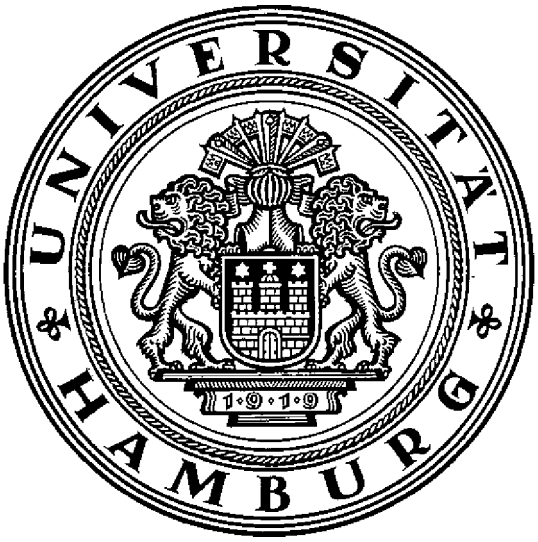
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Anwendungen in Geistes- und Naturwissenschaften
Vogt-Kölln-Straße 30
22537 Hamburg
Germany



Studienarbeit

”Konzeptionelle Migration
einer FoxPro-Datenbank an das WorldWideWeb”

Eingereicht von : Michael Stradt
E-Mail : 1stradt@informatik.uni-hamburg.de

Betreuer : Prof. Dr. Klaus Brunnstein
Datum : 09.01.2001

Inhalt

Das Ziel dieser Arbeit ist es, anhand einer realen, bereits existierenden Datenbank und Ihrer Anbindung an das WorldWideWeb, die Grundlagen und die unterschiedlichen vorhandenen Möglichkeiten bei einem solchen Projekt zu erläutern. Hierbei wird auch auf die verschiedenen Faktoren eingegangen, die die Auswahl der geeigneten Implementation beeinflussen. Aufgrund verschiedener Zielsetzungen soll das bestmögliche Vorgehen ermittelt und die Anbindung konzipiert werden.

Im ersten Teil wird zunächst eine Übersicht über Viren allgemein und den ursprünglichen Computer-Virus-Katalog, sowie die Notwendigkeit eines solchen Katalogs gegeben. Hierauf aufbauend wird die digitale Weiterentwicklung des Computer-Viren-Katalogs in Form des Programms „CMBase“ vorgestellt.

Auf der Grundlage des ersten Kapitels werden im zweiten Kapitel zunächst die eine Implementations-Entscheidung beeinflussenden Faktoren lokalisiert und deren Abhängigkeiten dargestellt. Auf die in dieser Arbeit angestrebte Implementation angewendet, wird im Folgenden ein Anforderungskatalog erstellt, dem die Anwendung bestmöglich gerecht werden soll. Dieser Katalogs wird später in dieser Arbeit die Entscheidunggrundlage für eine der dargestellten Möglichkeiten sein.

In Kapitel drei folgt dann eine Darstellung der verschiedenen Konfigurations-möglichkeiten bei der Planung, sowie vorhandene Techniken für Internet-Anwendungen mitsamt Ihrer Vor- und Nachteile.

Nach der Entscheidung für die geeigneten Mittel und Techniken geht es im vierten Teil um die eigentliche Implementation. Die gewählte Lösung wird umrissen, sowie detailliertere Begründungen für diese Wahl dargestellt.

Das abschließende Kapitel fünf gibt zunächst eine Zusammenfassung, sowie einen Ausblick auf Erweiterungen, Trends und Erfordernisse.

0 Abkürzungs- und Abbildungsverzeichnis

0.1 Abkürzungsverzeichnis

API	Application Programming Interface (Programmierschnittstelle)
BIOS	Basic Input/Output-System
CGI	Common Gateway Interface
DBMS	Database-Management-System
DLL	Dynamic Link Library
GUI	Graphical User Interface (Grafische Benutzeroberfläche)
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
JVM	Java Virtual Machine
ODBC	Open Database Connectivity
PC	Personal Computer
PERL	Practical Extraction and Reporting Language
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
URL	Uniform Resource Locator
VPN	Virtual Private Network
VTC	Virus Test Center (an der Universität Hamburg)
WWW	WorldWideWeb

0.2 Tabellen- und Abbildungsverzeichnis

1 Einleitung

Dieses Kapitel soll dem Leser einen Überblick über die vorhandene Datenbank, die der Öffentlichkeit zugänglich gemacht werden soll, geben. Für das bessere Verständnis des Aufbaus der Datenbank werden zunächst Computerviren und andere schädliche Programme sowie ihre Funktionsweisen und Eigenschaften vorgestellt, und in diesem Zusammenhang wichtige Begriffe eingeführt. Hierauf aufbauend wird dann das Format des Computerviren-Katalogs erläutert, der dann wiederum in Datenbankform vorliegt.

1.1 Computer-Viren und -Malware: Grundlagen

Seit den ersten theoretischen Arbeiten über Computerviren und den ersten real existierenden experimentellen Viren hat sich die Problematik maliziöser Software enorm verstärkt. Kaum ein größeres Unternehmen, daß nicht mindestens einen Virus in seinen Systemen gehabt hätte. Deshalb ist die Kenntnis über die einzelnen Viren, Ihre Wirkungsweise, die Beseitigung und ggf. die Behebung des dadurch entstandenen Schadens von großer Wichtigkeit.

Waren es zunächst vor allem die Viren die für Aufsehen sorgten, ist für Sicherheits-Experten inzwischen auch der Bereich der sog. Dropper, Trojanischen Pferde und der Würmer wichtig geworden. Alle Formen bösartiger Software werden deshalb inzwischen unter dem Oberbegriff „Malware“[1] zusammengefaßt. Die wichtigsten Grundlagen zu diesem Thema werden nachfolgend erklärt.

1.1.1 Computer-Viren

Ein Computervirus ist selbstreplizierender Programmcode, der sich an ein Wirtsprogramm anhängt, da er nicht eigenständig existieren kann. Aus dieser Beschreibung kann man leicht erkennen, daß die Bezeichnung ”Computervirus” nicht aus der Luft gegriffen ist, sondern in Anlehnung an die biologischen Viren gewählt wurde. Ebenso wie bei den biologischen Viren gibt es auch auf dem Computer Viren, die sich lediglich verbreiten, um zu überleben, und andere, die zusätzlich eine schadhafte Funktion (”Payload”) besitzen, die mitunter auch dazu führen kann, daß das Wirts-System nicht mehr korrekt funktioniert. Computer-Viren breiten sich per Definition nicht selbständig über Netze aus[2], können aber durch Kopier-Aktionen infizierter Dateien innerhalb von Unternehmen oder Netzen ihre Verbreitung finden.

Aufgrund der Tatsache, daß der Virus-Code an bestehende Programme angehängt wird[3], sind Viren teilweise auch leicht wieder zu entfernen, indem der zusätzliche Code einfach gelöscht bzw. durch Null-Operationen überschrieben wird und die zuvor modifizierten Sprungadressen korrigiert werden.

[1] Malware : Kurzform von „malicious software“ (schadhafte Software)
[2] s. auch S. 11, „Würmer“
[3] s. Seite 8f, „Klassifizierung nach Speicherort (Wirt)“

Man kann bei Computer-Viren fünf Hauptbestandteile unterscheiden, die, bis auf einen, aber nicht zwingend vorhanden sein müssen.

Replikationsteil : Dieser Teil ist zwingend notwendig für einen Computer-Virus, da die Replikation laut Definition Bestandteil eines Virus ist. Replikation ist hierbei die Fähigkeit, andere Programme so zu modifizieren, daß diese eine (eventuell modifizierte)[4] Kopie von ihm enthalten.

Erkennungsteil : Viele Viren überprüfen, ob die Datei, die infiziert werden soll, evtl. schon mit dem Virus infiziert ist. Ist dies der Fall, so muß die Datei nicht erneut infiziert werden. Dies hat u.a. den Sinn, daß der Virus nicht durch zu lange Verarbeitung beim Systemstart oder auffällig stark steigende Dateigrößen frühzeitig erkannt wird.

Schadensteil („Payload“) : Viren führen durch die Replikation alleine schon eine schadhafte[5] Funktion aus, indem sie Rechenzeit für die Verbreitung benötigen, Arbeits- und Festplattenspeicher belegen und Ressourcen für ihre Beseitigung gebraucht werden. Zusätzlich zu diesen, durch die Definition eines Virus vorhandenen Schäden, ist in vielen Viren explizit eine Schadensfunktion programmiert. Dies kann mit einfacher Tonerzeugung und Bildschirm-ausgabe anfangen und geht bis dahin, daß das System unbrauchbar wird und Daten verändert oder sogar vernichtet werden. Grob unterschieden wird hier zwischen dauerhaften (permanenten) und vorübergehenden (transienten) Schäden, wobei ein Virus auch sowohl transienten als auch permanenten Schaden anrichten kann. Permanente Schäden sind auch nach einem Neustart des Rechners noch vorhanden, wohingegen transiente zunächst verschwunden sind. Dies bedeutet aber nicht, daß der Schaden nicht erneut auftreten kann.

Bedingungsteil : Sowohl der Schadensteil als auch der Replikationsteil können an bestimmte Bedingungen gebunden sein. Bei beiden Teilen liegt der Sinn darin, eine frühzeitige Entdeckung zu verhindern, somit für eine weitere Verbreitung des Virus zu sorgen und damit sein Fortbestehen zu sichern. Der Schadensteil ist beispielsweise an bestimmte Kalenderdaten gebunden und wird nur an diesen ausgeführt. Somit hat er oft ein Jahr oder mehr Zeit sich auf möglichst vielen Systemen einzunisten. Im Replikationsteil wird durch Bedingungen ein auffälliges Verhalten gering gehalten. Auch hierdurch soll eine frühzeitige Erkennung verhindert und somit das Fortbestehen des Virus gesichert werden.

Tarnungsteil : Dieser Teil kam erst mit der verbesserten Systemkenntnis der Viren-Autoren und der Notwendigkeit, die Erkennung durch Virenscanner oder den Anwender zu verhindern oder zu erschweren.[6]

Aufgrund der Vielzahl verschiedener Computerviren kann man diese nach mehreren Merkmalen einordnen und sie dadurch zunächst grob klassifizieren.

1. Klassifizerung nach Speicherort (Wirt)
2. Klassifizerung nach Infektionsmechanismus
3. Klassifizerung nach Schutzmeachnismen

[4] s. S. 9, „Polymorphismus“
[5] schadhaft im Sinne von „Nicht vom Benutzer gewollt“
[6] s. S. 9 „Klassifizierung nach Tarnmechanismus“

Klassifizierung nach Speicherort (Wirt)

Dateiviren stammen ursprünglich aus der DOS-Zeit und nisten sich in ausführbaren Programmdateien ein. Meistens sind dieses Dateien mit den Endungen .EXE und .COM. Aber auch andere Dateien wie z.B. .SYS oder .DLL können befallen werden, was jeweils vom infizierenden Virus abhängt.

Meistens läuft die Infektion so ab, daß der Virus seinen eigenen Code an ein Wirtsprogramm anhängt und das Wirtsprogramm dahingehend verändert, daß bei der nächsten Ausführung zunächst der Viruscode angesprungen wird. Erst nach dessen Ausführung wird zur normalen Ausführung des Wirtsprogramms zurückgesprungen, damit die Infektion der Datei nicht durch Nichtausführung der selbigen frühzeitig bemerkt wird (Abb. 1.1-1).

Abbildung 1.1-1 "Ausführungsabfolge eines virus-infizierten Programms"

Eine andere Art von Viren sind die sogenannten Systemviren, die Eigenarten eines Betriebssystems oder eines Computer-Typs ausnutzen und sich in Teilen von diesem dann einnisten. Beispielsweise hängen sich viele Systemviren auf PCs an den Bootsektor von Disketten und Festplatten und werden somit vor jedem Start des Betriebssystems ausgeführt. Bei der Festplatte wird noch zwischen Partitionssektor und Bootsektor unterschieden. Im Partitionssektor befinden sich u.a. die Informationen wie groß jede einzelne Partition der Festplatte ist. Jede Partition kann außerdem einen Bootsektor enthalten. Beim Booten wird das Ladeprogramm im Partitionssektor auf jeden Fall ausgeführt und ist damit betriebssystem- und dateisystemunabhängig. Am Ende des Urladers des Partitionssektors ruft dieser das Ladeprogramm der aktiven Partition auf, welches dann die Kontrolle übernimmt. Meistens infizieren Systemviren den Bootsektor von Disketten und den Partitionssektor oder Bootsektor der Festplatte. Die Übertragung der Viren findet bei dieser Form durch Austausch von Disketten statt. Einer der bekanntesten Bootsektor-Viren ist der ”Form.A”-Virus.

Die sog. Path- oder Companion-Viren sind eine Ausnahme von der ursprünglichen Beschreibung, da sie sich nicht direkt an eine andere Datei anhängen. Vielmehr nutzen sie eine Eigenschaft des Betriebssystems DOS und seiner Nachfolger aus. Bei DOS wird im Suchpfad zunächst nach Dateien mit der Dateiendung .COM gesucht und dann erst nach denen mit der Endung.EXE. Ausführbare Dateien des Typs .EXE werden nun in der Art „infiziert“, daß in dem selben Verzeichnis eine Datei mit dem selben Namen, aber der Endung .COM angelegt wird. Wenn der Anwender nun in der Eingabeaufforderung den Namen ohne die Endung zum Starten des Programms angibt, wird zunächst die virale .COM-Datei aufgerufen, die ihren maliziösen Code ausführt und dann wiederum die .EXE-Datei startet.

Die oben beschriebene Einstufung des Speicherorts ist durch Entwicklungen der Viren-Programmierer nicht immer eindeutig zu ziehen. Um z.B. bei einer Entdeckung in Dateien trotzdem ein Fortbestehen des Virus auf dem befallenen System zu gewährleisten, infizieren einige Viren zusätzlich noch Teile des Betriebssystems wie z.B. den Bootsektor. Beim nächsten Starten des Computers verbreitet sich der Virus wiederum in zwischenzeitlich evtl. bereinigte Dateien aus. Diese Art von Viren werden als multipartit bezeichnet, da sie mehrere Infektionswege und Wirte benutzen.

Klassifizierung nach Infektionsmechanismus

Bei der Infektion gibt es mehrere Kriterien anhand derer man den Virus klassifizieren kann. Als erstes Merkmal gibt es hier die Infektionsart. Es gibt Viren, die direkt bei Aufruf nach potentiellen Wirten suchen, diese ggf. infizieren und dann wieder aus dem Speicher verschwinden, um nicht aufzufallen. Andere Viren hingegen nisten sich im Speicher ein und infizieren von dort die vom Anwender geöffneten Dateien. Diese Art wird als speicherresident bezeichnet, wohingegen erstere als „direct action“-Viren bezeichnet werden.

Zusätzlich zur Infektionsart ist die Infektionsgeschwindigkeit ein wichtiges Kriterium. Wie bereits erwähnt[7] ist die Infektion teilweise an Bedingungen gebunden, wodurch eine Entdeckung durch zu schnelle Verbreitung verhindert werden soll. Ist eine solche Bedingung vorhanden, handelt es sich um einen sog. „Slow Infector“, andernfalls um einen „Fast Infector“.

Klassifizierung nach Schutzmechanismen

Um eine Erkennung durch den Anwender oder einen Antivirenscanner zu verhindern, haben Virenautoren mit der Zeit unterschiedliche Mechanismen entwickelt. Beispielsweise hängt sich der Virus so in das System ein, daß dem Anwender falsche Informationen geliefert werden, die ihn nicht auf das Vorhandensein eines Virus schließen lassen, wie z.B. eine andere Dateigröße oder eine korrigierte Anzeige des Arbeitsspeichers. Wenn solch ein Tarnmechanismus vorhanden ist bezeichnet man den Virus als Tarnkappen- oder Stealth-Virus.

Die ersten Antivirenscanner arbeiteten mit einem einfachen Mustervergleich des Binärcodes eines Virus, um diesen aufzuspüren und zu identifizieren. Um eine Erkennung durch dieses System zu verhindern wurde die Technik des Polymorphismus von den Virenautoren entwickelt. Hierbei verändert sich das Aussehen des Virus, so daß es für Antivirenscanner schwierig ist ihn mit dem normalen Mustervergleich zu erkennen. Vielmehr waren hierfür neue Erkennungstechniken nötig, die z.B. alle möglichen Musterkombinationen eines Virus erzeugen können. Diese Technik kann dermaßen komplex sein, daß es auch heute noch einige Antivirenscanner gibt, die nicht alle möglichen Kombinationen eines polymorphen Virus erkennen.[8] Auch für den Polymorphismus gibt es evtl. einen Auslöser, so daß es Viren gibt, die sehr schnell die nächste polymorphische Generation erzeugen, wohingegen andere nur sehr langsam ihr Aussehen verändern. Wurden diese Viren am Anfang von den AV-Scannern noch erkannt, gelang dies kurze Zeit später nicht mehr, da den Analysten des Virus die Mutations-Engine nicht aufgefallen war. Hierdurch wurde anfangs meist nur die erste analysierte Generation erkannt.

[7] s. Seite 7, „Bedingungsteil“
[8] [VTC 2000], Tabellen FDOS.FA, W98.FA und WNT.FA

Tunneling als Schutzmechanismus arbeitet gezielt gegen die Erkennung durch sog. Monitor-[9] oder Wächter-Programme, die die Veränderung an Dateien oder Systembereichen entdecken sollen. Diese greifen vor allem auf die Beobachtung der sog. Software-Interrupts zurück, um hier ggf. kontrollierend einzuschreiten. Software-Interrupts stellen standardisierte Aufrufe von Funktionen im Basic Input/Output-System (BIOS) dar. Diese verweisen dann zumeist auf die entsprechende Speicheradresse im BIOS, an der die eigentliche Routine zu finden ist und rufen diese auf. Tunnelnde Viren versuchen nun gezielt diese Adressen zu lokalisieren, rufen die entsprechenden Routinen direkt auf und umgehen damit das jeweilige Schutzprogramm.

Die drei genannten Schutzmechanismen schließen sich nicht gegenseitig aus und können durchaus parallel in einem Virus vorkommen.

1.1.2 Trojanische Pferde

Trojanische Pferde, kurz Trojaner, beherbergen ebenso wie Viren eine maliziöse Funktion. Diese wird meist in ein ausschließlich nützlich scheinendes Programm eingebettet und im Hintergrund und ohne Wissen des Anwenders ausgeführt.

Im Gegensatz zu den Viren findet bei den Trojanern keine Selbstreplizierung statt, sondern die Verbreitung geschieht dadurch, daß durch die offensichtlichen Funktionen Anwender dazu gebracht werden, diese Software aus dem Internet herunterzuladen und ggf. sogar an Bekannte zu verteilen. Eine Entfernung der maliziösen Funktion alleine ist meist nicht möglich, da sie gewollt fest im Programmcode verankert ist.

Spezielle Unterarten von Trojanern sind Hintertüren (Backdoor oder Trapdoor) und logische Bomben. Backdoors öffnen im Hintergrund eine Tür für einen anderen Benutzer, so daß dieser Kontrolle über den fremden Rechner übernehmen kann. Logische Bomben haben eine schadhafte Funktion, die nur unter bestimmten Voraussetzungen, wie z.B. einem bestimmten Datum, eintritt.

1.1.3 Dropper

Bei dieser Form der Malware handelt es sich ebenfalls um eine spezielle Art von Trojanern. Ihre Aufgabe ist es einen anderen Trojaner oder Virus in das System zu bringen, wobei die Dropper selber nicht mit diesem infiziert sind, sondern ihn lediglich installieren. Bei Ausführung des Programmes findet die Infizierung des Systems statt. Hierbei ist nicht der Sachverhalt der Selbstreplizierung gegeben, womit es sich per Definition nicht um einen Virus handelt. Um ein System von einem durch einen Dropper infizierten Virus oder Trojaner zu bereinigen, genügt es also nicht selbigen zu entfernen, sondern der Dropper muss auch identifiziert und entfernt werden. Bzgl. der Entfernung von Droppern gilt das gleiche wie für Trojaner.

[9] s. S. 16, „1. Überwachung des Systems (Monitoring)“

1.1.4 Würmer

Diese Art von maliziöser Software ist zwar auch selbstreplizierend wie die Viren, allerdings als eigenständige Programme, d.h. Würmer hängen sich nicht an andere, gutartige Programme an. Oft findet die Verbreitung in Form einer trojanischen Funktion in einem witzigen Programm statt oder der Titel einer Mail verleitet den Anwender dazu, die maliziöse Anwendung auszuführen. Im Gegensatz zu Viren verbreiten sich Würmer aber selbständig über Netze. Mit den stark ineinander-greifenden Geschäftsprozessen einzelner Unternehmen und der damit einhergehenden zugenommenen Vernetzung, auch durch das Internet im privaten Bereich, sind Computer-Würmer besonders in den letzten Jahren zu einer starken Bedrohung geworden.

