Database and Distributed Systems Group

Fachbereich Informatik

Technische Universität Darmstadt

64283 Darmstadt

Diplomarbeit

SEC://HOUSE

Ein Data Warehouse für Software-Schwachstellen

eingereicht von

Michael Hurler

bei

Prof. Alejandro P. Buchmann, Ph.D.

16. August 2000

Betreuer : Dipl. Wirtsch.-Inform. Christian Haul
Dipl.-Ing. Markus Schumacher

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 16. August 2000 _____ Michael Hurler

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CERIAS	Center for Education and Research in Information Assurance and Security
CERT	Computer Emergency Response Team
CERT/CC	Computer Emergency Response Team Coordination Center
COAST	Computer Operations, Audit, and Security Technology Laboratories
CORBA	Common Object Request Broker Architecture
CVE	Common Vulnerabilities & Exposures
DBMS	Datenbankmanagementsystem
DDoS	Distributed Denial of Service
DFN	Deutsches Forschungsnetz
DoS	Denial of Service
GPL	GNU General Public License
ICQ	‘I Seek You’ - Weit verbreitetes, proprietäres Kommunikationssystem
IRC	Internet Relay Chat
IT	Informationstechnologie, engl. **I**nformation **T**echnology
JDBC	Java Database Connectivity
JSSE	Java Secure Socket Extension
NASL	Nessus Attack Scripting Language
NSA	National Security Agency
ODBC	Open Database Connectivity
OECD	Organisation for Economic Co-Operation and Development
OMG	Object Management Group
OPL	Open Content License
PGP	Pretty Good Privacy

RDBMS	Relationales Datenbank Managementsystem
RMI	Remote Method Invocation
SDB	Schwachstellendatenbank, engl. Vulnerability Database
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer
TRUSTED	Testbed for Reliable, Ubiquitous, Secure, Transactional, Event-driven and Distributed Systems
VAS	Value Added Services
VDB	Vulnerability Database, engl. für Schwachstellendatenbank
WWW	World Wide Web

Vorwort

Eine Systemkomponente weist eine Sicherheitslücke oder *Schwachstelle* auf, wenn sie nur in unzureichendem Maß gegen Mißbrauch geschützt ist. Wird diese Schwachstelle von Dritten ausgenutzt, so ist die geplante Sicherheit des betroffenen IT-Systems gefährdet. Dabei ist es unerheblich, ob hinter einem solchen Verstoß gegen die festgelegten Sicherheitsrichtlinien Absicht oder Zufall steckt.

Heute vergeht kaum ein Tag, an dem nicht eine Sicherheitslücke in einem IT-System bzw. einer IT-Komponente aufgedeckt wird. Obwohl es einige Richtlinien für den sicheren Entwurf von IT-Systemen gibt, hat es nicht den Anschein, daß eine spürbare Besserung in Sicht ist. Die Forschung über die Ursachen und die Eigenschaften von sicherheitsbedrohenden Schwachstellen auf dem Gebiet der IT Sicherheit ist daher von hoher Relevanz.

Ein grundlegendes Werkzeug für die Untersuchung von Schwachstellen stellen sogenannte *Schwachstellendatenbanken* (SDB) dar. In SDBs werden Informationen unterschiedlicher charakteristischer Informationsquellen wie Newsgroups, Mailing-Listen oder auch anderer (Schwachstellen-) Datenbanken möglichst strukturiert erfaßt. Mit Ausnahme der Datenbanken sind diese Informationsquellen jedoch i.d.R. unstrukturiert und daher nur schlecht für eine systematische Untersuchung von Schwachstellen geeignet.

Es gibt heute bereits eine Vielzahl unterschiedlichster Informationsquellen zu Schwachstellen, die vielfältige Informationen enthalten, die als Basis für wissenschaftliche Untersuchungen dienen könnten. Diese Datenbanken werden öffentlich und privat von den unterschiedlichsten Organisationen betrieben. Da Informationen über Schwachstellen einen Wettbewerbsvorteil darstellen können, werden allerdings wenige Anstrengungen unternommen, einheitliche Daten- und Betreibermodelle anzustreben, was u.a. zu einer hohen Datenredundanz und einer erschwerten Informationssuche führt, da regelmäßig mehrere Quellen für eine umfassende Recherche bezüglich einer Schwachstelle konsultiert werden müssen.

Wissenschaftliche und (partiell) öffentlich zugängliche Arbeiten zu diesem Thema finden bisher ausschließlich in den USA statt. Da jede Art der Forschung auf dem Gebiet der IT Sicherheit unweigerlich das Interesse amerikanischer Regierungsstellen, wie z.B. der NSA, weckt, werden nur wenige Erkenntnisse an Forscher außerhalb der Vereinigten Staaten weitergegeben. Aufgrund der Bedeutung des Themas ist es aber notwendig, unvoreingenommen an SDBs arbeiten und die ermittelten Erkenntnisse über Schwachstellen mit anderen Forschern und interessierten Gruppen teilen zu können. Möglicherweise sind in Europa, z.B. aufgrund anderer gesetzlicher Bestimmungen, auch einige Ansätze möglich, die in den USA grundsätzlich ausgeschlossen sind. Da es beispielsweise in Deutschland nahezu keine Exportrestriktionen gibt, dürfte sich eine internationale Ausrichtung wesentlich einfacher gestalten.

Aus den oben genannten Gründen heraus wurde am Fachbereich Informatik der Technischen Universität Darmstadt (TUD) die Forschungsgruppe **TRUSTED**[1] ins Leben gerufen. Teil von TRUSTED ist das Projekt **SEC://HOUSE**, das sich u.a. mit der Erforschung von Schwachstellen in IT-Systemen beschäftigt.

Die vorliegende Arbeit beschäftigt sich mit den notwendigen Forschungs- und Entwicklungsarbeiten für den Aufbau einer Schwachstellendatenbank, die dann als Grundlage für die Forschungen im Bereich der Schwachstellen dienen wird. Es liegt außerdem eine CD mit Kopien der verwendeten elektronischen Quellen und der Quellcodes für die Erzeugung und Nutzung der SEC://HOUSE-SDB bei.

Im Rahmen dieser Arbeit werden Fachbegriffe oder Bezeichnungen, insbesondere bei deren erstem Auftreten, *kursiv* hervorgehoben. Bei der ausführlichen Besprechung von einzelnen Punkten einer Liste werden diese Unterpunkte i.d.R. **fett** hervorgehoben. Bezeichnungen von Klassen werden in `Schreibmaschinenschrift` dargestellt und Elemente oder Felder von Klassen werden wiederum *kursiv* hervorgehoben.

[1] **T**estbed for **R**eliable, **U**biquitous, **S**ecure, **T**ransactional, **E**vent-driven and **D**istributed Systems

Kapitel 1

Einleitung

In diesem Kapitel soll ein Überblick über die aktuelle Situation im Umgang mit Schwachstellen von IT-Systemen gegeben werden. Dies betrifft zum einen die Fehlerursachen selbst und zum anderen die Informationsquellen, die man benutzen kann, um sich vor neuen Schwachstellen zu schützen oder zumindest die Ausnutzung dieser Schwachstelle bis zur Verfügbarkeit eines Patches oder einer neuen Version der betroffenen Software zu verhindern.

1.1 Sicherheit von IT-Systemen

Die OECD definiert in [OEC97] Sicherheit von Informationssystemen als:

> Sicherheit von Informationssystemen ist der Schutz von Verfügbarkeit, Vertraulichkeit und Integrität. Verfügbarkeit bedeutet, daß ein Informationssystem im vorgesehenen zeitlichen Rahmen erreichbar und nutzbar ist. Vertraulichkeit bedeutet die Freigabe von Daten und Informationen nur an authorisierte Personen, Gruppen und Prozesse zu einer vorgegebenen Zeit und in einer vorherbestimmten Art und Weise. Integrität bedeutet, daß Daten und Informationen vollständig und genau sind und daß die Vollständigkeit und Genauigkeit über die Zeit erhalten bleiben. Die Gewichtung und Signifikanz der drei Faktoren kann dabei in Abhängigkeit von der Natur des Informationssystems variieren.

Zwar sind Informationssysteme nur eine Untermenge der IT-Systemen, jedoch ist die Definition weitgehend übertragbar, wobei darauf hingewiesen werden soll, daß auch andere Definitionen von Sicherheit existieren (siehe beispielsweise [HS99]).

Abseits der theoretischen Definition von Sicherheit muß man jedoch die übliche Wahrnehmung der Sicherheit von IT-Systemen betrachten. Dabei wird ein IT-System als sicher angesehen, wenn keine Schwachstellen bekannt sind. Dieses Gefühl der Sicherheit ist jedoch trügerisch, denn nur das Fehlen öffentlich bekannter Schwachstellen bedeutet noch nicht, daß diese nicht vorhanden sind und daß es niemanden gibt, der eine Schwachstelle im entsprechenden IT-System kennt und ausnutzen kann. Somit spiegelt die fehlende Wahrnehmung einer Bedrohung, die mit dem Gefühl der Sicherheit einhergeht, nur eine falsche Sicherheit vor.

1.2 Schwachstellen von IT-Systemen

Neben dem Typus von Schwachstelle, der in dieser Arbeit betrachtet wird, können weitere Arten von Schwachstellen betrachtet werden, die hier kurz dargestellt werden sollen. Eine mögliche Einteilung, die in dieser oder ähnlicher Form weit verbreitet ist, ist in [Kni00] zu finden.

Benötigte Zeit zur Nutzung	**Betriff Person**	**Betriff Computer**
Unmittelbar	„Social Engineering“	Logischer Fehler
Benötigt eine gewisse Zeitspanne	Fehlerhafte Richtlinie	Schwäche

Tabelle 1.1: Schwachstellentypen nach Ausnutzungsdauer und Ziel

Es werden in dieser Einteilung vier grundlegende Arten von Schwachstellen unterschieden, die relativ zu zwei Faktoren sind. Der erste Faktor bezieht sich auf das spezifische Ziel eines Angriffs, wobei zwischen Person und Computer unterschieden wird. Der zweite Faktor bezieht sich auf die Schnelligkeit, mit der eine solche Schwachstelle ausgenutzt werden kann. Tabelle 1.1 stellt die Einordnung dar.

Ein **Logischer Fehler** stellt eine Möglichkeit dar, einen, die Sicherheit verändernden Effekt auszunutzen. Eine solche Möglichkeit wird i.d.R. als elementarer Fehler betrachtet. Solche Probleme entstehen durch spezielle Gegebenheiten, meist durch schlecht geschriebenen Programmcode, die einen Zugang zu einem System mit größeren Rechten als beachsichtigt erlauben. Dies ist der Typ von Schwachstelle, an den gewöhnlich zuerst gedacht wird.

Eine **Schwäche** stellt eine Sicherheitsmaßnahme dar, die durch einen Entwurfsfehler zu einem Sicherheitseinbruch führen kann. Unter diesen Typus fallen i.d.R. Maßnahmen, die zwar an sich geeignet sind, die Sicherheit zu erhöhen, die aber umgangen werden können. Auch der klassische Begriff der „Security through Obscurity“ fällt in diesen Bereich. Darunter ist zu verstehen, daß versucht wird, ein System zu sichern, indem ein spezielles Geheimnis benutzt wird, das die Sicherheit solange erhöht, bis ein Dritter versteht, wie das Geheimnis „funktioniert“. Auch die Verschlüsselungsverfahren fallen unter den Typ „Schwäche“, da es möglich ist, jeden Code mit genügend Zeit- und Rechenaufwand zu brechen.[1] Kernpunkt dieses Schwachstellentyps ist, daß zwar Sicherheit vorhanden ist, es jedoch auch mehr oder minder effiziente Verfahren gibt, um diese Sicherheit auszuhebeln.

Unter **Social Engineering** ist ein direkter Angriff auf die Sicherheitsrichtlinien eines Unternehmens zu verstehen, der versucht, die Richtlinien über die Mitarbeiter des Unternehmens selbst zu umgehen. Darunter ist zu verstehen, daß ein Mitarbeiter selbst vorsätzlich Sabotage begeht, ein naiver Mitarbeiter von einem Dritten dazu gebracht wird Geheimnisse preiszugeben, was dann zu einer Schwachstelle führt, oder auch die Suche nach Informationen im Abfall eines Unternehmens.

Fehlerhafte Richtlinien schließlich beziehen sich auf Fehler bei der Planung zur Vermeidung bestimmter Situationen. Beispielsweise ist es fatal, wenn bei der Erstellung von Backups ein Fehler vorliegt, der zur Unbrauchbarkeit des selben führt, oder wenn Personen Zugriffsrechte auf Daten haben, die nicht für sie bestimmt sind.

[1] Eine Ausnahme stellen ausschließlich die sogenannten One-Time-Pads dar.

Diese Einteilung ist sicher kritikwürdig, jedoch durchaus geeignet, eine grobe Gliederung von Schwachstellen durchzuführen. Beispielsweise ist es durchaus denkbar, daß man Social Engineering auch über einen längeren Zeitraum durchführt, um einen bestimmen Geheimnisträger dazu zu bringen, Informationen preiszugeben. Dies wäre das klassische „Spionageszenario“, bei dem zuerst das Vertrauen einer Person erworben wird, um diese dann zu Indiskretionen zu verleiten. Ebenso kritikwürdig ist die Abgrenzung von logischen Fehlern und Schwächen. Im Grenzbereich zwischen diesen Typen kann es Schwachstellen geben, die nicht eindeutig zu einem der beiden Typen zugeordnet werden können. Beispielsweise gerade beim von Eric Knight zitierten Beispiel mit der Umgehbarkeit einer Sicherheitsmaßnahme kann es sich einerseits tatsächlich um eine Schwäche handeln, es kann dabei jedoch auch sehr wohl ein logischer Fehler vorliegen.

Für die Betrachtung von Schwachstellen eines IT-Systems ist dabei i.d.R. der Bereich der logischen Fehler, sowie der Bereich der Schwächen, der sich direkt in Produkten widerspiegelt, beispielsweise die Verwendung eines zu schwachen Verschlüsselungsverfahrens, interessant. Daher ist auch dieser Bereich unter dem Begriff der Schwachstelle, wie er in dieser Arbeit verwandt wird, zu verstehen.

Dabei ist es möglich, die logischen Fehler weiter nach dem Zeitpunkt ihrer Entstehung zu untergliedern. Die Fehler gehen, ähnlich wie auch die alltäglichen, nicht direkt sicherheitsrelevanten Programmierfehler, auf eine von mehreren Ursachen zurück, dies sind:

- Verständnisfehler
- Designfehler
- Implementierungsfehler
- Systemintegrationsfehler
- Konfigurationsfehler

Implementierungs-, und Systemintegrationsfehler können sowohl durch „echte“ Programmierfehler entstehen, wie auch durch Unvermögen oder Unkenntnis auf Seiten des Entwicklers, während für Designfehler i.d.R. letzteres gilt. So wird beispielsweise auch heute noch sicherheitssensitive Software regelmäßig von Entwicklern realisiert, die keine ausgewiesene Expertise im Bereich Sicherheit haben [BS00a].

Einer der häufigsten sicherheitsrelevanten Fehler ist der sogenannte *Buffer Overflow*, bei dem ein Programm einen Puffer für die Eingabe von Daten benutzt, jedoch nicht sicherstellt, daß der zur Verfügung stehende Speicherbereich auch ausreicht. Dadurch ist es einem potentiellen Angreifer möglich, durch eine besonders lange und geschickt gewählte Zeichenfolge den Speicherbereich hinter dem Ende des Puffers zu überschreiben. Wenn der Angreifer hier geschickt vorgeht, so ist es ihm möglich, mit den von ihm gelieferten Daten, einerseits ein „eigenes“ Stück Programmcode zu liefern, als auch die Rücksprungadresse des Unterprogramms, in dem das Programm sich gerade befindet, so zu überschreiben, daß statt eines Rücksprungs ein Einsprung in sein Codefragment erfolgt.

Buffer Overflows werden schon seit den sechziger Jahren als Problem diskutiert [BS00a], doch zählen sie immer noch zu den häufigsten Angriffsmöglichkeiten auf IT-Systeme. Einer der ersten *Würmer* der Internetgeschichte, der Morris Wurm, nutze einen Buffer Overflow

im Finger-Daemon, einem häufig eingesetzten Dienst. Weiter wurden im Jahr 1999 56% der sogenannten CERT-Advisories aufgrund von Buffer Overflows veröffentlicht. Und im Mai 2000 wurde erst wieder eine Warnung vor mehreren (!) Buffer Overflows im Kerberos-Authentifizierungs-System herausgegeben [CER00b], also ausgerechnet in einem der bekanntesten, meistgenutzten und als verhältnismäßig ausgereift geltenden Sicherheitssysteme, die heute existieren.

Ein weiteres Beispiel für Schwachstellen von IT-Systemen sind die sogenannten *Denial of Service*-Attacken (DoS), die darauf abzielen, die Nutzung eines Systems unmöglich zu machen, ohne dabei direkt Zugriff auf das System haben zu müssen. DoS-Attacken machen sich den Ressourcenverbrauch, der durch bestimmte Aktionen bewirkt wird, zu nutze, um möglichst so viele Ressource zu binden, damit aufgrund des daraus resultierenden Ressourcenmangels kein Zugriff auf das System mehr möglich ist. Ein Beispiel dafür wäre eine immens hohe Anzahl von Anfragen an einen HTTP-Server, so daß dieser mit der Bearbeitung der Anfragen beschäftigt ist und keine neuen Anfragen mehr annehmen kann. Es gibt jedoch auch DoS-Attacken, die durch Designschwächen ermöglich werden. Ein Beispiel für einen solchen Designfehler ist der TCP/IP-Verbindungsaufbau, bei dem durch das sogenannte SYN-Flooding ein DoS-Angriff besonders erleichtert wird [HS99].

Ein letztes Beispiel für ein Problem, das immer wieder zu Schwachstellen führt, ist das Problem der Fehlkonfiguration. Dabei werden durch die falsche Konfiguration von Systemkomponenten Schwachstellen erzeugt, die es einem potentiellen Angreifer ermöglichen, Zugriff auf das System zu nehmen. Konfigurationsfehler können einerseits durch den Entwickler oder andererseits durch den Anwender, der die Software installiert und konfiguriert, entstehen. Häufig bestehen Konfigurationsfehler in zu großzügig vergebenen Zugriffsrechten. Diese können bei der Installation vom Anwender so vergeben worden sein oder aber der Entwickler hat hier nicht die richtigen Einstellungen vorgenommen. So wurde beispielsweise Microsoft Windows NT 4 mit viel zu großzügigen Zugriffsrechten auf die Paßwortdaten ausgeliefert, obwohl dazu prinzipiell kein Grund bestand und es normalerweise wünschenswert wäre, Systeme in der Grundkonfiguration möglichst sicher einzurichten. Ein weiteres Beispiel ist ein Konfigurationsfehler in Microsoft Office 2000. Mit dieser Anwendung wurde ein ActiveX-Control ausgeliefert, das fälschlicherweise als „safe for scripting“ markiert war. Dadurch konnte jeder Betreiber einer WWW-Seite mittels Microsofts Active Scripting Funktionalität Office-Funktionen ausführen. Da hinter den Office-Funktionen jedoch die komplette Microsoft Visual Basic for Applications Programmiersprache steht, hätte eine Webseite beliebige Aktionen auf dem Rechner des Anwenders ausführen können [MSNS00].

Bei Konfigurationsfehlern, die durch den Anwender entstehen, ist allerdings häufig unklar, warum die Anwendungen nicht Testroutinen bereitstellen, die überprüfen, ob die Konfiguration, beispielsweise die der Zugriffsrechte, korrekt ist. An diesen Stellen verlassen sich die Entwickler häufig auf Annahmen über die Umgebung, in der ihre Anwendung ablaufen wird, die sich hinterher als falsch herausstellen und eine Schwachstelle generieren.

