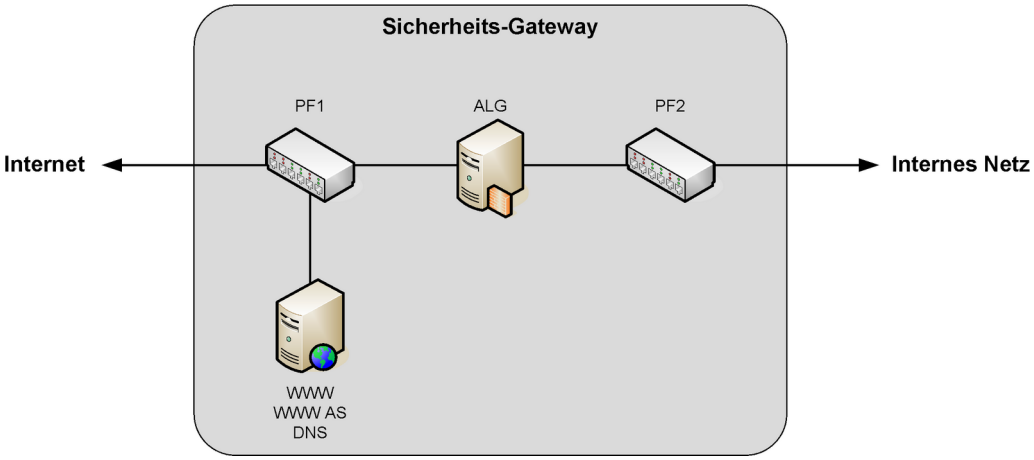


Abbildung 7.1.: Variante für ein kleines Unternehmen mit normalem Schutzbedarf (mit eigenem Webserver)

Änderungen der Grundarchitektur für den hohen Schutzbedarf

Besteht ein hoher Schutzbedarf bezüglich der Integrität und der Verfügbarkeit des Web-Angebotes, so können zusätzliche Maßnahmen getroffen werden. Aufgrund der Annahmen für kleine Unternehmen ergeben sich folgende Änderungen (vgl. Abbildung 7.2):

- Beschränkung auf statische Web-Inhalte (WWW AS entfällt).
- Der interne Webserver entfällt.
- Die Hintergrund-Systeme entfallen.
- Für das Protokollieren der Web-Anwendungen wird kein eigener Logging-Server verwendet (Beschränkung auf das Logging in der Management-Zone).

Konsequenzen für die Gefährdung des Unternehmens

Kleines Unternehmen – hoher Schutzbedarf		
Bedrohung	***Gefährdung***	***Bemerkung***
Eindringen	↓	Verringertes Risiko, da auf dynamische Inhalte verzichtet wird
Verändern/Täuschen	↔	Kein Erhöhtes Risiko im Vergleich zur Grundarchitektur
Entwenden/Ausspähen	↔	Kein erhöhtes Risiko im Vergleich zur Grundarchitektur
Verhindern/Zerstören	↔	Kein erhöhtes Risiko im Vergleich zur Grundarchitektur

Tabelle 3: Konsequenzen für die Gefährdung eines kleinen Unternehmens mit hohem Schutzbedarf