In den letzten Jahren verschickten sich Würmer bevorzugt über spezielle Mail-Programme, wie z.B. Outlook oder Outlook Express von Microsoft, an die dort im Adressbuch vorhandenen Empfänger. Diese Empfänger öffnen die erhaltene Mail und sorgen damit für eine weitere Verbreitung des Wurms im Hintergrund. Dieses Schneeball-System kann eine sehr schnelle Ausbreitung hervorrufen, die sowohl zu einer starken Belastung der Netzwerke als auch der Mail-Server führt, so daß mitunter kein produktives Arbeiten mit dem E-Mail-System mehr möglich ist.

1.1.5 Hoaxes

Die sog. Hoaxes oder schlechten Scherze sind eine Art E-Mail-Wurm. Hierin wird meist vor einem sehr gefährlichen Virus gewarnt, gegen den es noch kein Gegenmittel gibt. Zu den Hoaxes zählen auch die schon lange vor dem Informationszeitalter bekannten Kettenbriefe. Wie bei den Würmern handelt es sich zur Verbreitung um ein Schneeball-System. In dem E-Mail-Text wird der Empfänger meistens dazu aufgefordert, die Mail schnellstmöglich an alle oder einen Teil seiner Bekannten weiterzuleiten. Hier ist also bewußtes Handeln des Anwenders nötig, damit der Hoax sich verbreitet - im Gegensatz zum Wurm, wo die Verbreitung im Hintergund und ohne Wissen des Anwenders stattfindet.

1.1.6 Jokes

Die Jokes genannten Scherzprogramme haben an sich keine böshafte Funktion, sondern führen lediglich zu einer Verunsicherung des Anwenders, da sie vorgeben etwas schadhaftes zu tun, obwohl der Anwender dieses nicht möchte. Beispielsweise kann ein Programm, das vorgibt die Festplatte zu formatieren, mitunter beim Anwender zu einer Überreaktion führen, so daß er den Computer ausschaltet. Hierdurch könnte in Teilen das Dateisystem beschädigt werden und somit zu einem Datenverlust führen.

1.1.7 Makro- und Skript-Malware

Dies ist die jüngste Art der Malware. Im Gegensatz zu der oben beschriebenen „konventionellen“ Malware in Maschinensprache, liegt diese Form meist im Quelltext vor, da sie auf einer Skript-Sprache basiert, die interpretiert wird. Makrosprachen existieren vornehmlich in Anwendungs-Programmen wie z.B. Textverarbeitungen oder Tabellenkalkulationen und wurden ursprünglich dazu erdacht, dem Anwender die Möglichkeit zu geben, diese Programme für seine eigenen Bedürfnisse zu erweitern bzw. anzupassen.

Durch die Tatsache, daß der Programmcode für jeden lesbar ist und durch die Einfachheit der Programmierung mit einer solchen Skript-Sprache, kann nahezu jeder einen Virus o.ä. anschauen, analysieren, modifizieren oder ggf. einen völlig neuen Virus erstellen. Dies führte in den vergangenen Jahren zu einer sehr starken Zunahme der Makro-Malware. Ein weiterer Faktor für die starke Zunahme und Verbreitung ist, daß Office-Dokumente zunehmend per E-Mail verteilt werden bzw. über Server gemeinsam genutzt werden und dadurch mehr Wirts-Systeme erreichen können. Hinzu kommt, daß Makro- bzw. Skript-Sprachen betriebssystem-unabhängig sind, sondern von der benutzten Anwendungs-Software abhängen. Oft gibt es die gleiche Software für unterschiedliche Betriebssysteme, wie z.B. Microsoft Word oder Excel. Dies führt zusätzlich zu einer Zunahme der potentiell infizierbaren Systeme. War in der Anfangszeit der Viren nur von „betroffenen Betriebssystemen“ die Rede, ist durch die Skript- und Makro-Malware in den letzten Jahren eher die Sprache von „betroffenen Plattformen“, da die Grundlage für den Virus das Anwendungsprogramm ist, das völlig vom Betriebssystem abstrahiert.

1.1.8 Hostile Agents

Mit dem steigenden Interesse am WorldWideWeb (WWW) wurde der Bedarf nach mehr Funktionalität zunehmend größer. Der Funktionsumfang der Formatierungssprache HTML reichte nicht aus, um grafisch aufwendigere und interaktive Applikationen bereitzustellen. Daher wurden Schnittstellen wie ActiveX entwickelt, um die Funktionalität der Internet-Browser entsprechend zu erweitern.

Agenten bedienen sich der o.g. Schnittstellen bzw. Programmiersprachen. Hostile wird ein Agent genannt, wenn er „Handlungen ausführt, die nicht im Sinne des Benutzers liegen oder der den Benutzer dazu bringt, diese auszuführen“[10]. Hierzu zählen beispielsweise das Ausspähen von Paßwörtern oder das Löschen oder Verändern von Daten. Am bekanntesten sind sicherlich folgende zwei Arten von Schnittstellen bzw. Programmiersprachen zur Gestaltung von Agenten :

Java Applets

Die Programmiersprache JAVA wurde von der Firma SUN zur Erweiterung der Browser entwickelt. JAVA-Applets liegen auf dem Server in einem plattformunabhängigen kompilierten Zwischencode vor, der dann auf dem jeweiligen Client durch die „JAVA Virtual Machine“ (JVM) ausgeführt wird. Per Definition soll dieses im Internet in einem abgeschotteten Bereich auf dem Client, dem sog. Sandkasten[11], stattfinden, der keinerlei Zugriff auf lokale Ressourcen außerhalb dieses Bereichs zulassen soll. Trotz dieser Einschränkungen können immer noch „Denial of service“-Attacken hiermit gefahren werden, indem z.B. die Prozessorauslastung durch das JAVA-Applet dermaßen ansteigt, daß andere Anwendungen kaum noch Prozessorzeit erhalten und somit ein normales Arbeiten nicht mehr möglich ist. Außerdem existieren in dem Modell der Sprache JAVA oder in den Implementationen der JVM Fehler. Hierdurch ist es möglich über die Grenzen der Sandbox hinweg auf lokale Ressourcen zuzugreifen und somit die Sicherheitsvorkehrungen zu umgehen.

[10] s. [VTC 99]
[11] englisch : „Sandbox“

ActiveX-Controls

Im Gegensatz zu JAVA ist ActiveX keine Programmiersprache, sondern eine Schnittstelle. Außerdem ist sie nicht plattformunabhängig und besitzt auch kein Sicherheits-Modell. ActiveX wurde von Microsoft entwickelt und vor allem mit deren Internet-Browser, dem Internet Explorer, zur Verfügung gestellt. Mittlerweile gibt es auch für andere Browser sog. PlugIns, so daß dieselben Funktionalitäten auch hiermit genutzt werden können.

Da die Technik an sich sicherheitsgefährdend sein kann, können ActiveX-Controls im Internet bei den entsprechenden Stellen zertifiziert werden. Mit den entsprechenden Einstellungen im Browser werden dann nur solche zertifizierten und meist gutartigen Controls ausgeführt. Einen Schutz gibt die Zertifizierung allerdings lediglich gegen Verfälschung während der Übertragung, aber nicht gegen die von vornherein böswillige Programmierung.

1.1.9 Java-Script

JavaScript ist eine leistungs- und umfangsärmere Sprache als JAVA und wurde für kleinere Ablaufsteuerungen auf dem Client konzipiert. In ihrer ersten Version wurde sie direkt in den HTML-Code mit eingebaut und von daher auf jeden Fall mit auf den Client übertragen. Neuere Versionen erlauben das Einbinden von Skripten aus Dateien, die auch auf dem Server abgelegt sind. Mit JavaScript ist es u.a. möglich vom User unbemerkt E-Mails in seinem Namen zu versenden. Mit evtl. steigenden Möglichkeiten von JavaScript würden sich ähnliche Probleme wie bei ActiveX ergeben.

1.1.10 Mischformen

Die oben vorgenommene Einteilung ist eine Klassifizierung, die durch Definitionen zustande kam. In der Realität ist es teilweise schwer eine genaue Einstufung vorzunehmen, da viele Mischformen der verschiedenen Arten existieren. Beispielsweise gibt es Makro-Dropper, die normale Betriebssystem-Viren im System absetzen, Skript-Würmer, die normale Trojaner aus dem Internet nachladen wollen, um Passwörter auszuspähen o.ä.

1.1.11 Wanzen (Bugs)

Abhängig von der Definition gehört auch diese Programm-Anomalie ggf. zur Malware. Wanzen sind Programmierfehler, die unter bestimmten Umständen zu einem nicht spezifiziertem Verhalten führen. Unter der Betrachtungsweise, daß Malware vom Anwender nicht gewollte Funktionen ausführt, könnte man Wanzen also als Malware bezeichnen. Erweitert man die Definition von Malware um die Intention des Programmierers, der die schadhafte Funktion bewußt in den Code programmiert hat, so fallen die Wanzen nicht mehr unter diese Definition. Dies macht durchaus Sinn. Sämtliche Programmierfehler als Malware zu betrachten würde jeglichen Rahmen sprengen, denn es wird geschätzt, daß pro 5.000 Zeilen Code mindestens ein schwerwiegender Programmfehler auftritt.[12] Da bei den aktuellen kommerziellen Programmen diese Grenze mehrfach überschritten wird wäre also jedes größere Anwendungsprogramm als Malware klassifizierbar.

[12] [Brunnstein 1991], S. 19f

1.2 Der Computer-Virus-Katalog

1.2.1 Die Geschichte des Computer-Virus-Katalogs

Mit der Einrichtung des Virus Test Centers an der Universität Hamburg (VTC) im Frühjahr 1988 wurde eine der ersten Anlaufstellen für Personen geschaffen, die den Verdacht einer Computervirus-Infektion hatten. Zur exakten Beseitigung des Virus und, soweit möglich, der Auswirkungen seiner Schadensfunktion, bedurfte es sowohl für den Betroffenen als auch für die Mitarbeiter des VTC einer möglichst guten Kenntnis des jeweiligen Virus.

Zu diesem Zweck schien die Einrichtung eines Computer-Virus-Katalogs sinnvoll. Hierin sollten alle wichtigen Merkmale eines Virus aufgenommen und somit Unterschiede zwischen den verschiedenen Viren leicht erkennbar werden. Vergleichbar mit einem biologischen Bestimmungsbuch kann anhand spezifischer Merkmale eines Virenvorfalls auf den jeweiligen Virus geschlossen werden. Da der Katalog öffentlich zugänglich gemacht werden sollte, durfte er aber keinerlei Information liefern, die zur Virus-Programmierung benutzt werden könnte. Zunächst nur lokal auf Deutschland beschränkt wurde der Katalog später durch internationale Zusammenarbeit erweitert. Hierfür mußte er vorher in die englische Sprache übersetzt und validiert werden. Durch diese internationale Kooperation konnten bereits kurz nach ihrer Entdeckung neue Viren im Katalog dargestellt werden.[13] Um eine effiziente Arbeit zu ermöglichen sollten aber nur Viren für den Katalog klassifiziert werden, die an mindestens drei verschiedenen Orten aufgetaucht sind. Alle anderen würden als nicht verbreitet gelten und somit für die Öffentlichkeit uninteressant sein.

1.2.2 Das Format des Computer-Virus-Katalogs

Das aktuelle Format ist derzeit in der Version 1.2 spezifiziert. Im Gegensatz zu den vorherigen Versionen ist es in Englisch spezifiziert, so daß Einträge von weltweit verteilt tätigen Virenforschern vorgenommen werden können.

Ein Eintrag in dem Katalog enthält die folgenden drei Hauptteile, die für die Identifikation und die Maßnahmen gegen einen Virus maßgeblich sind:

- Benennung, Art und betroffene Systeme
- Wesentliche Merkmale
- Erprobte Gegenmaßnahmen

Zusätzlich zu diesen drei Abschnitten existiert noch ein weiterer, in dem die Historie des jeweiligen Katalogeintrags betreffende Informationen gespeichert werden. Alle Eigenschaften werden der Vollständigkeit halber aufgeführt, aber nicht ausgefüllt, wenn hierzu keine Angaben erforderlich sind. Hierdurch ist eine einheitliche Darstellung gewährleistet.

[13] [Brunnstein 1991], S. 127f

1.2.2.1 Benennung, Art und betroffene Systeme

In diesem Teil sind im Wesentlichen die Attribute enthalten, die die Klassifizierung betreffen. Neben den betroffenen Betriebssystemen bzw. Plattformen, wenn man in Hinblick auf Makro- und Skriptviren von Betriebssystemen abstrahiert, ist hier auch bereits der Name des Virus aufgeführt. Bei der Namensvergabe wird im Allgemeinen versucht eine spezifische Eigenschaft des Virus hervorzubringen. Dies können Bildschirm-Darstellungen, ertönende Melodien, Entdeckungsorte oder auch spezielle Auslösemechanismen sein, die maßgeblich für die Bezeichnung des Virus sind. Oftmals werden an verschiedenen Orten gleichzeitig Vorschläge gemacht, so daß diese dann als Alias-Namen im Katalog-Eintrag geführt werden. Es wird außerdem eine Klassifikation in eine der in Kapitel 1.1 aufgeführten Klassen angegeben.

Eine wichtige Eigenschaft zur schnellen Identifikation eines Virus ist seine Länge. Diese kann auf dem permanenten Speicher und im Arbeitsspeicher unterschiedlich sein, weshalb ggf. beide Längen im Katalog geführt werden. Oftmals sind zwei Viren vom gleichen Stamm, d.h. ihre Infektionsart und ihr Verhalten innerhalb des Systems unterscheiden sich kaum merkbar. Eine exakt gleiche Länge wird es aber bei kleinen Unterschieden im Verhalten selten geben. Daher wurde auch diese Eigenschaft in den ersten Teil des Katalogeintrags mit aufgenommen. Dies liegt daran, daß manchmal auch die Länge des Virus Teil des Namens oder eines Alias-Namens wird. In seltenen Fällen ergibt sich aus der Länge der Name, wie z.B. beim „5120“-Virus, der Dateien um 5120 Bytes größer macht.[14]

1.2.2.2 Wesentliche Merkmale

Zur Bestimmung der im vorigen Teil erwähnten Attribute bedarf es einer genauen Analyse des jeweiligen Virus. Hierbei wird, früher meist auf Maschinensprache-Ebene, heutzutage zunehmend auf Skript-Ebene, das Verhalten des Virus so exakt wie möglich nachvollzogen.

Infektionsmechanismus, Infektionsauslöser, Schadensfunktion und deren Auslöser, sowie die befallenen Speichermedien, die alle bereits in Kapitel 1.1 hinreichend erläutert wurden, zählen zu den hier beschriebenen Eigenschaften. Zu diesen Eigenschaften kommen ggf. noch Hinweise zur einfachen Identifikation hinzu. Dies können Texte sein, die im Maschinen-Code in Klarschrift stehen oder ein bestimmter Disketten-Name, den der Virus bei der Infektion vergibt.

Einige Viren nutzen, wie bereits erwähnt, Eigenarten spezieller Betriebssysteme oder Computer-Arten aus. Auf Personal Computern (PCs) sind dies beispielsweise die sog. Interrupts. Bei Interrupts unterscheidet man zwischen Software- und Hardware-Interrupts. Hardware-Interrupts sind durch die Hardware ausgelöste Unterbrechungsanforderungen, die gegenüber normalen Programmabläufen priorisiert werden. Die interne Uhr eines PCs wird über solch einen Interrupt 18 mal pro Sekunde weitergeschaltet. Würde dieses nicht über Interrupts geregelt werden, wäre eine ordnungsgemäße Funktion der Uhr nicht gewährleistet. Kundige Viren-Programmierer können diese Interrupts für Ihre Zwecke ausnutzen und so umleiten, daß zunächst die Routine des Virus aufgerufen wird und danach die eigentliche Interrupt-Routine.

[14] [Martin 1997] S.115f

Software-Interrupts hingegen ermöglichen eine computer-unabhängige Programmierung. Beim Aufruf eines Software-Interrupts wird an einer bestimmten Stelle im Speicher des Computers die eigentliche Sprungadresse der benötigten Routine ermittelt und dann zu dieser weiter verzweigt. Über die Software-Interrupts werden z.B. auch Schreib- und Lese-Aktionen auf die Datenträger gesteuert.

Von einem Virus benutzte Interrupts werden unter der Eigenschaft „Interrupts hooked“ im Virus-Katalog aufgeführt. Durch die Ausnutzung eines Interrupts kann es zu Auffälligkeiten im Systemverhalten kommen, die auf das Vorhandensein eines Virus schließen lassen, wie z.B. eine Verlangsamung des Rechners. Da Interrupts unter einigen neueren Betriebssystemen nicht direkt zugreifbar sind und ggf. Funktionen des Betriebssystems genutzt werden, ist diese Eigenschaft teilweise in „APIs[15] hooked“ umbenannt worden.

Oftmals läßt sich zwischen einigen Viren nicht ohne weiteres unterscheiden. Zwischen Viren besteht oftmals die Gefahr, diese miteinander zu verwechseln. Um dieses zu vermeiden, wird auf Ähnlichkeiten zu anderen Viren verwiesen, damit auf Unterschiede zwischen diesen Viren verstärkt geachtet werden kann. Die Gefahr bei einer Fehleinstufung besteht darin, daß ggf. falsche Informationen zur Entfernung des Virus und zur Beseitigung der Schäden vorliegen. Durch unkorrektes Vorgehen bei der Entfernung können die Schäden evtl. vergrößert werden. Dies kann bis zum endgültigen Datenverlust und zur Unbrauchbarkeit des Systems führen. Auch für aktuelle Antiviren-Scanner ist eine exakte Einstufung des Virus zur korrekten Entfernung ein entscheidendes Kriterium, welches auch auf die Qualität des Scanners schließen läßt.

Die digitale Version des Katalogs, CMBASE[16], zeigt hier noch zusätzliche Felder für exaktere Beschreibungen für Polymorphismus-, Stealth-, Tunneling- und Verschlüsselungstechniken. Durch diese Anpassungen wurde den aktuelleren Entwicklungen in der Virus-Programmierung Rechnung getragen.

1.2.2.3 Erprobte Gegenmaßnahmen

Aufgrund der während der Analyse eines Virus gewonnenen und in den Computer-Virus-Katalog eingetragenen Erkenntnisse über dessen Wirkungsweise lassen sich entsprechende zusätzliche Maßnahmen gegen eine Infektion entwickeln und testen. Im Allgemeinen gibt es sechs mögliche Wege, neben denen des menschlichen Verhaltens, mit denen sich die Ausbreitung oder Auswirkung eines Virus verhindern läßt. Diese sollen im Folgenden kurz dargestellt werden. Hierzu sei angemerkt, daß diese lediglich gegen Viren bzw. replizierende Malware wirken. Trojanische Funktionen können hierdurch nicht immer zuverlässig abgedeckt und verhindert werden.

1. Überwachung des Systems (Monitoring)

Bei dieser Methode soll bereits der Eindring-Versuch eines Virus in eine Datei oder in das System festgestellt werden. Hierbei wird bei einem versuchten, kritischen Zugriff eine Erlaubnis dieser Veränderung durch den Anwender verlangt. Um diese Funktionalität zu gewährleisten, muß das Verfahren im Betriebssystem fest verankert sein.

[15] API = Application Programming Interface
[16] s. S. 18, Kapitel 1.3 „CMBASE“

2. Entdeckung einer Veränderung
Im Gegensatz zum vorherigen kann dieses Verfahren eine Veränderung erst nach bereits erfolgter Infektion und evtl. bereits eingetretenem Schaden feststellen. Hierbei wird eine Prüfsumme nach einem geheimen Algorithmus für jede Datei im uninfizierten Zustand angelegt. Bei der nächsten Prüfung werden diese gespeicherten Daten mit den aktuell berechneten Prüfsummen verglichen. Bei einer Infektion einer Datei durch einen Virus wird sich deren Prüfsumme verändern. Durch die Diskrepanz läßt sich auf eine Virusinfektion schließen, die aber nicht zwingend die Ursache sein muß. Eventuell handelt es sich z.B. um eine Datendatei an der selbstverständlich Änderungen vorgenommen werden müssen, da sich der Datenbestand eines Unternehmens ständig ändert. Angreifbar wird dieses Verfahren in dem Augenblick, wo ein Virus-Programmierer den Algorithmus zur Berechnung der Prüfsumme kennt. Hierdurch wäre es möglich dafür zu sorgen, daß eine infizierte Datei die selbe Prüfsumme besitzt wie die unveränderte Datei.

3. Vernichtung des Virus („Anti-Viren“)
In der Zeit in der noch wenige Viren bekannt waren, wurde u.a. auch am VTC speziell für jeden Virus ein Antivirus entwickelt. Bereits leichte Modifikationen am Virus selber machten den Antivirus allerdings evtl. unbrauchbar oder es mußten Änderungen an ihm vorgenommen werden. Der starke Zuwachs an Viren ließ den Bedarf an Antiviren steigen. Hierdurch entwickelte sich die heutige Anti-Virus-Branche, in der die Anbieter von Anti-Virus-Lösungen sich meistens zunächst nur auf dieses Gebiet konzentrierten, inzwischen aber auch andere Sicherheitslösungen im Angebot haben. Um einen möglichst geringen Aufwand an Programmierung und Systemressourcen zu benötigen, verwenden die meisten Hersteller sogenannte generische Treiber, die viele Viren in einem Programmsegment zusammenfassen. Hierfür ist eine genaue Kenntnis der Wirkungsweise und der Ähnlichkeiten von Viren nötig.