1.3 Charakteristische Informationsquellen

Zur Ermittlung von Informationen zu Schwachstellen von IT-Systemen lassen sich, in Anlehnung an [SMR00], verschiedene charakteristische Informationsquellen unterscheiden. Dabei können zwei Dimensionen betrachtet werden:

- Typ der Informationsquelle
- Informationsstruktur

1.3.1 Typ der Informationsquelle

Der Typ der Informationsquelle kennzeichnet den Verfasser bzw. die Quelle einer Meldung bzw. eines Beitrags zu einer Schwachstelle. Dabei sind eine Reihe unterschiedlicher Informationsquellen zu unterscheiden, auf die im folgenden genauer eingegangen werden soll.

Wichtige Attribute bei der Betrachtung von Typen von Informationsquellen sind:

- Seriosität
- Zuverlässigkeit
- Reaktionszeit

Hinsichtlich der im folgenden erläuterten Typen von Informationsquellen zeigt Tabelle 1.2 die Einordnung bezüglich der Attribute.

Typ der Informationsquelle	Seriosität	Zuverlässigkeit	Reaktionszeit
CERT (Advisories)	hoch	hoch	niedrig
CERT (sonstige Meldungen)	hoch	hoch	mittel
Hackergruppen	niedrig	i.d.R. hoch	hoch
Sicherheitsunternehmen	mittel - hoch	mittel - hoch	niedrig - mittel
Newsgroups / Mailinglisten	niedrig - hoch	niedrig - hoch	hoch
Bücher	mittel - hoch	mittel - hoch	sehr niedrig

Tabelle 1.2: Einordnung der Typen von Informationsquellen

1.3.1.1 Computer Emergency Response Teams

Die *Computer Emergency Response Teams* (CERTs) sind häufig nicht gewinnorientierte Organisation, die sich aus größeren Projekten oder Organisationen heraus entwickelt haben. Der Ursprung der CERTs liegt im amerikanischen CERT Coordination Center (CERT/CC) [CER00e]. In Deutschland wäre hier beispielsweise das DFN-CERT [DFN00] zu nennen, das sich aus dem Deutschen Forschungsnetz (DFN) heraus entwickelt hat. Ein weiteres Beispiel für ein CERT ist das australische CERT [CER00a]. Ziel der CERTs ist es, Informationen über Schwachstellen zu sammeln und bereitzustellen. Dies geschieht i.d.R. im Rahmen der sogenannten *Advisories*. Dabei warnen die CERTs regelmäßig nur vor extrem gefährlichen oder aber einen besonders breiten Personenkreis betreffenden Schwachstellen und beobachten darüber hinaus auch Gebiete wie *Viren* oder *Trojanische Pferde*. Da die CERTs

jedoch großen Wert auf Vollständigkeit der Informationen innerhalb der Advisories legen, sind diese nicht immer aktuell, da eine gewisse Zeit benötigt wird, um die Informationen zu allen Aspekten einer Schwachstelle sowie den Möglichkeiten, die Schwachstelle zu beseitigen, zusammenzutragen. Außerdem verfolgen viele CERTs die Strategie, Schwachstellen erst dann zu veröffentlichen, wenn deren Beseitigung sichergestellt ist, so daß oft eine Veröffentlichung eines Advisorys weiter verzögert wird.

Über die Advisories hinaus gibt es beim CERT/CC zwei weitere regelmäßige Mitteilungstypen: einerseits gibt es die *Vulnerability Notes* [CER00j], die Informationen über neu entdeckte Schwachstellen enthalten. Dabei können sich aus Vulnerability Notes zu einem späteren Zeitpunkt Advisories entwickeln. Zum zweiten gibt es die *Incident Notes* [CER00i], die Informationen über ungewöhnliche Vorfälle „im Internet“ enthalten, die sich möglicherweise später als Versuche zur Ausnutzung einer Schwachstelle erweisen könnten.

Um die aktuellen CERT-Advisories zu erhalten kann man entweder die WWW-Seiten des jeweiligen CERT nutzen oder die i.d.R. vorhandenen Mailinglisten abonnieren.

Neben diesen öffentlichen, nationalen CERTs gibt es noch eine Reihe weiterer Organisationen, die sich auf Informationen und Beratung zum Thema IT-Sicherheit spezialisiert haben. Dabei handelt es sich nicht selten um staatliche Organisationen. Beispielsweise ist „*CIAC*“ [CIA00], die Computer Incident Advisory Capability, eine solche Organisation. Sie ist eine Einrichtung des amerikanischen Energieministeriums und unterstützt auf Anfrage alle Einrichtungen des Energieministeriums bei Vorfällen, die die IT-Sicherheit betreffen. Eine ähnliche Rolle, nur für die amerikanische Bundesregierung und ihre Einrichtungen spielt *FedCIRC* [Fed00], die Federal Computer Incident Response Capability.

1.3.1.2 Hackergruppen

Eine weitere wichtige Quelle für Informationen über Schwachstellen stellen *Hackergruppen* dar. Dabei handelt es sich um Gruppen von Individuen, die sich, häufig mit nicht eindeutig legalen Zielen, aber mit sehr hohem Engagement und Know-How, damit beschäftigen, Schwachstellen in IT-Systemen aufzuspüren. Dieses Wissen wird dann zumindest von einigen Hackergruppen veröffentlicht. Dabei wird dann i.d.R. nicht darauf geachtet, ob Dritten durch das frühzeitig veröffentlichte Wissen über Angriffsmöglichkeiten möglicherweise Schaden zugefügt werden könnte.

Die Hackergruppen nutzen zum Informationsaustausch häufig eigene WWW-Seiten, manchmal auch öffentliche Newsgroups im USENET oder auch IRC-Kanäle und ICQ-Kommunikation[2].

Beispiele für Hackergruppen, die im Internet eigene Web-Seiten betreiben, sind der Chaos Computer Club (CCC) [CCC00], Phrack [Phr00] oder auch L0phT [L0p00], eine Gruppe, die inzwischen auch als IT-Sicherheits Beratungsunternehmen tätig ist.

[2] Siehe auch `http://www.icq.com`

1.3.1.3 Sicherheitsunternehmen

Viele Beratungsfirmen für IT-Sicherheit und Hersteller von Sicherheitssoftware veröffentlichen regelmäßig Informationen zu Schwachstellen. Dies geschieht allerdings nicht uneigennützig, sondern dient als Werbung, als „Beweis“ für die eigene Kompetenz und, im Falle der Softwareherseller, natürlich auch, um Werbung für das eigene, Schutz bietende Produkt zu machen.

INFILSEC Systems Security [Sec00a] bezeichnet seine Schwachstellendatenbank beispielsweise als eine „Vulnerability Engine“, die als Werkzeug für Hersteller, Systemadministratoren, Sicherheitsberater und Analysten dienen soll und verfolgt damit den Aufbau und Betrieb eines zentralen Repositorys für Schwachstellen von Betriebssystemen, Anwendungen und Protokollen. Außerdem werden Informationen darüber, wie diesen Schwachstellen begegnet werden kann, gespeichert. INFILSEC will die Ergebnisse von Mailing-Listen wie Bugtraq extrahieren und über seine Suchmaschine zur Verfügung stellen. Dazu stellt INFILSEC ein Online-Update-System zur Verfügung, das die Möglichkeit bietet, Informationen in das System einzuspeisen.

Ein weiteres Beispiel ist das Unternehmen Internet Security Systems (ISS) [ISS00d]. ISS verkauft Sicherheitssoftware und -beratung und betreibt daneben die Schwachstellendatenbank X-Force [ISS00a].

1.3.1.4 IT-Unternehmen

Eine weitere Quelle stellen IT-Unternehmen selbst dar, die Informationen zu Schwachstellen in ihren Produkten veröffentlichen. Dabei gibt es jedoch nahezu kein Unternehmen, das solche Informationen aus freien Stücken heraus veröffentlicht. Meist werden diese Informationen erst veröffentlicht, wenn eine Schwachstelle durch eine der anderen Quellen bekannt wird. Dabei muß davon ausgegangen werden, daß im Regelfall nur die Informationen veröffentlicht werden, die im großen und ganzen schon bekannt sind und daß keine initiative Meldung von Schwachstellen, die bisher vermeintlich nur der Hersteller kennt, stattfindet. Als Beispiel für die herstellerseitige Veröffentlichung von Informationen über Schwachstellen kann die Microsoft Security Mailingliste [MC00] dienen.

1.3.1.5 Newsgroups und öffentliche Mailinglisten

Unterstellt man, daß die „echten“ Hacker sowieso über aktuelle Informationen zu Schwachstellen verfügen, stellen Newsgroups im USENET und spezielle Mailinglisten die aktuellsten, öffentlich zugänglichen Quellen für Schwachstelleninformationen dar. Informationen zu diesen Quellen liefern Hacker, Mitarbeiter von IT-Unternehmen und andere IT-Profis. Newsgroups, in denen sicherheitsrelevante Themen diskutiert werden, sind beispielsweise:

- `comp.security.unix`
- `comp.security.ssh`
- `comp.security.misc`

- `de.comp.security`
- `comp.lang.java.security`
- `comp.os.ms-windows.nt.admin.security`
- `comp.os.netware.security`
- `comp.security.firewalls`

Eine Liste von Quellen, die über sicherheitsrelevante Newsgroups informieren, findet sich im Anhang dieser Arbeit in Abschnitt A.9.

Sicherheitsbezogene Mailinglisten existieren ebenfalls in großer Anzahl. Zu den bedeutendsten zählen:

- Bugtraq [Sec00b]
- NT Bugtraq [NTB00]
- Alert [ISS00c]
- NT Security [ISS00c]

Eine Liste von Quellen, die über sicherheitsrelevante Mailinglisten informieren, findet sich im Anhang dieser Arbeit in Abschnitt A.8.

1.3.1.6 Bücher

Abschließend sollen noch Bücher als Quelle für Informationen zu Schwachstellen von IT-Systemen genannt werden. Dabei können Bücher naturgemäß keine hochaktuellen Themen behandeln. Dafür können Sie sehr ausführlich Konzepte von *klassischen*, aber immer wieder begangenen Fehler beim Softwaredesign oder bei Implementierungen der IT-Systeme darstellen. Ein Beispiel für ein (Online-)Buch stellt die Abhandlung über Computer Schwachstellen von Eric Knight dar [Kni00].

1.3.2 Informationsstruktur

Informationen zu Schwachstellen können in verschiedenen Formen vorliegen, die eine systematische Recherche oder auch eine Analyse von Informationen über Schwachstellen unterschiedlich gut unterstützt. Auf die entsprechenden Möglichkeiten soll im folgenden kurz eingegangen werden. Dabei ist es prinzipiell unerheblich, ob die Informationen in Form von reinen Textdateien (Text, e-Mail, Newsgroupbeiträge), HTML-Seiten oder als „Datenbank-Einträge“ vorliegen, da hier prinzipiell nur die Effizienz des Zugriffs und der Recherche zu unterscheiden ist, nicht aber die Struktur selbst.

Die vorgestellte Gliederung wurde im Rahmen dieser Arbeit entworfen, um bereits frühzeitig in der Lage zu sein, die Eignung von verschiedenen Informationsquellen als Quellen für eine (semi-)automatische Übernahme von Daten in die zu entwickelnde SEC://HOUSE-SDB beurteilen zu können.

1.3.2.1 Unstrukturierte Informationsquellen

Quellen, die in freier From vorliegen und keinerlei vorgegebene Struktur besitzen, können nur schwer zu einer automatisierten Verarbeitung und Analyse herangezogen werden. Unter die unstrukturierten Quellen fallen hierbei viele „Veröffentlichungen“ von Hackergruppen, viele HTML-Seiten, die die textuelle Information quasi nur mit bestimmten „Auszeichnungen“ für die Anzeige versehen, die jedoch keine echte Struktur erzeugt, IRC- und ICQ-Kommunikation und teilweise e-Mails.

1.3.2.2 Schwach strukturierte Informationsquellen

Eine minimal höhere Struktur kann man häufig aus Diskussionen in Newsgroups und Mailing-Listen ableiten, indem man die Hierarchie der Diskussionen anhand Betreff-Zeilen oder der *References*- bzw. *In-Reply-To*-Einträge in den Headern der Diskussionsbeiträge nutzt. Außerdem kann man, bei Vorhandensein der häufig verwandten Zitierweise in e-Mails und Newsgroup-Beiträgen möglicherweise auch noch Strukturen innerhalb der Nachrichten nutzen. Diese Zitierweise zeichnet sich dadurch aus, daß der Teil einer Nachricht, auf den sich ein Kommentar bezieht, mittels „>“-Zeichen eingerückt wird und dann unter dem entsprechenden Abschnitt der Kommentar geschrieben wird. Nicht benötigte Abschnitte der Originalnachricht sollten dabei gelöscht werden.

Da jedoch die beschriebenen Strukturen nicht zwingend vorhanden sind, kann eine automatisierte oder auch nur semi-automatisierte Analyse hier häufig versagen.

1.3.2.3 Strukturierte Informationsquellen

Für eine (semi-)automatische Verarbeitung eignen sich strukturierte Quellen wesentlich besser als unstrukturierte bzw. schwach strukturierte, da i.d.R. bestimmte Informationsabschnitte in einer bestimmte Reihenfolge vorliegen, so daß man den Inhalt dieser Abschnitte relativ einfach für eine Weiterverarbeitung extrahieren kann. Dabei handelt es sich bei den einzelnen Abschnitten meist um beschreibenden Text und nicht um kurze und prägnante Attribute.

Insbesondere Advisories, unabhängig davon, ob in Form einer e-Mail oder als HTML-Seite, liegen häufig als auf diese Weise strukturierte Quellen vor, darunter fallen sowohl die CERT-Advisories [CER00e], als auch die im Web abrufbaren Advisories von L0pht [L0p00], die in ihrer Kurzform als HTML-Seiten vorliegen und in einer ausführlichen Fassung als Textdateien heruntergeladen werden können.

Im Falle eines Advisories des CERT/CC sieht dabei die grobe Struktur wie folgt aus:

- Betroffene Systeme
- Überblick
- Beschreibung
- Original Advisory (optional)

- Auswirkung
- Lösung
- Anhang A. Informationen zu einzelnen Herstellern des Produkts
- Weitere Anhänge (optional)

Ebenfalls unter diese Kategorie der Informationsquellen können Datenbanken fallen, nämlich dann, wenn im Endeffekt nur die einzelnen Abschnitte der meist sehr „textlastigen“ Advisories in einzelne Datenbankfelder übernommen werden. Durch die Speicherung in der Datenbank ist zwar ein schneller und einfacher Zugriff auf die einzelnen Abschnitte der Advisories möglich, jedoch werden Analysen der gespeicherten Daten kaum erleichtert.

Beispiele für solche Datenbanken sind INFILSEC [Sec00a] und X-Force [ISS00a].

1.3.2.4 Stark strukturierte Informationsquellen

Unter einer stark strukturierten Informationsquelle sollen solche Quellen verstanden werden, die nicht nur benutzt werden, um textbasierte Informationen zu speichern, sondern neben den notwendigen, beschreibenden Texten auch möglichst viele bewertende Attribute zu Schwachstellen enthalten, die auch ohne die Informationen aus den Texten eine möglichst eindeutige Klassifikation der Schwachstelle erlauben. Dadurch sollten Analysen wesentlich einfacher möglich sein, als bei rein textbasierten Quellen.

Keine der öffentlich bekannten Schwachstellendatenbanken erfüllt diese Anforderung an eine stark strukturierte Datenbank. Einzig die Schwachstellendatenbank, die im Rahmen der Dissertation *Software Vulnerability Analysis* von Ivan Victor Krsul [Krs98] bei den COAST Laboratories, inzwischen Teil des „Center for Education and Research in Information Assurance and Security“ (CERIAS), entwickelt wurde, kann als stark strukturierte Informationsquelle eingestuft werden.

1.3.3 Nichtöffentliche Quellen

Neben den weiter oben erwähnten und häufig öffentlichen Quellen zu Software-Schwachstellen, existieren vermutlich noch eine große Anzahl nicht öffentlich zugänglicher SDBs, deren Existenz teilweise noch nicht einmal bekannt sein dürfte. Eine Datenbank, deren Existenz bekannt ist, ist *VulDa* von IBM, die nach Auskunft von IBM ausschließlich für den internen Gebrauch bei IBM erstellt wurde, aber keine Spezialisierung auf bestimmte IBM-spezifische Bereiche aufweist [AD99]. Allein die von IBM für die Aktualisierung von VulDa genutzten öffentlichen Quellen vermitteln einen Eindruck über den zu betreibenden Aufwand: mehr als 30 Newsgroups, 60 Mailinglisten, Spiegelungen von über 45 FTP-Servern und Kopien von mehreren Dutzend „Hacker-Seiten“. Vermutlich werden auch öffentlich zugängliche SDBs als Quellen genutzt. Im Juli 2000 umfaßte VulDa circa 18 GByte komprimierte Daten.

Zusätzlich zu ihren öffentlichen Informationsangeboten dürften viele Softwarehersteller Datenbanken von Schwachstellen ihrer Produkte unterhalten, deren Existenz bisher nicht an die Öffentlichkeit gelangt ist.

Als wichtige Betreiber weiterer nicht öffentlicher SDBs kommen beispielsweise Geheimdienste in Frage, die ihre SDBs möglicherweise auch zu anderen Zwecken als der bloßen Abwehr von Angriffen auf ihre IT-Systeme nutzen möchten (z.B. Gegenangriffe, aber auch „Information Warfare“) und wenig Interesse daran haben dürften, diese Informationen allgemein zugänglich zu machen.

1.3.4 Einordnung von Informationsquellen nach ihrem Strukturierungsgrad

Abbildung 1.1 zeigt eine exemplarische Einordnung verschiedener betrachteter Informationsquellen nach ihrem Strukturierungsgrad und der Fähigkeit zur (semi-)automatischen Weiterverarbeitung der Daten.

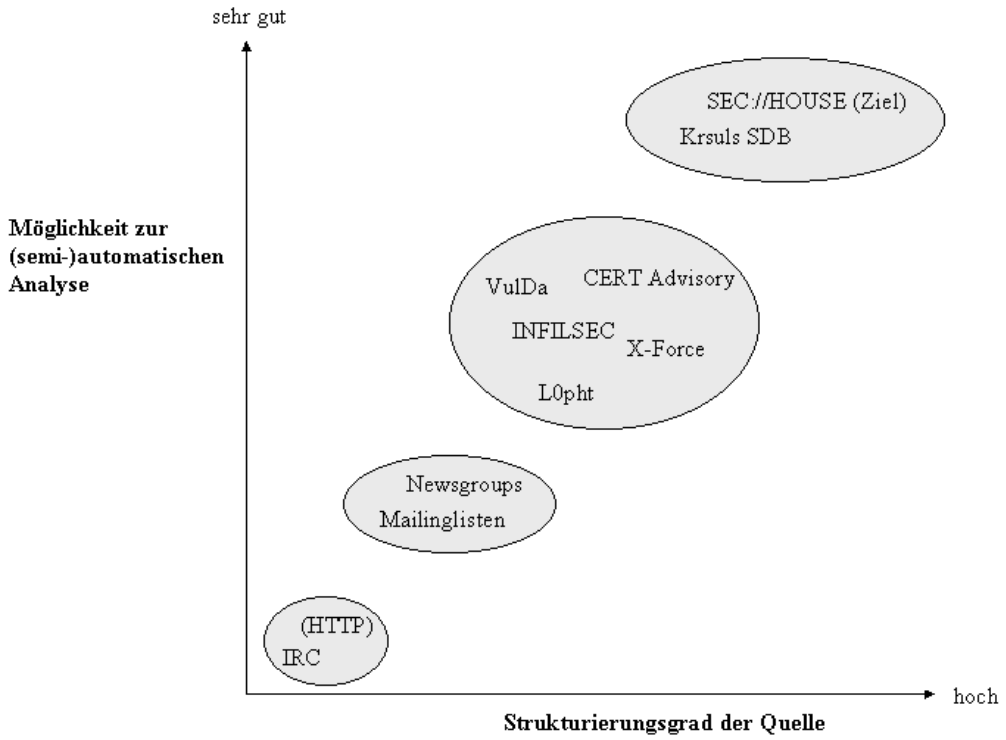

Abbildung 1.1: Einordnung von Informationsquellen nach ihrem Strukturierungsgrad

1.4 Überblick über die weitere Arbeit

Kapitel 2 befaßt sich mit bisherigen Arbeiten auf dem Gebiet der Erfassung von Schwachstellen bzw. der Schwachstellendatenbanken. Dabei werden einige der wichtigsten Informationsquellen zur Erfassung von Schwachstellen betrachtet.