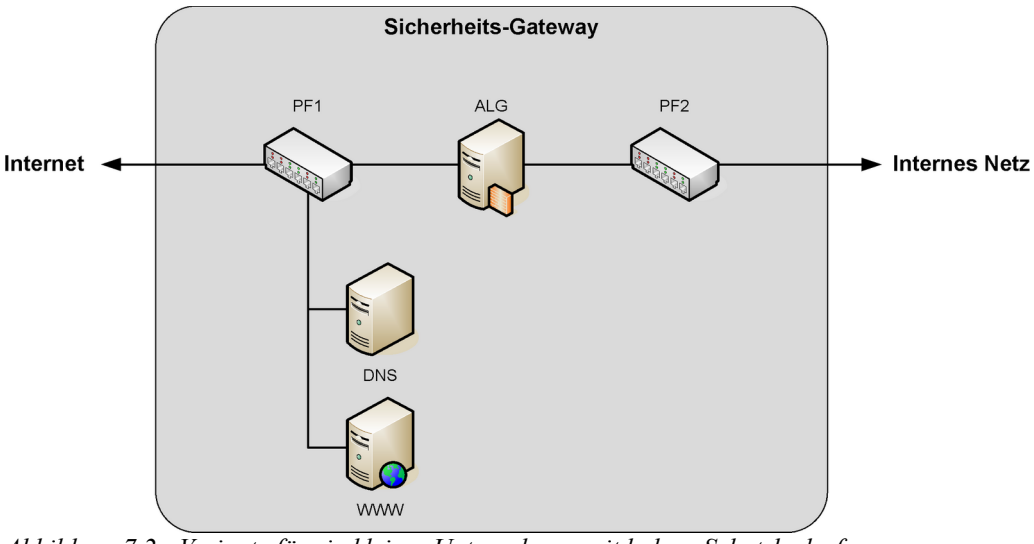


Abbildung 7.2.: Variante für ein kleines Unternehmen mit hohem Schutzbedarf

7.1.2 Mittelgroßes Unternehmen

Annahmen

Für ein mittelgroßes Unternehmen werden folgende Annahmen getroffen:

- Mittlere Anzahl (100-400) von Clients und Servern im internen Netz.
- Einsatz eines internen Webserver.
- Einsatz eines Datenbank-Server.
- Verwendung eines Content-Management-Systems (CMS).

Änderungen für den normalen Schutzbedarf

Für den normalen Schutzbedarf können folgende Varianten verwendet werden (vgl. Abbildung 7.3):

- Die Paketfilter PF3 und PF5 werden zusammengelegt.
- Der externe Webserver und der externen DNS-Server werden zusammengelegt.

Konsequenzen für die Gefährdung des Unternehmens

Mittelgroßes Unternehmen – normaler Schutzbedarf		
Bedrohung	***Gefähr-dung***	***Bemerkung***
Eindringen	↑	Vergrößerte Angriffsfläche wegen Verstoß gegen das Minimalprinzip („Ein Dienst pro Server“) CMS für dynamische Inhalte ist zusätzlicher Angriffspunkt
Verändern/Täuschen	↔	Kein erhöhtes Risiko im Vergleich zur Grundarchitektur
Entwenden/Ausspähen	↑	Erhöhtes Risiko, da der Datenbank-Server nicht separat abgeschirmt ist
Verhindern/Zerstören	↔	Kein erhöhtes Risiko im Vergleich zur Grundarchitektur

Tabelle 4: Konsequenzen für die Gefährdung eines mittelgroßen Unternehmens mit normalem Schutzbedarf

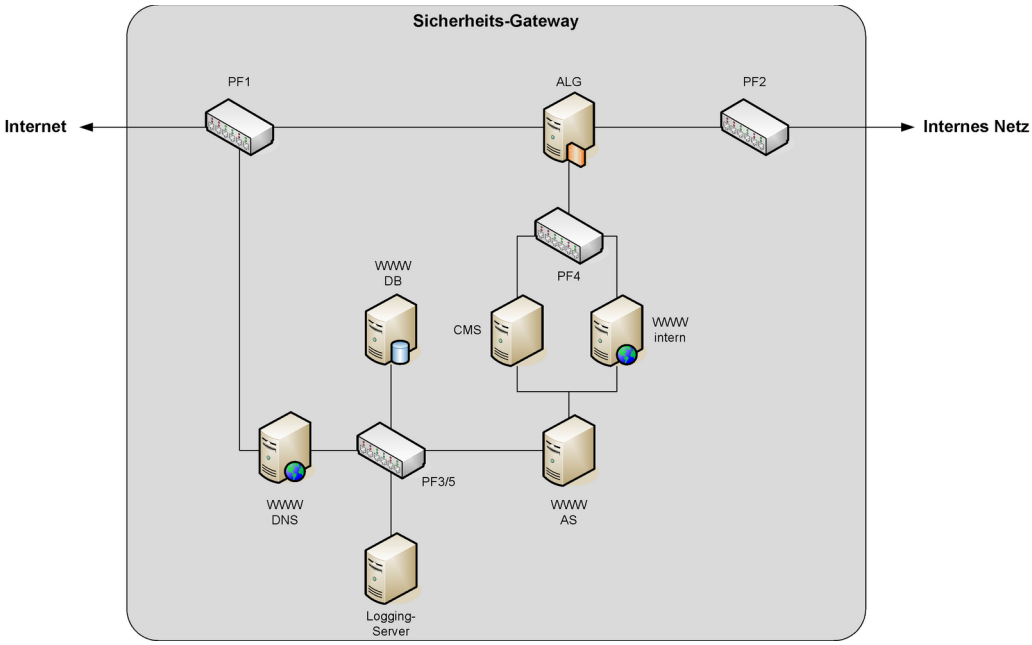


Abbildung 7.3: Variante für ein mittelgroßes Unternehmen mit normalem Schutzbedarf