4. Impfung der Datei
Auch dieses Verfahren stammt aus der Anfangszeit der Viren-Problematik. Viele Viren haben bestimmte Merkmale die sie selber nutzen, um zu erkennen, daß eine Datei bereits mit dem Virus infiziert ist.[17] Dieses kann man sich zunutze machen, indem man allen potentiell infizierbaren Dateien genau dieses Merkmal zuweist, so daß der bestimmte Virus (oder zumindest eine Familie) diese Datei nicht infizieren würde, da sie für ihn bereits infiziert ist.

Mit der starken Zunahme der Viren ist dieses Verfahren nicht mehr anwendbar, da man nicht alle Dateien gegen alle Viren impfen könnte. Außerdem hilft dieses Verfahren lediglich gegen bereits bekannte Viren, die außerdem selber eine Erkennung der Infektion besitzen müssen. Gegen neue oder Abänderungen bereits bekannter Viren hilft es hingegen nicht.

5. Hardware-Methoden
Da die meisten Betriebssysteme keinen ausreichenden Schutz gegen Viren bieten, können Hardware-Methoden zum Ausfüllen dieser Schwachstelle benutzt werden. Hierbei werden durch die Hardware die Schreibzugriffe auf bestimmte Bereiche kontrolliert, wodurch Viren sich nicht in diese Bereiche replizieren können. Diese Schutzmechanismen müssen bei geplanten Installationsoperationen durch den Anwender oder den Administrator kurzzeitig außer Funktion gesetzt werden.

[17] s. S. 7, “Erkennungsteil“

6. Verschlüsselungsmethoden
Eine andere Art der Impfung stellt die Verschlüsselung dar. Hierbei werden alle Dateien mit einem bestimmten Algorithmus verschlüsselt. Ein anderes, im Speicher befindliches Programm sorgt dann bei Lesezugriffen für die Entschlüsselung und die Ausführbarkeit im Arbeitspeicher. Eine Infektion einer verschlüsselten Datei führt dazu, daß diese nicht ordnungsgemäß entschlüsselt werden und somit auch der Virus nicht aktiv werden kann. Zwar kann die reguläre Datei auch nicht ausgeführt und muß neu installiert werden, jedoch sind die Verbreitung und die Schadensauswirkungen, abgesehen vom administrativen Aufwand der Neuinstallation, eingegrenzt.

1.3 CMBase

Waren die Einträge im Computer-Virus-Katalog zu Anfang noch eine Sammlung einzelner Textdateien wurde mit der Zunahme an Viren eine besser handhabbare Lösung nötig. Da man u.a. die Möglichkeit haben wollte nach bestimmten Merkmalen suchen zu können und damit die Übersichtlichkeit zu erhöhen, war eine Datenbank die naheliegende Lösung. Diese wurde dann auch am VTC programmiert und auf den Namen „CMBase“ (**C**omputer **M**alware Data**base**) getauft.

Die Anwendung ist eine rein textbasierte Lösung, die auf MS-Dos und allen kompatiblen Betriebssystemen oder Emulationen läuft. Die gesamte Benutzerführung und Hilfe ist in englischer Sprache. Diese wurde gewählt, da die gespeicherten Katalogeinträge ebenfalls in englischer Sprache sind und CMBASE internationalen Experten zur Verfügung stehen sollte.[18]
Innerhalb vom CMBASE kann man zwischen den verschiedenen Plattformen für Viren wählen. Beim Programmstart wird eine Standard-Plattform geöffnet die vom Anwender konfigurierbar ist. Somit ist gewährleistet, daß immer eine Plattform die Aktive ist. Alle nachfolgenden Aktionen sind nur für die aktuell gewählte Plattform innerhalb von CMBASE verfügbar.
Für die jeweils aktive Plattform stehen verschiedene Ansichten zur Verfügung, zwischen denen der Anwender wählen kann. Zunächst einmal gibt es einen alphabetischen Index, mit dessen Hilfe man durch die Vielzahl der Einträge navigieren kann. Innerhalb dieses Indexes sind lediglich die Viren verfügbar für die ein Katalog-Eintrag existiert.

Eine bessere Übersicht über die verschiedenen Stämme von Viren gibt die nächste verfügbare Ansicht „Strain Information“ (Virenstamm-Information). Hierbei werden die unterschiedlichen Virenstämme hierarchisch angezeigt. Interessiert sich der Anwender für einen Stamm näher, kann er sich in einem zweiten Fenster die dazugehörigen Viren mit Namen anzeigen lassen. Dieses geht bis zu einer Tiefe von 4 Ebenen, wäre aber bei Bedarf durchaus problemlos erweiterbar.

Einen anderen Blick auf die Virenstämme bietet die Baum-Ansicht. Ähnlich wie in einem Stammbaum werden hierbei, im Gegensatz zur Virenstamm-Ansicht, alle vorhandenen Ebenen auf einmal eingeblendet.

[18] s. S. 14

Stehen bei dem alphabetischen Index nur die Viren zur Auswahl für die tatsächlich Virenbeschreibungen vorhanden sind, sind sowohl bei der Virenstamm- als auch bei der Baum-Ansicht alle Viren und ihre Familienzugehörigkeit vorhanden. In diesen Ansichten sind die Viren zu denen detaillierte Informationen vorliegen, speziell kenntlich gemacht, so daß der Anwender dieses sofort erkennen kann.

Da die drei Ansichten bereits die Kenntnis des Virusnamens voraussetzen, ist hiermit bei der Identifikation eines bestimmten Virus nicht geholfen. Für diesen Zweck gibt es die Möglichkeit in bestimmten Merkmalen des Virus-Katalogs suchen zu lassen. Die Suche erstreckt sich hierbei nur auf die Viren für die eine detaillierte Beschreibung vorliegt. Also jene, die auch im alphabetischen Index aufgeführt werden.

Zusätzlich zu den verschiedenen Ansichten bereits existierender Virenbeschreibungen können neue Katalog-Eintragungen der Datenbank hinzugefügt werden. Im Umkehrschluß hierzu können auch Einträge aus der Datenbank in die altbekannte Textdarstellung exportiert werden, beispielsweise um diese per E-Mail zu verschicken.

Sowohl die Vireninformationen als auch die Hierarchie der einzelnen Virenfamilien und der Viren sind in einer Datenbank gespeichert. Die Datenbank an sich ist im DBase IV-Format. Für jede Plattform existiert eine eigene Tabelle, wobei die Struktur aller Tabellen gleich ist und sich sehr gut aus der ASCII-Darstellung eines Katalogeintrages ableiten läßt. DBase IV-Datenbanken werden nicht in einer einzelnen Datei gehalten, sondern in separaten Dateien innerhalb eines Verzeichnisses. Für jede Tabelle existiert hierbei mindestens eine Datei, ggf. sogar mehr.

Bislang existierte diese Anwendung nur lokal am Fachbereich Informatik der Uni Hamburg und wurde für externe Anfragen genutzt. Oft wurden diese Anfragen von Studenten der Universität beantwortet. Da dieses nicht immer vor Ort waren und Zugriff auf die Daten hatten, gelangten die angeforderten Informationen manchmal nicht schnellstmöglich zu den Anfragenden. Um diese Lücke zu stopfen würde es Sinn machen, eine Anwendung, die auf der bereits existierenden basiert, der Öffentlichkeit zur Verfügung zu stellen. Mit welchen Mitteln dieses am besten geschehen sollte, wird im Folgenden diskutiert.

2 Anforderungen

Bei der Erstellung einer neuen Anwendung oder der Portierung einer vorhandenen haben vielerlei Faktoren Einfluß auf die letztendliche Implementation. Die Faktoren und ihre Abhängigkeiten voneinander sollen in diesem Kapitel dargestellt werden. Anschließend werden die Gegebenheiten des Projektes erläutert, das dieser Arbeit zugrundeliegt.

2.1 Entscheidungsfaktoren

Die wohl wichtigsten Faktoren bei der Entscheidungsfindung sind sicherlich die Kosten, die Sicherheit und die erwartete und tatsächliche Performanz des Systems. Diese Faktoren haben zentrale Auswirkungen auf die Ausprägung anderer Faktoren. Diese anderen Faktoren wiederum beeinflussen auf andere Art die auf sie einwirkenden Faktoren und ggf. andere. Ein optimales Zusammenspiel aller Faktoren mit allen vorab gewünschten Anforderungen zu einer Lösung ist oftmals nicht möglich, sondern es gilt, durch Abwägen die bestmögliche Lösung aus der Kombination aller, vielfach historisch bedingter, Faktoren zu finden. Eine Skizze der Zusammenhänge ist in Abbildung 2.1.1 dargestellt. Der Einfachheit halber wurden indirekte, also transitive, Abhängigkeiten in der Darstellung weggelassen. Auf einzelne Faktoren soll im Folgenden näher eingegangen werden.

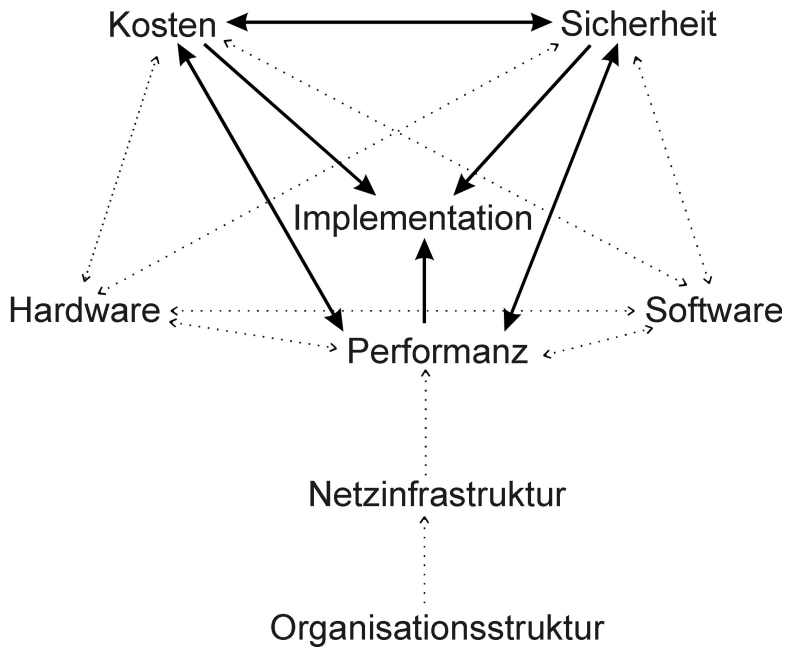


Abbildung 2.1-1 „Interdependenzen bei der Entscheidungsfindung“

2.1.1 Faktor „Kosten“

Zu den wichtigsten Faktoren bei Entscheidungsfindungen jeglicher Art zählen zweifelsohne die Kosten bzw. die Wirtschaftlichkeit. Als zentraler Punkt wird hieraus die Nützlichkeit einer Entscheidung klar. Andere Parameter wie z.B. die Performanz hängen hiermit eng zusammen. Um eine Implementation rechtfertigen zu können, muß normalerweise im unternehmerischen Umfeld ein positiver Nutzen herausspringen, welcher aber nicht immer direkt meßbar ist, wie z.B. ein Image-Gewinn.

Der Nutzen wird normalerweise im Begriff der Wirtschaftlichkeit zusammengefaßt. Damit ein gewisses Maß an Wirtschaftlichkeit erreicht werden kann, darf eine bestimmte Investitionssumme nicht überschritten werden. Dieser Betrag schlägt sich dann in dem Budget für ein Projekt nieder. Somit kann man zur Vereinfachung die Wirtschaftlichkeit und das Budget bzw. die Kosten durchaus gleichsetzen.

2.1.2 Faktor „Performanz“

Um mit einem bestimmten Budget eine bestimmte Wirtschaftlichkeit erreichen zu können bedarf es auch eines gewissen Zugewinns. Dieser schlägt sich bei Projekten im informationstechnischen Umfeld normalerweise in Produktivitätsgewinnen gegenüber der vorherigen Lösung nieder. Hierbei ist es unwichtig, ob es sich um eine Lösung mit oder ohne IT-Einsatz handelte. Der Produktivitätsgewinn wird sowohl durch eine evtl. mögliche Umstrukturierung der betriebswirtschaftlichen Abläufe als auch durch die meist schnellere, da (teil-)automatisierte Verarbeitung erzielt.

Performanz steht aus Vereinfachungsgründen als übergeordneter Begriff für die Summe mehrerer kleinerer Faktoren, da diese in unterschiedlichen Szenarien verschiedene Bedeutungen erhalten. Außerdem hängen sie von den jeweiligen wirtschaftlichen Zielsetzungen ab und erhalten damit unterschiedliche Gewichtungen. Zu den o.g. Punkten zählen z.B. Geschwindigkeit, Kundenzufriedenheit (die meist in direktem Zusammenhang mit dem Geschwindigkeitsgewinn steht), Kostenminimierung auf unterschiedlichen Ebenen (z.B. Personaleinsparungen und/oder Minimierung der Lagerbestände) und Zugewinn an Informationen als Entscheidungsgrundlage. Weitere Punkte sind durchaus denkbar. Bei Bedarf ist eine genauere Betrachtung evtl. vorhandener Unterpunkte möglich. So kann bei Geschwindigkeit noch zwischen Antwortzeit des Servers und Verarbeitungsgeschwindkeit des Clients unterschieden werden, wenn dies aufgrund der Rechnerausstattung Sinn macht und man Engpässe des Systems lokalisieren will.

In Abbildung 2.1.1 werden unter dem entsprechenden Punkt die aus wirtschaftlicher Sicht gewünschte Performanz als auch die tatsächlich Erzielte bzw. Erzielbare zusammengefaßt. Sollte die Erzielte unter der Gewünschten liegen, hat dieses entweder Einfluß auf die gesteckten Ziele, die heruntergesetzt werden müssen, oder auf das Budget, da ggf. durch eine kleine Aufstockung desselbigen Performanzgewinne erzielbar wären.

2.1.3 Faktor „Sicherheit“

Je nachdem um was für eine Anwendung es sich bei der zu implementierenden handelt ist dieser Faktor mehr oder minder wichtig. Daran richten sich die drei verschiedenen Sicherheitsmerkmale – Vertraulichkeit, Verfügbarkeit und Integrität – aus.
Der Schutz der Daten, die Vertraulichkeit, ist oft auch durch das Datenschutzgesetz vorgegeben. Hierbei ist sowohl der Zugriff auf die Daten selber als auch das Abhören der Daten von Übertragungswegen zu achten. Vor allem externe Übertragungswege über WAN-Verbindungen sind hierbei besonders zu sichern, z.B. durch Virtual Private Networks (VPN) oder allgemein durch Verschlüsselungsverfahren.

Essentielle Anwendungen, die das Tagesgeschäft bestimmen, wie z.B. Informationssysteme in einer Telefonauskunft, benötigen eine sehr hohe Verfügbarkeit, da ohne diese Anwendungen kein weiterer Betrieb möglich ist. Zudem ist die Integrität der Daten hierbei sehr wichtig. Ein funktionierendes Informationssystem, das aufgrund eines falschen Datenbestandes inkorrekte Daten liefert, ist für ein Unternehmen existenzbedrohend.

2.1.4 Faktoren „Hardware“, „Software“ und „Netz-Infrastruktur“

Für die Erreichung der gewünschten Leistung durch die jeweilige Implementation spielt natürlich die Leistungsfähigkeit der jeweiligen Hardware eine entscheidende Rolle. Diese, sowohl auf der Server- als auch auf der Clientseite, hat hierbei direkte Auswirkungen auf die Verarbeitungsgeschwindigkeit. Normalerweise wird zunächst die bereits vorhandene Ausstattung betrachtet. Neuanschaffungen sollten erst an zweiter Stelle stehen.

Auch die Auswahl der eingesetzten Software hat Auswirkungen auf die Performanz. Pakete mit weniger Optionen sind meist schlanker und daher auch performanter. Sollten die Voraussetzungen an die Vielfalt der Software nicht so hoch gesteckt sein, empfiehlt sich hier die schlankere Lösung, die dann meist auch weniger Anforderungen an die Hardware stellt.

Natürlich interdependieren die Auswahl der einzusetzenden Hard- und Software sowie die Netz-Infrastruktur miteinander. Die Auswahl der Hardware hat Auswirkungen auf die verfügbare Software. Andererseits kann die Entscheidung für eine benötigte Software an bestimmte Hardwarevoraussetzungen gekoppelt sein. Die bereits vorhandene Infrastruktur hat ggf. Auswirkungen auf die zu wählende Hardware, da sie durch vorherige Entscheidungen z.B. keinen offenen Standard darstellt, sondern herstellerspezifisch ist. Oftmals zieht diese Abhängigkeit dann auch eine Einschränkung der zur Auswahl stehenden Software nach sich. Andererseits ziehen Entscheidungen für benötigte Hard- und Software ggf. auch Änderungen in der Infrastruktur nach sich.

Die Netz-Infrastruktur selber hängt in weiten Teilen auch von der Organisation innerhalb des Unternehmens ab. So werden geographisch weit voneinander entfernt liegende Standorte durch WAN (Wide Area Network)-Leitungen miteinander verbunden. Meist bieten diese Leitungen nicht genügend Bandbreite, um ein effizientes Arbeiten mit einer server-gestützten Anwendung[19] zu ermöglichen.

2.2 Implementationsumgebung

2.2.1 Serverseitige Umgebung

Da der Arbeitsbereich AGN (Anwendungen in Geistes- und Naturwissenschaften) am Fachbereich Informatik der Universität Hamburg einen eigenen Web-Server betreibt, sollte dieser für die Anbindung an das Internet genutzt werden. Dieser Server wurde für die Studenten des Arbeitsbereichs zu Studiumszwecken installiert und auch von diesen administriert.

[19] s. Kapitel 3

Primär ist dieser Server dafür gedacht, den Studenten Praxiserfahrung in der Administration eines solchen Rechners zu vermitteln und außerdem den Arbeitsbereich durch Informationen nach außen zu repräsentieren. Viele außer-universitäre Institutionen, Firmen und Privatpersonen nutzen diesen Dienst weltweit auch zur Information und Kontaktaufnahme. Außerdem haben die Mitarbeiter und Studenten des Arbeitsbereichs die Möglichkeit hier ihre persönliche Homepage der Öffentlichkeit zu präsentieren.

Bei dem Server handelt es sich um einen handelsüblichen Personal Computer mit einem Intel Pentium II-Prozessor mit 333 MHz. Der Rechner verfügt über 128 MB RAM Arbeitsspeicher und eine Grafikkarte mit 4 MB RAM. Außerdem ist ein SCSI-Controller vorhanden, an dem eine entsprechende Festplatte mit insgesamt vier Partitionen und einem Speichervolumen von insgesamt 6,8 GigaByte (GB) installiert ist. Die Partitionen beinhalten jeweils einen bestimmten Teil der gesamten Software und Daten des Web-Servers als ganzen. Zur Datensicherung verfügt der Computer außerdem noch über ein fest eingebautes Bandlaufwerk.

Als Betriebssystem wurde Microsoft Windows NT 4.0 in englischer Sprache installiert, das mit dem zum gegenwärtigen Zeitpunkt aktuellen ServicePack 6a aktualisiert wurde. Die Wahl fiel auf Englisch, da hierfür sicherheitsrelevante Patches meist eher verfügbar sind als für anderssprachige Versionen. Oftmals gibt es diese Patches auch nur in Englisch. Um hier schnellstmöglich eine erneute Absicherung des Servers vornehmen zu können wurde diese Entscheidung getroffen, wie übrigens in vielen anderen Institutionen auch, da hier die Sicherheit und Verfügbarkeit der von Servern bereitgestellten Dienste essentiell wichtig sind.

Aufsetzend auf dem Betriebssystem wurde als WebServer-Software der Internet Information Server in der Version 4.0 installiert. Als nach außen verfügbare Dienste werden durch diesen WWW und FTP (File Transfer Protocol) angeboten. Über letzteren haben Interessierte die Möglichkeit Publikationen des Arbeitsbereichs oder evtl. auch aktuelle Antiviren-Programme herunterzuladen.

Zusätzlich zu der WebServer-Software ist außerdem ein speicherresidentes Antiviren-Programm installiert, um die Infektion mit z.B. einem trojanischen Pferd zu unterbinden, mit dessen Hilfe ein Angreifer die Kontrolle über den Server übernehmen könnte. Die Antiviren-Software verfügt außerdem über einen Zeitplaner, über den üblicherweise regelmäßige Aktionen, den Virus-Schutz betreffend, automatisch ausgeführt werden. Auf dem hier vorliegenden Server wird über diesen Zeitplaner die tägliche Sicherung des Datenbestandes auf dem bereits oben erwähnten Bandlaufwerk gestartet. Die Speicherbänder werden täglich durch Mitarbeiter der Arbeitsbereichs gewechselt und dezentral aufbewahrt, so daß immer eine tagesaktuelle Kopie des Datenbestandes existiert.