Im 3. Kapitel werden Anforderungen und Voraussetzungen für einen erfolgreichen Betrieb einer Schwachstellendatenbank untersucht. Dabei geht es insbesondere um den Entwurf eines tragfähigen Geschäftsmodell für eine SDB. Außerdem werden in diesem Abschnitt die

Ergebnisse einer öffentlichen Umfrage zum Betrieb einer Schwachstellendatenbank vorgestellt und schließlich zwei alternative Geschäftsmodelle vorgeschlagen. Weiterhin werden die notwendigen Zugangswege eines potentiellen Nutzers zu einer SDB dargestellt. Danach werden kurz auf die Möglichkeiten für eine initale Füllung der SDB betrachtet. Schließlich werden die notwendigen Informationen betrachtet, die bezüglich Schwachstellen erfaßt werden können sollten, um ein umfassendes „Bild“ einer Schwachstelle erfassen zu können. Und abschließend werden einige Anwendungsfälle bei der Benutzung dargelegt.

Kapitel 4 betrachtet die Implementierung der SEC://HOUSE Schwachstellendatenbank. Nach der Präsentation des Datenmodells der SDB, wird zuerst eine geeignete Architektur für den Zugriff auf die Schwachstellendatenbank vorgestellt. Und schließlich wird auf die prototypische Implementierung der SEC://HOUSE-Administrator-Schnittstelle eingegangen, die im Rahmen dieser Diplomarbeit entwickelt wurde.

Im Anhang schließlich findet sich, neben den Verweisen auf Quellen und Quellcodes, die auf der, dieser Diplomarbeit beiliegenden CD-ROM zu finden sind, eine ausführliche Darstellung der Ergebnisse der im Rahmen dieser Diplomarbeit durchgeführten Umfrage. Danach folgt eine ausführliche Erläuterung der Klassen des Datenmodells. Abschließend folgt eine Übersicht über die bestehenden Mailinglisten und Newsgroups, die sich (u.a.) mit dem Thema Schwachstellen befassen.

Kapitel 2

Verwandte Arbeiten

Der Schwerpunkt der bisherigen wissenschaftlichen Forschung auf dem Gebiet der SDBs liegt in den USA. Dabei dienten den Forschern zwei „invitational“ Workshops als Foren. Der erste Workshop fand im Juni 1996 unter dem Titel „Workshop on Computer Vulnerability Data Sharing“ statt. Die Nachfolgeveranstaltung im Januar 1999 wurde unter dem Titel „2nd Workshop on Research with Security Vulnerability Databases“ durchgeführt. Der zweite Workshop behandelte neben technischen Fragen, motivierende Aspekte und mögliche Folgen der Unterhaltung von SDBs. Ein weiterer wichtiger Teil des Workshops beschäftigte sich mit Vor- und Nachteilen von unterschiedlichen Realisierungsmodellen für SDBs [MS99].

Eine zentrale Rolle bei der Forschung auf dem Gebiet der SDBs spielt das *Center for Education and Research in Information Assurance and Security* an der Purdue University. Eine grundlegende Arbeit über Schwachstellen stellt, wie bereits erwähnt, die Dissertation von Ivan Victor Krsul mit dem Thema „Software Vulnerability Analysis“ [Krs98] dar, die in Abschnitt 2.1.1 ausführlich dargestellt wird.

Weitere Arbeiten fanden auch bei MITRE, einer privaten, staatlich geförderten Forschungsanstalt, statt. Dort wird an „Common Vulnerabilities & Exposures“ (CVE) [MC99] gearbeitet, das eine Austauschbarkeit der Einträge verschiedener SDBs über eine gemeinsame Bezeichnung gewährleisten soll.

Außerdem wird die Forschung auf dem Gebiet der SDBs von verschiedenen staatlichen Organisationen, großen Industrieunternehmen und Unternehmen, die auf das Gebiet der IT Sicherheit spezialisiert sind, vorangetrieben. Aufgrund der Beteiligung der amerikanischen Regierung kann leider nicht immer davon ausgegangen werden, daß alle Forschungsergebnisse bezüglich der Konzeption und der Nutzung von SDBs öffentlich zugänglich gemacht werden.

2.1 Ausgewählte Beispiele bestehender SDBs

Im folgenden soll eine Übersicht über eine Auswahl von charakteristischen Informationsquellen über Schwachstellen gegeben werden. Dabei soll betrachtet werden, welche Daten

über Schwachstellen selbst, aber gegebenenfalls auch über Vorfälle bei laufenden Systemen (*Incidents*) erfaßt werden. Außerdem soll kurz auf die Schnittstellen zu diesen Informationsquellen eingegangen werden, d.h. auf die Möglichkeiten auf diese Quellen zuzugreifen.

Der Betrachtung von Vorfällen sollte deshalb Aufmerksamkeit zugedacht werden, da zum einen Vorfälle häufig vorliegen, bevor eine dazugehörige Schwachstelle bekannt ist und zum anderen, weil eine gute Beschreibung eines Incidents möglicherweise dafür genutzt werden kann, um ähnliche, potentiell angreifbare Systeme ermitteln zu können.

2.1.1 Krsuls SDB

Ivan Victor Krsul implementierte im Rahmen seiner Dissertation *Software Vulnerability Analysis* [Krs98] an der Purdue Univeristät in der USA eine Schwachstellendatenbank, die er für die Untersuchung von Schwachstellen nutzte. Diese ist im CERIAS anscheinend auch im Einsatz, sie ist jedoch nicht ohne weiteres von außen nutzbar.

Das Ziel Krsuls bestand in einer einheitlichen Definition von Software-Schwachstellen, die auf der Idee basiert, daß es die Sicherheitsrichtlinien (*Security Policy*) sind, die definieren, was in einem System gestattet werden kann oder wünschenswert ist. Auf dieser Idee aufbauend entwickelte Krsul eine Klassifikation von Schwachstellen, die sich u.a. auf die Annahmen bezüglich der Laufzeitumgebung der Software konzentriert, die Entwickler während der Implementierung treffen und die regelmäßig während der Ausführung der Software nicht zutreffen.

Dabei baute Krsul auf früheren Arbeiten zu Software-Schwachstellen auf und versuchte eine Taxonomie für Software-Schwachstellen aufzustellen. Tatsächlich stellt Krsul dabei zwei Taxonomien auf: Die erste ermöglicht eine a posteriori Klassifikation von Schwachstellen, die sich sehr gut für die Einordnung von tatsächlich aufgetretenen Schwachstellen eignet. Die zweite Taxonomie stellt eine a priori Klassifikation von Schwachstellen dar und ist für das bessere Verständnis und die Vermeidung derselben geeignet. Diese Taxonomie wird jedoch nur grob skizziert.

Krsul betrachtet bei seiner Analyse ausschließlich Schwachstellen, Vorfälle finden keine Berücksichtigung.

2.1.1.1 Klassifikation von Schwachstellen

Für die Einordnung von Schwachstellen verwendet Krsul verschiedene, beschreibende Kategorien [Krs98, S. 39 ff. und S. 162 ff.]:

- Merkmale der Bedrohung (Threat Features)
- Merkmale der Annahmen über die Laufzeitumgebung (Environmental Assumption Features)
- Merkmale der Art der Schwachstellen (Features on the Nature of Vulnerabilities)
  - Betroffene Objekte (Objects Affected)

  - Auswirkung auf Objekte (Effect on Objects)
  - Benutzte Methode oder benutzter Mechanismus (Method or Mechanism Used)
  - Typ der Eingabe (Input Type)
- Unmittelbare Auswirkungen (Direct Impact)
- Mittelbare Auswirkungen (Indirect Impact)
- Benötigter Zugang (Access Required)
- Schwierigkeitsgrad der Ausnutzung der Schwachstelle (Complexity of Exploit)
- Kategorie der Systemkomponente (Category)
- Betriebssystem (OS Type)

Einige dieser Kategorien enthalten eine Reihe von Attributen, die verschiedenen Merkmalen entsprechen. Jedes dieser Merkmale kann mit der Aussage *Ja, Nein, Nicht anwendbar* oder *Unbekannt* belegt werden. Andere Kategorien führen anhand von Entscheidungsbäumen zu einer eindeutigen Aussage.

Unter **Threat Features** sind die unmittelbaren Bedrohungen durch das Vorhandensein der Schwachstelle zu verstehen. Ein Beispiel hierfür wäre, daß die Ausnutzung der Schwachstelle zum Ergebnis hat, daß ein Benutzer, unter Verletzung der Sicherheitsrichtlinien, bestimmte Daten einsehen kann, die nicht für ihn bestimmt sind.

Die **Environmental Assumption Features** repräsentieren die Annahmen, die ein Entwickler bei der Implementierung getroffen hat und die dann zur Laufzeit nicht zutreffen. Beispielsweise könnte ein Entwickler voraussetzen, daß ein bestimmtes Objekt zum Zeitpunkt der Ausführung nicht existiert.

Die Kategorie **Objects Affected** bestimmt die Objekte, die durch die Schwachstelle unmittelbar betroffen sind, dies können beispielsweise Benutzerdateien, Verzeichnisse oder ausführbarer Code auf dem Stack eines laufenden Programms sein.

Effects on Objects betrifft die Auswirkungen, die die Schwachstelle auf die betroffenen Objekte hat. Diese können beispielsweise sein, daß das Objekt ersetzt oder gelesen werden kann oder daß die Zugriffsrechte auf das Objekt geändert werden können.

Die Kategorie **Method or Mechanism Used** beschreibt die Methode oder den Mechanismus, der verwendet wird, um ein betroffenes Objekt zu beeinflussen. Dies kann beispielsweise die Verwendung eines symbolischen Links unter Unix sein, es kann ein Konfigurationsfehler vorliegen oder auch eine fehlerhafte Implementierung bezüglich der aktuellen Umgebung, was i.d.R. einer falschen Annahme über die Laufzeitumgebung entspricht.

Input Type identifiziert die Quelle der Eingabe, die verwendet werden kann, um durch Ausnutzung der Schwachstelle ein oder mehrere Objekte zu beeinflussen.

Direct Impact identifiziert die unmittelbare Auswirkung bei Ausnutzung der Schwachstelle.

Indirect Impact bewertet die mittelbare, „ultimative“ Auswirkung der Schwachstelle.

Access Required gibt an, welchen Zugang ein Angreifer zu dem angegriffenen System mindestens haben muß, um die Schwachstelle ausnutzen zu können.

Complexity of Exploit stuft die Ausnutzung der Schwachstelle nach ihrem Schwierigkeitsgrad ein.

Category beschreibt die Systemkomponete, der die Schwachstelle zuzurechnen ist.

OS Type gibt an, welche Betriebssysteme von der Schwachstelle betroffen sind.

Für die Charakteristiken Indirect Impact, Direct Impact, Access Required, Complexity of Exploit, Category und OS Type hat Krsul Entscheidungsbäume entwickelt, die es erlauben, Schwachstellen eindeutig einem der möglichen Werte der jeweiligen Kategorie zuzuordnen. Bei den Charakterisierungen hat Krsul dabei die unzähligen, bereits bestehenden Charakterisierungen untersucht, ihre Schwachstellen analysiert und daraus abgeleitet die Entscheidungsbäume gemäß des *fundamentum divisionis* Prinzips [Krs98, S. 23] entwickelt, so daß es keine Mehrdeutigkeiten beim „Abschreiten“ der Entscheidungsknoten des Entscheidungsbaumes gibt und man zu einer deterministischen Einstufung gelangt. Dies stellt einen der wichtigsten Teile der Dissertation dar.

2.1.1.2 Datenbankschema

Obwohl sich für die die Verarbeitung der Datensätze bei Krsuls komplexem Schema die Verwendung eines Datenbankmanagementsystems (DBMS) geradezu aufgedrängt hätte, verwendet Krsul zur Speicherung der Daten eine Textdatei.

Krsul verwendet zur Speicherung der Daten folgendes Schema:

- Bezeichnung der Schwachstelle (Identification Section)
- Verzeichnis der Modifikationen des Eintrags (Modification Section)
- Beschreibung und Auswirkung (Description and Impact)
- Quelle der Informationen (Information regarding the Source of the Information)
- Identifikation der von der Schwachstelle betroffenen Systeme (System Identification)
- Information zur betroffenen Anwendung (Application Information)
- Referenzen (References)
- Detaillierte Analyse, Möglichkeiten zur Erkennung der Schwachstelle und Gegenmaßnahmen (Detailed Analysis, Detection Techniques, and Fixes)
- Detaillierte Informationen über die Ausnutzung der Schwachstelle (Detailed Information about Exploitation)
- Quellcode oder Verweise auf den Quellcode der Systeme, die die Schwachstellen enthalten (Source code and pointers to source code for the systems that contain the vulnerabilities)

- Klassifikation des Fehlers (Fault Classification)
- Kategorie und Komponentenklassifikation (Category and Component Classification)
- Identifikation der Art der Schwachstelle (Identification of Nature of the Vulnerability)
- Bestätigung der Überprüfung der Schwachstelle (Verification of the Vulnerability)
- Identifikation der Verletzung der Sicherheitsrichtlinien (Identification of Policy Violation)
- Identifikation der Umgebungsfaktoren, die zu der Schwachstelle geführt haben (Identification of Environmental Factors)
- Identifikation der Art der Bedrohung (Identification of the Nature of Threat)

Category and Component Classification Der Abschnitt enthält Informationen über das System oder die Komponente, dem / der die Schwachstelle zuzuordnen ist, beispielsweise dem Betriebssystem-Kernel.

Identification of Nature of the Vulnerability Dieser Abschnitt enthält die oben beschriebenen Einstufungen der Schwachstellen.

Identification of the Nature of Threat „Identification of the Nature of Threat“ enthält die Angaben über die Threat Features.

2.1.2 CERT

Anhand der Prozesse im CERT Coordination Center (CERT/CC), dem ältesten und bekanntesten CERT, soll die Erfassung eines Vorfalls und die Schwachstellenmeldung eines CERT analysiert werden.

Bei den Angaben, die das CERT erhält, wird auf ein hohes Maß an Vertrauenswürdigkeit des CERT gesetzt, da Systeminformationen in allen Details abgefragt werden und beispielsweise ein Unternehmen ein Interesse hat, in der Öffentlichkeit in Zusammenhang mit Einbrüchen bzw. Einbruchsversuchen in IT-Systeme nicht genannt zu werden.

2.1.2.1 Erfassung von Vorfällen

Zur Erfassung von Vorfällen bietet das CERT/CC eine Vorlage für ein Meldeformular an. Die hier betrachtete Fassung dieses Formulars ist Version 5.2 vom April 2000 [CER00f]. Darüber hinaus stellt das CERT/CC ausführliche Richtlinien für die Meldung von Vorfällen bereit [CER00g].

Die Daten, die zu Vorfällen erfaßt werden, sind in fünf Kategorien aufgeteilt, die im folgenden beschrieben werden:

Kontaktinformationen Zu den Kontaktinformationen zählt das CERT den Namen des Melders und der Organisation, für die er arbeitet und den Wirtschaftssektor, in dem die Organisation tätig ist. Zur Kontaktaufnahme dient die Angabe einer e-Mail-Adresse und einer Telefonnummer. Außerdem können noch beliebige andere Informationen vom Melder angegeben werden.

Betroffener Rechner In dieser Kategorie, die für jeden betroffenen Rechner angegeben werden soll, werden der Rechnername respektive die IP-Adresse, die Zeitzone, in der der Rechner arbeitet, sowie eine möglichst exakte Beschreibung der Funktion des Rechners erfaßt.

Ursprung des Angriffs Auch diese Kategorie soll für jeden angreifenden Rechner angegeben werden. Erfaßt werden dabei der Rechnername bzw. die IP-Adresse des angreifenden Rechners, die Zeitzone, in der dieser Rechner arbeitet, sowie Angaben darüber, ob man mit dem zuständigen Systemoperator bereits in Kontakt steht. Letzteres ist deshalb sinnvoll, da heute die meisten Angriffe von Rechnern aus stattfinden, die bereits durch Ausnutzung von Schwachstellen unter Kontrolle des Angreifers gebracht worden sind.

Geschätzte Kosten Die, soweit möglich, geschätzten Gesamtkosten, die durch den Vorfall verursacht werden.

Ausführliche Beschreibung des Vorfalls Die Beschreibung des Vorfalls soll so viele Details zu dem Vorfall enthalten, wie möglich. Dazu zählen:

- Angaben zu den Zeitpunkten der Angriffe
- Methoden, die bei den Einbruchsversuchen angewandt wurden
- Tools, die der Angreifer eingesetzt hat
- Eingesetzte Softwareversionen und Patchlevel
- Ausgaben, die von den Tools, die der Angreifer eingesetzt hat, erzeugt wurden
- Details zu den Schwachstellen, die ausgenutzt wurden
- Ursprung des Angriffs
- Alle weiteren möglicherweise relevanten Informationen

2.1.2.2 Meldung von Schwachstellen

Wird in einem IT-System eine Schwachstelle gefunden, so kann sie dem CERT gemeldet werden. Dazu ist wiederum ein Formular vorgegeben, das die, nach Ansicht des CERT, notwendigen Informationen enthält [CER00h].

Neben den Kontaktinformationen werden dort folgende Punkte aufgeführt:

- Ist eine *Meldung an den Hersteller* erfolgt, wenn ja, wann und an wen (Datum, Name des Kontakts, Telefonnummer, e-Mail-Adresse und Referenz-Nummer der Meldung)
- Festlegung der *Kommunikations-Richtlinien* (Policy) zwischen CERT und Hersteller der Software, bezüglich der Preisgabe des Namens des Melders der Schwachstelle
- Technische Informationen, die sich in folgende Abschnitte gliedern, zu denen, soweit möglich, Angaben gemacht werden sollen:
  - Vorhandensein einer *CERT Vulnerability Tracking Number*
  - Ausführliche *Beschreibung der Schwachstelle*
  - Ausführliche Beschreibung der möglichen *Auswirkungen* der Schwachstelle
  - Angabe, ob die *Schwachstelle schon ausgenutzt* wurde
  - Angabe, welche *Systeme* bzw. *Systemkonfigurationen* von der Schwachstelle betroffen sind
  - Angaben zu *Workarounds* oder *Beseitungsmöglichkeiten*
- Weitere Informationen, die nicht unter einen der bisher genannten Punkte fallen

2.1.2.3 Warnung vor Schwachstellen

Die Warnungen des CERT vor Schwachstellen oder Bedrohungen der IT-Infrastruktur, wie beispielsweise „Viren“ oder „Trojanische Pferde“, werden als Advisories bezeichnet.

Der Aufbau der Advisories entspricht dem einer strukturierten Informationsquelle, ist jedoch im höchsten Maße textgebunden.