Änderungen für den hohen Schutzbedarf

Für ein mittelgroßes Unternehmen mit hohem Schutzbedarf ist von einer Vereinfachung der Grundarchitektur abzuraten. Um den Schutz weiter zu erhöhen können beispielsweise folgende Varianten verwendet werden (vgl. Abbildung 7.4):

- Verwendung eines CMS, das ausschließlich statische Inhalte generiert.

Konsequenzen für die Gefährdung des Unternehmens

Mittelgroßes Unternehmen – hoher Schutzbedarf		
Bedrohung	***Gefährdung***	***Bemerkung***
Eindringen	↓	CMS produziert nur statische Seiten und ist somit schwerer angreifbar
Verändern/Täuschen	↔	Kein erhöhtes Risiko im Vergleich zur Grundarchitektur
Entwenden/Ausspähen	↔	Kein erhöhtes Risiko im Vergleich zur Grundarchitektur
Verhindern/Zerstören	↔	Kein erhöhtes Risiko im Vergleich zur Grundarchitektur

Tabelle 5: Konsequenzen für die Gefährdung eines mittelgroßen Unternehmens mit hohem Schutzbedarf

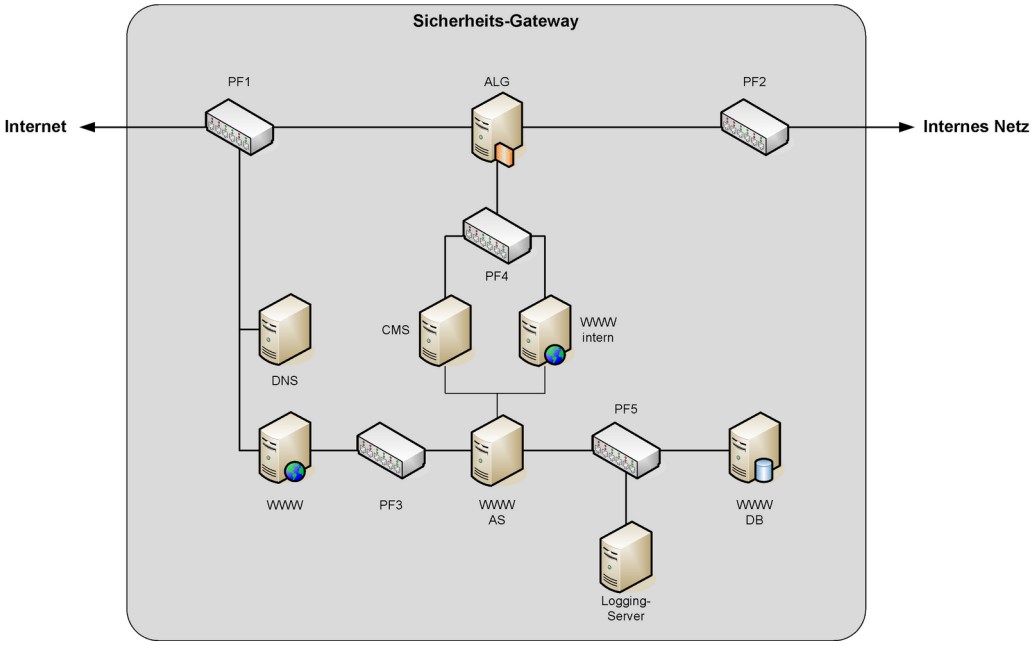


Abbildung 7.4: Variante für ein mittelgroßes Unternehmen mit hohem Schutzbedarf

7.1.3 Großes Unternehmen

Für ein großes Unternehmen werden folgende Annahmen getroffen:

- Große Anzahl (1000+) von Clients und Servern im internen Netz.