Die Anbindung an das Internet erfolgt über das Rechenzentrum des Fachbereichs Informatik. Der Rechner hat eine feste IP-Adresse. Um ihn gegen Angriffe von außen abzusichern steht er hinter einer Linux-Firewall, die mit einfachem Paketfilter arbeitet. Durch diese Firewall werden nur die TCP/IP-Pakete an die Ports durchgelassen, auf denen der WebServer auch Dienste anbietet. Im einzelnen sind dieses die Ports 20 und 21 für FTP und Port 80 für HTTP.

Zusätzlich zu dem WebServer werden noch ein paar Mailinglisten über diesen Computer verwaltet. Außerdem wird später wahrscheinlich eine weitere Internet-basierte Anwendung auf dem Computer betrieben, die Zugriff auf die Ports der E-Mail-Protokolle POP3 (Port 25) und SMTP (Port 110) von innen durch die Firewall benötigt. Diese Anwendung wird im Rahmen einer weiteren Ausarbeitung am Arbeitsbereich AGN konzipiert und entwickelt.

2.2.2 Clientseitige Umgebung

Einer der großen Vorteile des WWW[20] birgt auch einen gewichtigen Nachteil. Im Gegensatz zu unternehmens-internen Intranets, in denen man sich auf einen gewissen Standard bzgl. der auf den Clients vorhandenen Software stützen kann, ist dieses im offenen Internet nicht gegeben. Gerade die Unabhängigkeit von einer bestimmten Plattform ist für die Anwender ein großer Vorteil, aber für die Anbieter von Inhalten ein großes Problem.

Obwohl HTML einen Standard darstellt, implementiert doch nahezu jeder Anbieter der Browser-Software eigene Erweiterungen, die er gerne in zukünftige Standards aufgenommen haben möchte. Hierdurch sollen Anteile auf dem Browser-Markt gesichert und neue erobert werden. Zu diesen Erweiterungen kommt hinzu, daß auch jeder Hersteller eine eigene Implementation der Interpretation und daraus resultierend eine Darstellung erzielt, die sich von der Darstellung anderer Hersteller unterscheidet. Es kommt durchaus vor, daß sogar ein und dieselbe HTML-Seite, die nur Standard-Formatierungen enthält, in den Browsern unterschiedlicher Hersteller verschieden dargestellt wird. Viele Entwickler und Designer von Internet-Seiten müssen deshalb bei der Erstellung das Aussehen auf unterschiedlichen Browsern und auch Betriebssystemen kontrollieren und daraufhin die Gestaltung der Seiten ausrichten. Am verbreitetsten ist hier der Navigator von Netscape ab Version 4 aufwärts, genauso wie der Internet Explorer von Microsoft, ebenfalls ab Version 4 aufwärts. Zu den verbreitetsten Betriebssystemen bei den Internet-Usern zählen alle Varianten von Windows von Microsoft, MacOS für den Apple Macintosh und in letzter Zeit stark zunehmend Linux.

Eine zusätzliche Schwierigkeit ist, daß auf den Computern der Clients nicht immer die gleiche Bildschirmauflösung vorausgesetzt werden kann. Somit ist die Darstellung auch nicht immer die gleiche. Um eine einheitliche Darstellung zu erreichen sollte hier die geringste Auflösung gewählt werden, da diese den maximal benutzbaren Bildschirmbereich definiert. Alle höheren Auflösungen benutzen den nicht benötigten Bildschirmplatz dann nicht. Am gebräuchlichsten ist es, die Auflösung von 800 x 600 Bildpunkten als Entwicklungsgrundlage zu benutzen, da nahezu alle noch eingesetzten Grafikkarten und Monitore diese Auflösung darstellen können. Der Anteil der Computer der dieses nicht kann, kann vernachlässigt werden. Eine Farbtiefe von 256 Farben kann clientseitig ebenfalls als Minimum angenommen werden, da viele lokal installierte Anwendungen diese bereits benötigen und somit von den Anwendern als Standard-Einstellung mindestens vorhanden ist.

Die Übertragungsraten der Anwender im Internet sind sehr unterschiedlich. Viele Firmen haben großzügig dimensionierte Anbindungen für ihre Mitarbeiter, andere teilen sich eine Einwähl-Verbindung über ein Modem oder eine ISDN-Leitung. Obwohl die Tendenz zu immer schnelleren Internet-Anbindungen geht, gibt es dennoch, vor allem in Privathaushalten, recht langsame Verbindungen. Um das Gros aller Nutzer abzudecken, sollte man hier, ähnlich wie bei der Bildschirmauflösung, von einer Übertragungsrate von 28.800 Baud ausgehen.

[20] s. Kapitel 3.2, Seite 35

2.3 Anforderungskatalog

Aufgrund der o.g. dargestellten Gegebenheiten läßt sich ein Anforderungskatalog erstellen, an dem sich die Auswahl einer geeigneten Lösung ausrichten sollte.

2.3.1 Kosten

Da die finanziellen Mittel einer Universität meist begrenzt sind und auch kein finanzieller Nutzen aus dem Projekt entsteht, sollten sich die Kosten in einem möglichst geringen Rahmen halten. Die Kosten für die eigentliche Implementation sind für den Arbeitsbereich effektiv nicht vorhanden, da sie im Rahmen dieser Arbeit übernommen wird. Desweiteren kann auf bereits vorhandene Hardware zurückgegriffen werden, die durchaus noch Kapazität für diesen zusätzlichen Dienst zur Verfügung hat. Ebenfalls bereits vorhanden ist die WebServer-Software, so daß durch die Anwendung keine Lizenzkosten hierfür entstehen.

Um die zukünftigen Wartungskosten gering zu halten, sollte das System so programmiert werden, daß es durch Erweiterungen der Einträge sich selber ebenfalls erweitert und keine zusätzlichen Anpassungen an der Programmierung erforderlich machen. Damit bei einem Umstieg auf eine andere WebServer-Software die zu erstellende Anwendung ebenfalls ohne großen Aufwand benutzt werden kann, wäre es wünschenswert hierbei auf Standards zurückzugreifen, die auch auf anderen Systemen verfügbar sind. Dabei sollte auch darauf Rücksicht genommen werden, daß durch zusätzlich erforderliche Software die Lizenzkosten nicht unnötig erhöht werden.

2.3.2 Antwortzeiten und Darstellung

Da der Dienst der Universität unentgeltlich angeboten wird sind die Antwortzeiten eher unwichtig. Zudem ist die Anzahl der Zugriffe auf diesen Dienst mit heutigem Stand nicht abschätzbar, werden aber verhältnismäßig gering erwartet. Sollte die erzeugte Last auf den WebServer durch diesen Dienst zu hoch, könnte durch eine Aufstockung der Hardware oder durch eine funktionale Trennung diese wieder kompensiert werden. Dennoch sollte auf die Attraktivität des Angebots geachtet werden. Schließlich stellt der WebServer mit der angebotenen Datenbank-Anwendung einen Dienst dar, der den Arbeitsbereich AGN nach außen repräsentieren soll und somit für den Ruf eine wichtige Rolle spielt. Lange Antwortzeiten sorgen bei Anwendern eher für Frust und machen somit das Angebot uninteressant. Wie bereits oben erwähnt ist der Engpaß bei vielen Anwendern bei der Übertragungsrate zu sehen. Deshalb sollte das zu übertragene Datenvolumen möglichst gering gehalten, aber dennoch eine optisch ansprechende Darstellung gefunden werden. Eine schlichte und grafisch nicht überladene Darstellung würde auch dem Image einer Universität gerechter werden. Die Darstellung sollte auf eine Bildschirmauflösung von 800 x 600 Punkten angepaßt werden. Da der Großteil der Viren für IBM-kompatible PCs existiert und diese derzeit nahezu alle windows-basiert betrieben werden, sollte man ein Hauptaugenmerk auf die für diese Plattformen gängigen Internet-Browser wie Netscape Navigator und Microsoft Internet Explorer legen.

2.3.3 Sicherheit

2.3.3.1 Vertraulichkeit

Die Daten in der Datenbank sind nicht schützenswert gegen Ansicht durch Dritte. Vielmehr sollen diese Daten der Allgemeinheit zugänglich gemacht werden, die vorher schon auf anderen Wegen uneingeschränkt für jeden einsehbar waren. Somit sind hierbei in Hinsicht auf die Vertraulichkeit keine Dinge zu beachten.

2.3.3.2 Verfügbarkeit

Obwohl kein Vertrag über die Dienstverfügbarkeit mit jemandem existiert, aufgrund dessen der Arbeitsbereich an einen bestimmten Faktor der Verfügbarkeit gebunden ist, ist eine hohe Verfügbarkeit wünschenswert. Diese ist gleichzusetzen mit der Verfügbarkeit des WebServers an sich, da die Datenbank einen Teil des Gesamtangebotes ausmacht. Der WebServer repräsentiert die Arbeit des Arbeitsbereiches nach außen hin. Maßnahmen, um diesen zugreifbar zu erhalten liegen außerhalb des Rahmens dieser Arbeit. Sie sind von den Administratoren und Arbeitsbereichs-Verantwortlichen zu bestimmen.

Durch die regelmäßige Sicherung der Daten des Servers ist eine rasche Neuaufnahme des Betriebes der Datenbank-Anwendung, bei einem Einbruch oder einem Rechnerausfall, fast genauso schnell bewerkstelligbar wie die des gesamten Servers.

Gegen Angriffe von außen schützt primär die vorgeschaltete Firewall[21], die einen Großteil der möglichen Attacken schon abwehren sollte. Lediglich die bereits in Kapitel 2.2.1 erwähnten Ports werden von außen von der Firewall an den WebServer weitergeleitet. Dieses ist nötig, um Anfragen an den Server überhaupt zuzulassen. Gefahr für die Verfügbarkeit besteht also nur über die von außen zugänglichen Ports 20,21 und 80. Über diese gelangen die Anfragen weiterhin an den WebServer und werden von diesem verarbeitet.

Durch Schwächen im WebServer-Design oder Fehler in der Implementation desselbigen kommt es zu sog. Vulnerabilities. Diese Schwachpunkte können für mögliche Angriffe von außen genutzt werden. Beispielsweise kann durch Senden einer bestimmten Anfrage-URL ein Puffer-Überlauf erzeugt werden, der die WebServer-Software zum Stillstand bringt. Durch diesen „Denial of Service“-Angriff steht das Angebot des Arbeitsbereichs so lange nicht mehr zur Verfügung, bis der WebServer bzw. der Rechner neu gestartet wird. Nach Bekanntwerden solcher Schwachstellen veröffentlicht der jeweilige Hersteller meistens innerhalb kurzer Zeit einen Korrektur-Patch[22]. Hier ist es Aufgabe der Administratoren die einschlägigen Foren zu beobachten und die entsprechenden Patches schnellstmöglich zu installieren.

Eine Analyse der Verfügbarkeits-Anforderungen der Anwendung „VMServ“ werden im Rahmen dieser Arbeit nicht betrachtet.

[21] s. Seite 23
[22] s. Seite 23

2.3.3.3 Integrität

Die Korrektheit der Daten innerhalb der Datenbank und der darauf aufbauenden Ausgabe durch die Anwendung ist sehr wichtig. Aufgrund der gelieferten Daten versucht evtl. ein Viren-Geschädigter sein System von der Malware zu reinigen und den Ursprungszustand wiederherzustellen. Fehlerhafte oder unvollständige Informationen über den jeweiligen Virus würden hierbei weniger helfen als vielmehr die spätere Rettung gänzlich unmöglich machen.

Durch die regelmäßigen Sicherheitskopien wird auch jeweils der letzte Stand der Datenbank gesichert. Allerdings ist nicht gewährleistet, daß die gesicherten Daten immer korrekt sind, da durch evtl. auftretende Festplattenfehler o.ä. der Inhalt der Datenbank-Dateien selber nicht mehr korrekt ist. Wenn dieser Fehler zu spät bemerkt wird ist ggf. keine korrekte Kopie der Datenbank mehr vorhanden. Hier kann die Anwendung selber durch Checksummen-Bildung die Integrität auf dem Datenträger validieren und ggf. Warnungen ausgeben.

Auch wenn die Daten auf dem Server selber intakt sind und die Anwendung diese auch korrekt verarbeitet, stellt dieses nicht sicher, daß die Daten auch korrekt beim Empfänger ankommen. Da die Daten durch das Internet geleitet werden ist es an den entsprechenden Knotenpunkten für einen potentiellen Angreifer möglich, die Original-Daten abzufangen, zu verändern und die so modifizierten Daten an den Empfänger zu senden. Bekannt ist dieses als Man-in-the-middle-Attack. Der Empfänger bekommt von diesen Modifikationen nichts mit und ist in der Annahme, daß er die Original-Daten vom Arbeitsbereich AGN vor sich hat. Gegen solche Manipulationen helfen nur elektronische Verschlüsselungs- und Signaturverfahren. Diese gibt es auch in der WebServer- und Browser-Software. Da man hierfür ein offiziell beglaubigtes digitales Zertifikat auf Seiten des Servers benötigt und dieses durch die Zertifizierung hohe Kosten verursacht, wurde auf den Einsatz eines solchen Zertifikats auf dem WebServer des Arbeitsbereichs verzichtet.

2.4 Risikoanalyse

In diesem Abschnitt soll ein kurzes Resümee der zuvor erläuterten Aspekte bzgl. der Gefährdung einer webbasierten Anwendung auf dem WebServer des Arbeitsbereichs AGN erfolgen. Hierbei werden die Anforderungen und Aspekte bzgl. der Anwendung „VMServ“ außer acht gelassen, da diese erstens zum Zeitpunkt der Erstellung dieser Arbeit nicht endgültig fertiggestellt ist und zweitens auch nicht sichergestellt ist, daß diese wirklich endgültig auf dem gleichen Rechner laufen wird. Sicherheitsbetrachtungen bzgl. dieser Anwendung werden dann in einer weiteren Arbeit ausgeführt werden.

Beinhaltete das WWW zu Anfang nur statische HTML-Seiten wuchs der Datenbestand mit der Zeit, womit auch die Unübersichtlichkeit für Benutzer des Internets zunahm. Der Bedarf nach interaktiven und somit auch nach dynamisch generierten Seiten durch Anwendungen wurde größer. Hierdurch entstand aber auf der Seite der Inhaltsanbieter auch ein größeres Sicherheitsrisiko. Viele Anwendungen laufen zum größten Teil servergestützt, d.h. die Verarbeitung der Daten findet auf dem Server statt und nur die Darstellung und Eingabe erfolgt auf den Clients.[23] Somit erlauben die Anbieter den oftmals anonymen Benutzern auf ihrem Server auch Code auszuführen.

[23] s. Kapitel 3.1, S. 29ff

Der Programm-Code selber ist zwar durch den Anbieter vorgegeben, doch kann durch ungeschickte oder unbedachte Programmierung ein Sicherheitsloch entstehen, welches durch entsprechend gewählte Parameterübergabe durch einen Internet-Benutzer ausgenutzt werden kann. Diese entsprechenden Löcher, am besten schon im Vornherein, zu stopfen und die entsprechenden Absicherungs-Mechanismen gut zu nutzen, obliegt der Verantwortung des Anbieters bzw. des jeweiligen beauftragten Programmierers.

Fehler in der WebServer-Software selber sind nicht ohne weiteres behebbar. Allerdings sollten hier die Administratoren wie bereits erwähnt auf neu bekanntgewordene Sicherheitslücken achten, die entweder durch Konfigurationsänderungen oder Einspielen von Patches behoben werden können. Die einzige wirkliche Gefahr besteht durch fehlerhafte Konfiguration oder fehlerhafte WebServer-Software. Alles andere sollte durch die vorgeschaltete Firewall vom Server ferngehalten werden. Sollte es jemandem gelingen Kontrolle über die Firewall zu erlangen so sind alle bekannten Angriffe auf den WebServer denkbar.

Die Daten an sich sind nicht vertraulich und somit nicht gegen die Einsicht Dritter zu schützen. Allerdings ist die Korrektheit der beim Benutzer empfangenen Daten evtl. von entscheidener Bedeutung. Die Integrität auf Server-Seite ist durch entsprechende Admininstration schon gut abgesichert und kann durch die Anwendung selber noch zusätzlich gesichert werden. Die Korrektheit der empfangenden Daten ist nur durch Verschlüsselungs- und Signatur-Verfahren zu gewährleisten.

3 Dynamische Inhalte

Das nun folgende Kapitel soll Grundlagen der Client/Server-Architektur und mögliche Konfigurationen aufzeigen, da diese eine wichtige Grundlage für die zu wählende Implementationsmethode darstellen.

Aufgrund der zuvor aufgezeigten Gestaltungsmöglichkeiten soll anschliessend die Eignung des WWW als Implementationsumgebung mitsamt der damit verbundenen Vor- und Nachteile aufgezeigt werden. Hierbei wird auch auf spezielle Eigenschaften des HTTP-Protokolls, damit verbundene Schwierigkeiten bei der Implementation und auf mögliche Auswege eingegangen.

3.1 Client-Server-Architektur

Abgesehen von einigen Einzelplatzanwendungen sind moderne Datenbankanwendungen verteilte Applikationen mit einem zentralen Datenspeicher. Dieser kann zwar auch auf mehrere physische Rechner verteilt sein, ist aber dennoch zentralisiert, da sie der Obhut der Systemadministratoren unterliegt. Dies bildet das typische Szenario für eine Client/Server-Umgebung. Üblicherweise trennt man solche Systeme in drei Hauptkomponenten (Abb. 3.1.1). :

1. Präsentations-Logik;
2. Applikations-Logik (oder auch Anwendungslogik);
3. Datenverwaltungs-Logik

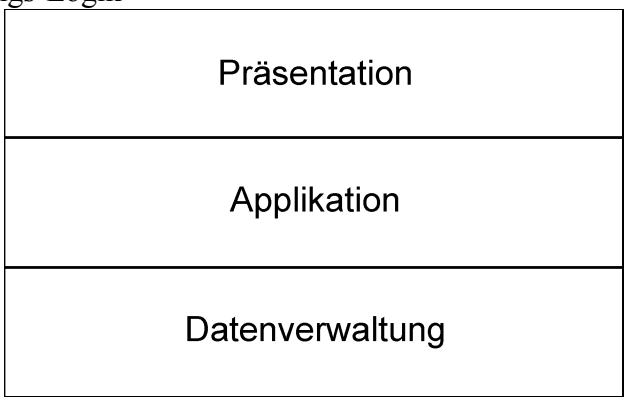


Abbildung 3.1-1 „Klassische Schichten der Client/Server-Architektur“

In einer verteilten Umgebung ist die konzeptionelle Hauptaufgabe darin zu sehen, inwieweit diese Schichten einzelnen Rechnern zugeteilt werden sollen.

Zu den Hauptaufgaben der Präsentationslogik zählen Darstellung und Benutzereingaben, also nahezu die gesamte Benutzerinteraktion. Validation der Eingaben und Ablaufsteuerung der Applikation gehören schon nicht mehr dazu, da sie bereits Teil der Applikationslogik selbst sind.

Der Hauptteil der Präsentationskomponente liegt üblicherweise auf dem Client. Hier erfolgt die Eingabe durch den Anwender und die Darstellung der gewünschten Ergebnisse. Die angefragten Daten werden hier endgültig formatiert und für den Anwender in einer geeigneten Form dargestellt.

Der eigentlich Teil, der eine Datenbankanwendung von einer anderen unterscheidet, ist die Anwendungslogik. Sie ist verantwortlich für die Plausibilitätskontrolle der eingegebenen Daten, die Benutzerführung innerhalb der Anwendung und die Weiterverarbeitung der Daten, eingeschlossen Weitergabe von Anfragen an das DBMS und die entsprechende Verarbeitung der von dort empfangenden Daten.

Die Präsentations- und die Applikationslogik sind recht eng miteinander verzahnt, da die Darstellung von den verarbeiteten Daten abhängt und die Darstellung wiederum die Eingabe mitbestimmt. Eine klare Trennung ist oftmals schwer wenn nicht sogar unmöglich.

Also ist in vielen Konfigurationen ist ein gewisser Teil der Präsentations-Komponente auf dem selben Rechner implementiert auf dem auch die Applikationslogik läuft. Bei WWW-Anwendungen werden auf dem Applikations-Server die HTML-Seiten erstellt, was auch als Teil der Präsentationslogik aufzufassen ist. Diese werden dann auf dem Client im Internet-Browser, abhängig von den lokalen Darstellungseigenschaften, entsprechend umgesetzt und dargestellt. Selbst bei Terminal/Mainframe-Systemen ist die Grenze nicht eindeutig zu ziehen, da bereits vom Hauptrechner Steuerzeichen gesendet werden, die dann die Formatierung auf dem Terminal beeinflussen. Nur in dem Fall, wo die Anwendungslogik vollends auf dem Darstellungsrechner läuft, kann auch die Präsentation komplett vorhanden sein. Obwohl eine klare Trennung wie bereits erwähnt oft nicht zu vollziehen ist, wird im folgenden diese Trennung zur Vereinfachung angenommen.