Nach den grundlegenden Angaben

- Advisory-Nummer
- Bezeichnung der Schwachstelle
- Datum des ersten und letzten Releases des Advisorys
- Quelle der Informationen
- Liste von betroffenen Systemen
- Überblick über den Inhalt des Advisorys

folgen stets mindestens die im folgenden erläuterten drei Abschnitte sowie eventuelle Anhänge.

Beschreibung Der Abschnitt Beschreibung (Description) enthält, wie der Name schon vermuten läßt, eine ausführliche Beschreibung der Schwachstelle. Dazu kommen, sofern vorhanden, zusätzliche Informationen, beispielsweise der Inhalt des Advisorys, auf dem das CERT-Advisory basiert.

Auswirkung Der Abschnitt Auswirkung (Impact) enthält die Beschreibung der möglichen Auswirkungen, bei Ausnutzung der Schwachstelle.

Lösung Der Abschnitt Lösung (Solution) enthält die Beschreibung für die Beseitigung des Problems.

Anhänge Die Anhänge enthalten verschiedene zusätzliche Angaben. So enthält der meist vorhandene Anhang A (Appendix A) i.d.R. Angaben zu den Herstellern der betroffenen Produkte.

2.1.2.4 Zugriff auf die CERT-Advisories

Auf die Advisories des CERT/CC kann auf zweierlei Weise zugegriffen werden. Zum einen ist es möglich sich in eine Mailing-Liste des CERT einzutragen, über die die Advisories verschickt werden [CER00c], dadurch erhält der Abonnent die Meldungen frühstmöglich. Zum anderen ist es möglich über eine Webschnittstelle eine Volltextsuche in den Advisories durchzuführen [CER00d].

2.1.3 Bugtraq

Die moderierte Mailing-Liste *Bugtraq* [Sec00b], die von der „IT-Sicherheitscommunity“ Securityfocus unterhalten wird, ist neben den CERTs die wohl bekannteste und bedeutendste Quelle für Informationen zu Schwachstellen. Außerdem kann man Bugtraq wohl als eine der schnellsten öffentlichen Informationsquellen zu Schwachstellen bezeichnen. Dabei kommt nahezu keiner der großen Softwarehersteller weder darum herum, die Liste von seinen Mitarbeitern überwachen zu lassen, noch sich bei entsprechenden Meldungen zu seinen Produkten dazu zu äußern. Auch werden viele der separat zugänglichen Advisories von Sicherheitsorganisationen und -unternehmen und von Herstellern selbst automatisch mit an Bugtraq geschickt. Bugtraq hat keine festen Vorgaben für die Meldung von Vorfällen oder Schwachstellen. Entsprechende Meldungen können an die Mailing-Liste geschickt werden und werden dann dort diskutiert bzw. analysiert. Als Mailing-Liste ist Bugtraq eine schwach strukturierte Informationsquelle. Jedoch existiert eine SDB[1], die die „Ergebnisse“ der Diskussionen in der Mailing-Liste enthält. Dabei ist der Zugriff über ein Webinterface möglich. Die SDB selbst ist eine strukturierte Informationsquelle. Die Informationen sind in die sechs Abschnitte *Info*, *Diskussion*, *Ausnutzung*, *Lösung*, *Verdienste bei der Analyse* und *Hilfe* eingeteilt, die im folgenden kurz erläutert werde:

2.1.3.1 Info

Der Abschnitt Info enthält neben dem Namen die sogenannte *Bugtraq ID*, die Fehlerklasse, die CVE-Referenz (siehe Abschnitt 2.2 auf Seite 25), Informationen, ob die Schwachstelle

[1] Auf der Securityfocus-Homepage ([Sec00c]) zu erreichen über die Menüpunkte Vulnerabilies→Database

nur lokal oder auch remote ausgenutzt werden kann, das Datum der Erstveröffentlichung und des letzten Updates, die Software und die entsprechenden Versionen, die durch die Schwachstelle angreifbar sind und die, die durch die Schwachstelle explizit nicht angreifbar sind.

Die Fehlerklasse, gibt dabei Aufschluß über die Ursache der Schwachstelle. Bugtraq verwendet dabei die folgenden Fehlerklassen:

- Boundary Condition Error
  1. Versuch, über eine zulässige Adressgrenze hinauszulesen
  2. Erschöpfung einer Systemressource
  3. Überlauf einer Datenstruktur statischer Größe (der „klassische“ Buffer Overflow)
- Access Validation Error
  1. Versuch einer Operation auf einem Objekt, für das keine Zugriffsrechte bestehen
  2. Auftreten eines Fehlers durch den Versuch eines Lese- oder Schreibzugriffs auf eine Datei oder ein Gerät, für das keine Zugriffsrechte bestehen
  3. Auftreten eines Fehlers, weil ein Subjekt eine Eingabe von einem Objekt akzeptiert, das dafür nicht berechtigt ist
  4. Auftreten eines Fehlers, weil das System nicht in der Lage ist, ein Subjekt vollständig und korrekt zu identifizieren und authentifizieren
- Input Validation Error
  1. Auftreten eines Fehlers, weil das Programm nicht in der Lage war, eine syntaktisch nicht zulässige Eingabe zu erkennen
  2. Auftreten eines Fehlers, weil ein Modul „fremde“ Eingabefelder akzeptiert hat
  3. Auftreten eines Fehlers, weil ein Modul nicht in der Lage war, das Fehlen von Eingabefeldern zu verarbeiten
  4. Auftreten eines Fehlers aufgrund von Fehlern beim Inhalt korrelierter Felder
- Failure to Handel Exception Conditions

  Auftreten eines Fehlers, weil das System nicht in der Lage war, einen Fehler zu verarbeiten, der von einem Modul, einem Gerät oder einer Benutzereingabe erzeugt wurde
- Race Condition Errors

  Möglichkeit, einen Fehler auszunutzen, der durch ein Zeitfenster zwischen zwei Operationen entsteht
- Serialization Errors

  Auftreten eines Fehlers, durch eine ungeeignete Serialisierung von Operationen
- Atomicity Errors

1. Auftreten eines Fehlers, der dadurch erzeugt wird, daß ein anderer Prozeß eine unvollständig modifizierte Datenstruktur „beobachtet“ hat
2. Auftreten eines Fehlers, weil der Code in einer Operation terminiert, die atomar sein sollte und dabei unvollständig modifizierte Datenstrukturen hinterläßt

- Environment Errors

1. Auftreten eines Fehler durch eine Interaktion von per se korrekten Modulen in einer speziellen Umgebung
2. Auftreten eines Fehlers, durch das Ausführen eines Programms auf einem bestimmten Rechner unter einer speziellen Konfiguration
3. Auftreten eines Fehlers, weil die Laufzeitumgebung sich von der unterscheidet, für die das Programm entwickelt wurde

- Configuration Errors

1. Auftreten eines Fehlers, weil ein System-Utility mit falschen Setup-Parametern installiert wurde
2. Auftreten eines Fehler, weil ein System-Utility am falschen „Ort“ installiert wurde
3. Auftreten eines Fehlers, weil Zugriffsrechte auf ein Utility falsch gesetzt wurden, so daß die Sicherheitsrichtlinien durchbrochen wurden

2.1.3.2 Diskussion

Der Abschnitt Diskussion (Discussion) enthält eine ausführliche Erörterung der Schwachstelle und des Rahmens, in dem die Schwachstelle ausgenutzt werden kann.

2.1.3.3 Ausnutzung

Der Abschnitt Ausnutzung (Exploit) enthält Beschreibungen und Verweise aus Quellcode, der die Ausnutzung der Schwachstelle ermöglicht. Ausnutzung kann dabei bedeuten, eine Schwachstelle nur soweit auszunutzen, so daß die betroffene Software abstürzt oder aber, die Schwachstelle so auszunutzen, daß man tatsächlich, beispielsweise mit Administratorrechten, Zugriff auf das angegriffene System erhält.

2.1.3.4 Lösung

Der Abschnitt Lösung (Solution) enthält, soweit entsprechende Informationen vorhanden sind, Hinweise, wie die Schwachstelle beseitigt werden kann, beispielsweise Beschreibungen von Workarounds, Links auf Patches für die Software oder auch Hinweise auf Versionen der Software, die von der Schwachstelle nicht betroffen sind.

2.1.3.5 Verdienste bei der Analyse

Der Abschnitt Verdienste bei der Analyse (Credit) enthält sowohl die Angabe der Person oder Organisation, die den Fehler als erster gemeldet hat, als auch Referenzen auf Nachrichten, Advisories oder Webseiten, auf denen ebenfalls Informationen zu der Schwachstelle gefunden werden können. Dieser Abschnitt dient sicherlich einerseits der Angabe zusätzlicher Informationsquellen zu einer Schwachstelle, andererseits hat die Nennung als „Entdecker“ sicher auch einen motivierenden Aspekt.

2.1.3.6 Hilfe

Der Abschnitt Hilfe (Help) enthält Hinweise zur Bedeutung der Informationen aus dem Info-Abschnitt.

2.1.4 NTBugtraq

NTBugtraq [NTB00] ist eine Mailing-Liste, die „in der Tradition“ von Bugtraq gegründet wurde, um Microsoft Windows NT spezifische Fehler zu diskutieren, wobei auch Fehler in nicht Microsoft Produkten für Windows NT diskutiert werden können und sollen. NTBugtraq ist wie Bugtraq eine schwach strukturierte Informationsquelle, es exisitiert, im Gegensatz zu Bugtraq jedoch keine SDB, die für Recherchen genutzt werden könnte. Für Recherchen kann man nur das Archiv der Mailing-Liste nutzen und dort in den e-Mails textorientiert suchen.

2.1.5 INFILSEC

INFILSEC Systems Security [Sec00a] ist ein australisches Beratungsunternehmen für Systemsicherheit. Im Rahmen ihrer Arbeit stellt INFILSEC eine öffentliche SDB zur Verfügung. Die SDB zählt zu den strukturierten Informationsquellen und kann von beliebigen Personen über eine Web-Schnittstelle abgefragt werden. Eine Erfassung von Vorfällen in der öffentlichen SDB ist dabei nicht vorgesehen. Schwachstellen können von beliebigen Nutzern gemeldet werden und werden über ein simples Web-Interface eingegeben, genauso können Änderungen vorgenommen werden.

Folgende Attribute werden in der INFILSEC-SDB erfaßt:

- Schwachstelle (Vulnerability) - Name der Schwachstelle
- Betroffene Systeme (Systems affected)
- Komponente (Component)
- Auswirkung (Impact)
- Autor (Author)

- Beschreibung (Description)
- Ausnutzung der Schwachstelle (Exploit by)
- Autor des Skripts zur Ausnutzung der Schwachstelle (Exploit Author)
- Behebung der Schwachstelle (Fix by)
- Autor der Behebungsmöglichkeit (Fix Author)
- Referenzen (References)
- Datum der Erzeugung des Eintrags
- Datum der letzten Modifikation des Eintrags

Die Einträge bestehen aus Text, so daß für Recherchen auch nur reine Textsuche genutzt werden kann. Der Eintrag *Ausnutzung der Schwachstelle* enthält dabei Informationen dazu, wie die Schwachstelle von einem Angreifer ausgenutzt werden kann. Häufig ist in diesem Feld ein Listing eines Programms gespeichert, das die Schwachstelle ausnutzt. Im Eintrag *Behebung der Schwachstelle* wird in der Regel auf Patches, neue Version oder Workarounds verwiesen.

2.1.6 X-Force

Die X-Force Schwachstellendatenbank [ISS00a] gehört zu den strukturierten Informationsquellen und ist als solche weitgehend textbasiert. Der Zugriff auf die SDB erfolgt dabei über eine Web-Schnittstelle.

Es gibt zweierlei Arten von Einträgen in der SDB. Einerseits die Einträge, die auf die Schwachstellenwarnungen von ISS, dem Betreiber der SDB zurückgehen, und andererseits gibt es Einträge, die auf fremde Quellen zurückgehen. Diese Einträge bestehen nur aus einem Minimum an Informationen: zum einen eine Beschreibung der Schwachstelle, eine kurze textuelle Angabe darüber, welche Systeme betroffen sind, eine knappe Ausführung zu möglichen Gegenmaßnahmen, eine kurze Beschreibung der Konsequenzen, die sich aus der Schwachstelle ergeben und eine Referenz auf die Meldung, auf die der Eintrag zurückgeht. Bei dieser Art von Einträgen ist es nahezu nicht möglich, Informationen innerhalb der X-Force-SDB zu recherchieren, da keine ausreichenden Angaben vorhanden sind. Ein Nutzer ist gezwungen, zusätzlich mindestens die angegebene Quelle zu konsultieren.

Die ISS-Schwachstellenwarnungen, ISS Security Advisories genannt, sind ausführlicher als die von anderen Quellen übernommenen Einträge. Sie bestehen aus sechs Abschnitten: der erste Abschnitt stellt eine kurze Inhaltsangabe dar (Synopsis), der zweite Abschnitt enthält eine Beschreibung der Schwachstelle (Description). Daraufhin folgt eine Angabe der von der Schwachstelle betroffenen Versionen der Software (Affected Versions). Danach werden Empfehlungen gegeben, wie der Schwachstelle entgegengewirkt werden kann (Recommendations). Dort werden wiederum Verweise in Form von Links auf verschiedene Quellen gegeben, wobei es sich um interne ISS-Quellen und externe Quellen handeln kann. Der nächste Abschnitt verweist auf zusätzliche, externe Informationsquellen zu der Schwachstelle (Additional Information). Schließlich folgt noch ein Abschnitt mit Angabe der Quellen, die für diese Schwachstellenmeldung verwandt wurden (Credits).

2.2 Common Vulnerabilities & Exposures

Common Vulnerabilities & Exposures [2] (CVE) ist eine Liste von standardisierten Namen für Schwachstellen [MIT00a]. Damit handelt es sich bei CVE nicht um eine SDB, sondern um ein Dictionary. Ziel von CVE ist es, daß möglichst viele SDBs zu den bei ihnen gespeicherten Schwachstellen den standardisierten CVE-Namen speichern. Dadurch ist es leichter möglich, die Informationen zu einer Schwachstelle in mehreren SDBs, die CVE unterstützen, zu vergleichen bzw. zusätzliche Informationen zu einem Schwachstelleneintrag einer SDB aus einer anderen SDB zu erhalten. CVE wurde von MITRE, einer privaten, staatlich geförderten Forschungsanstalt in den USA, initiiert. CVE wird heute vom *CVE Editorial Board* verwaltet. Mitglieder des Board sind Repräsentanten von mehr als zwanzig Organisationen, die sich mit Sicherheit beschäftigen, wie beispielsweise Hersteller von Tools, akademische Einrichtungen, Regierungsvertreter und anerkannte Sicherheitsexperten.[3]

In 1999 wurde 1011 CVE-Bezeichner für Schwachstellen vergeben, Ende Juli 2000 waren es bereits 465 CVE-Bezeichner für das Jahr 2000. Zusätzlich existieren Listen mit Kandidaten für die Vergabe von CVE-Bezeichnern.

Folgende Organisationen haben sich verpflichtet, ihre Produkte CVE-kompatibel zu gestalten: Advanced Research Corporation, Alliance Qualité Logiciel, AXENT Technologies Inc., BindView Development, CERIAS/Purdue University, CERT Coordination Center, Cisco Systems, Computer Security Laboratory (Dept. of Computer Science, UC Davis), CyberSafe, CYRANO, Ernest & Young, Harric Corporation, Hiverworld Inc., Intrusion.com, Internet Security Systems Inc., Max Vision Network Security/Whitehats, National Institute of Standards and Technology, The Nessus Project (Renaud Deraison & Jordan Hrycaj), Network Security Wizards, NTBugtraq, PGP Security (Network Associates), SANS, Security Focus Inc., World Wide Digital Security, Symantec. Eine genau Übersicht über den Grad der Anpassung der jeweiligen Produkte findet sich auf der CVE-Homepage [MIT00b].

2.3 Software im Bereich von SDBs

Sowohl viele Betreiber von öffentlichen SDBs, als auch vermutlich viele Beratungsunternehmen, die auf dem Gebiet IT-Sicherheit tätig sind und private SDBs betreiben, nutzen die Informationen der SDBs für Sicherheitsanalysen von IT-Systemen und die Beratung von Kunden.

Wie die Nutzung der Informationen einer SDB bzw. die Nutzung von Informationen über Schwachstellen im Rahmen von Softwareprodukten aussehen kann, soll anhand von zwei

[2] Bei CVE werden unter *Exposures* solche Schwächen in Systemen verstanden, die zwar keine Schwachstellen darstellen, aber beispielsweise Informationen über Nutzer eines Systems verraten, wenn dies nicht notwendig ist. Beispielsweise wäre die Bekanntgabe der Nutzer eines Systems über den finger-Daemon ein Exposure im Sinne des CVE.

[3] Mitglieder des Boards sind Vertreter folgender Organisationen: AXENT, BindView, CanCERT, CERIAS, CERT, Cisco, CyberSafe, Ernest & Young, GTE Internetworking, Harris, Hiverworld, IBM, Intrusion.com, ISS, L-3 Security, Microsoft, The MITRE Corporation, NFR, NTBugtraq, PGP Security (Network Associates), SANS, Security Focus, Silicon Defense, Sun Microsystems, Symantec, UC-Davis, Vista IT und Zero-Knowledge Systems.

Beispielen skizziert werden. Dafür wurde ein kommerzielles Produkt namens WebTrends Security Analyzer und ein Open Source Produkt namens Nessus ausgewählt. Ältere, ebenfalls sehr bekannte, nicht-kommerzielle Produkte wie SATAN oder SAINT wurde dabei nicht berücksichtigt, da sie seit Erscheinen von Nessus als veraltet gelten dürfen.

Eines der Produkte der *WebTrends Corporation* [WC00b, WC00a] ist der *Security Analyzer* [WC00c]. Dieses Programm untersucht weitgehend autonom Rechner, die über Netzwerk erreichbar sind, auf Schwachstellen, die durch Netzwerkzugriffe ausgenutzt werden können. Das Programm ist für die Microsoft Windows Produktfamilie und Unix-Systeme verfügbar und bietet Sicherheitsanalysen für verschiedene Dienste und Anwendungen, wie Web-, Mail-, FTP- und Game-Server, Webbrowser, e-Mail-Clients, Proxies und Firewalls sowie Dateizugriffsrechte und Paßwörter. Dabei greift das Programm auf eine interne SDB zurück, die Schwachstellen, Möglichkeiten zur Ausnutzung bzw. zum Testen von Schwachstellen und Vorschläge oder Hinweise zur Beseitung der Schwachstelle enthält. Diese SDB wird sogar via Internet mit den Daten von neuen Schwachstellen versorgt, so daß das Programm prinzipiell immer auf dem neusten Stand sein kann. Dabei ist der Security Analyzer durch die Vorgabe von Sicherheitsrichtlinien konfigurierbar. Als Ergebnis werden Berichte im HTML-Format erstellt. Zusätzlich können die „eingebauten“ Analysen durch zusätzliche Tests mittels eines *Software Development Kit* (SDK) erweitert werden.

Ein Produkt, das eine ähnliche Funktion erfüllt wie der WebTrends Security Analyzer ist *Nessus* [Nes00]. Im Gegensatz zum Produkt von WebTrends ist Nessus jedoch frei verfügbar, in dem Sinne, daß es unter der GNU General Public License [FSF00b] steht. Nessus wird von seinen Entwicklern selbst als ein Security Auditing Tool bezeichnet, das dazu dienen soll, Risiken die den Betrieb eines IT-Systems bedrohen, zu erkennen und zu melden. Dabei wollen die Entwickler einem, wie sie ausführen, negativen Trend bei Security Scannern entgegenwirken, nämlich, daß diese langsam aber sicher zu „PowerPoint Präsentationsgeneratoren“ verkommen und sich dabei der Focus stetig vom Thema Sicherheit entfernt. Außerdem wird kritisiert, daß kommerzielle Security Scanner Hersteller viel zu lange benötigen, um Tests für neue Schwachstellen in ihre Produkte zu integrieren [HD99].