Änderungen für den normalen Schutzbedarf

Auch bei einem großen Unternehmen ist von einer Vereinfachung der Grundarchitektur abzuraten. Die Grundarchitektur kann jedoch um weitere Komponenten erweitert werden. Für den normalen Schutzbedarf können beispielsweise folgende Änderungen verwendet werden (vgl. Abbildung 7.5):

- Verwendung eines Virtuellen Patching Systems (VPS)

Konsequenzen für die Gefährdung des Unternehmens

Großes Unternehmen – normaler Schutzbedarf		
Bedrohung	***Gefähr-dung***	***Bemerkung***
Eindringen	↓	Verwendung eines Virtuellen Patching Systems
Verändern/Täuschen	↓	Verwendung eines Virtuellen Patching Systems
Entwenden/Ausspähen	↓	Verwendung eines Virtuellen Patching Systems
Verhindern/Zerstören	↓	Verwendung eines Virtuellen Patching Systems

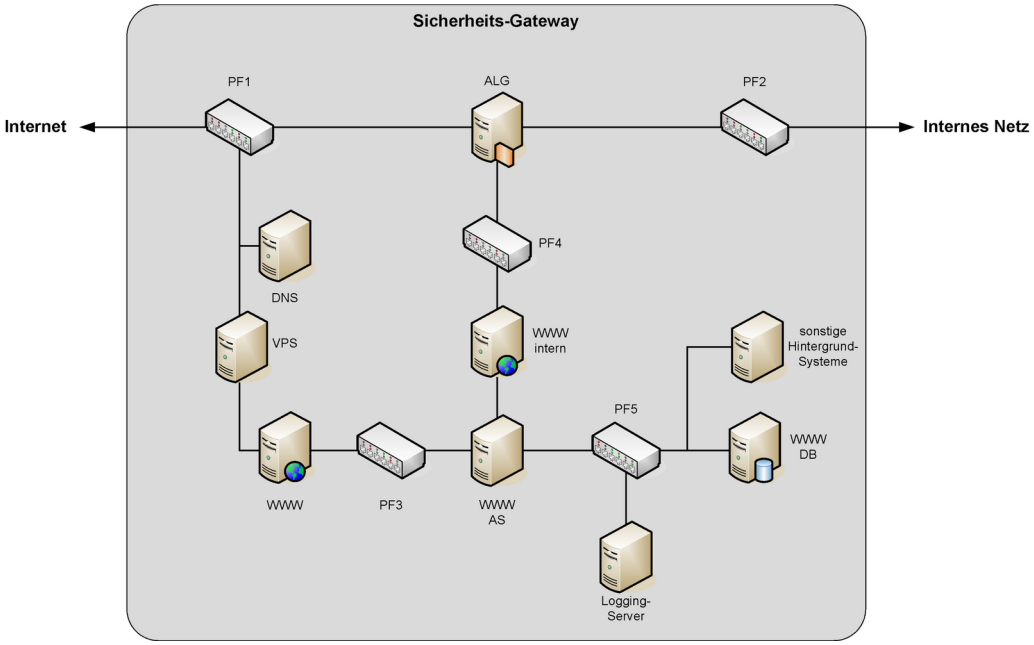


Abbildung 7.5.: Variante für ein großes Unternehmen mit normalem Schutzbedarf

Tabelle 6: Konsequenzen für die Gefährdung eines großen Unternehmens mit normalem Schutzbedarf

Änderungen für den hohen Schutzbedarf

Besteht ein erhöhter Schutzbedarf für die Vertraulichkeit und Integrität von Daten, so können, zusätzlich zu der bereits in der Grundarchitektur vorhandenen verschlüsselten Übertragung über HTTPS, z. B. folgende Varianten verwendet werden (vgl. Abbildung 7.6)

- Verwendung von Prepared Statements und Stored Procedures
- Mehr-Faktoren-Authentisierung
- Software zur Erkennung von System-Anomalien auf den Servern WWW, WWW AS, WWW int.
- Pfad-Normalisierung
- Log-Rotation
- Zeitstempel für Anfragen

Konsequenzen für die Gefährdung des Unternehmens

Großes Unternehmen – hoher Schutzbedarf		
Bedrohung	***Gefähr-dung***	***Bemerkung***
Eindringen	↓	Erkennung von System-Anomalien
Verändern/Täuschen	↓	Zeitstempel für Anfragen verhindern Replay-Angriffe
Entwenden/Ausspähen	↓	Verwendung von Prepared Statements und Pfad-Normalisierung erschweren Angriffe auf Hintergrund-Systeme
Verhindern/Zerstören	↓	Log-Rotation verhindert Log-Überläufe Redundante Internet-Anbindung