Die dritte Komponente, die Datenverwaltung, übernimmt das Datenbankmanagementsystem (DBMS). Dieses kann auf dem gleichen Rechner implementiert sein, auf dem der eigentliche Datenspeicher vorhanden ist, könnte aber unter Umständen auch getrennt von diesem sein. In einer verteilten Umgebung ist diese Komponente der Gesamtanwendung immer zentralisiert auf Servern zu finden. Zur Vereinfachung wird im Folgenden angenommen, daß sich sowohl das DBMS als auch der Datenspeicher auf dem selben Server befinden. Zu den wichtigsten Funktionen des DBMS gehören :

- Zugriffskontrolle auf Daten,
- Berechtigungsverwaltung,
- Unterstützung von Transaktionen,
- Wahrung der Konsistenz des Datenspeichers,
- Verwaltung der Dateien und Datenträger,
- Backup und Recovery ermöglichen

Meistens ist die Anwendungslogik so programmiert, daß spezifische Vorteile des eingesetzten DBMS ausgenützt werden oder Nachteile umgangen werden. Da sich der Hersteller eines DBMS nicht nur an Standards hält, sondern sich von Konkurrenten durch Innovation abheben will, ist eine Umstellung auf ein anderes DBMS meistens mit neuem Programmieraufwand verbunden, da die Anwendungs-Programmierer oft die Vorteile des DMBS ausnützen wollen. Sollte dieses nicht der Fall sein gibt es oft sowieso meistens keinen Grund eines DBMS-Wechsels.

Eine Unabhängigkeit vom eingesetzten DBMS kann man erreichen indem man eine Datenbank-Middleware benutzt, die einen standardisierten Satz an Befehlen von der Anwendung erhält und diese ggf. in spezielle Befehle für das eingesetzte DBMS umsetzt. Dies geht jedoch oft zu Lasten der Performance und für die Programmierung steht ein eingeschränkter Satz an Befehlen zur Verfügung, so daß hier innerhalb der Anwendungslogik Teile des DBMS mit übernommen werden müssen. Beispielsweise unterstützt nicht jede Middleware die Übergabe von Transaktionsbefehlen an jedes DBMS, da diese nicht dem normalen Standardsprachsatz von SQL (Structured Query Language) entsprechen. Der bekannteste Vertreter im Bereich der Datenbank-Middleware ist wohl ODBC (Open Database Connectivity) von Microsoft. ODBC basiert auf den Call-Level-Interface Spezifikationen von X/Open und ISO/IEC. Den Mittelpunkt bildet der ODBC-Treiber-Manager. Eine Anwendung sendet seine Daten an ihn, dieser gibt sie dann weiter an den entsprechenden Treiber für das benutzte DBMS und leitet die Ergebnisse dann wieder zurück an die Anwendung (Abb. 3.1.2)

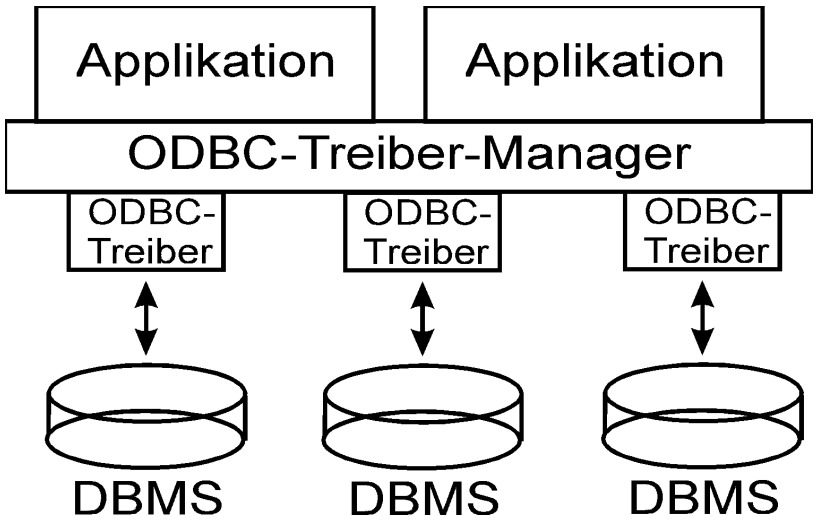


Abbildung 3.1-2 „ODBC-Architektur“

Aufgrund der o.g. gezeigten Aufteilung in drei Komponenten gibt es auch drei vereinfachte unterschiedliche Möglichkeiten der Konfiguration einer verteilten Anwendung. Diese richtet sich maßgeblich nach dem Implementationsort der Applikationslogik. Die möglichen Kombinationen sind

a. client-zentrierte Verarbeitung
b. server-zentrierte Verarbeitung
c. verteilte Client/Server-Verarbeitung

a. Client-zentrierte Verarbeitung

Wie die Bezeichnung schon verrät steht bei dieser Konfigurations-Variante der Client im Vordergrund. Auf dem Client befinden sich hier sowohl die Präsentations- als auch die Applikations-Logik. Die Anwendung auf dem Client kontaktiert für Abfragen hier das DBMS auf dem Server, welches den Part der Datenverwaltungslogik übernimmt, typischerweise über die Abfragesprache SQL (Structured Query Language). Wie bereits weiter oben erwähnt kann der Zugriff hier auch über Datenbank-Middleware geschehen.

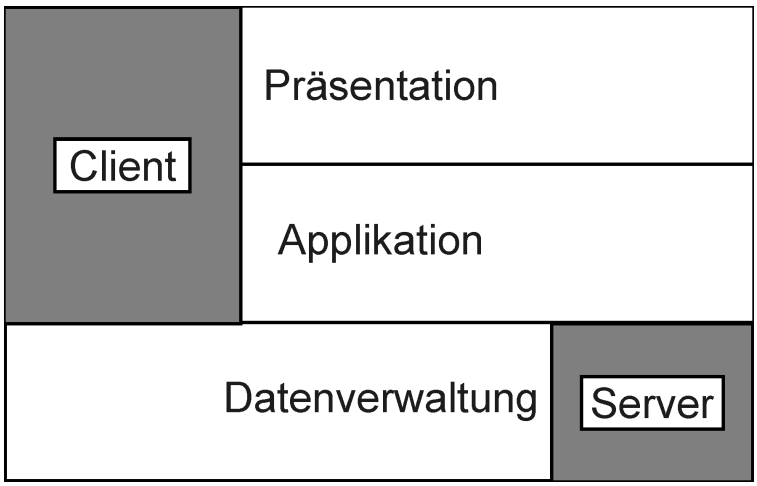


Abbildung 3.1-3 „Client-zentrierte Verarbeitung“

Diese Art der Konfiguration setzt verhältnismäßig schnelle Client-Computer voraus, wohingegen der Server selber relativ gering dimensioniert sein kann. In diesem Fall spricht man auch von einer “fat client/thin server”-Architektur.

Als Nachteile dieser Konfigurations-Strategie sind anzusehen, daß

- die Datenbankabfragen ggf. sehr große Ergebnismengen vom DBMS ergeben können, die dann zu einer erhöhten Netzlast führen. Ab einer gewissen Userzahl ist das Netz überlastet und bietet nicht mehr die gewohnten bzw. gewünschten Antwortzeiten.

- jeder Client innerhalb der Anwendung eine Verbindung zum DBMS aufbauen muß. Jede dieser Verbindungen muß innerhalb des DBMS verwaltet werden, was sowohl Prozessorzeit als auch Speicher auf dem Server in Anspruch nimmt. Beispielsweise verbraucht der Microsoft SQL-Server pro bestehender Verbindung 37 KByte, was schon relativ wenig im Vergleich zu anderen DBMS ist.

- bei einem Update der Anwendung diese auf allen Clients aktualisiert werden muß. Dies trägt einen hohen Adminstrationsaufwand mit sich und führt ggf. zu Inkonsistenzen innerhalb der Anwendung wenn nicht alle Clients in gleicher Weise rechtzeitig umgestellt wurden.

- die Flexibilität bei evtl. notwendigen Ausbaustufen nicht gegeben ist.

Weiterhin bleibt bei dieser Implementation die effektive Nutzung des vorhandenen Datenbestandes außen vor. Um eine noch bessere Auswertung der Daten zu erreichen, bedienen sich viele Unternehmen sog. Data-Mining-Technologien. Diese Techniken extrahieren Erkenntnisse, Zusammenhänge und Trends, die normalerweise nicht aus dem Datenbestand ohne großen Aufwand zu erzielen sind. Eine Analyse der Daten kann aber nur zentral auf dem Datenbestand erfolgen. Dies würde im Falle einer client-zentrierten Konfiguration bedeuten, daß sich trotzdem ein Prozeß um die Analyse des Datenbestandes kümmern müßte. Besser wäre hier eine Integration des DataMinings in eine zentrale Anwendung, so daß lediglich aufgrund der geänderten Datensätze im Datenbestand die Analysewerte korrigiert werden müssen. Vor allem für die Trenderkennung ist dies wichtig.

b. Server-zentrierte Verarbeitung

Im Gegensatz zur client-zentrierte Konfiguration steht die im folgenden beschriebene. Hierbei ist die Anwendungs-Logik zusammen mit der Datenverwaltungslogik auf einem einzigen Server implementiert. Die Clients übernehmen lediglich die Komponente der Präsentation. Daher müssen die Clients nicht so performant sein wie in der client-zentrierten Konfiguration. Die Last der Anwendungslogik vieler Clients aus der client-zentrierten Verarbeitung wird hierbei auf einen einzelnen Server verlagert, wodurch dessen Belastung deutlich stärker wird. Deshalb muß der Server sehr leistungsfähig sein, weshalb man hier auch von einer “thin client/fat server”-Konfiguration spricht.

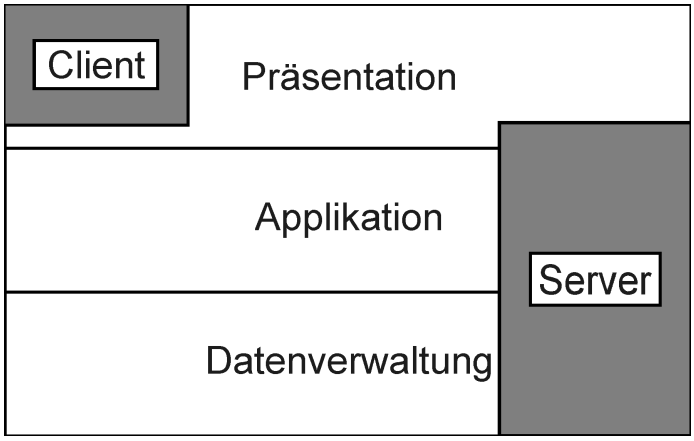


Abbildung 3.1-4 „Server-zentrierte Verarbeitung“

Auch bei dieser Systemgestaltung ist es möglich die Verbindung von Anwendungs-Logik zum DBMS mittels Middleware zu implementieren, was dann aber zu erhöhtem Ressourcenbedarf auf dem Server führt.

Die server-zentrierte Konstellation ist freier skalierbar als die client-zentrierte und kann ggf. umkonfiguriert werden, so daß Anwendungs- und Datenverwaltungs-Logik auf getrennten Rechnern ablaufen. Dieses würde dann die nachfolgend beschriebene Konfiguration darstellen.

verteilte Client/Server-Verarbeitung
Wie bereits angedeutet ist die Trennung von allen drei Komponenten - Präsentation, Anwendung und Datenverwaltung - auf jeweils eigene Rechner die letzte mögliche Systemgestaltung innerhalb einer Client/Server-Umgebung. Hierbei findet eine Lastverteilung auf alle Rechner statt, so daß sowohl die Performanz der Anwendung verbessert als auch die Netzbelastung verringert werden kann.

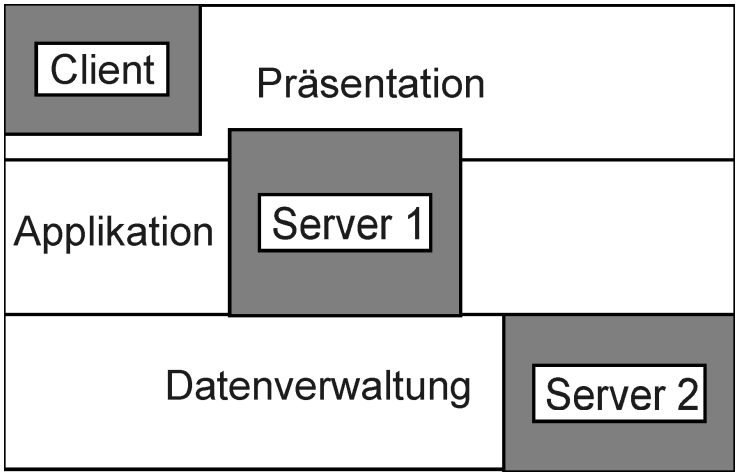


Abbildung 3.1-5 „Verteilte Client/Server Verarbeitung“

Die einzelnen Komponenten können ausgetauscht werden ohne die anderen jeweils zu beeinflussen. Dies würde allerdings voraussetzen, daß die benötigten Schnittstellen zwischen den Teilkomponenten standardisiert sind. Die Wahrscheinlichkeit, daß bei einer solchen Konfiguration Interdependenzen zwischen den Einzelkomponenten auftreten, die sich negativ auf das System auswirken, ist als gering anzusehen. Eine Unverträglichkeit über eine Standard-Schnittstelle auf einen anderen Rechner ist unwahrscheinlicher als wenn zwei evtl. unabhängig voneinander entwickelte Systeme beim Zugriff auf gemeinsam genutzte Ressourcen einen Konflikt hervorrufen. Sollte ein Update für eine der Teilkomponenten vorgenommen werden müssen ist dieses meist reibungsloser möglich.

Als deutlicher Nachteil ist der erhöhte Entwicklungsaufwand zu sehen, da mit einer komplexeren Systemgestaltung meist auch die Komplexität der Anwendung und der Schnittstellen zunimmt. Mit dem Entwicklungsaufwand steigen natürlich auch die Kosten einer solchen Implementation.

Die dreistufige Client/Server-Verarbeitung ist als die erste Stufe einer nach oben unbegrenzten Skalierung zu sehen. Für spezielle Teile der Anwendung können auch jeweils eigenständige Server verantwortlich sein. Beispielsweise könnte es in einem Buchhaltungssystem einen Server geben, der nur für die Buchungen innerhalb der Konten abgestellt ist und den Monats- oder Jahresabschluß durchführt. Diese Aufgaben würden von dem Server, der die Benutzer-eingaben entgegennimmt an den Buchungsserver weitergereicht werden, so daß die Antwortzeiten für die Anwender nicht zu groß sind. Dieses Szenario läßt sich je nach gegebenen Anforderungen ausdehnen.

3.2 Das WorldWideWeb als Anwendungsumgebung

Das WorldWideWeb stellt per se schon eine verteilte Umgebung dar. Clients aus allen Teilen der Welt greifen auf Server an anderen Standorten zu. In den Anfängen wurden zumeist statische Seiten vom Benutzer über die grafische Benutzeroberfläche (englisch “Graphical User Interface” = GUI) von einem Server angefordert, die dann auf dem Client dargestellt wurden. Hierbei agierte der Web-Server mehr wie ein Datei- als ein Anwendungs- oder Datenbankserver.

Durch die steigende Zahl der Informationen und der Benutzer des Internets war sowohl die Informationsdichte vom Anwender nicht mehr zu überschauen, als auch die Dokumente auf einem Server vom jeweiligen Anbieter nicht mehr zu verwalten. Daher wurde der Bedarf an dynamisch erzeugten Seiten und damit von Anwendungen zunehmend größer.

Das WWW bietet durch seine Standardisierung und Systemunabhängigkeit eine gute Plattform für die Entwicklung von Client/Server-Anwendungen. Zu den wichtigsten Standards zählen hierbei

- die eindeutige Identifikation von Daten und Anwendungen durch die URL (Uniform ressource location);
- die einheitliche hypermediale Seitenbeschreibungssprache HTML;
- ein Kommunikationsprotokoll auf Netzwerkebene in Form von TCP/IP;
- HTTP als Protokoll zum Übertragen der Dokumente, der binären Datenströme und der damit verbundenen Informationen, speziell zwischen Internet-Browser und WWW-Server;

Durch diese Standards offeriert das WWW viele Vorteile, die auch Unternehmen recht schnell für sich entdeckten. Die Vorzüge gegenüber sonst üblichen Anwendungen sollen in der folgenden Übersicht aufgezeigt werden :

	Client/Server-Anwendungen	**WWW-Anwendungen**
GUI	- hardwarespezifische Programmierung erforderlich	+ durch WWW-Browser bereitgestellt + einheitliche GUI für alle Anwendungen => geringerer Schulungsaufwand - einfache GUI bei Beschränkung auf HTML - uneinheitliche Darstellung
Standardisierung	- selten standardisiert, sondern Eigenentwicklungen oder speziell angepaßt (customtailored)	+ basierend auf vielen, sehr breit unterstützten Standards, wie z.B. HTML, URL, HTTP, TCP/IP, FTP, SMTP u.ä.

	Client/Server-Anwendungen	WWW-Anwendungen
Unterstützung heterogener Systeme	- ggf. Abhängigkeit von einem Hersteller	+ weitgehend Unabhängigkeit auf Client-Seite und in Teilen auch Server-Seite, da sowohl Browser als auch Server für nahezu jedes Betriebssystem verfügbar sind
Administration	- hoher Aufwand bei Updates durch Installation auf Clients	+ problemloses Hinzufügen neuer Clients. + weniger Administrations-aufwand durch zentrales Update - ggf. größerer Schaden durch zentrales Update

Tabelle 3.2-1 "Vorteile von WWW-Anwendungen gegenüber Client/Server-Anwendungen"

Durch die in Kapitel 3.1 und die in Tabelle 3.2-1 dargestellten Vorzüge ist das WWW sehr gut für die Entwicklung verteilter Anwendungen geeignet. Vor allem die Unterstützung heterogener Systeme durch die Standardisierung ist in diesem Zusammenhang hervorzuheben.

Im nächsten Abschnitt sollen nun speziell Aspekte einer Implementation im WWW-Umfeld mit Schwerpunkt auf Datenbankaspekten erläutert werden.

3.2.1 Das HTTP-Protokoll

Das WWW basiert zum einen auf dem HTML-Standard, in dem die Seiten beschrieben werden, zum anderen auf dem HTTP-Protokoll, welches für die Kommunikation zwischen Client und Server verantwortlich ist. Da das HTTP-Protokoll für die Entwicklung insbesondere von Datenbankanwendungen im Internet wichtige Eigenschaften besitzt sollen diese im Folgenden dargestellt werden.

Eine grundlegende Eigenschaft des HTTP-Protokolls ist, daß es zustandslos ist und nicht sitzungsorientiert arbeitet. Bei einem nicht-sitzungsorientierten Protokoll fordert der Client eine Datei bei einem Web-Server an, diese Anforderung wird vom Server bearbeitet und an den Client zurückgeliefert. Aus dem Inhalt der gelieferten Datei können dann neue Anforderungen entstehen, z.B. wenn in einem HTML-Dokument eine Grafik eingebunden wird. Hierbei bekommt der Client die HTML-Datei und versucht diese darzustellen. Während der Abarbeitung wird dann festgestellt, daß zur kompletten Darstellung noch eine Grafik-Datei fehlt, die dann als neue Anfrage an einen Server geht. Dieser Server muß nicht zwangsläufig derjenige sein von dem die HTML-Datei kam.

Für die Übertragung zwischen Client und Server wird eine Verbindung aufgebaut, die nach erfolgreicher Übertragung oder nach Zeitüberschreitung (Time-Out) abgebaut wird. Dieses stellt den Sachverhalt der nicht-sitzungsorientierten Kommunikation dar, da keine logische Verbindung gehalten wird nachdem die Daten übertragen worden sind.

Anders verhält sich dieses z.B. bei Telnet oder FTP, wo der Anwender die Sitzung und damit die Verbindung normalerweise explizit beenden muß.

Viel entscheidender für eine WWW-Anwendung ist allerdings die Tatsache, daß das HTTP-Protokoll zustandslos ist. Dies bedeutet, daß das HTTP-Protokoll selber keinerlei Informationen beinhaltet, wann zu welchem Server welche Daten gesendet wurden und welche empfangen wurden. Auch auf der Serverseite werden solche Daten nicht bereitgehalten. Die Zustandslosigkeit hängt direkt mit der o.g. Eigenschaft der Nicht-Sitzungsorientierung zusammen.