Nessus zielt dabei darauf ab, regelmäßig, beispielsweise mittels eines CRON-Auftrags, in einem Netzwerk aktiviert zu werden. Außerdem sollen die Zeiten, die benötigt werden, um Test für neue, bekannt gewordene Schwachstellen bereit zu stellen, auf wenige Stunden schrumpfen. Dabei bedient sich Nessus, wann immer möglich, der speziell für Nessus entwickelten Skriptsprache NASL (Nessus Attack Scripting Language), ansonsten wird C für die Implementierung der Tests verwandt, was sich allerdings negativ auf die Portabilität auswirkt. NASL beinhaltet dabei Sicherheitsmachanismen, um zu verhindern, daß Skripte selbst Schaden anrichten können. Beispielsweise ist es nicht möglich, auf dem lokalen Rechner, auf dem ein Skript ausgeführt wird, Befehle auszuführen. Ebenfalls ist es nicht möglich Datenpakete an andere Rechner zu schicken, als den aktuelle geprüften Rechner.

Nessus basiert auf einer Client-Server-Architektur. Auf einem oder mehreren Rechnern ist der Nessus-Daemon *nessusd* installiert. Dieser führt die Scans im Netzwerks aus. Gesteuert werden kann er von einem oder mehreren Clients. Clients müssen sich dabei authentifizieren. Die Kommunikation zwischen Client und Server wird aus Sicherheitsgründen verschlüsselt durchgeführt. Während der Server nur für Unix-Systeme verfügbar ist, existieren Clients für Unix, Windows und Java.

Als Quelle für die Schwachstellentests benutzt Nessus ebenfalls eine eigene Schwachstellendatenbank. Die Einträge der Datenbank werden *Plugins* genannt. Diese können einzeln

oder automatisch vom Nessus-Server heruntergeladen werden, so daß die SDB immer auf dem aktuellen Stand ist. Die Plugins sind dabei in folgende Kategorien unterteilt: *Backdoors*, *CGI abuses*, *Denial of Service*, *Finger abuses*, Firewalls, FTP, Gain a shell remotely, Gain root remotely, General, *Miscellaneous*, *NIS*, *Port scanners*, *Remote file access*, *RPC*, *SMTP problems*, *Useless services* und *Windows*. Ein Plugin enthält dabei einen Test für eine einzelne Schwachstelle oder eine Gruppe ähnlicher Schwachstellen.

Neben solchen relativ neuen, automatisierten Lösungen wird meist eine „manuelle“ Beratung durch Spezialisten durchgeführt. Gründe dafür dürften darin liegen, daß automatisierte Lösungen noch nicht in der Lage sind alle Schwachstellen aufzudecken. Insbesondere Schwachstellen zu denen sehr komplexe Angriffsszenarien gehören, die beispielsweise auf ein sehr kompliziertes Timing angewiesen sind oder aber nicht als einfache Skripte repräsentiert werden können, bereiten Probleme. Es ist allerdings absehbar, daß es nur eine Frage der Zeit ist, bis Sicherheitsscanner auch solche Angriffe nachvollziehen können.

Ein weiteres Problem stellen „destruktive“ Angriffe dar, d.h. Angriffe, bei denen ein Dienst oder gar ein ganzer Rechner gestört werden. Werden Produktivsysteme untersucht, ist es nicht möglich einen Scanner zu verwenden, der im Zweifelsfall den Produktivrechner stört. In einem solchen Fall muß auf eine „klassische“, manuelle Sicherheitsanalyse zurückgriffen werden.

Es kann abschließend festgestellt werden, daß schon heute verschiedene Softwarelösungen existieren, die geeignet sind, den Anwender bei der Suche nach Schwachstellen zu unterstützen. Dabei gibt es jedoch einerseits noch Einschränkungen in der Leistungsfähigkeit und andererseits benutzt jedes dieser Werkzeuge seine eigene SDB, so daß regelmäßig mehrere SDBs genutzt werden müssen: mindestens eine SDB, damit sich die „menschlichen“ Nutzer über Schwachstellen informieren können und eine SDB, die ein Softwarewerkzeuge nutzt, um nach Schwachstellen zu suchen. Dabei ist jedoch nicht gewährleistet und auch nicht leicht erkennbar, ob die SDBs Informationen zu den selben Schwachstellen enthalten.

Kapitel 3

Anforderung an die Schwachstellendatenbank

Dieses Kapitel bietet eine Problemanalyse für den Aufbau und Betrieb einer Schwachstellendatenbank. Die wichtigste Voraussetzung für eine Schwachstellendatenbank ist, daß der Betrieb sichergestellt ist. Dies setzt voraus, daß insbesondere die Finanzierung sichergestellt ist, was primär durch eine hohe Qualität des Inhalts zu erreichen sein dürfte. Die hohe Qualität wiederum sollte aufgrund der dadurch erreichten hohen Attraktivität ein Garant für eine große Anzahl von Nutzern sein, was wiederum mittelfristig zumindest einen kostendeckende Betrieb sicherstellen sollte. Der erste Teil des Kapitel beschäftigt sich mit dem Problem, ein geeignetes Geschäftsmodell für den Aufbau und den Betrieb einer SDB zu erarbeiten. Der zweite Teil befaßt sich primär mit den Informationen, die in der SDB erfaßt werden müssen und der Art und Weise, wie diese Informationen erfaßt und genutzt werden können.

Bei der Diskussion bezüglich des Geschäftsmodells darf nie außer acht gelassen werden, daß gegenüber den meisten, im vorhergehenden Kapitel beschriebenen SDBs eine Erhöhung des Nutzens erreicht werden soll. Würde dies nicht erreicht werden, stellte sich sicherlich sehr schnell die Frage, ob nicht „nur noch eine weitere“ SDB entwickelt wurde. Dieser Aspekt wird in Abschnitt 3.2 betrachtet. Außerdem gehen die Abschnitte 3.3.1 und 3.3.2 auf Nachteile von Krsuls SDB und der X-Force-SDB ein.

3.1 Geschäftsmodell für den Betrieb einer SDB

Das Geschäftsmodell soll sicherstellen, daß die SDB erfolgreich, d.h. insbesondere dauerhaft, betrieben werden kann. Dazu müssen organisatorische, aber auch technische Aspekte berücksichtigt werden. Im folgenden sollen die wichtigsten Aspekte für den Betrieb einer SDB untersucht werden.

Dabei kann das Geschäftsmodell einer SDB unter den Aspekten *Organisationsmodell, Zugriff auf die SDB, Veröffentlichungspolitik, Qualitätssicherung, Finanzierung, Rechtliche Organisation, Rechtliche Aspekte, Mitarbeiter, Infrastruktur* betrachtet werden. Diese werden in den nächsten Abschnitten ausführlich untersucht.

3.1.1 Organisationsmodell

Das Organisationsmodell der SDB beschäftigt sich mit der grundlegenden Positionierung der SDB gegenüber anderen Quellen für Informationen zu Schwachstellen.

Dabei sind verschiedene Aspekte zu berücksichtigen:

- *Verwaltungsaufwand* Unter Verwaltungsaufwand ist der administrative Aufwand zu verstehen, der aus der Organisation des Betriebs der SDB entsteht. Er kann sich in der Höhe, je nach gewähltem Organisationsmodell, deutlich unterscheiden.
- *Aufwand für Unterhaltung* Der Aufwand für die Unterhaltung betrachtet den zu erwartenden Aufwand, der sich aus dem Betrieb, der Pflege und Weiterentwicklung der SDB ergibt.
- *Technische Durchführbarkeit* Die technische Durchführbarkeit befaßt sich mit dem Aufwand und der Realisierbarkeit der technischen Anforderungen an den Betrieb der SDB.
- *Kosten* Die Kosten für den Betrieb können, abhängig vom Organisationsmodell, aber auch von anderen Faktoren, deutlich schwanken und somit die Finanzierbarkeit des SDB-Betriebs beeinflussen.
- *Akzeptanz* Verschiedene denkbare Nutzergruppen, können auf verschiedene Organisationsmodelle unterschiedlich reagieren, bis hin zur Ablehnung.
- *Spezialisierungsmöglichkeit* Unter Umständen kann es wünschenswert sein, die Möglichkeit zur Spezialisierung auf einen bestimmten Themenkreis, beispielsweise auf ein spezielles Betriebssystem, in einem Organisationsmodell vorzusehen oder zumindest möglich zu machen.
- *Recherchierbarkeit* Speziell für die Forschung, aber auch für andere denkbare Einsatzgebiete, ist es notwendig, komplexe Recherchen, beispielsweise für Data-Mining, auf den Daten der Datenbank möglichst effizient durchführen zu können. Dies kann, je nach Organisationsmodell unterschiedlich gut möglich sein. Außerdem muß bezüglich der Recherchierbarkeit noch berücksichtigt werden, wieviel Zeit benötigt wird, um ein möglichst umfassendes Bild einer Schwachstelle und ihrer Auswirkungen zu erhalten.
- *Potentielle Qualitätsunterschiede* Jedes Organisationsmodell kann inherent bestimmte Vor- und Nachteile bezüglich der später zu erwartenden Qualität der Daten haben. Die Qualität kann einerseits in der „Vollständigkeit“ der in der SDB erfaßten Schwachstellen gesehen werden, andererseits auch darin, wie vollständig und verläßlich die Daten zu den einzelnen Schwachstellen sind.
- *Potentielle Widersprüche* Bestimmte Organisationsmodelle führen u.U. eher zu Widersprüchen bezüglich der gespeicherten Daten.
- *Redundanzen* Je nach Organisationsmodell kann es zu Redundanzen bezüglich der gespeicherten Daten kommen. Dies ist ineffizient und erschwert u.U. Recherchen.

- *Verfügbarkeit* Eine Schwachstellendatenbank ist für viele Nutzer ein wichtiges Instrument zur Sicherung von IT-Systemen. Daher ist es notwendig, daß eine SDB eine möglichst hohe Verfügbarkeit erreicht. Hinzu kommt die Tatsache, daß bestimmte Organisationsmodelle für Sabotage, beispielsweise für *Distributed Denial of Service*-Angriffe (DDoS) anfälliger sind, als andere.

In Erweiterung von [MS99] können die vier Grundtypen *Balkanisiert*, *Zentralisiert*, *Föderiert* und *Open Data* eines Organisationsmodells für eine Schwachstellendatenbank unterschieden werden. Auf diese Modelle wird im folgenden detailliert eingegangen.

3.1.1.1 Balkanisiertes Modell

Das balkanisierte Modell, das diesen Name in Analogie zur politischen Situation in der gleichnamigen Region Südosteuropas erhielt, stellt zugleich den aktuellen Zustand im Bereich der Schwachstellendatenbanken dar. Es existiert eine Vielzahl von SDB unterschiedlichster Struktur. Zwischen den SDBs gibt es keine Koordination und keine Absprachen[1].

Da in einem balkanisierten Modell jeder Betreiber einer SDB den gesamten Aufwand für den Betrieb einer SDB selbst übernehmen muß, ist der Aufwand für jeden Betreiber sehr hoch und über alle Betreiber gesehen maximal. Dies betrifft sowohl den Verwaltungsaufwand, als auch den Aufwand für die Unterhaltung der SDB.

Die technische Durchführbarkeit stellt kein Problem dar. Da sich die Anfragen über verschiedene SDBs und Server verteilen, sollten keine Probleme durch übermäßige Last auftreten. Für die Kosten gilt, was für den Verwaltungsaufwand gilt. Da jeder Betreiber alle Aufgaben im Rahmen des Betriebs selbst übernehmen muß, sind auch die Kosten pro Betreiber, als auch die Kosten über alle SDBs sehr hoch. Dabei muß natürlich berücksichtigt werden, daß manche der öffentlich zugänglichen SDBs nur ein „Abfallprodukt“ einer gewinnorientierten wirtschaftlichen Tätigkeit darstellen.

Die Akzeptanz eines balkanisierten Modells dürfte relativ schwach sein, wobei einzelne SDBs durchaus hohe Akzeptanzwerte erhalten können, wie beispielsweise Bugtraq. Es existieren keine Alternativen, aber häufig müssen Daten über viele SDBs hinweg recherchiert werden, um ein umfassendes Bild zu erhalten. Diese Recherche wird zusätzlich dadurch erschwert, daß meistens die gleichen Schwachstellen unterschiedlich benannt sind, so daß die betreffende Schwachstelle in jeder SDB von neuem gesucht werden muß. Hinzu kommt, daß aufwendige Anfragen, beispielsweise im Bereich der Forschung, i.d.R. nicht möglich sind, da es für externe Nutzer keine dokumentierten Schnittstellen gibt, die es erlauben, auf die SDB zuzugreifen. Außerdem dürfte die geringe Akzeptanz und das damit einhergehende mangelnde Vertrauen ein Grund dafür sein, daß so viele private SDBs in Unternehmen existieren. Darüber hinaus liegen die SDBs grundsätzlich nicht lokal vor, was die Geschwindigkeit komplexer Anfragen weiter reduziert.

Eine klare Stärke des balkanisierten Modells ist die Fähigkeit zur Spezialisierung. Wenn eine SDB „benötigt“ wird, die sich auf einen speziellen Problemkreis, beispielsweise ein Betriebssystem oder eine spezielle Anwendung, konzentriert, so wird sie, einen entsprechenden Bedarf vorausgesetzt, aufgebaut.

[1] Die einzige Ausnahme stellt CVE dar, vgl. dazu Abschnitt 2.2

Eine klare Schwäche des balkanisierten Modells ist, daß über die Qualität der Daten der einzelnen SDBs von vornherein keine Aussagen getroffen werden können. Alle Qualitätsbeurteilungen sind Erfahrungswerte und zwischen SDBs können teilweise große Qualitätsunterschiede bestehen, sowohl was den Umfang der erfaßten Schwachstellen angeht, als auch die Qualität der Informationen zu einer einzelnen Schwachstelle. Aufgrund der fehlenden Koordination zwischen den verschiedenen SDBs muß damit gerechnet werden, daß es starke Widersprüche zwischen den Informationen verschiedener SDBs gibt. Daraus ergibt sich der Zwang entweder die Daten, soweit möglich, selbst zu verifizieren oder aber sich auf Erfahrungswerte[2] zu verlassen. Außerdem exisitieren in diesem Modell hohe Redundanzen, da die selben Informationen mehrfach gespeichert werden. Dies ist das Ergebnis fehlender Koordination und ist einer der Gründe für den hohen Gesamtaufwand des Modells, da die gleichen Arbeiten mehrfach durchgeführt werden müssen.

Eine weitere Stärke stellt die Verfügbarkeit im balkanisierten Modell dar. Da es unzählige SDBs gibt, ist es kaum wahrscheinlich, daß alle SDBs gleichzeitig nicht verfügbar sind oder daß alle SDBs gleichzeitig durch Angriffe ausgeschaltet werden.

3.1.1.2 Zentralisiertes Modell

Das zentralisierte Modell ist mehr ein hypothetisches Denkmodell, denn eine realistische Variante für ein Organisationsmodell für den Betrieb von SDBs. Das Modell geht vom Gedanken einer einzigen SDB aus und stellt damit gegenüber dem balkanisierten Modell das zweite Extrem des Spektrums denkbarer Modelle für SDBs dar. Es ist natürlich auch möglich eine zentralisierte SDB unter vielen betrachten, daraus resultiert allerdings wieder das balkanisierte Modell. Die Aussagen können jedoch größtenteils analog für den Betrieb einer zentralisierten SDB in einem balkanisierten Umfeld übernommen werden.

Der Verwaltungsaufwand für den Betrieb der zentralisierten SDB ist hoch, da alle administrativen Aufgaben, die durch den Betrieb anfallen, komplett bearbeiten werden müssen. Da jedoch nur eine SDB existiert, ist der Gesamtaufwand über alle SDBs gesehen, gering. Das gleiche gilt für den Aufwand für die Unterhaltung der SDB.

Die technische Durchführbarkeit ist prinzipiell gegeben, jedoch ist mit verschiedenen Problemen, wie Bandbreiten- und Lastproblemen zu rechnen, da alle Anfragen an *die eine* SDB gerichtet werden. Selbstverständlich könnte diesen Problemen mittels geeigneter Replikationsmechanismen entgegengewirkt werden, was dann wieder zu einer speziellen Form des föderierten Modells führen würde. Die Kosten für den Betrieb der zentralisierten SDB sind hoch, da alle Arbeiten für den Betrieb und die Verwaltung durchgeführt werden müssen. „Gesamtwirtschaftlich“ betrachtet wären die Kosten jedoch gering, da dieser Aufwand nur einmal entstehen würde.

Die Akzeptanz dieses Modell dürfte sehr gering sein, zu groß dürfte das Mißtrauen gegenüber den Betreibern der einen SDB sein. Nutzer könnten mutmaßen, daß die SDB-Administration beispielsweise durch Bestechung bestimmte Schwachstellen verschweigt. Somit würden sich vermutlich automatisch wieder andere Quellen für Schwachstelleninformationen herausbilden, die zur Verifikation der Angaben der SDB herangezogen werden

[2] Beispiel: „SDB A ist zuverlässiger als SDB B“

könnten. Zwar ließe sich dieser Effekt vermutlich verringern, wenn die Geschäftspolitik öffentlich festlegt und die Prozesse innerhalb der SDB-Administration transparent gestaltet, jedoch dürfte ein Rest Mißtrauen bestehen bleiben.

Ein weiteres Problem stellt eine zentralisierte SDB für Unternehmen dar. Diese wollen vermutlich erreichen, daß Schwachstellen selbsterstellter und ausschließlich selbst genutzer Software, die das Unternehmen u.U. von außen angreifbar machen würden, bekannt werden. Daher würden diese Unternehmen vermutlich zumindest für diese Informationen wieder private SDBs einrichten, die nur ihnen zugänglich wären. Damit wäre schon wieder ein Schritt Richtung Balkanisierung getan. Auch die Möglichkeit der Spezialisierung, die u.U. wegen der bedauerlicherweise hohen Informationsmengen zu Schwachstellen sinnvoll erscheinen kann, ist bei einer zentralisierten SDB nicht gegeben, da per Definition in diesem Modell nur eine SDB existiert.

Die Recherchierbarkeit der zentralisierten SDB ist definitionsgemäß gut, da alle Informationen an einem zentralen Ort gespeichert sind. Andererseits ergeben sich durch den *Remote*-Zugriff und die bereits erwähnten Lastprobleme möglicherweise Probleme bei der Ausführung komplizierter und aufwendiger Anfragen an die Datenbank. Qualitätsunterschiede zwischen SDBs exisiteren in diesem Modell ebenfalls nicht, genauso wenig wie Widersprüche und Redundanzen zwischen verschiedenen SDBs.

Die hohe Verfügbarkeit wäre wiederum ein potentielles Problem für die SDB, da, je nach Architektur der SDB, durchaus mit Ausfällen der Hardware, der Software oder auch des Netzzugangs zu rechnen wäre. Außerdem könnte die zentralisierte SDB beispielsweise mittels DDoS-Angriffen unbrauchbar gemacht werden und auch eine gezielte „Verschmutzung“ des in diesem Modell „einzigen“ Datenbestands könnte durch die Übermittlung falscher Informationen versucht werden. Dem könnte nur durch einen (partiell sowieso notwendigen) umfangreichen Verifikationsprozeß Einhalt geboten werden. Bei Erfolg eines solchen „Verschmutzungsversuchs“ wären die Folgen jedoch u.U. gravierend, da es definitionsgemäß keine „andere“ Informationsquelle gäbe und somit falsche Informationen verbreitet würden.