Tabelle 7: Konsequenzen für die Gefährdung eines großen Unternehmens mit hohem Schutzbedarf

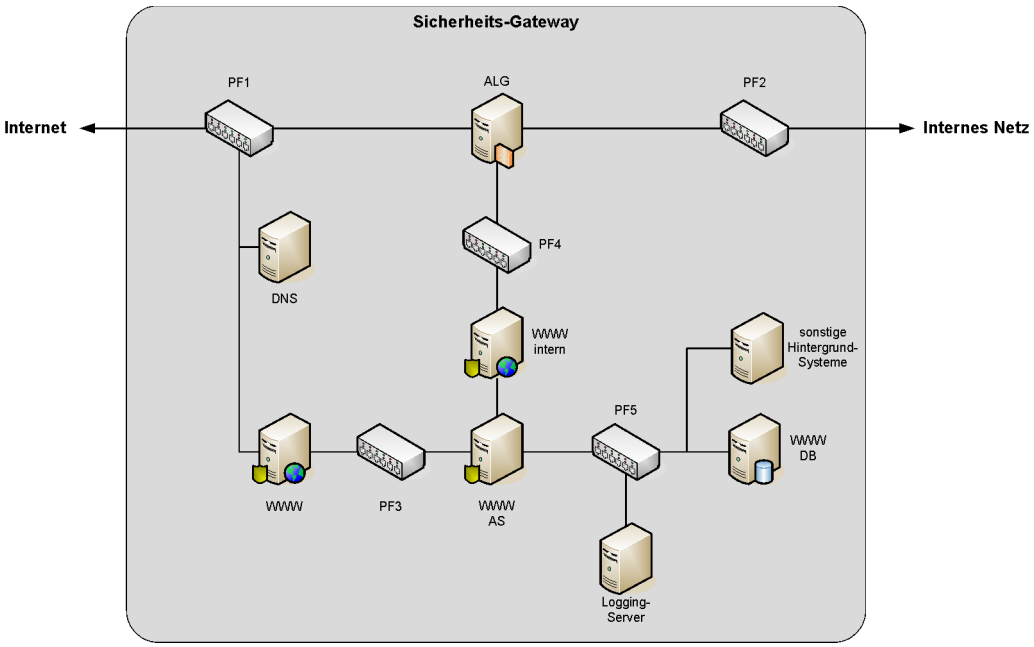


Abbildung 7.6: Variante für ein großes Unternehmen mit hohem Schutzbedarf

7.2 Literaturverzeichnis

ISi-Web-Client: Bundesamt für Sicherheit in der Informationstechnik (BSI), BSI-Schriftenreihe zur Internet-Sicherheit: Sichere Nutzung von Web-Angeboten, 2008

RFC 7540: Internet Engineering Task Force (IETF), Hypertext Transfer Protocol -- HTTP/2, 2015

BSI-TR-02102: Bundesamt für Sicherheit in der Informationstechnik (BSI), Kryptographische Verfahren: Empfehlungen und Schlüssellängen, 2017, https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html

ISi-Mail-Server: Bundesamt für Sicherheit in der Informationstechnik (BSI) , Sicherer Betrieb von E-Mail-Servern, 2009, https://www.bsi.bund.de/DE/Themen/StandardsKriterien/ISi-Reihe/ISi-Mail-Server/mail_server_node.html

RFC 1945: Internet Engineering Task Force (IETF), Hypertext Transfer Protocol -- HTTP/1.0, 1996

RFC 2617: Internet Engineering Task Force (IETF), HTTP Authentication: Basic and Digest Access Authentication, 1999

TMG-Papier: Bundesamt für Sicherheit in der Informationstechnik, Absicherung von Telemediendiensten nach Stand der Technik, 2016

ISi-LANA: Bundesamt für Sicherheit in der Informationstechnik (BSI), BSI-Schriftenreihe zur Internet-Sicherheit: Sichere Anbindung lokaler Netze an das Internet, 2007

CMS-Studie: Bundesamt für Sicherheit in der Informationstechnik, Studie „Sicherheitsuntersuchung von Content-Management-Systemen“, 2015

ISi-E: Bundesamt für Sicherheit in der Informationstechnik (BSI), BSI-Schriftenreihe zur Internet-Sicherheit: Einführung, Grundlagen, Vorgehensweise, in Bearbeitung