Für eine Anwendung ist es aber mitunter recht wichtig, daß Informationen über die vorherigen Aktionen zur Verfügung stehen. Insbesondere bei Datenbankanwendungen könnte es zu Problemen führen wenn dieses Informationen nicht zur Verfügung stehen. Wie bei normalen Datenbankanwendungen, die innerhalb eines Unternehmens zur Verfügung stehen, kann auch bei Internet/Intranet-Anwendungen die Notwendigkeit bestehen die Eingabe auf mehrere Bildschirme zu verteilen. Dies kann verschiedene Gründe haben. Zum einen wegen der besseren Übersicht, zum anderen weil die Dateneingabe in einer Maske für den Aufbau der Folgemaske entscheidend ist. Manchmal müssen aber bereits nach der ersten Maske Aktionen auf der Datenbank stattfinden, die allerdings erst nach der erfolgreichen Abarbeitung der zweiten Maske endgültig vollzogen werden dürfen. Um bei Fehlern oder Abbruch durch den Benutzer trotzdem eine konsistente Datenbank zu behalten, wurde für Datenbanken das sog. Transaktions-Konzept entwickelt. Eine Transaktion besteht dabei aus mehreren atomaren Aktionen auf der Datenbank. Hierbei gilt die Devise ”Alles oder nichts”, d.h. die Änderungen werden endgültig erst nach erfolgreichem Abschluß der letzten Aktion vollzogen. In normalen Datenbanksystemen ist dies möglich, da hier der Server einen Status für jeden angemeldeten Client halten kann und auf diesen bei der nächsten Anforderung zugreifen kann. Aufgrund der beiden o.g. Eigenschaften von HTTP ist dieses bei Internet-/Intranet-Anwendungen nicht ohne weiteres möglich. Lösungsmöglichkeiten hierzu müssen also außerhalb von HTTP gefunden werden.

Bezug nehmend auf die Ausführungen in Kapitel 3.1 konzentriert sich der nächste Teil dieser Arbeit vor allem auf die Frage auf, wieviel Last der Applikation auf die Clients mit welchem Aufwand verlagert werden kann (s. Abb. 3.1.1 und 3.1.2). Im Folgenden wird beim Zugriff auf die Datenbank davon abstrahiert, ob sich das DBMS und der Datenspeicher auf einem Rechner befinden und ggf. sogar auf dem gleichen Server wie die Anwendung implementiert sind. Zur Vereinfachung geht diese Arbeit davon aus, daß serverbasierte Applikations-Logik, DBMS und der Datenspeicher auf dem selben Rechner installiert sind.

Bei Anwendungen, die einer Vielzahl von Anwendern zur Verfügung stehen, kann der Server zum Engpaß werden. Im Internet sind die meisten Anwendungen frei zugänglich. Das fängt an mit Suchmaschinen für das WorldWideWeb und geht über Fahrplanauskünfte und Stadtplandienste bis hin zu weitaus komplizierteren Anwendungen. Ein entscheidendes Kriterium für die Wahl der Implementation ist also auch, inwieweit es möglich ist den Server zu entlasten und Vorverarbeitung oder sogar einen Großteil der eigentlichen Anwendung auf dem Client stattfinden zu lassen. Bei der heutigen Leistungsfähigkeit der als Clients eingesetzten Computer sollte man diese Rechenleistung mit in die Konzeption einbeziehen.

Als einfachstes Beispiel für Vorverarbeitung sei hier die Plausibilitätsprüfung von Eingaben genannt. Kann diese schon auf dem Client durchgeführt werden spart dieses Anfragen an den Server, der diese ansonsten prozessieren und den entsprechenden Fehler ggf. zurückmelden würde.

Eine Verarbeitung vor Ort auf dem Client würde aber nicht nur den Server entlasten, sondern auch die Antwortzeit für den Anwender verkürzen, da die zeitaufwendige Kommunikation über das Netzwerk entfallen würde und er nahezu ohne Verzögerung eine Reaktion der Anwendung hätte, daß eine weitere oder korrigierte Eingabe nötig ist.

3.2.2 Serverseitige Verarbeitung

Bei dieser Form der Datenbank-Anbindung erfolgt normalerweise keinerlei Informations-verarbeitung auf dem Client. Der Web-Client dient lediglich zur Dateneingabe und -ausgabe, welches durch HTML-Dokumente realisiert wird. Die eigentliche Applikation liegt auf dem Server und wird dort ausgeführt.

3.2.2.1 Parameterübergabe

Wie bereits in Kapitel 3.2.1 erwähnt werden im HTTP-Protokoll keinerlei Informationen bzgl. der Historie vorheriger Anforderungen gespeichert. Bei Anwendungen die serverbasiert laufen ist ein gemeinsamer Ablauf allen Möglichkeiten gleich : Die Anfrage erreicht den Web-Server, ein Prozeß zur Bearbeitung wird instanziiert, die Anfrage wird verarbeitet, die Resultate werden an den Client zurückgeschickt und die vorher erzeugte Instanz wird wieder aus dem System entfernt. Da hierbei keine Möglichkeit besteht Informationen für die weitergehenden Schritte innerhalb der Anwendung bereitzuhalten, müssen diese Daten auf andere Weise bereitgestellt werden. Diese Daten müssen mit den aufbereiteten Daten an den Client gegeben werden, damit diese dann bei der nächsten Übertragung wieder vorhanden sind. Für die Speicherung auf dem Client gibt es verschiedene Möglichkeiten :

a. versteckte Formularfelder, über die die Eingabe des vorhergehenden Formulars mitgereicht werden
b. Parameterübergabe über Links, die dynamisch erzeugt wurden
c. Cookies, die auf Clientseite gespeichert werden und dann bei der nächsten Übertragung den neuen Informationen hinzugefügt werden.

a. Versteckte Formularfelder

Bei den versteckten Formularfeldern wird mit dem Übersenden der Informationen das nächste Formular erzeugt. Dieses beinhaltet sowohl neue, für den Anwender sichtbare Eingabefelder als auch versteckte Felder, die lediglich dazu dienen, die zuvor erfaßten Daten oder Zwischenergebnisse zu beherbergen und somit für den nächsten Verarbeitungsschritt zur Verfügung zu stehen. Die Übergabe der Daten erfolgt hierbei meistens über die Methode, die als “Post” bekannt ist. Die übergebenen Daten werden anschließend über die Standardeingabe STDIN der Anwendung zum Zugriff zur Verfügung gestellt.

Der Nachteil dieser Übergabetechnik besteht darin, daß die Menge der übertragenen Daten mit der Zahl der erfaßten Felder wächst und somit ggf. eine Vielzahl von Daten unnötigerweise über das Netzwerk verschickt werden. Außerdem müssen die Daten nach jeder Übertragung sowohl auf dem Client als auch auf dem Server in irgendeiner Form verarbeitet werden. Auf dem Server müssen die Daten wieder in das Folge-Formular codiert werden und dieses muß

dann auf dem Client entsprechend interpretiert werden. Der Aufwand auf der Client-Seite und im Netzwerk könnte hier verringert werden, indem man die Daten auf dem Server beispielsweise in einer Datenbank zwischenspeichert und lediglich eine Sitzungsnummer o.ä. in dem Formular versteckt. Auf die in dem Formularfeld versteckte Sitzungsnummer wird dann bei der nächsten Anfrage an den Web-Server wieder zugegriffen. So wird zwar der Datenverkehr verkleinert, jedoch zu Lasten von mehr Verarbeitung und Verwaltung auf dem Server.

Durch die Einsehbarkeit des HTML-Codes ist es für jeden User möglich auch die versteckten Felder sichtbar zu machen und ansatzweise zu sehen was an Verarbeitung auf dem Server stattfindet. Prinzipiell könnte jedermann sich den Quellcode anschauen, modifizieren und diese modifizierte Struktur an den Server als Anfrage schicken. Eine sicher programmierte Anwendung kontrolliertt hierfür entweder die Gültigkeit der benötigten Werte, verarbeitet auch nur die erwarteten Parameter und/oder kontrolliert die mit dem HTTP-Protokoll übergebene Variable „HTTP_REFERRER“, die als Inhalt die URL der Seite enthält von der der Aufruf erfolgt. Hierdurch kann sichergestellt werden, daß wirklich die Originalseite vom Server vorlag und nicht durch den Benutzer zunächst geladen, lokal gespeichert und modifiziert wurde und diese modifizierte Version dann zur Dateneingabe benutzt wurde. Eine Täuschung dieser Variablen ist zwar nicht unmöglich, aber zumindest mit dem Wissen des Großteils der Anwender im Internet nicht ohne weiteres machbar.

Mit o.g. Vorgehen würde es beispielsweise auch möglich sein, Eingabe-Prüfungen durch JavaScript zu umgehen. Aus diesem Grunde ist es unerläßlich die Eingaben auf dem Server nochmals auf Plausibilität zu prüfen.

b. Parameterübergabe über Links

Bei der Parameterübergabe durch Links - auch als Übergabemethode “Get” bekannt - verhält es sich ähnlich wie bei den versteckten Formularfeldern. Hier wird die Folgeseite auch dynamisch erzeugt. Allerdings werden hier in aufrufenden Links die zuvor übertragenen Werte mit eingebunden. Es besteht der gleiche Nachteil wie oben bei den versteckten Formularfeldern: Höhere Netzlast oder höhere Belastung des Web-Servers. Hinzu kommt noch, daß die Anzahl der Zeichen, die über diese Möglichkeit übergeben werden können, limitiert ist. Sie variiert von Client zu Client und Server zu Server, aber im allgemeinen liegt die Obergrenze bei 1024 Byte. Dies liegt daran, daß die Daten dem aufgerufenen Programm über die Umgebungsvariable „QUERY_STRING“ zur Verfügung gestellt werden. Außerdem ist eine aufgerufene Seite mit solch einer langen URL erstens nicht sehr einprägsam und verwirrt ggf. den Anwender und zweitens wird bei vielen Web-Servern die aufgerufene URL in den Log-Dateien festgehalten. Mehrere übertragene Formulare können hier also zu einem schnell wachsenden Log-File führen. Ein Vorteil dieser Art ist allerdings, daß solche Links auch als Lesezeichen gespeichert werden können und somit z.B. Standard-Abfragen für den Anwender erleichtert werden. Dies verleitet allerdings User auch dazu Login-Informationen als Bookmark abzuspeichern, was bei sicherheits-sensiblen Anwendungen zweifellos ein Risiko darstellt.

Durch die Codierung in der URL sind auch alle übergebenen Variablen mit Namen für den User leichter einsehbar, modifizierbar und damit ggf. für Angriffe auf die Anwendung ausnutzbar. Prinzipiell gilt auch hier, daß alle auf dem Server zur Verarbeitung empfangenden Parameter auf gültige Werte überprüft und nur erwartete Variablen verarbeitet werden sollten.

c. Cookies

Cookies wurden ursprünglich von Netscape eingeführt, um genau die oben beschriebenen Probleme Zustandslosigkeit und doppelte Übertragung der Daten zu umgehen. Hierzu kann auf dem Client ein sog. Cookie gesetzt werden. Der Inhalt kann durch die Anwendung selber gesetzt werden und ist frei definierbar. Ebenso gibt es ein Verfallsdatum. Wird dieses überschritten wird der Cookie auf dem Client gelöscht. Allerdings bleibt er für eine laufende Sitzung des Browsers aktiv und verfällt erst beim Beenden des Browsers. Die Daten dieser Cookies können nur von Seiten oder Skripten von dem Server gelesen werden von dem sie auch geschrieben wurden. Dies ist bei verteilten Anwendungen, die intern von einem Server auf einen anderen verweisen, evtl. mit Problemen verbunden.

Leider können die Cookies auch dazu mißbraucht werden, daß man Informationen über den Anwender speichert und damit ggf. ein Profil desjenigen anlegt. Da der Inhalt der Cookies frei setzbar ist können hierin auch Informationen codiert werden, die der Anwender nicht als Profildaten seiner selbst ansieht. Beispielsweise können das Datum des letzten Besuchs oder die zuvor besuchte Seite gespeichert werden und damit Rückschlüsse auf sein Internet-Verhalten und seine Interessen gezogen werden.

Diese Tatsache des möglichen Mißbrauchs hat dazu geführt, daß viele Anwender Cookies nicht mehr akzeptieren. Früher wurden die Cookies ohne Zutun des Anwenders im Hintergrund einfach gespeichert. Die Möglichkeit der Zweckentfremdung hat aber bei vielen Usern dazu geführt, Cookies per se zu verbieten. Dieses ist entweder im Internet-Browser ausstellbar oder durch kleine Zusatzprogramme erreichbar. Viele Cookies können auch bei einigen Anwendern zu Ärger führen, wenn bei jedem Versuch ein Fenster erscheint mit dem die Akzeptierung des Cookies bestätigt werden soll. Dies führt dazu, daß die entsprechende Seite oder Anwendung für den User unattraktiv wird. Man kann und sollte also bei einer WWW-basierten Anwendung nicht davon ausgehen, daß Cookies immer zur Verfügung stehen und akzeptiert werden. Dieses ist auch als Hauptnachteil dieser Möglichkeit zu sehen.

3.2.2.2 Verarbeitung

Bei genauerem Betrachten des Aufbaus der Server-Struktur sieht man, daß es zwei mögliche Punkte gibt, an denen der Applikationscode ausgeführt werden kann: Außerhalb des Web-Server-Adreßraums oder innerhalb desselbigen.

Beide Lösungen bieten verschiedene Vor- und Nachteile :

Webserver-externe Verarbeitung:

Hierbei findet die Kommunikation zwischen Server und Applikation über die CGI-Schnittstelle (Common Gateway Interface) statt. Die CGI-Schnittstelle ist eine Standard-Schnittstelle, die von nahezu allen Web-Servern unterstützt wird. Ein Umstieg auf einen anderen Web-Server oder das Update auf eine aktuellere Version eines Servers sollten also keine Probleme bereiten, da CGI ein Standard ist.

Die Kommunikation zwischen WebServer und dem CGI-Programm findet über die Standard-Ein-/Ausgabe sowie über Umgebungsvariablen statt. Von daher ist CGI sehr flexibel, da es unabhängig von der gewählten Sprache des externen Programms ist. Die Programmiersprache muß lediglich fähig sein,

1. in die Standardausgabe zu schreiben,
2. von der Standardeingabe zu lesen und
3. Umgebungsvariablen einlesen zu können.

Da diese Bedingungen nahezu von jeder Programmiersprache erfüllt werden steht eine große Breite an Sprachen für die Implementation zur Verfügung. Eigentlich kann jeder Programmierer, der eine Programmiersprache beherrscht, eine CGI-Applikation schreiben, sofern er sich mit dem Mechanismus der CGI-Kommunikation und dem HTTP-Protokoll auskennt. In Bezug auf die Projektplanung und die Human Ressources ist dieses ein Vorteil der nicht außer acht gelassen werden darf, da der Markt an Programmierern recht dünn ist und eine Festlegung auf eine Sprache die Auswahl aus diesem Markt noch zusätzlich drastisch verkleinern würde. Bei einer CGI-Anwendung ist es so, daß das an den Anwender zurückgelieferte Internet-Dokument in einem Programm als Ausgabe (an die Standardausgabe) mittels print-Befehlen erzeugt wird. Dieses macht es relativ schwierig evtl. angedachte Formatierungs- oder Design-Änderungen in das Programm einzupflegen, da es recht unübersichtlich ist.

Wenn man die Implementation einer WWW-Anwendung mittels CGI in Betracht zieht, sollte man sich jedoch vorher Gedanken über die gewünschte Performanz machen. Interpretierte Skript-Sprachen wie Perl, awk oder Tcl bieten den Vorteil einer schnellen und leichten Programmierung, wohingegen kompilierte Sprachen wie C/C++ einen deutlichen Vorteil in der Geschwindigkeit haben, da sie nicht mehr interpretiert werden müssen. Bei größeren und rechenintensiven Anwendungen ist auf jeden Fall eine kompilierte Sprache zu wählen, während bei kleineren Applikationen Skript-Sprachen zu bevorzugen sind. Dies ist damit begründbar, daß bei CGI die Performanz alleine schon durch den Aufruf des externen Programms leidet und bei kurzen Berechnungen ein kompiliertes Programm diesen Aufwand nicht lohnt. Der Aufruf eines CGI-Programms ist so ressourcenverbrauchend, weil diese Anfrage vom Client an den Web-Server gesendet wird. Dieser erkennt, daß es sich um ein externes Programm handelt und ruft dieses auf. Für jeden Aufruf wird hierbei ein eigener Prozeß generiert. Diese Instanziierung braucht eine gewisse Zeit und der Prozeß selber benötigt einen Adreßraum in dem er laufen kann. Die durch das Programm erzeugte Ausgabe wird an den Web-Server zurückgeliefert, der diese dann an den anfragenden Client zurückgibt. Nach Beendigung des CGI-Programms wird der Prozeß wieder gelöscht und der Speicher freigegeben.

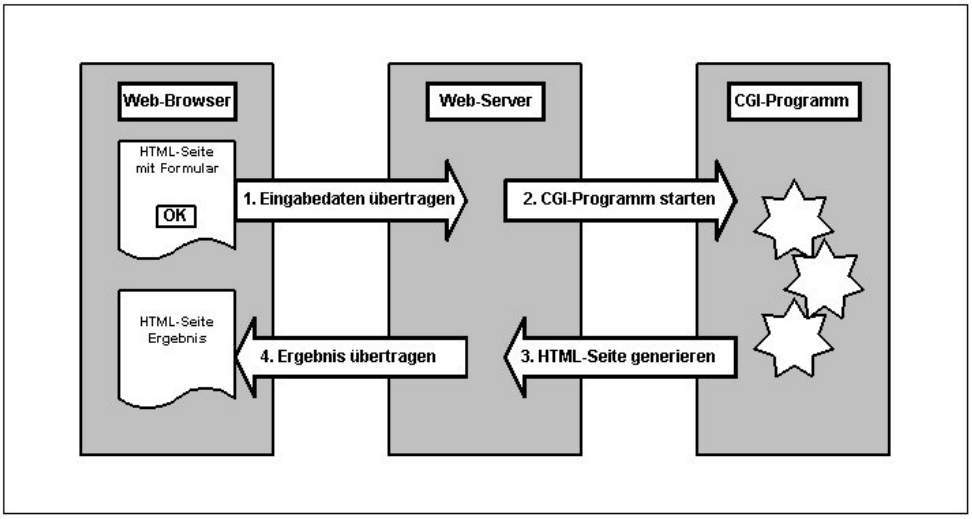


Abbildung 3.2-1 „Auftragsfluß bei Aufruf über CGI“

Dieses Verfahren benötigt also sehr viel Verwaltungsaufwand und zusätzliche Ressourcen. Bei vereinzelt auftretenden Anfragen stellt dieses kein großes Problem dar. Wenn jedoch sehr viele Anfragen auf diesem Server parallel laufen wird der Rechner durch die Prozeßverwaltung und durch den erhöhten Speicherbedarf stark belastet. In Umgebungen mit einem erwarteten hohen Aufkommen an Anfragen ist CGI also nicht sehr geeignet .

Als Vorteile von CGI sind also folgende Punkte hervorzuheben :

- hohe Flexibilität, da nahezu jede Sprache für CGI geeignet ist.
- hohes Maß an Portabilität, da eine Vielzahl von Web-Servern CGI unterstützen und CGI-Applikationen dermaßen verbreitet sind, daß auch zukünftige Web-Server diesen Defacto-Standard unterstützen werden.
- einfache Schnittstelle

Demgegenüber stehen die Nachteile :

- pro Aufruf eigener Prozeß => viel Ressourcenverwaltung durch das Betriebssystem, viele benutzte Ressourcen bei hoher Last
- Sitzungs- und Transaktionsmanagement bei DB-Anwendungen schwierig
- Mit jedem Aufruf erneut Verbindung zum DBMS aufbauen
- Rechteverwaltung im DBMS kann nicht gut genutzt werden und muß ggf. selber innerhalb der CGI-Programme implementiert werden
- Unübersichtlicher Code

Webserver-interne Verarbeitung:

Wie bereits erwähnt kann die dynamische Erzeugung von Formularen oder Ergebnisseiten auch im Adreßraum des Web-Servers stattfinden. Hierfür wird von vielen Herstellern eines Web-Servers eine API zur Verfügung gestellt. Hierüber können, meist in C/C++, Anwendungen programmiert werden, die die Funktionalität des Servers erweitern. Im Windows-Bereich bieten die großen Konkurrenten Microsoft und Netscape jeweils eine solche API für ihren Web-Server an, die ISAPI (Microsoft) und die NSAPI (Netscape). Da auf dem vorliegenden System ein Microsoft-Server installiert ist konzentriert sich der folgende Teil auf die ISAPI. Server-APIs haben gegenüber CGI den Vorteil, daß sie deutlich performanter sind. Mit dem ersten Aufruf der API wird diese als DLL in den Speicher geladen und steht von dem Zeitpunkt an dort zur Verfügung, ohne erneut geladen werden zu müssen. Durch Aufrufen der API wird lediglich ein neuer Thread innerhalb des Web-Server-Prozesses erzeugt. Dadurch findet die Instanziierung innerhalb des Adreßraums des Web-Servers statt und erzielt dadurch die Performanz-Vorteile gegenüber CGI, da der Verwaltungs-Overhead deutlich geringer ausfällt.