3.1.1.3 Föderiertes Modell

Unter dem föderierten Modell ist zu verstehen, daß *die* SDB in verschiedene, beispielsweise auf bestimmte Betriebssysteme spezialisierte Instanzen aufgeteilt ist. Dabei findet jedoch zwischen den Instanzen nach wie vor ein Koordinationsprozeß statt, bei dem beispielsweise das Datenbankschema oder die weiter unten erläuterte Veröffentlichungspolitik, aber auch die Beurteilung und Verifikation von Schwachstellen untereinander abgestimmt wird. Die Teilnehmer an der Föderation müßten im Idealfall durch Absprachen respektive Verträge an die Einhaltung bestimmter Vorgaben bezüglich Administration und Betrieb gebunden sein. Dazu müßte eine Art *Verwaltungsrat* oder ein *Change Management Board* existieren, das über Veränderungen von Datenbankschema oder Prozessen bestimmt. Zusätzlich können in dem föderierten Modell private SDBs für ausschließlich privat genutzte Informationen, beispielsweise bezüglich Schwachstellen proprietärer, selbsterstellter Software, existieren. Somit wäre auch dieses potentielle Problem von Unternehmen, das bereits in Verbindung mit dem zentralisierten Modell besprochen wurde, lösbar. Weiterhin wäre als Spezialfall ein föderiertes Modell denkbar, das nur eine zentrale, öffentliche Instanz besitzt und das von

den privaten Instanzen ergänzt wird. Diese Variante könnte auch als zentralisierte SDB, die um private Instanzen ergänzt wurde, angesehen werden. Eine weitere denkbare Variante des föderierten Modells wäre eine Föderation, bei der alle SDBs die gesamten Informationen untereinander synchronisieren, so daß quasi mehrere Kopien der gleichen SDB betrieben würden. Ein Austausch der Informationen wäre aufgrund des identischen Schemas relativ problemlos möglich, jedoch müßte hierbei insbesondere auf ein gleiches Qualitätsniveau zwischen den Föderationsmitgliedern geachtet werden, damit es mittelfristig nicht zu einer Verschlechterung der Informationsqualität kommt.

Der Verwaltungsaufwand für eine föderierte SDB ist sehr hoch, da zusätzlich zu den bei den anderen Modellen ebenfalls vorhandenen Aufgaben, Koordinationsaufgaben hinzukommen. Über die gesamte Föderation betrachtet, wäre der Administrationsaufwand vermutlich jedoch nicht so hoch wie beim balkanisierten Modell. Auch der Aufwand für den Betrieb wäre im Vergleich zum balkanisierten Modell geringer, da notwendige Entwicklungsarbeiten über die Mitglieder der Föderation verteilt werden könnten, so daß sich für die gesamte Föderation die Arbeit auf mehrere Gruppe verteilt. Die technische Durchführbarkeit sollte ebenfalls kein Problem darstellen. Insbesondere Lastprobleme sollten, mit Ausnahme des Modells mit nur einer zentralen und mehreren privaten Instanzen, in einer Föderation weniger häufig auftreten, da sich die „Gesamtmenge der Anfragen“ auf mehrere Instanzen der Datenbank verteilt, unabhängig davon, ob die Föderation in Form von Kopien oder spezialisierten SDBs organisiert ist.

Die Kosten, die in einem föderierten Modell entstehen, dürften gegenüber dem balkanisierten Modell sowohl bezüglich einer Instanz als auch bezüglich der gesamten Föderation niedriger liegen, da viele Arbeiten koordiniert erledigt werden können und viele redundante Tätigkeiten wegfallen. Dies dürfte beispielsweise für Unternehmen interessant sein, die nur noch ihre privaten Instanzen pflegen müssen und in „öffentlicher“ Software entdeckte Schwachstellen direkt an die zuständige öffentliche Instanz melden, was den Aufwand, den Unternehmen betreiben müssen, deutlich senken dürfte. Aufgrund verschiedener Vorteile, beispielsweise dem Fehlen proprietärer Instanzen in der Föderation oder der offenen Beschaffenheit der SDB, ihrer Daten und Prozesse und der Berücksichtigung spezieller Interessen von Unternehmen, sollte eine Föderation auf ein hohes Maß an Akzeptanz treffen.

Auch dem Aspekt der Spezialisierungsmöglichkeit wird im föderierten Modell genüge getan. Die Recherchierbarkeit in diesem Modell ist ebenfalls gut. Zwar ist es u.U. nach wie vor notwendig auf entfernte Datenbanken zuzugreifen, was bei aufwendigen Recherchen ein Problem sein kann, jedoch ist aufgrund des einheitlichen Schemas ein verhältnismäßig geringer Aufwand zu betreiben, um Ergebnisse zu Anfragen an mehrere SDBs zu erhalten.

Bezüglich der Qualität der Informationen aus den Datenbanken der Föderationsmitglieder ist festzustellen, daß es die notwendige Aufgabe des Verwaltungsrats und eine Frage des Inhalts der Verträge zwischen den Föderationsmitgliedern ist, dafür zu sorgen, daß durch geeignete Qualitätssicherungsmaßnahmen ein einheitliches Qualitätsniveau herrscht, da die Glaub- und Vertrauenswürdigkeit der gesamten Föderation vom schwächsten Föderationsmitglied bestimmt wird. Bei verantwortungsvollem Umgang mit dem Thema Qualität kann sogar eine sehr hohe Gesamtqualität erreicht werden, da im Falle der thematischen Spezialisierung der einzelen Instanzen, bei den jeweiligen Instanzen höher spezialisierte Mitarbeiter zu erwarten sind, da diese nur ein kleineres Spektrum des breiten IT-Bereichs abdecken müssen.

Die gleichen Aussagen, die bezüglich der Qualität zutreffen, treffen auf potentielle Widersprüche und Redundanzen zwischen den SDBs der Föderation zu. Nur wenn Widersprüche verhindert werden, ist damit zu rechnen, daß sich die SDB erfolgreich am Markt etablieren kann. Redundanzen sollten, wenn sie notwendig oder gewünscht sind, nur in der Form vorhanden sein, daß tatsächlich Informationen repliziert werden, nicht aber, daß Informationen in verschiedenen SDBs mehrfach erfaßt sind.

Die Verfügbarkeit der SDB hängt eng mit der gewählten Variante zusammen: für die Variante mit nur einer zentralen Instanz und die Variante mit spezialisierten SDBs gelten die Ausführungen zum zentralisierten Modell. Hier ist es möglich, den teilweisen oder auch gesamten Betrieb der SDB vorsätzlich zu behindern, was im Fall der „replizierten“ Datenbank nicht der Fall ist. Da sich bei den Varianten mit mehreren öffentlichen SDBs die Last wiederum auf viele Systeme verteilt, ist es wiederum wahrscheinlich, daß dadurch eine höhere Verfügbarkeit erreicht wird.

3.1.1.4 Open Data Modell

Das *Open Data*-Modell, ist an *Open Source*-Software [Ope00b], die unter der *GNU General Public License* [FSF00b] (GPL) steht, angelehnt. Das Ziel ist es, daß die Datenbank und möglicherweise auch Tools zur Nutzung der Datenbank frei verfügbar sind. Jeder Interessent kann die Datenbank kopieren und für seine Zwecke einsetzen, wird jedoch durch eine geeignete Lizenz verpflichtet, neue Informationen an die „Öffentlichkeit“ zurückzugeben. Die zu definierende Lizenz für die Nutzung der Datenbank könnte dabei an die *Open Content*-Lizenz (OPL) angelehnen werden [Ope00a]. Die OPL beinhaltet Vorgaben über das Kopieren, Verändern und Weiterverbreiten von „Content“, also von Inhalten, von „Dokumenten“, die unter der OPL stehen.[3] Ähnlich wie die GPL ist dabei auch sichergestellt, daß veränderte Dokumente, die verteilt oder publiziert werden, wiederum unter der OPL stehen. Dabei müßte zusätzlich zur OPL festgelegt werden, daß neue oder geänderte Datensätze markiert werden, so daß Dritte die neuen oder veränderten Daten relativ schnell auffinden, überprüfen und gegebenenfalls übernehmen können.

Der Open Data-Ansatz birgt weiterhin die Gefahr, daß es in kurzer Zeit eine große Anzahl von Ablegern bzw. Varianten der SDB gibt, so daß quasi eine Balkanisierung gefördert werden könnte. Hier wäre zu prüfen, ob ein ähnlicher Effekt wie bei Open Source-Software erreicht werden kann, bei der nur selten eine Spaltung der Entwicklungsarbeiten in zwei oder mehr eigenständige Projekte stattfindet. Allerdings muß hier ein wichtiger Unterschied zwischen Open Source und Open Data herausgestellt werden. Bei großen Open Source-Projekten ist es nicht ohne sehr viel Aufwand möglich, sich in den Quellcode einzuarbeiten und dann eigenständige Änderungen durchzuführen. Außerdem sind Änderungen in großen Projekten selbst wieder so komplex und schwierig, so daß hier quasi inherent ein wirksamer Schutzwall gegen eine große Anzahl von abgespaltenen Projekten existiert. Bei einer Open Data-Datenbank ist dies nicht so. Bei einer guten Dokumentation von Datenbank und Tools, ist es möglich, verhältnismäßig schnell und leicht den Wert, den die Datenbank darstellt, zu nutzen, so daß hier ein höheres Risiko für die Entstehung von Ablegern existieren dürfte. In wie weit die „Originalversion“ dann von diesen Ablegern noch profitieren kann,

[3] Eine weitere Lizenz für Dokumente, die sich jedoch nicht grundlegend von der OPL unterscheidet ist die GNU Free Documentation License [FSF00a]

dürfe i.d.R. davon abhängig, wie weit sich diese Ableger, insbesondere in Bezug auf das Datenbankschema, vom Original entfernen.

Der Verwaltungsaufwand für die Open Data-SDB dürfte gegenüber einer zentralisierten SDB ungefähr gleich hoch sein, da zwar zu der eigentlichen Verwaltung einer Instanz der SDB noch Koordinationsaufgaben zwischen den i.d.R. verteilt arbeitenden Entwicklern hinzukommen, dafür aber ähnliche Projekte im GPL-Umfeld häufig ein hohes Maß an selbständiger Koordination aufweisen. Sofern davon ausgegangen wird, daß es nicht zu einer starken Aufspaltung des Projekts käme, kann auch davon ausgegangen werden, daß der Gesamtaufwand über die Instanzen hinweg verhältnismäßig gering ist. Der Aufwand für den Betrieb und die Unterhaltung sollte ebenfall relativ niedrig sein, da sich die notwendigen Tätigkeiten auf viele freiwillige „Mitarbeiter“ verteilen.

Für die technische Durchführbarkeit gilt das gleiche, das auch für die zentralisierte SDB gilt. Die Kosten sollten sich beim Open Data-Modell als sehr niedrig erweisen, da, entsprechende Interessenten vorausgesetzt, mit viel freiwilliger Mitarbeit zu rechnen sein dürfte, so daß sich die Kosten primär auf Technik und Infrastruktur beschränken sollten.

Bezüglich der Akzeptanz durch potentielle Nutzer ist festzustellen, daß Open Source-Projekte in jüngster Vergangenheit ein hohes Maß an Akzeptanz sowohl privater als auch kommerzieller Nutzer erreicht haben, da sie sich weitgehend durch ein hohes Qualitätsniveau auszeichnen. Gelingt es, für das Open Data-Projekt ein ähnlich hohes Qualitätsniveau zu erreichen, so ist damit zu rechnen, daß dann auch eine ähnliche Akzeptanz erreicht wird. Weiterhin bietet der Open Data-Ansatz gute Möglichkeiten für Spezialisierungen, da Dritten die Datenbank und eventuell Verwaltungssoftware zugänglich ist und somit auch spezialisierte Instanzen entstehen können. Würde dabei nur der Inhalt und nicht beispielsweise das Schema der Datenbank geändert, so wäre dies wiederum ein Schritt in Richtung einer Föderation.

Recherchen könnten in diesem Modell in einer Instanz aufgrund des vollständig dokumentierten Schemas von allen Nutzern sehr gut durchgeführt werden. Insbesondere die Möglichkeit, komplexe Anfrage an eine lokale Kopie der Datenbank zu schicken, bietet sich für Forschungszwecke an. Sofern sich verschiedene Entwicklungszweige der Open Data-SDB herausbilden, ist bei identischen Schemata immernoch eine gute Recherchierbarkeit über alle Instanzen gegeben, wird jedoch auch das Schema modifiziert, so werden die Anfragen komplizierter, da diese an die verschiedenen Schemata angepaßt werden müßten. Nichtsdestotrotz blieben die Anfragen möglich, da man aufgrund der Lizenz auf alle Instanzen frei zugreifen könnte.

Bezüglich der Qualität der Daten in der bzw. den SDBs und der Widersprüche zwischen den Instanzen der SDB muß unterstellt werden, daß die Qualität der Daten massiv davon abhängt, wieviele Entwicklungszweige entstehen. Es ist zu erwarten, daß die Qualität bei nur einer öffentlich unterstützten SDB höher ist, als wenn sich die „gemeinschaftlichen Anstrengungen“ auf viele verschiedene SDBs verteilen. Bei massiver Unterstützung der „original“ Open Data-SDB ist zu erwarten, daß der qualitätssteigernde Effekt ähnlich ausfällt, wie bei Open Source-Projekten. Selbst dann müssen jedoch Überlegungen angestellt werden, ob wirklich *jeder* Nutzer Daten in eine Instanz einer SDB einfügen darf oder ob ein sich aufgrund von Qualifikationen herausbildendes *Change Management Board* die Aufnahme neuer Daten in die SDB überwachen sollte. So könnte besser sichergestellt werden, daß die SDB konsistente Daten enthält. Redundanzen bei der Erfassung von Daten wären

nur in dem Maß zu erwarten, in dem sich mehrere verschiedene Instanzen der SDB herausbilden.

Potentiell wäre eine hohe Verfügbarkeit der SDB zu erwarten. Allein schon die Tatsache, daß jeder Interessent die SDB regelmäßig kopieren und dann lokal nutzen kann, läßt einen dauerhaften Ausfall oder eine Sabotage aussichtslos erscheinen. Dazu wäre es denkbar, daß mehrere identische Kopien der SDB von verschiedenen Gruppen zum Zugriff bereitgestellt werden. Auch in diesem Fall wäre eine längere Nichtverfügbarkeit unwahrscheinlich.

3.1.1.5 Einordnung der Organisationsmodelle

Wie auch schon in [SHHB00] ausgeführt, können die verschiedenen Organisationsmodelle bezüglich der Aspekte Kontrolle über *die* Datenbank und Anzahl der zu erwartenden Instanzen von SDBs eingeordnet werden. Abbildung 3.1 zeigt diese Einordnung. Bei einer zentralisierten SDB ist ein Höchstmaß an Kontrolle über Zustand und Betrieb der SDB zu erwarten. Mit zunehmender Variablität und Flexibilität des Organisationsmodells geht ein zunehmender Kontrollverlust einher und die potentielle Anzahl der zu erwartenden Datenbanken nimmt zu, mit allen bereits beschriebenen Konsequenzen.

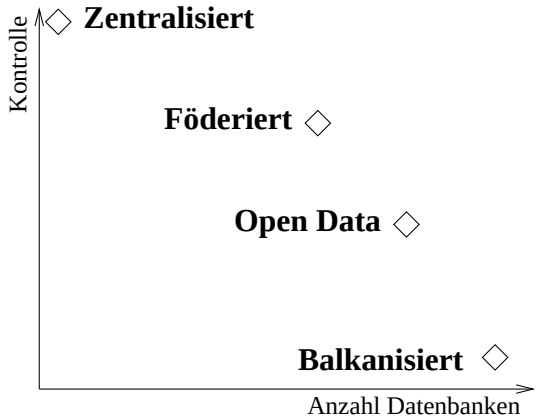

Abbildung 3.1: Einordnung der Organisationsmodelle

Eine Einordnung der verschiedenen Organisationsmodelle (der Tendenz nach) bezüglich verschiedener, oben erläuterter Kriterien gibt Tabelle 3.1 wieder. Dabei wird immer das Gesamtmodell und nicht nur einzelne SDBs betrachtet.

3.1.2 Zugriff

Der Aspekt *Zugriff* behandelt die Frage, wer auf die SDB zugreifen kann bzw. darf. Dabei ist zu unterscheiden, ob und wie überwacht werden soll,

- wer Beiträge zur SDB bereit stellt und
- wer Daten aus der SDB abruft.

Kriterium	Balkanisiert	Zentralisiert	Föderiert	Open Data
Verwaltungsaufwand	hoch	mittel	mittel	niedrig
Aufwand für die Unterhaltung	hoch	mittel	mittel	niedrig
Technische Durchführbarkeit	gut	mittel	gut	gut
Kosten	hoch	mittel	mittel	niedrig
Akzeptanz	mittel	niedrig	hoch	hoch
Spezialisierungsmöglichkeit	hoch	niedrig	hoch	hoch
Recherchierbarkeit	schlecht	gut	gut	mittel
Potentielle Qualitätsunterschiede	hoch	niedrig	niedrig	mittel
Potentielle Widersprüche	hoch	niedrig	niedrig	mittel
Redundanzen	hoch	niedrig	niedrig	mittel
Verfügbarkeit	hoch	mittel	hoch	hoch

Tabelle 3.1: Einordnung der Organisationsmodelle

Die Entscheidung über diese beiden Punkte hat u.a. Einfluß auf die Anzahl erfaßbarer Schwachstellen und die Möglichkeit eine Qualitätssicherung zu etablieren. Die Anzahl erfaßbarer Schwachstellen ist deshalb betroffen, weil durch die Behandlung des Themas Zugriff, die Akzeptanz der und damit die Beteiligung an der SDB durch kommerzielle und nichtkommerzielle Nutzer beeinflußt wird. Qualitätssicherung wiederum setzt voraus, daß Beiträge Nutzern zugeordnet werden können, so daß beispielsweise die Möglichkeit besteht, zu erkennen, wenn ein bestimmter Teilnehmer über einen längeren Zeitraum falsche Informationen an die SDB meldet.

Es können die drei Zugriffsalternativen *anonym*, *individualisiert* und *personalisiert*, jeweils für lesenden und schreibenden Zugriff auf die SDB, unterschieden werden. Anonymer Zugriff bedeutet unbeschränkter Zugang zur SDB, ohne daß eine Kontrolle über den Zugriff möglich wäre. Individualisierter Zugriff wiederum bedeutet, daß Nutzer ein Pseudonym verwenden, um auf die SDB zuzugreifen. So können Zugriffe zumindest dem Pseudonym zugeordnet werden. Unter personalisiertem Zugriff ist zu verstehen, daß der „reale“ Benutzer dem Betreiber tatsächlich bekannt ist und daß der Betreiber die Daten des Nutzers auf geeignete Weise überprüft hat. Dabei sind zwei Varianten denkbar: der Nutzer kann bei der Nutzung der SDB mit einem Pseudonym auftreten, das nur der Betreiber dem Nutzer zuordnen kann, oder aber der Nutzer tritt in der SDB grundsätzlich unter seinem echten Namen auf. Bei der Abwägung zwischen den Möglichkeiten sind verschiedene Aspekte zu berücksichtigen:

- *Mißbrauchsmöglichkeit* - Kann durch den Zugriff auf die SDB Mißbrauch stattfinden, beispielsweise durch das Eintragen von falschen Daten in die SDB?
- *Privatsphäre* - Jeder Nutzer hat einen Anspruch auf Privatsphäre. Außerdem kann es sein, daß bestimmte Nutzer, beispielsweise Hacker, zwar Informationen beisteu-

ern möchten, jedoch nicht wünschen, daß ihre Identität bekannt wird, um sich beispielsweise nicht der Strafverfolgung auszusetzen. Genauso wäre es denkbar, daß ein Mitarbeiter eines Softwareunternehmens eine Schwachstellen in einem Produkt seines Arbeitgebers melden möchte, ohne daß er Gefahr läuft, dafür entlassen oder in Regreß genommen zu werden. Es ist allerdings auch möglich die Auffassung zu vertreten, daß ein Nutzer der SDB für seine Äußerungen verantwortlich gemacht werden können sollte. Dabei würde jedoch riskiert, daß bestimmte Nutzer keine Beiträge zur SDB beisteuern würden.