Die Instanziierung im Adreßraum des Web-Servers birgt allerdings auch das Risiko, daß durch eine fehlerhaft programmierte API-Anwendung der gesamt Web-Server in Mitleidenschaft gezogen wird und somit keinerlei Dienste im Internet angeboten werden können.

In Bezug auf ISAPI unterscheidet man zwischen Anwendungen und Filtern. Die ISAPI-Anwendungen sind DLLs, die im Funktionsumfang in etwa der von CGI-Anwendungen entsprechen. ISAPI-Filter sind ebenfalls DLLs, die sowohl vom Server aktiviert werden als auch diesen beeinflussen können. Die Aktivierung des Filters hängt von der Konfiguration des Web-Servers ab. Als häufigstes Aktivierungskriterium gilt allgemein die Datei-Erweiterung der angeforderten Datei. Der Web-Server gibt bei Übereinstimmung die Anfrage an den zu aktivierenden Filter weiter, der dann das Ergebnis abliefert. Dieses Ergebnis wird dann an den Client zurückgeliefert. Active Server Pages (ASP) von Microsoft stellen einen solchen ISAPI-Filter dar. Hierbei wird aufgrund der Dateiendung erkannt, daß es sich um eine ASP-Anwendung handelt und die Kontrolle an die entsprechende DLL übergeben. Die ASP-DLL tut nichts weiter, als die angeforderte Datei einzulesen, zu parsen und entsprechend zu verarbeiten. In ASP-Files können in normalen HTML-Files VisualBasic- oder JScripts abgearbeitet werden. Diese Skripte werden in die normalen HTML-Formatierung mit eingebaut, welches die Lesbarkeit und Wartbarkeit im Gegensatz zu CGI-Anwendungen erhöht. Das Ergebnis der Prozessierung wird dann an Stelle des eigentlichen Skripte im Dokument plaziert, d.h. als Ergebnis kommt bei einer korrekten Programmierung reiner HTML-Code, evtl. mit integrierten clientseitigen Skripten, herraus. Dies ist das was der User zu sehen bekommt. Die genaue Programmierung, wie das Ergebnis erstellt wurde, bleibt ihm verborgen.

Viele Hersteller von Programmiersprachen haben den Vorteil von APIs erkannt und entsprechend reagiert. Beispielsweise gibt es von “PERL for Win32” eine entsprechende Erweiterung für den Internet Information Server (IIS). Durch Einsatz dieser API fallen die Antwortzeiten geringer aus und eine Neuprogrammierung der vorhandenen Skripte ist kaum nötig. Man kann die Skripte auch von Anfang an so programmieren, daß das Skript selber feststellen kann über welchen Aufruf (CGI oder API) es gestartet wurde und ich welcher Speicherumgebung es läuft.

Als Vorteile von API-Aufrufen sind besonders folgende Punkte hervorzuheben :

- schnellerer Aufruf durch den Web-Server
- weniger Overhead durch Verwaltung
- weniger Ressourcen-Verbrauch
- Zugriff auf WebServer-Funktionen
- übersichtlicher HTML-Code

Demgegenüber stehen die Nachteile :

- Gefahr des kompletten Absturzes des WebServers
- Inkompatibel zu anderen WebServern, sofern spezielle API-Zugriffe benutzt werden => Höhere Kosten bei WebServer-Wechsel
- Sitzungs- und Transaktionsmanagement bei DB-Anwendungen schwierig
- Mit jedem Aufruf erneut Verbindung zum DBMS aufbauen
- Rechteverwaltung im DBMS kann nicht gut genutzt werden und muß ggf. selber innerhalb der CGI-Programme implementiert werden.

3.2.3 Clientseitige Verarbeitung

Um eine Verarbeitung auf dem Server kommt man im Internet nicht herum, da sämtliche Informationen auf dem Server gespeichert sind. Selbst beim Abruf einer statischen Seite hat der Server etwas zu tun, indem er die Seite vom Festplattenspeicher liest und sie über das Internet zum Anwender schickt.

Bei großen und beliebten Webseiten kommen hunderte von Anfragen pro Minute. Dementsprechend hoch ist dann auch die Last auf dem Server/ den Servern und entsprechend lange dauern dann auch Antworten an den Client. Eine Last-Abgabe an die anfragenden Clients läge also sowohl im Sinne des Seiten-Betreibers, der damit um eine Aufstockung seiner Hardware herumkommen würde, als auch im Sinne des Benutzers, der dadurch kürzere Wartezeiten zu erwarten hätte.

Je nach Anforderungen an die GUI gibt es unterschiedliche Stufen der client-seitigen Verarbeitung. Als erste Stufe gibt es hier die Skript-Sprachen, die rudimentäre Ablaufsteuerungen übernehmen können. Richtig ausgenutzt können Sie dem Benutzer aber auch viel Arbeit abnehmen. Bekanntester und am stärksten verbreitetster Vertreter dieser Art ist JavaScript[24]. JavaScript ist in den meisten gängigen Browsern wie Internet Explorer und Netscape Navigator implementiert. Allerdings sind hier, ebenso wie bei der Interpretation von HTML-Seiten, die Implementationen ein wenig unterschiedlich und somit nicht eindeutig kompatibel zueinander[25]. Zudem haben viele Anwender JavaScript auch deaktiviert, da hiermit auch Aktionen ausgeführt werden können, von denen der Anwender nichts mitbekommt oder es kommt ein schädliches Skript auf den Rechner, das beispielsweise hunderte von Browser-Fenstern öffnet, so daß entweder der Speicher voll ist oder der Anwender alle Fenster wieder per Hand schließen müßte. Meist führt dieses aber dazu, daß der Anwender den Computer neu startet. Ein anderes Skript berechnet im Hintergrund die Fibonacci-Zahlen und bremst damit den Computer massiv aus. Werbefenster, die beim Verlassen einer Web-Seite automatisch geöffnet werden, stellen ein weiteres unschönes Verhalten dar, das Anwender dazu bringt JavaScript zu deaktivieren, zumal dieses oft Seiten mit pornographischem Inhalt sind. Aufgrund von Fehlern in den Browsern ist es teilweise auch möglich über bestimmte Skripte Dateien vom Computer des Anwenders zu übertragen, wie z.B. die Datei mit den Lesezeichen des Anwenders. Wenn gleichzeitig die E-Mail-Adresse und andere Daten aus dem Browser-Profil des Anwenders ausgelesen werden, läßt sich hierüber ein Verhaltens- und Vorlieben-Profil erstellen.

Auch die bekannte Sprache Java[26] ist aus ähnlichen Gründen nicht auf allen Clients aktiviert. Dieses hat zum einen auch die o.g. Gründe der nicht-kontrollierbaren Aktionen, aber auch die benötigten Ressourcen sind hierbei größer. Während bei JavaScript der Interpreter im Browser implementiert ist, wird bei Java die sog. Java Virtual Machine separat gestartet. Außerdem beanspruchen diese Anwendungen und Applets recht viel Rechenzeit und Übertragungs-Bandbreite.

[24] s. Kapitel 1.1.9 „Java-Script“, S.13
[25] s: Kapitel 2.2.2 “Clientseitige Umgebung“, S.24
[26] s. Kapitel 1.1.8 “Hostile Agents“, S.12

Java hat gegenüber JavaScript deutlich mehr Möglichkeiten. Wenn es mehr als einer Ablaufsteuerung bedarf kommt man an Java kaum vorbei. Durch Java sind Inhalte aktiv während der Laufzeit änderbar. Im Gegensatz zu reinen HTML-Seiten sind hier die Darstellungsmöglichkeiten wesentlich höher.

Ebenso verhält es sich bei ActiveX-Controls. Diese erlauben, im Gegensatz zu Java-Applets, Zugriff auf die System-Ressourcen der Clients, d.h. auf den Bildschirm, Festplatte, Drucker u.ä.. Aus diesem Grunde ist auch diese Komponente nicht bei allen Internet-Anwendern aktiviert.

Innerhalb des Internets können die o.g. Komponenten nicht immer als vorausgesetzt angesehen werden und sollten bei einer Anwendung daher auch nicht als Standard benutzt werden. Viele Benutzer besuchen Seiten, die eine dieser Scripting-Komponenten voraussetzen, aus genau diesem Grunde nicht wieder. Vielmehr geben sich Anwender oft aus Sicherheitsgründen auch mit weniger grafischen Spielereien zufrieden.

Sofern im Intranet für eine interne Anwendung die o.g. Fähigkeiten zwingend aktiviert sein müssen, können diese durch die System-Administration und durch entsprechende Arbeits-Anweisungen vorgegeben und damit vorausgesetzt werden. Wenn die Anwender aber gleichzeitig Internet-Zugang haben birgt dieses ein großes Sicherheitsrisiko. Die hier diskutierten clientseitigen aktiven Inhalte sollten dann z.B. durch eine Firewall geblockt werden.

4 Implementation

Bereits in Kapitel 2.1 wurden die Abhängigkeiten bestimmter Faktoren bei der Entscheidungsfindung dargestellt. Außerdem wurde aufgrund der vorhandenen Umgebung ein Anforderungskatalog erstellt. Mit Hilfe dieser Informationen soll aus den vorhandenen Techniken die am geeignetsten erscheinende ermittelt werden. Diese Diskussion soll Thema dieses Kapitels sein.

4.1 Datenbank

Die Aufteilung nach betroffenen Plattformen in jeweils eine eigene Tabelle machte zur Anfangszeit der Malware-Thematik durchaus Sinn. Durch das Internet und aktuelle Entwicklungen in der Virenprogrammierung wird diese Trennung nicht länger den derzeitigen und zukünftigen Anforderungen gerecht.

Eine Erweiterung des Datenbestandes um eine neue infizierbare Plattform würde bei der aktuellen Struktur eine neue Tabelle erfordern. Die Suche über mehrere Tabellen wäre sowohl umständlich als auch unperformant. Da die Struktur aller Tabellen gleich ist, ist es durchaus sinnvoll die Tabellen zu einer einzigen zusammenzufassen. Die zuvor im Titel der Tabelle enthaltene Information über die betroffene Plattform muß dennoch erhalten bleiben. Zu diesem Zweck wird der neu gewonnenen Tabelle ein weiteres Feld hinzugefügt, in der diese Information bei der Zusammenführung gespeichert wird.

Macht ein neu entdeckter Virus es nun nötig, daß eine neue Plattform in die Datenbank aufgenommen wird, muß lediglich in dem Eintrag diese neue Plattform hinzugefügt werden. Bei einer Suche innerhalb der Datenbank nach einer speziellen Plattform wird diese neue automatisch mit bedacht. Bei der vorherigen Aufteilung in jeweils eine eigene Tabelle mußte die Tabelle erzeugt werden, die Tabelle der Anwendung hinzugefügt werden und zudem wurde die Suche deutlich langsamer.

Ein Wechsel von einem relationalen Datenbanksystem auf ein anderes Datenbank-Konzept ist unnötig.

4.2 Client/Server-Struktur

Wie bereits oben beschrieben ist aus unterschiedlichen Gründen im Internet auf Client-Seite kein Standard vorauszusetzen, außer der Fähigkeit zur Darstellung von HTML-Seiten. Da die Datenbank einer großen Anzahl von Anwendern zugänglich sein soll, sollte hier aus diesem Grund auch die Darstellung und Eingabe nur durch HTML-Elemente und erfolgen. Dies bedeutet, daß die komplette Verarbeitung serverseitig geschehen muß[27].

Um dennoch den Server so weit wie möglich zu entlasten kann bei Bedarf zur Eingabe-Prüfung o.ä. JavaScript eingesetzt werden, da dieses in den meisten Browsern vorhanden ist und die Größe der Übertragungsdaten nicht stark zunehmen läßt. Um auch diese unnötige Übertragung zu verhindern ist es bei JavaScript ab der Version 1.1 möglich die Skripte in einer separaten Datei abzulegen und diese bei Bedarf in ein HTML-Dokument einzubinden. Dies geschieht nur, wenn JavaScript auch aktiviert ist.

[27] s. Abbildung 3.1-4 „Server-zentrierte Verarbeitung“, S. 33

Zudem bietet diese Möglichkeit den großen Vorteil, daß, wenn die selben Skripte in mehreren HTML-Seiten Anwendung finden und eine Änderung des Skriptes nötig ist, lediglich die Skript-Datei aktualisiert werden muß. Hierdurch werden automatisch auch die darauf zugreifenden HTML-Seiten einheitlich und zum selben Zeitpunkt auf den aktuellen Stand bzgl. der Skripte gebracht.

4.3 Prozessierung auf dem Server

Nachdem die Entscheidung für eine server-zentrierte, aber durch den Client wenn möglich unterstützte, Verarbeitung gefallen ist, muß nunmehr die Art der Implementation gewählt werden.

4.3.1 Einbindung in den WebServer

Wie in Kapitel 3.2.2 erläutert wurde, gibt es zwei Möglichkeiten des Aufruf der HTML-Seiten-erzeugenden Programme: Innerhalb oder außerhalb des Adreßraums des WebServers. Beide Möglichkeiten hatten sowohl ihre Vor- als auch ihre Nachteile. Wenn man diese Vorteile in Verbindung mit den Anforderungen und dem bereits vorhandenen System sieht, kommt man zu dem Schluß, daß die beste Wahl für eine Anbindung über CGI spricht.

Die Nachteile, die CGI gegenüber der API-Methode hat, fallen in der vorhandenen Umgebung nicht so sehr ins Gewicht, während die Vorteile besser zum Tragen kommen. Die schlechteren Performanz-Werte gegenüber einer Integration über die API werden bei den erwarteten Zugriffen und dem nichtkommerziellen Charakter der Anwendung nicht auffallen. Bei der Erwartung von vielen parallelen Zugriffen wäre die Entscheidung für die API-Lösung sicherlich zu bevorzugen.

Die Vorteile von CGI decken vor allem in Bezug auf den Kostenfaktor völlig die Anforderungen. Portabilität, Standardisierung und „Zukunfts-Sicherheit“ garantieren einen Fortbestand der Anwendung auch bei Wechsel des Betriebssystems und/oder des WebServers. Hierdurch wird eine Re-Programmierung nicht nötig werden. Zudem ist CGI wie gesagt ein Standard, der über nahezu jede Programmiersprache nutzbar ist.

4.3.2 Wahl der Sprache

Bei der Wahl der zu verwendenden Programmiersprache spielen verschiedene Faktoren eine Rolle. Zum einen sind dieses wieder die Kosten, die vor allem in Form von Lizenzkosten anfallen würden. Aufgrund der beschränkten finanziellen Mittel sollte hier ein kostengünstige, wenn nicht sogar kostenfreie, Lösung gewählt werden.

Da CGI an sich durch die Aufrufe schon ein wenig an Performanz einbüßt, kann hier durch die Wahl einer performanten Programmiersprache ggf. der Verlust an der Leistungsfähigkeit der Anwendung in Maßen gehalten werden.

Die Programmiersprache sollte nicht auf ein Betriebssystem beschränkt sein, sondern auch ein hohes Maß an Portabilität aufweisen. Tut sie dieses nicht wäre der Portabilitäts-Vorteil durch CGI zunichte gemacht, da bei einer System-Umstellung doch wieder eine Neu-Programmierung nötig wäre. Selbstverständlich muß die Sprache auch auf dem in Kapitel 2.2.1 beschriebenen System vorhanden und lauffähig sein.

Zu diesen, durch die Wahl der Integration in den WebServer bedingten, Faktoren kommt noch der, daß die Sprache immer der Art der Anwendung und damit auch der Programmierung sehr entgegenkommen sollte. Im vorliegenden Fall wird die Anwendung primär Eingaben verarbeiten und eine entsprechende Ausgabe erzeugen. Diese Tätigkeit an sich ist nichts besonderes, kann aber durch entsprechende Elemente der Programmiersprache unterstützt, wenn nicht sogar beschleunigt werden. Einige Programmiersprachen sind auf die Zeichenketten-Verarbeitung spezialisiert bzw. hierfür entworfen worden. Eine fest implementierte Funktion ist meistens schneller als wenn diese durch die zur Verfügung stehenden Sprachelemente extra programmiert werden muß.

Für den vorliegenden Fall muß natürlich auch eine Schnittstelle zu mindestens einer Datenbank existieren. Ansonsten ist ein Ansprechen derselbigen nicht möglich.

Unter der Vielzahl von verfügbaren Sprachen wurde sich für Perl entschieden. Der Name Perl ist eine Abkürzung und steht für „**P**ractial **e**xtraction and **r**eporting **l**anguage“. Dieses heißt in etwa „Praktische Extraktions- und Berichtssprache“[28]. Ursprünglich kam Perl vom Unix-Betriebssystem, wo es vorzugweise als Skript-Sprache innerhalb der Shell benutzt wurde. Viele Unix-Programmierer hatten die Zeichen der Zeit erkannt und spezialisierten sich auf andere Betriebssysteme. Dennoch wollten Sie auf den neuen Plattformen nicht auf Perl verzichten, weshalb Perl auf vielen verschiedenen Betriebssystem zur Verfügung steht. Perl gilt als eine der portabelsten heute verfügbaren Programmiersprachen.

Zu dem hohen Portabilitäts-Faktor kommt noch, daß Perl Freeware ist und somit also ohne Lizenzkosten nutzbar ist.[29]

Perl ist ursprünglich eine interpretierte Sprache, die aber dennoch recht performant ist, da sie zunächst vorkompiliert wird. Beim ersten Aufruf eines geänderten Perl-Programms verzögert sich der Start je nach Ausmaß des Codes, da hier die Übersetzung in den Zwischencode stattfindet. Alle nachfolgenden Starts greifen auf diesen Zwischencode zu, welcher dann interpretiert wird. Mittlerweile gibt es aufgrund der starken Nachfrage bereits für vereinzelte Plattformen Compiler für Perl, so daß hierbei die Einfachheit der Programmierung mit recht schnellen Programmen gepaart wird. Diese Compiler sind aber meistens keine Freeware und gehören nicht zum Standard-Lieferumfang von Perl.

Als Implementation für Perl für Windows NT 4.0 wurde die von Activeware, auch bekannt als „Perl for Win32“ gewählt. Ursprünglich gehörte diese zum Ressource-Kit zu Windows NT 4.0, wurde aber später ausgegliedert, um die Entwicklung aufgrund des starken Interesses stärker vorantreiben zu können. Vor allem in Internet-Kreisen erfreut sich diese Implementation, wie Perl allgemein, großer Beliebtheit. Aufgrund der Performanz-Verluste durch den Aufruf über CGI wünschten sich viele Programmierer eine Möglichkeit Perl-Programme über eine performantere Schnittstelle ansprechen zu können. Speziell für den Microsoft Internet Information Server 4.0 und höher existiert eine solche Schnittstelle in Form einer ISAPI-fähigen DLL. Hierdurch wird zumindest der Aufruf der Skripte beschleunigt.

[28] [Wall 1997], S. xv
[29] [Wall 1997], S. xiii ff

4.3.3 Verbindung zur Datenbank

Ein entscheidender Faktor bei der Wahl der Programmiersprache war auch die Forderung nach einer Schnittstelle zu einer Datenbank. Um die bei den Anforderungen betonte Portabilität auch in Bezug auf die Datenbank zu gewährleisten empfiehlt sich der Einsatz einer sog. Middleware[30]. „Perl for Win32“ bietet eine Schnittstelle in Form eines ODBC-fähigen Moduls, auf das Skripte zugreifen können. Die Ansteuerung erfolgt objektorientiert, so daß über mehrere unterschiedliche Instanziierungen eines Objektes auch parallele Zugriffe innerhalb eines Programmes möglich sind.

Da ODBC vom eigentlichen DBMS abstrahiert und nur Standard-SQL-Befehle verwandt werden können, gewährleistet dieses auch die Portabilität mit anderen Systemen, sofern ein ODBC-Modul für Perl zur Verfügung steht, das auch über die gleichen Perl-Befehle den Zugriff steuert. Eine vollständige Kompatibilität ist hier wahrscheinlich leider nicht gegeben, was eine Anpassung der Befehle, die für den Datenbank-Zugriffs verantwortlich sind, nötig macht.

Da das eigentlich benutzte Datenbank-Format durch ODBC für die Programmierung unwichtig ist, kann hier jedes benutzt werden, für das ein ODBC-fähiger Treiber zur Verfügung steht. Ein späterer Wechsel, z.B. wegen Speicherplatzproblemen oder einer erhofften Leistungs-steigerung ist hier ohne Änderung der Programmierung möglich.

4.4 Grundlegendes der Implementation

Zur besseren Übersicht wird dieses Kapitel in anhand der unterschiedlichen möglichen Tätigkeiten in zwei Unterkapitel unterteilt.

4.4.1 Suchen und Betrachten von Einträgen

Diese Funktionen werden allen Internet-Anwendern frei zugänglich sein, was die eigentliche Intention der Migration in das WWW ist.

Das Verhalten dieser Anwender ist nicht vorhersagbar und die Suche nach einer Virenbeschreibung kann auch manigfaltige Art geschehen. Zunächst einmal sollte man davon ausgehen, daß die Anwender nicht zwischen den verschiedenen Plattformen so exakt unterscheiden können wie z.B. ein Antiviren-Experte. Aus diesem Grunde kann man auch nicht davon ausgehen, daß der Anwender weiß, bei welcher Plattform er suchen muß oder um was für eine Art von Malware es sich exakt handelt. Inzwischen haben sogar Antiviren-Experten hier oftmals Diskussionen über die genaue Einstufung.

Anwender erhalten oft von einem Virus nur Symptome oder durch einen Virenscanner einen Namen. Letzterer kann von Hersteller zu Hersteller unterschiedlich sein und muß nicht mit dem im Computer-Virus-Katalog übereinstimmen. Mit den Makro- und Skript-Viren wurde dieses Manko durch eine Gruppe von Antivirus-Forschern ansatzweise behoben, die vom Auftreten des ersten Makrovirus ein klares Namensschema aufgebaut haben.

[30] s. Seite 31

Die Kommunikation dieser Gruppe findet größtenteils über eine an der Universität Hamburg administrierte Mailingliste namens VMACRO statt. Diese Liste soll durch die zu entwickelnde Anwendung „VMServ“ unterstützt werden, die im Rahmen einer Diplomarbeit am Arbeitsbereich AGN konzipiert und implementiert werden soll.

Durch die Manigfaltigkeit des Malware-Themas sollte es für Anwender möglich sein eine Suche durchzuführen. Diese sollte so gestaltet sein, daß erfahrene Anwender durch eine exakte Eingabe in bestimmten Feldern schnell zu einem Ergebnis kommen, aber auch unerfahrene Anwender zu einem Ergebnis kommen können, indem ihnen eine Vorauswahl präsentiert wird.
Das Ergebnis einer Suche sollte in Form einer Liste mit Verweisen zu dem entsprechenden Virus erstellt werden.

Eine generelle Liste aller in der Datenbank enthaltenen Viren sollte ebenfalls verfügbar sein, damit der interessierte Anwender ohne konkretes Suchanliegen dennoch die Einträge der Datenbank erforschen kann.

Die beiden zuvor genannten Punkte waren so ähnlich, bedingt durch unterschiedliche grafische Möglichkeiten, auch in CMBASE vorhanden. Als dritte Möglichkeit gab es in CMBASE auch noch die Baum-Ansicht[31]. Diese nur mit HTML auf der grafischen Ebene zu implementieren ist schwierig, da es beispielsweise keine vertikalen Linien gibt. Diese Lücke müßten extra erzeugte Grafiken ausfüllen. Eine mögliche Implementation wäre, daß zunächst nur die erste Hierarchie-Ebene eingeblendet wird und jedes Öffnen eines Eintrags dieser Ebene die nächste Ebene öffnet und man durch geschicktes Ausnutzen der kleinen Grafiken, die die vertikalen Linien und Abzweigungen darstellen, die Baum-Struktur grafisch realisiert.

Wenn ein Anwender einen entsprechenden Eintrag gefunden hat und er auf der Originalität der Daten großen Wert legt, muß sichergestellt werden, daß diese auf dem Übertragungsweg zu ihm nicht verfälscht wurden. Dies kann durch Signatur- und Verschlüsselungsverfahren gewährleistet werden, was aber bei der Übertragung über den Browser ein digitales Zertifikat benötigt[32] und außerdem die Performanz durch die Ver- und Entschlüsselung senken würde. Da nur die endgültig gefundenen Einträge eine gesicherte Übertragung benötigen und hier auch auch noch viel zu wenig Menschen sich dieser potentiellen Beeinträchtigung ihrer Virenentfernung bewußt sind, würde es sich anbieten, daß man die Möglichkeit anbietet sich einen Katalog-Eintrag per Mail zukommen zu lassen. Diese Mail könnte dann mit einem privaten Schlüssel des Arbeitsbereichs signiert und dem anfragenden Anwender automatisch zugesandt werden. Hierfür werden keine Lizenzkosten fällig, da z.B. PGP (Pretty Good Privacy) bis zur Version 2.6.2i frei erhältlich und der Quell-Code offen einsehbar war. Außerdem fallen für die Erzeugung und Zertifizierung des Schlüssels keine Kosten an, was auch im Einklang mit dem gesteckten Anforderungskatalog steht.

[31] s. Seite 18
[32] s. Kapitel 2.3.3.3 “Integrität“, S.27

4.4.2 Hinzufügen und Editieren von Einträgen

Über diese Funktion soll es authorisoierten Personen möglich sein, den Datenstamm der Anwendung über ein Web-Interface zu ergänzen bzw. zu korrigieren. Diese Funktionalität ist dem normalen Anwender aus Sicherheitsgründen verborgen und ist nur über eine spezielle URL zugreifbar. Auch der Zugriff auf diese URL muß zunächst durch einen Authentifizierungs-Vorgang in Form von Einloggen erlangt werden. Nach der Authorisierung erhält der jeweilige Anwender abhängig von seinen Rechten ein entsprechendes Auswahlmenü, durch das er dann die jeweilige Funktion auswählen kann. Der Zugriff sollte, vom Benutzer abhängig, nur von bestimmten IP-Adressen oder von einem IP-Adressen-Bereich erlaubt sein.

Bei dieser weltweit verfügbaren Funktionalität tritt wieder die Schwäche des HTTP-Protokolls zutage. Der Zustand der Anwendung ist nicht ohne weiteres erfaßbar, da HTTP zustandslos ist. Hier muß die Anwendung dafür sorgen, daß durch Modifikationen am Datenbestand keine Inkonsistenzen erzeugt werden. Diese könnten z.B. dadurch auftreten, daß gleichzeitig zwei unterschiedliche Anwender den gleichen Datensatz zum Bearbeiten öffnen. Speichert nunmehr einer von beiden seine Daten erscheint diese Änderung nicht bei dem anderen Anwender. Dieser modifiziert die alten Daten und überschreibt letztendlich die Änderungen des ersten Benutzers. Dieses zu verhindern ist auch Aufgabe der Anwendung. Da diese Funktion wahrscheinlich nicht extrem parallel benutzt werden wird genügt es hier einen Zugang zu implementieren, der es nur einem Anwender erlaubt überhaupt Modifikationen an der Datenbank vornehmen zu können. Dies setzt allerdings voraus, daß er sich nach erledigter Arbeit ordentlich ausloggt, um somit den Zugang wieder für andere Anwender freizugeben.

Alle durch einen Anwender getätigten Aktionen werden protokolliert. Dieses Protokoll ist auch nur entsprechend berechtigten Personen zugänglich.

4.4.2.1 Hinzufügen

Die jeweils authorisierten Anwender können über diese Funktion neue Einträge zu der Datenbank hinzufügen. Um hierbei die Eingabe so einfach wie möglich für den Anwender zu machen kann ein im ASCII-Format vorliegender Eintrag in ein Eingabefeld kopiert werden. Anschließend wird kontrolliert, ob alle geforderten Felder vorhanden sind. Ist dieses nicht der Fall wird eine Maske erzeugt, in der die zuordbaren Felder aufgelistet sind, nicht zuordbare ebenfalls erscheinen und dem Anwender darin Auswahlmöglichkeiten von evtl. nicht vorhandenen, aber geforderten Felder gegeben werden, so daß hierdurch eine Eindeutigkeit erzielt werden kann. In diesem Fall kann, wenn möglich eine Unterstützung durch JavaScript erfolgen, die, nachdem in einem Fall ein Zuordnung erfolgt ist, die Auswahlmöglichkeiten in den übriggebliebenen, nicht zugeordneten Feldern einschränkt. Nur eindeutige Datensätze werden der Datenbank dann hinzugefügt. Eventuelle Fehler in der Zuordnung zu einer Plattform können hierbei durch die Notwendigkeit einer erneuten Bestätigung durch den Anwender validiert werden.

4.4.2.2 Bearbeiten

Neuere Erkenntnisse zu einem Virus machen es evtl. erforderlich, daß der entsprechende Eintrag in der Datenbank ergänzt wird. Hierzu kann sich der Anwender den Datensatz zu einem Virus auf den Bildschirm anzeigen lassen und die entsprechenden Korrekturen vornehmen. Nach erfolgter Korrektur sendet er die Daten an die Datenbank, damit der Eintrag ab diesem Zeitpunkt für alle anderen Anwender, sowohl zur Betrachtung als auch zur Modifikation zur Verfügung steht.

4.4.2.3 Löschen

Dieser Menüpunkt wird voraussichtlich eher selten benutzt werden, da die existenten Viren nicht verschwinden. Zwar kann es sein, daß einige nicht mehr bei Anwendern auftreten, dennoch existieren diese Viren noch in einigen Virendatenbanken zu Testzwecken, wie z.B. am Arbeitsbereich AGN. Potentiell wäre es, wenn auch durch ein Versehen, durchaus möglich, daß dieser Virus wieder in Umlauf gerät. Die vermehrte Meldung eines Computervirus erforderte auch nur die Erstellung eines Eintrags in den Computer-Virus-Katalog, sagt aber nichts darüber aus, daß ein Eintrag für einen Virus nur dann enthalten sein soll, wenn dieser Virus verbreitet ist.

Um Manipulationen an der Datenbank ohne Kontrolle durch eine Anwendung zu verhindern sollte dies Funktion dennoch implementiert werden, da es durchaus einmal vorkommen könnte, daß ein Eintrag doppelt vorhanden ist und einer von beiden gelöscht werden muß.

4.4.3 Benutzer-Verwaltung

Durch diese Funktion können die Rechte der jeweiligen Anwender neu hinzugefügt, editiert oder gelöscht werden. Außerdem kann ein für jeden Anwender ein IP-Adreß-Bereich angegeben werden, von wo aus der Zugriff erfolgen darf. Dieses schränkt die Möglichkeiten eines Angriffs stark ein, da die IP-Adressen und auch Adreßbereiche eindeutig zuordbar sind.

Um diese Funktion ausüben zu können müssen auch hier dem jeweiligen Anwender die entsprechenden Rechte erteilt worden sein.

5 Zusammenfassung und Ausblick

Die zunehmende Verbreitung des Internets auch im privaten Bereich macht es sinnvoll Informationen der Öffentlichkeit zugänglich zu machen, sofern diese nicht dem Datenschutz unterliegen oder aus anderen Gründen vertraulich sind. Um den Anwender bei der Suche der gewünschten Informationen in der Vielzahl der vorhandenen Daten zu unterstützen sind hierfür Anwendungen nötig.

Bei der Implementation oder Migration einer WWW-Anwendung sollte viel Augenmerk bereits auf die Planungsphase gelegt werden. Viele Faktoren beeinflussen die Entscheidung, die den Grundstock legt, um die gesetzten Ziele bestmöglich zu erreichen. Evtl. tauchen nach der Planungsphase und während der Implementationsphase neue Faktoren mit anderen Parametern auf, die einen Rückschritt zur Planungsphase erfordern. Dieser zusätzliche Zeit und Geldaufwand kann im Vorfelde bereits durch eine gründliche Analyse des gegebenen Szenarios und der gestellten Anforderungen in Grenzen gehalten werden.

Grundlage eines jeden Internet-Präsenz ist die Hardware, die WebServer-Software und das Betriebssystem, mit dem der Rechner betrieben wird. Während man eine Aufstockung der Hardware durch entsprechende finanzielle Mittel gewährleisten kann, ist die Entscheidung für ein adhoc unsicheres Betriebssystem oder einen unflexiblen und unperformanten WebServer später nur durch eine komplette Umstellung erreichbar. Dies zieht meist auch eine Neu-programmierung der entsprechenden Anwendungen nach sich.

Ein sicheres Betriebssystem und der entsprechende Internet-Dienst können durch schlechte Administration schnell zu einer weit geöffneten Tür für Angreifer werden. Die entsprechenden Administratoren sollten sich sowohl mit den Sicherheitsmechanismen des Betriebssystems als auch der Software auskennen und sich auf diese entsprechend spezialisieren. Auch hier ist eine gute Planung viel wert. Von vornherein nur als WebServer geplant und ohne zusätzliche Komponenten muß das System selten gewartet werden. Entsprechend stabil läuft dann auch meistens das ganze System und die Administratoren haben dann nur noch die Pflege der User (sofern es hierfür spezielle gibt) und evtl. das Einspielen von Patches als Aufgaben zu erledigen.

Eine gute Administration steht aber reicht nicht alleine für eine sichere und zuverlässige Anwendung. Das was die Administratoren abgesichert haben kann ein schlechter Programmierer durch seinen Programm-Code wieder zunichte machen und damit Dritten wieder Zugang zu dem entsprechenden Rechner und ggf. sogar dem internen Netz öffnen. Häufigster Fehler hierbei ist das Vertrauen darauf, daß nur die angeforderten Parameter korrekt eingegeben wurden und nur die Original-Seite benutzt wurde. Weiterreichende Kenntnisse des WWW, seiner Möglichkeiten und der Gefahren sollten für jeden Programmierer in diesem Umfeld zum Basiswissen gehören.

Auf der Grundlage des bestehenden Systems, eines erstellten Anforderungskatalogs und Kenntnis der vorhandenen Techniken und Implementationsmöglichkeiten ist es gelungen, die zum aktuellen Zeitpunkt am geeignetsten erscheinende Lösung zu finden. Ob diese den zukünftigen Anforderungen gerecht werden wird ist nicht vorhersagbar, da diese Anforderungen durch alle Anwender des Internets z.B. durch verstärkte Zugriffszahlen beeinflussbar sind. In diesem Fall ist ggf. eine Neukonzipierung nötig.

Dadurch, daß jeder Antivirus-Hersteller inzwischen seine eigene Datenbank mit Virenbeschreibungen pflegt und durch Umbrüche innerhalb des Arbeitsbereiches sind viele neuere, verbreitete Viren nicht mehr in dem Katalog erfaßt. Am Arbeitsbereich AGN findet ein sog. Reverse Engineering-Praktikum statt, in dessen Rahmen vor allem Viren-Analyse-Techniken erlernt werden. Innerhalb dieser Veranstaltung wäre es denkbar durch die Studierenden verbreitete Viren analysieren und die entsprechenden Katalog-Einträge in digitaler Form erstellen zu lassen, so daß der Datenbestand in dem Katalog wieder reichhaltiger und somit interessanter für Internet-Anwender wird.

6 Literaturliste

[Brunnstein 1991] Brunnstein, Klaus: Computer-Viren-Report, WRS Verlag Wirtschaft,Recht und Steuern 1991, ISBN 3-8092-0704-7

[Kim 1997] Kim, Eugene Eric: CGI - Developer’s Guide, SAMS Publishing 1997, ISBN 3-87791-891-3

[Loeser 1997] Loeser, Henrik : Datenbankanbindung an das WWW - Techniken, Tools und Trends in K.R. Dittrich, A.Geppert (Hrsg.) : Datenbanksysteme in Büro, Technik und Wissenschaft - GI-Fachtagung 1997, S.83 - 99 Springer, 1997, ISBN 3-540-62569-0

[Martin 1997] Martin, Felix : Virus Report ’98, Franzis-Verlag, 1997, ISBN 3-7723-5234-0

[Sommer 1997] Ulrike Sommer, Peter Zoller : Online-Datenbanken in Michael Barabas, Klaus-Peter Boden(Hrsg.): Internet - von der Technologie zum Wirtschaftsfaktor, Deutscher Internet Kongreß ‘97, dpunkt-Verlag, 1997, ISBN 3-920993-91-8

[Taylor 1997] Taylor, Art : JDBC Developer’s Resource : Database programming on the internet, Prentice Hall, 1997, ISBN 0-13-842352-0

[Vromans 1996] Vromans, Johan : Perl 5 Schnellübersicht, O’Reilly/International Thomson Verlag, 1996 ISBN 3-930673-50-9

[VTC 1999] Broschüre „Viren und Malware“, Mario Ticak und Martin Kittel, Universität Hamburg, FB Informatik, Arbeitsbereich AGN

[VTC 2000] Testergebnisse des Anti-Virus-Testcenters an der Universität Hamburg : http://agn-www.informatik.uni-hamburg.de/vtc/en0004.htm

[Wall 1997] L.Wall, T.Christiansen, R.L. Schwartz : Programmieren mit Perl, O’Reilly Verlag, 1997, ISBN 3-930673-48-7

Anhang I : Das Format des Computer-Virus-Katalogs

------ Computer Virus Catalog 1.2: "Virusname" (Date of Entry) --------

```
Entry..............:  "Virusname" (=Name of virus)
Alias(es)..........:  Alternate Name(s)
Virus Strain.......:  "Family" (if any) to which this virus belongs
Virus detected when.:  Date of first appearance
              where.:  Where was Virus produced or first detected
                       (both entries only if well-known)
Classification.....:  System Virus (BootSector, Command.Com, BAT V.)
                       Link or Program Virus (Overwriting/Extending V.)
                       Resident, Direct Action
Length of Virus....:  1.Length (Byte) on storage medium
                       2.Length (Byte) in RAM

--------------------- Preconditions ----------------------------------

Operating System(s).:  e.g. AMIGA-DOS, ATARI-TOS, MacOS, MS-DOS,
                       UNIX, VMS, MVS, VM
Version/Release.....:  Special Version of OS (e.g. UNIX System V,
                       UNIX BSD, VMS etc) if needed, and Release
                       (e.g. MS-DOS 3.2, UNIX BSD 4.2)
Computer model(s)...:  The Computer models (e.g. ROM BIOS versions)
                       on which the Virus runs.

--------------------- Attributes -------------------------------------

Easy Identification.:  if applicable: Typical texts, either messages
                       (e.g. screen), or texts in Virus body (readable
                       with HexDump-facilities), Volume Labels etc.
                       by which viruses may easily identified

Type of infection...:  Self-Identification methods;
                       Executable File infection(.COM,.EXE):overwriting,
                       extending; resident; (RAM/File) Direct Action;
                       WCS infection (e.g. CMOS at initialisation setup);
                       System infection: RAM-Resident, Reset-Resident,
                       Bootblock/Bootsectors, Command.Com, BAT, Device
                       Handlers/Libraries etc;
                       Infection of unlinked Object Files;
                       Source Code Infection.

Infection Trigger...:  e.g. time/date, other events, random, reset (CTRL+
                       ALT+DEL), operations such as: DIR, execution of
                       specific program (.COM/.EXE).

Storage media affected: Infection of (particular) diskettes, hard disks,
                       DiskPacks, etc.

Interrupts hooked...:  Interrupts used and changed by this virus.
```

```
Damage.............: Permanent Damage: e.g. overwriting bootblock,
                     repeated restart/format, zeroing of sectors,
                     Bad Sectors in FAT etc;
                     Transient Damage: e.g. screen buffer manipulation,
                     audio effects, blinking LEDs;
                     Transient/Permanent Damage: viruses which (under
                     specified conditions) produce parmanent damage
                     while "normally" producing transient damage.

Damage Trigger.....: e.g. time/date, value of infection counter, other
                     events, random, reset, operations.

Particularities....: special effects e.g. process velocity slowed-down

Similarities.......: dis/similarities to other viruses ( either from
                     same "family" (=strain) or different viruses);
                     names of related viruses.

-------------------- Agents ----------------------------------------

Countermeasures....: Names of tested products of Category 1-5:
                     Category 1: .1 Monitoring Files: program which
                                    monitors (attempted) changes in
                                    files
                                 .2 Monitoring System Vectors: program
                                    which monitors changes in vectors
                                    (e.g. resident, interrupt vectors)
                                 .3 Monitoring System Areas: program
                                    which monitors System Areas such
                                    as BootSectors/Blocks.
                     Category 2: Alteration Detection: a program which
                                 detects changes in given files
                     Category 3: Eradication: a program which erases
                                 a specific virus code from files or
                                 from RAM (if resident)
                     Category 4: Vaccine: a program which alters files
                                 (on permanent storage) or RAM resident
                                 programs such that viruses regard them
                                 as already infected
                     Category 5: Hardware Methods: methods to detect or
                                 prevent alteration or infection of
                                 files, vectors or system areas.
                     Category 6: Cryptographic Methods (Hard/Software):
                                 methods keeping programs on storage
                                 in encrypted form, and decrypting
                                 them before execution.

Countermeasures successful: Names of those countermeasures (of given
                     category) which, without (or with known "small")
                     restrictions or side effects, were "successful"
                     to detect, identify, inactivate or erase the
                     given virus or exclude infection by it.

Standard means.....: Means in the respective System which may be
                     used to identify/destroy this virus.
```

```
---------------------- Acknowledgement --------------------------------

Location.............: e.g. Virus Test Center, University Hamburg, FRG
Classification by...: Author(s) of Reverse-Engineering Document
Documentation by....: Author(s) of this Catalog Entry;
                      Translator of Non-English document (if applicable)
Date................: Production/last Update of this Catalog Entry
                      (this information also in the 1st line)
Information Source..: Information used for Documentation (only in cases
                      where Reverse-Analysis was not possible).

-------------------------End of "Virusname"-Virus---------------------
```