- *Schutz* - Meldet ein Nutzer Daten an die SDB, so muß sichergestellt werden können, daß niemand diese Meldung gegen den Melder nutzt. Dies könnte beispielsweise dadurch geschehen, daß Dritte davon ausgehen, daß eine Person, die ein Schwachstelle meldet, möglicherweise von dieser Schwachstelle auch betroffen ist. Auch könnten Dritte der Auffassung sein, daß ja schon die Tatsache, daß Informationen über bestimmte Schwachstellen recherchiert werden, zeigt, daß der Recherchierende über diese Schwachstelle angreifbar sein könnte. Beide Punkte würden dann ein legitimes Interesse begründen, bei Abfragen auf einen nicht personalisierten Zugangsweg zurückgreifen zu können.

Eine sinnvolle Lösung für das Zugriffsproblem muß aus oben genannten Gründen in einer Koexistenz von individualisiertem und personalisiertem schreibendem Zugriff bestehen. Anonymer Zugriff sollte nur lesend gewährt werden, da sonst keinerlei Mißbrauchskontrolle möglich wäre. Andererseits bietet die Wahl der Zugriffsmethode durch einen Nutzer durchaus eine gewisses Indiz für die Vertrauenswürdigkeit eines Nutzers, denn ein Nutzer, der der SDB-Administration bzw. jedem bekannt ist, wird kaum leichtfertig falsche Meldungen in Umlauf setzen.

Weiterhin ist zu berücksichtigen, daß eine Kontrolle des Zugriffs in nicht zentralisierten Modellen schwierig ist. Im föderierten Modell könnte noch versucht werden die Föderationsmitglieder durch Vereinbarungen an eine bestimmte Zugriffspolitik zu binden. Im Open Data- und im balkanisierten Modell können Zugriffskontrollen jedoch nur noch bei einzelnen individuellen SDBs durchgeführt werden, jedoch nicht mehr parallel im gesamten „Modell“. Es wäre zwar theoretisch denkbar, daß sich in diesen Fällen die SDB-Instanzen durchsetzen, die ein entsprechendes Konzept verfolgen, sicher wäre dies jedoch nicht.

3.1.3 Veröffentlichungspolitik

Unter Veröffentlichungspolitik ist zu verstehen, wie insbesondere brisante Informationen über Schwachstellen, wie beispielsweise die sogenannten Exploit-Skripten, also Beispielprogramme, die demonstrieren, wie eine Schwachstelle ausgenutzt werden kann, veröffentlicht werden.

Es sind drei verschiedene Veröffentlichungspolitiken denkbar, die angewandt werden können, wenn Informationen über eine potentielle neue Schwachstelle bekannt werden:

- *Unmittelbare Veröffentlichung aller verfügbaren Informationen* - Hierbei werden grundsätzlich alle verfügbaren Informationen veröffentlicht; gleichzeitig sollte der Hersteller bzw. Betreuer der Software informiert werden.

- *Einstufige Veröffentlichungspolitik* - Hierbei werden Informationen über die Schwachstelle für eine bestimmte Zeit (Gnadefrist, Grace Period) zurückgehalten. In dieser Zeit hätte der Hersteller / Betreuer, der sofort kontaktiert würde, die Möglichkeit, die Schwachstelle zu beheben. Gleichzeitig können im Rahmen des SDB-Betriebs die Informationen zur Schwachstelle genauer geprüft und verifiziert werden. Mit Ablauf der Grace Period werden alle bekannten Informationen veröffentlicht.

- *Mehrstufige Veröffentlichungspolitik* - Bei der mehrstufigen Veröffentlichungspolitik werden mit Bekanntwerden einer potentiellen Schwachstelle alle bis dahin bekannten Informationen veröffentlicht, die nicht für Angriffe mißbraucht werden können. Es würden alle Informationen über die Schwachstelle selbst und über Möglichkeiten sich gegen die Ausnutzung der Schwachstelle zu schützen, veröffentlicht, jedoch beispielsweise Exploit-Skripten zurückgehalten. Gleichzeitig würde der Hersteller / Betreuer der Software über die potentielle Schwachstelle informiert. Nach Ablauf der Grace Period werden alle noch nicht veröffentlichten Informationen über die Schwachstelle bereitgestellt. Problem dieser Veröffentlichungspolitik ist, daß häufig schwer abzuschätzen sein dürfte, wieviel Informationen bekannt gegeben werden müssen, damit ernsthafte Nutzer in der Lage sind, zu verifizieren, ob sie überhaupt betroffen sind, ohne dabei bereits durch die veröffentlichte Information die Bedrohung herbeizuführen, die eigentlich durch die mehrstufige Veröffentlichungspolitik verhindert werden soll.

Die ein- und mehrstufigen Veröffentlichungspolitiken sind dabei nur realisierbar, wenn Beiträge zur SDB nur nach vorhergehender „Kontrolle“ zur Diskussion gestellt würden, d.h. wenn ein kontrollierender Moderator vorhanden wäre.[4]

Des weiteren ist zu diskutieren, ob die Grace Period, sofern eine solche einführt werden soll, je nach Schwere der Schwachstelle in der Länge variabel gestaltet werden sollte. Hierbei wäre wiederum zu diskutieren, ob bei einem hohen Gefährdungspotential die Schwachstelle früher veröffentlicht werden sollte, damit sich potentielle Opfer schnell schützen können oder ob man die Grace Period in solch einem Fall verlängert. Bei dieser Diskussion spielt die Hauptursache für die Diskussion um eine Grace Period ein wichtige Rolle. Es kann ohne Zweifel davon ausgegangen werden, daß die *echten* Hacker, ob nun gut- oder böswillig, von Schwachstellen frühzeitig, i.d.R. selbst vor vielen „regulären“ Spezialisten, Kenntnis erlangen. Daher ist es sinnlos, eine Grace Period einzuführen, um Anwender vor Attacken von Dritten zu schützen, die sowieso Kenntnis von der Schwachstelle haben. Das Hauptproblem, das Anlaß gibt, über die Einführung einer Grace Period nachzudenken, sind die sogenannten *Script Kiddies*. Script Kiddies sind eine Gruppe böswilliger Amateure, die, wenn sie beispielsweise Zugriff auf Exploit-Skripte haben, Dritten mutwillig Schaden durch Verwendung dieser Exploit-Skripten zufügen. Die Diskussion bezüglich einer Grace Period muß sich daher primär um die Gefährdung durch diese Gruppe von Angreifern drehen.

Die Diskussion um die Veröffentlichung von vollständigen Schwachstellendetails wird schon lange und heftig geführt. Eine gute Diskussion der Vor- und Nachteile ist in [BS00b, BS00c] zu finden.

Ein Argument gegen die einstufige Veröffentlichungspolitik ist im übrigen die Tatsache, daß viele Hersteller grundsätzlich erst Lösungen für Schwachstellen entwickeln, wenn diese öffentlich bekannt sind [BS00c]. Somit müßte im Rahmen der einstufigen Veröffentlichungspolitik erwartet werden, daß die Grace Period ohne Wirkung „verpufft“.

[4] Vgl. dazu Abschnitt 3.1.4

Stellt man abschließend die Überlegung an, daß die Anzahl der Quellen, die Exploit-Skripten veröffentlichen, stetig steigt (Newsgroups, Mailing-Listen, IRC), so daß erstens die Script-Kiddies höchstwahrscheinlich auf die eine oder andere Art sowieso Zugriff auf Exploit-Skripten erhalten und zweitens das Nichtveröffentlichen von detaillierten Informationen potentielle Nutzer dazu bringen könnte, andere Quellen als die entsprechende SDB zu nutzen, muß festgestellt werden, daß es sinnvoll erscheint, einer unmittelbaren Veröffentlichungspolitik zu folgen. Im übrigen muß festgestellt werden, daß eine ein- oder mehrstufige Veröffentlichungspolitik nur im zentralisierten Modell und eventuell, bei entsprechenden Vorgaben, im föderierten Modell allgemein durchzusetzen ist. Ansonsten könnte die Veröffentlichungspolitik nur in einzelnen Instanzen implementiert werden.

3.1.4 Qualitätssicherung

Eine wichtige Frage für den Betrieb einer SDB ist, wie die Qualität der in der SDB gespeicherten Daten auf Dauer auf hohem Niveau gehalten werden kann. Beim Themenkreis der Qualitätssicherung können die drei Aspekte *Moderation*, *Bewertung* und *Bonussystem* unterschieden werden:

Moderation Wenn innerhalb der SDB über neue, potentielle Schwachstellen diskutiert wird, so muß genau überlegt werden, wie man diese Diskussionen kontrolliert, damit sie sich auf einem sachlichen Niveau bewegen. Dazu bietet es sich an, sich eines Moderators bzw. einer Gruppe von Moderatoren zu bedienen. Dabei kann an einen „steuernd“ in die Diskussion eingreifenden Moderator gedacht werden, der im äußersten Fall eine Diskussion beendet, oder aber an einen Moderator, der Beiträge zur Diskussion erst freigeben muß, damit sie die Teilnehmer der Diskussion überhaupt sehen. Der letztgenannte Typ des Moderators entspräche dem Moderator der Bugtraq-Mailing-Liste, Elias Levy (Spitzname: aleph1). Bei der Diskussion über den Einsatz eines Moderators ist allerdings der damit verbundene, u.U. sehr hohe Aufwand zu berücksichtigen. Außerdem muß der „freigebende“ Moderator genügend Zeit für seine Tätigkeit haben, da ansonsten möglicherweise die Aktualität der Diskussion leidet. Zusätzlich muß der „freigebende“ Moderator sehr behutsam bei seiner Tätigkeit vorgehen, da er ansonsten u.U. in den Verdacht der Zensur gerät. Zur Vermeidung dieses Verdachts wäre außerdem eine Charta hilfreich, die, neben anderen Vorgaben und Regelungen für die SDB, die genauen Kriterien für das Eingreifen des Moderators enthält.

Für den Betrieb der SDB ist festzustellen, daß zumindest ein steuernd eingreifender Moderator sicherlich wünschenswert ist. Bezüglich der Veröffentlichungspolitik ist festzustellen, daß die ein- und mehrstufige Veröffentlichungspolitik einen freigebenden Moderator voraussetzt, da sie ansonsten nicht realisierbar ist. Außerdem kann Moderation nur in einem zentralisierten oder föderierten System durchgesetzt werden; in den übrigen Systemen könnte Moderation nur für jeweils eine SDB eingeführt werden.

Bewertung Ein weiteres hilfreiches Werkzeug für die Qualitätssicherung wäre eine Bewertung von Beiträgen der Teilnehmer. Dabei könnten einerseits die Beiträge selbst bewertet werden, andererseits können damit die Nutzer bewertet werden, so daß mittelfristig die Qualität eines Beitrags aus der Bewertung des Nutzers antizipiert werden kann. Eine solche Bewertung könnte auf drei verschiedene Arten vorgenommen werden:

- durch ein Bewertungsteam aus qualifizierten Personen, das von den SDB-Betreibern bestimmt wird,

- durch ein Bewertungsteam aus qualifizierten Freiwilligen, das von den Nutzern der SDB in regelmäßigen Abständen gewählt wird oder

- durch ein *Web of Trust*, das ähnlich wie beim Verschlüsselungsprogramm Pretty Good Privacy (PGP) auf Bewertungen der Nutzer selbst basiert.

Die ersten beiden Ansätze sind verhältnismäßig ähnlich, jedoch bietet der erste Ansatz zwei grundlegende Probleme: erstens müßten die Nutzer den Mitarbeitern der SDB vertrauen, was bei Sicherheitsfragen u.U. nur schwer zu erreichen sein dürfte, und zweitens muß die SDB-Administration dafür auch geeignete Mitarbeiter aufbieten können, denn es handelt sich bei dabei um eine sehr anspruchs- und verantwortungsvolle Tätigkeit. Beim zweiten Ansatz ist das prinzipielle Problem vorhanden, daß Freiwillige für diese zeitaufwendige Tätigkeit gewonnen werden müssen, die dann auch noch für diese Tätigkeit qualifiziert sind. Die dritte Möglichkeit erscheint auf den ersten Blick verlockend, auch funktioniert das Prinzip bei PGP relativ zuverlässig. Andererseits bietet diese Variante auch viel Spielraum für Mißbrauch, indem sich beispielsweise Gruppen von Nutzern gegenseitig „hochwerten“, ohne daß eine tatsächliche Qualifikation oder Leistung vorliegt.

Im Rahmen der Qualitätssicherung und insbesondere der Bewertung bietet es sich weiterhin an, bei den Schwachstellenberichten zwischen zwei Zuständen zu unterscheiden: *Gegenstand der Diskussion* und *Offizielle Meldung*. So kann unterschieden werden, was als offizielle Warnung der SDB gilt und was zwar gemeldet wurde, aber noch nicht als verifiziert bzw. zuverlässig gilt.

Weiterhin besteht die Möglichkeit verschiedene Beiträge zu bewerten. Einerseits ist es denkbar, nur die initialen Schwachstellenmeldungen zu bewerten. Dies wäre jedoch gegenüber den Diskussionteilnehmern, die Schwachstellen auf anderen Konfigurationen überprüfen oder aber wichtige Diskussionsbeiträge beisteuern, ungerecht. Daher sollten sowohl Originalbeiträge, als auch wichtige Beiträge zu dieser Schwachstelle bewertet werden. Eine Bewertung könnte auf Basis einer entsprechenden Skala durchgeführt werden, so daß schlechte, unnütze oder gar kontraproduktive Beiträge auch negativ bewertet werden können. Für die Bewertung eines Teilnehmers könnte dann beispielsweise ein gewichteter Mittelwert aus den Bewertungen seiner Beiträge verwendet werden, der weiter zurückliegende Beiträge mit abnehmender Gewichtung berücksichtigt. Problematisch an einer Bewertung ist, daß wiederum einige Teilnehmer annehmen könnten, daß Nutzungsprofile von ihnen erstellt werden. Andererseits ist dies die einzige, einfach realisierbare Möglichkeit, anhand der a posteriori Bewertung von Beiträgen eine a priori Annahme über die Qualität eines neuen Beitrags zu erhalten.

Abschließend betrachtet wäre eine Bewertung durch ein gewähltes Team qualifizierter Freiwilliger ein sinnvoller Zusatz zur SDB. Sollte dies mangels Freiwilliger nicht möglich sein, kann immernoch darüber nachgedacht werden, auf eine Bewertung zu verzichten. Für den Einsatz eines konsistenten Bewertungssystems gelten bezüglich der Organisationsmodelle die gleichen Aussagen wie für die Moderation.

Bonussystem Eine denkbare Möglichkeit zur Steigerung der Qualität und Attraktivität der SDB wäre ein Bonussystem, das auf der Bewertung der Beiträge zur SDB basieren würde. Einsetzbar ist ein Bonussystem nur, wenn durch die Nutzung der SDB Kosten für Individuen und Unternehmen entstehen. Auf den Finanzierungsaspekt wird im nächsten Abschnitt genauer eingegangen. Es wäre möglich, die Kostenbeiträge eines Nutzers durch „Anrechnung“ guter Beiträge zu senken oder gar abzugelten. Somit könnten aktive Teilnehmer, die nützliche Beiträge zur SDB beitragen, die in diesem Szenario durch die Nutzung der SDB entstehenden Kosten senken. Das Risiko eines Bonussystems besteht allerdings darin, daß künstlich Beiträge generiert werden, die nur dazu dienen sollen, den Kostenbeitrag eines Teilnehmers zu senken. Selbst wenn dabei qualitativ schlechte Beiträge durch das Bewertungssystem herausgefiltert werden, ergäbe sich ein störendes „Rauschen“, das die Bearbeitung von wichtigen Beiträgen verzögern würde. Daher sollte von der Einführung eines Bonussystems abgesehen werden. Bezüglich des Organisationsmodells bietet sich ein Bonussystem für den Open Data-Ansatz im Prinzip nicht an. Beim zentralisierten und eventuell beim föderierten Modell könnte ein Bonussystem eingeführt werden. Beim balkanisierten Modell kann es nur für einzelnen Instanzen eingeführt werden.

3.1.5 Finanzierung

Es sind verschiedene Finanzierungsmöglichkeiten denkbar. Dies sind:

- Jeder Nutzer (privat und kommerziell) zahlt einen monatlichen Beitrag
- Einführung eines Pay-per-Query-Systems mittels eines Micropayment-Systems
- Nur Unternehmen zahlen einen monatlichen Beitrag
- Unternehmen bzw. Organisationen treten als „Sponsoren“ auf
- Staatliche Unterstützung
- Einnahmen aus der Zurverfügungstellung von Value Added Services
- Spenden
- Werbung

Gegen einen monatlichen Beitrag für jeden Nutzer spricht die Vermutung, daß ein solches System die nichtkommerziellen Nutzer wahrscheinlich von der Nutzung abhielte. In diesem Zusammenhang muß außerdem vermutet werden, daß auch Mitarbeiter von Unternehmen oder Hacker sich nicht an der SDB beteiligen würden. Außerdem wäre ein solches Mitgliedersystem mit einem enormen Verwaltungsaufwand verbunden.

Dieser Aufwand könnte mit einem Pay-per-Query-System reduziert werden, allerdings gibt es hier einerseits technische Probleme, denn noch immer hat sich keines der am Markt befindlichen Systeme etablieren können, und andererseits dürfte es für dieses Modell außer von privaten auch von kommerziellen Nutzer erheblichen Widerstand geben, da bei intensiver Nutzung unter Umständen sehr hohe Kosten entstehen könnten, so daß mit diesem System auch die Akzeptanz der SDB gefährdet wäre.

Die nächste Möglichkeit, daß nämlich nur Unternehmen Beiträge zahlen, ist, abgesehen vom administrativen Aufwand, eine Variante die in Erwägung gezogen werden könnte. Allerdings ist es dann notwendig, einen hochprofessionellen und kommerziellen Service zu bieten, so daß dieser Schritt sehr gut durchdacht werden sollte.

Die Variante, daß Unternehmen bzw. Organisationen quasi als Sponsoren der SDB auftreten, bietet einige interessante Optionen. Die Finanzierung der SDB könnte dabei einerseits sichergestellt werden, auf der anderen Seite könnte durch die enge Zusammenarbeit mit diesen Unternehmen auch speziell auf deren Interessen und Bedürftnisse eingegangen werden, so daß beiden Seiten einen Vorteil aus der Zusammenarbeit ziehen könnten. Im Falle großer IT-Organisationen könnte eine solche Zusammenarbeit möglicherweise auch zu einer Standardisierung des Formats von Schwachstelleninformationen führen. Natürlich müßte sichergestellt sein, daß die engere Zusammenarbeit mit den Unternehmen keinen Einfluß auf die objektive Tätigkeit der SDB-Betreiber hat. Würde dieser Eindruck bei anderen Nutzern der SDB entstehen, wäre es leicht möglich, daß die breite Unterstützung für die SDB ausbleiben könnte.

Staatliche Unterstützung, beispielsweise in Form von Zuschüssen, wäre eine weitere Möglichkeit zur Finanzierung der SDB. Auch hier müßte darauf geachtet werden, daß nicht der Eindruck einer Einflußnahme von Regierungen auf die SDB-Betreiber entstehen würde. Andererseits ist in Deutschland zumindest nicht mit einer potentiell ähnlich starken Einflußnahme zu rechnen, wie in den USA.

Die nächste denkbare Finanzierungsmöglichkeit besteht darin, einen kostenlosen, freien Zugang zur SDB zu gewähren und die Finanzierung durch das kostenpflichtige Anbieten von Mehrwertdiensten sicherzustellen, beispielsweise durch die Zurverfügungstellung von speziellen Sicherheitsanalyse-Werkzeugen, die direkt auf die Daten der SDB zugreifen. Im Falle des Open Data-Modells muß dann jedoch mit potentiell beliebig vielen Wettbewerbern konkurriert werden, die auf die gleichen Daten zugreifen können. In diesem Fall herrschte ein Qualitätswettbewerb, wie es bereits bei vielen Open Source-Projekten in Bezug auf Beratungstätigkeiten der Fall ist.

Die letzten beiden Finanzierungsmöglichkeiten sind Spenden und Werbung. Spenden werden schon heute für die Finanzierung erfolgreicher Open Source-Projekte genutzt und stellen somit auch für die SDB eine Option dar. Online-Werbung ist im Fall des Erfolges der SDB ebenfalls eine Möglichkeit, den Betrieb teilweise zu finanzieren. Mögliche Werbekunden könnten beispielsweise Hersteller von Sicherheitswerkzeugen sein.

3.1.6 Rechtliche Organisation

Bezüglich der rechtlichen Organisation der SDB stehen, je nach verfolgtem Geschäftsziel, unterschiedliche Möglichkeiten offen. Im Falle eines Open Data-Modells bei Verzicht auf kommerzielle Interessen könnte auf jegliche besondere rechtliche Ausgestaltung verzichtet werden.[5] Soll ein „non-profit“-Ansatz verfolgt und die Finanzierung beispielsweise über Sponsoren sichergestellt werden, so könnte eine Organisation verwendet werden, die beispielsweise der der Object Management Group (OMG) ähnelt. Es könnte ein Verein gegründet und die dauerhaft interessierten Sponsoren könnten als Mitglieder des Vereins „tätig“

[5] Dies könnte jedoch in der Bundesrepublik Deutschland je nach Art der Tätigkeit dazu führen, daß im Falle rechtlicher Auseinandersetzungen die Existenz einer Gesellschaft bürgerlichen Rechts unterstellt würde.

werden. Bei einer gewinnorientierten Ausrichtung böte sich, eine entsprechende Anschubfinazierung vorausgesetzt, die Rechtsform einer GmbH an, da sich dabei das finanzielle Risiko für die Gesellschafter im vertretbaren Rahmen hielte.

3.1.7 Rechtliche Aspekte

Zwei wichtige rechtliche Aspekte, die betrachtet werden müssen, sind *Haftung* und *Urheberrecht* bzw. *Copyright*.

Es muß geklärt werden, in wie weit der Betreiber einer SDB dafür haftbar gemacht werden kann, wenn mit Informationen aus der SDB ein Angriff auf einen Dritten gestartet wird. Es sind dabei durchaus Szenarien denkbar, bei denen nachvollziehbar wäre, daß die für den Angriff verwandte Information aus der SDB stammt.

Bezüglich des Urheberrechts bzw. Copyrights stellt sich die Frage nach dem Status von Beiträgen. Welche Regelungen sind notwendig bzw. möglich für Beiträge, die direkt zur SDB gemeldet werden? Können in den Nutzungsbedingungen der SDB Regelungen aufgenommen werden, so daß mit der Meldung das Urheberrecht oder zumindest das Copyright automatisch auf die SDB übergeht. Kann dabei ein Melder verlangen, daß sein Name, sofern bekannt, genannt oder nicht genannt wird? Wäre es denkbar, daß der Teilnehmer zumindest das Nutzungsrecht für die Informationen an die SDB abtritt und was würde geschehen, zöge er diese Erlaubnis wieder zurück? Desweiteren wäre zu klären, wie die rechtliche Lage bezüglich der Beiträge aussieht, die aus Newsgroups und Mailing-Listen übernommen werden? Wie sieht die Lage bezüglich des Urheberrecht und des Copyright bei Beiträgen im USENET und in Mailing-Listen generell aus? Wie gestaltet sich die Übernahme von Daten aus anderen SDBs rechtlich?

Die rechtlichen Aspekte wurde im Rahmen dieser Arbeit nicht weiter untersucht und werden voraussichtlich Inhalt einer anderen Diplomarbeit sein.

3.1.8 Mitarbeiter

Ähnlich wie bei der rechtlichen Organisation stellt sich das Bild bezüglich der Mitarbeiter des Projekts dar. Kann im Rahmen des Open Data-Modells eine breite Beteiligung von Freiwilligen erreicht werden, so ist zu erwarten, daß der Personalbedarf für den eigentlichen Betreiber der Hard- und Software verhältnismäßig gering bleibt. Notwendig wäre ein Mitarbeiter, der die Entwicklungstätigkeiten koordiniert und sich möglichst selbst an der Entwicklung beteiligt. Im Fall eines geschlossenen Modells, wie beispielsweise einem zentralisierten oder föderierten Modell, muß damit gerechnet werden, daß der Personalaufwand höher ist, da man Systemadministratoren, Manager, Entwickler und IT-Sicherheits-Experten für den Betrieb der SDB benötigt.

3.1.9 Infrastruktur

Für den initialen Betrieb der SDB ist im Minimum ein leistungsfähiger Rechner notwendig, der in der Lage ist, ein Datenbankmanagementsystem, einige Serverprozesse, wie beispielsweise einen Webserver oder auch Teile der Zugangssoftware zur SDB, auszuführen.

Da Kommunikation zu diesem Server von außen möglichst verschlüsselt stattfinden sollte, ist es notwendig, daß der Rechner mit ausreichend Prozessor-Leistung ausgestattet ist. Außerdem sollte der Rechner breitbandig an das Internet angeschlossen werden, damit auch bei starker Nutzung kein Engpaß bei der Netzwerkübertragung entsteht. Sollte es später zu Engpässen bei der Prozessorleistung des Servers kommen, so kann in einem ersten Ausbauschritt ein getrennter Datenbank-Server und Anwendungs-Server verwendet werden.

3.1.10 Öffentliche Umfrage

Basierend auf den Vorüberlegungen für ein geeignetes Geschäftsmodell wurde ein verhältnismäßig umfangreicher Fragebogen erstellt [TRU00], der ermitteln sollte, wie *IT-Experten* aus verschiedensten Bereichen zu einzelnen, alternativ oder parallel realisierbaren Optionen, die sich aus den Vorüberlegungen ergaben, stehen.

3.1.10.1 Planung und Durchführung

Um eine möglichst große Gruppe von potentiellen Nutzern zu erreichen, wurde die Umfrage komplett in Englisch verfaßt. Zusätzlich wurde die auf einem Workshop des DFN-CERT vorgestellte erste Veröffentlichung der SEC://HOUSE-Gruppe [SHHB00] vollständig ins Englische übersetzt.

Um potentielle Teilnehmer auf die Umfrage aufmerksam zu machen, wurden Beiträge an diverse sicherheitsbezogene Mailinglisten[6] und Newsgroups versandt. Außerdem wurde versucht verschiedene Unternehmen der IT-, TK- und Finanz-Branche zu kontaktieren, um auch hier Teilnehmer für die Umfrage zu gewinnen. Schließlich wurden auch die über 300 Teilnehmer des DFN-CERT-Workshops angeschrieben.

Die Umfrage begann am 10. März 2000, in Anschluß an den Workshop des DFN-CERT. Während die Umfrage zum Zeitpunkt der Erstellung dieser Arbeit noch weitergeführt wird, wird hier der Stand der Umfrage vom 1. Juni 2000 präsentiert.

Um eine aussagekräftige Auswertung der Umfrage zu gewährleisten, wurden grundsätzlich alle Antworten vorgegeben, wobei die Standardeinstellung der Antworten eine neutrale Stellung war. So sollte eine Beeinflussung bei den Antworten vermieden werden, außerdem war es so möglich zu erkennen, ob eine Antwort gegeben wurde oder nicht.

Der Fragebogen wurde, neben einem kurzen Abschnitt zur Identität des Teilnehmers selbst, in sechs thematische Abschnitte aufgeteilt:

- Business model of the VDB
- Use of / Access to the VDB

[6] Bedauerlich war in diesem Zusammenhang, daß Elias Levy, der Moderator von Bugtraq, die Bitte um Teilnahme an der Umfrage nicht an die Mailing-Liste weiterleitete, obwohl die Bugtraq-Charta neue Forschungsvorhaben als zulässige Beiträge auf Bugtraq explizit nennt. Es erschien nicht sinnvoll eine Diskussion mit Levy zu beginnen, da dieser im IT-Sicherheitsbereich durchaus eine mächtige Position inne hat. Daher wurde auf weitere Versuche verzichtet, den *Request for Participation* über Bugtraq zu verteilen.

- Publication Policy
- Quality Assurance
- Financing
- “Last few questions”

Um den Teilnehmern Anmerkungen zu den einzelnen Abschnitten zu erlauben, wurde jeder Abschnitt mit einem Feld zur freien Texteingabe beendet.

3.1.10.2 Das Ergebnis

Nach aufwendigen Bemühungen gelang es bis zum 1. Juni 2000 immerhin 60 Teilnehmer für die Umfrage zu gewinnen.

Zu Beginn des Fragebogens wurden die Teilnehmer nach ihrer e-Mail-Adresse, der Branche in der sie arbeiten und ihrer aktuellen Tätigkeit gefragt. Dabei dient die e-Mail-Adresse dazu, die Teilnehmer, die dies wünschen, über die Ergebnisse der Umfrage zu informieren.[7]

Tabelle 3.2 zeigt die Verteilung der Teilnehmer nach Branchen. Auffällig ist dabei der mit 43% hohe Anteil aus dem Ausbildungs- & Forschungssektor, der jedoch bei einem universitären Forschungsprojekt kaum verwundern kann. Immerhin erfreulich ist, daß es gelang, ein Drittel der Teilnehmer aus dem Umfeld der IT-Unternehmen zu gewinnen.

Tabelle 3.3 gliedert die Teilnehmer der Umfrage bezüglich ihrer Tätigkeit in ihrem Unternehmen bzw. ihrer Organisation auf.

Im folgenden werden die Ergebnisse der Umfrage zusammengefaßt präsentiert. Für die ausführlichen Details sei auf Abschnitt A.1, Seite 95, verwiesen. Aufgrund der Teilnehmerzahl werden in den meisten Fällen drei aggregierte Gruppen betrachtet und zwar die Gruppe der Computer-related-Teilnehmer, die sich aus den beiden Gruppen Computer-related (IS, MIS, DP, Internet) und Computer-related (Software) zusammensetzt, der Gruppe *Education / Research* und den übrigen Teilnehmern. Nur in einzelnen Fällen, bei deutlichen Unterschieden innerhalb der aggregierten Gruppen, wird auf diese Unterschiede eingegangen.

Ergebnis Abschnitt 1: Organisationsmodell der SDB Abschnitt 1 des Fragebogens enthielt Fragen bezüglich des Organisationsmodells der SDB.

Auf die Frage, ob die Nutzer bereit wären, eine zentralisierte SDB zu nutzen, antwortete eine große Mehrheit von 74% der Befragten mit Ja. Es liegt jedoch ein deutlicher Unterschied zwischen den Antworten der Computer-related-Nutzer und allen übrigen Nutzern vor. Während sich 95% der Computer-related-Nutzer zur Nutzung einer zentralisierten SDB bereit erklärten, waren es aus den anderen Gruppe nur ca. 63%.

[7] Von den 60 Teilnehmern haben immerhin 46 eine zumindest syntaktisch korrekte e-Mail-Adresse angegeben.

Branche	Abs. Häufigkeit	Rel. Häufigkeit [%]
education, research	26	43,33
computer-related (IS, MIS, DP, Internet)	13	21,67
computer-related (software)	7	11,67
no details please	7	11,67
banking/finance/real estate	1	1,67
business supplies or services	1	1,67
engineering/construction	1	1,67
entertainment/media/publishing	1	1,67
government	1	1,67
manufacturing/distribution	1	1,67
medical/health services	1	1,67

Tabelle 3.2: Aufschlüsselung der Teilnehmer der Umfrage nach Branche

Tätigkeit	Abs. Häufigkeit	Rel. Häufigkeit [%]
computer technical/engineering	25	41,67
no details please	13	21,67
academic/educator	11	18,33
college/graduate student	4	6,67
executive/managerial	2	3,33
service/customer support	2	3,33
self-employed/own company	1	1,67
other	1	1,67
n/a	1	1,67

Tabelle 3.3: Aufschlüsselung der Teilnehmer der Umfrage nach ihrer Tätigkeit

Auf die zweite Frage, ob die Befragten eine föderierte SDB bevorzugen würden, antworteten sogar 83% mit Ja. Hierbei fielen keine signifikanten Unterschiede zwischen den drei Gruppen der Computer-related-Nutzer, der „Forscher“ und der übrigen Befragten auf.

Die dritte Frage des ersten Abschnitts beschäftigte sich damit, ob eine Spezialisierung nützlich oder hilfreich für den SDB-Betrieb sein könnte. Die Zustimmung von 90% für eine Spezialisierung der Datenbank war sehr groß. Zwischen den Teilnehmergruppen ergaben sich keine signifikanten Unterschiede.

Die vierte Frage zielte auf die Bereitschaft zur Nutzung einer „Open Data“-SDB. Von allen Befragten sprachen sich für diese Option 89% aus. Erstaunlicherweise war der Anteil unter den Befragten aus dem Bereich *Computer-related* am höchsten, denn hier stimmten 100% für diese Lösung, gegenüber 85% des Bereichs *Education / Research* und 79% bei den übrigen Befragten. Offensichtlich hat sich bei den Computer-related-Nutzern das „Open Source - GNU GPL“-Paradigma schon etabliert, während es selbst im Forschungs-Bereich eine höhere Skepsis gegenüber diesem Modell zu geben scheint.

Die nächste Frage befaßte sich mit einer möglichen Erhöhung der Mißbrauchswahrscheinlichkeit durch die Verwendung des Open Data-Modells. Während 60% der Teilnehmer der Auffassung waren, daß mit dem *Open Data*-Ansatz kein erhöhtes Risiko für eine höhere „Verschmutzung“ der Daten durch zufälliges oder vorsätzliches Einspeisen falscher Daten einhergehe, war es doch überraschend, daß von den Teilnehmern der Gruppe *Education / Research* 53,85% eine solche Möglichkeit bejahten, während aus der Gruppe *Computer-related* nur 25% diese Gefahr sahen. Hier wäre eigentlich ein umgekehrtes Ergebnis zu erwarten gewesen, wenn ein verstärktes Mißtrauen kommerzieller IT-Nutzer gegenüber offenen Lösungen unterstellt würde, wie es in der Vergangenheit häufig gegenüber entsprechenden Lösungen wie dem Betriebssystem Linux oder der HTTP-Server Apache existierte. Offensichtlich hat hier bereits ein Prozeß des Umdenkens eingesetzt.

Die sechste Frage bezog sich auf die Bereitschaft an Erweiterungsarbeiten der SDB teilzunehmen. Immerhin 40% der Teilnehmer antworteten auf diese Frage, daß sie bereit wären, an Arbeiten an der SDB mitzuwirken. Dabei gab es keine auffälligen Abweichungen zwischen den Branchen der Teilnehmer.

Die siebte Frage des ersten Abschnitts bezog sich auf die Notwendigkeit, Beiträge zur SDB so zu anonymisieren, daß aus Schutzgründen kein Rückschluß auf den Melder möglich ist. Insgesamt ergab sich bei dieser Frage kein eindeutiges Bild. Von den Teilnehmern sagten 50%, daß Anonymisierung der Daten nicht zwingend notwendig sei, 45% sprachen sich dagegen für eine Anonymisierung aus. Interessant ist allerdings, daß nur 25% der Teilnehmer aus dem *Computer-releated*-Bereich sich für eine Anonymisierung aussprachen, aus dem Bereich *Ausbildung / Forschung* dagegen, sprachen sich 61,5% dafür aus; bei den restlichen Teilnehmern waren es 42,9%.

Frage 8 bezog sich auf die Attraktivität der Unterstützung von privaten Instanzen in einem föderierten Modell. Insgesamt 69% aller Teilnehmer sprachen sich für die Unterstützung privater Instanzen aus. Allerdings ist auch in diesem Fall ein Unterschied zwischen den „Forschern“ und den restlichen Teilnehmern zu beobachten. Die Forscher sprachen sich zu 81% dafür aus, während bei den restlichen Teilnehmern nur ca. 60% dafür stimmten.

Auf die Frage nach einer höheren Effizienz einer SDB-Föderation, die ihre Schwachstelleninformationen untereinander abgleicht, im Vergleich zum balkanisierten Modell, antworteten 92% aller Teilnehmer mit Ja. Bei den einzelnen Gruppen gab es keine auffälligen Unterschiede.

Die letzte Frage des ersten Abschnitts befaßte sich mit der möglichen Vorteilhaftigkeit eines homogenen Datenbankschemas für die Analyse von Schwachstelleninformationen. Auch bei dieser Frage zeigte sich ein eindeutiges Bild: 88% der Teilnehmer bejahten die Frage. Der computer- und der forschungsbezogene Bereich urteilten mit 95% bzw. 92,3% für die Vorteile eines homogenen Schemas, während jedoch die restlichen Teilnehmer darin nur zu 71% Vorteile erkennen konnten.

Ergebnis Abschnitt 2: Nutzung / Zugriff auf die SDB Abschnitt 2 des Fragebogens enthielt Fragen bezüglich der Nutzung und des Zugriffs auf die SDB.

In der erste Frage des Abschnitts wurde nach der Nutzung irgendeiner Quelle für Schwachstelleninformationen gefragt. Aus den Antworten ergab sich, daß zum Zeitpunkt der Umfrage 92% aller Teilnehmer mindestens eine Quelle über Schwachstellen nutzten. Teilnehmer der Gruppe Computer-related nutzten solche Quellen zu 95%, Forscher zu 92,3% und die restlichen Teilnehmer zu 85,7%.

Die nächste Frage befaßte sich mit der Nutzung existierender Schwachstellendatenbanken. Insgesamt 67% aller Teilnehmer nutzten mindestens eine SDBs. Allerdings lag der Anteil in der Gruppe Computer-related bei 85%, während bei den restlichen Teilnehmern der Anteil nur bei ca. 57,5% lag. Hier zeigt sich, daß die kommerziellen IT-Nutzer scheinbar eine höhere Notwendigkeit sahen, sich systematisch über Schwachstellen zu informieren.

Bezüglich der Nutzung von Newsgroups ergab sich dabei ein etwas ausgeglicheneres Bild als bei der Nutzung einer SDB. So nutzten 69% aller Teilnehmer zum Zeitpunkt der Umfrage Newsgroups als Quelle von Informationen zu Softwareschwachstellen. Teilnehmer der Gruppe Computer-related nutzten diese Quelle zu 70%, Forscher zu 61,5% und von den restlichen Teilnehmern nutzten 78,5% das USENET. Interessanterweise zeigte sich hier, entgegen den bisherigen Fällen ein deutlicher Unterschied zwischen den beiden Teilgruppen der Gruppe Computer-related. Die Teilnehmer, die im Software-Bereich arbeiten, nutzten Newsgroups nur zu 42%, während die Teilnehmer der Gruppe IS/MIS/DP/Internet dieses Medium zu 84% nutzten.

Die Nachfrage bezüglich der Nutzung von Mailing-Listen wie Bugtraq oder NTBugtraq ergab sich wieder ein ähnliches Bild wie bei der Nutzung einer SDB. Von allen Teilnehmern nutzten 83% Mailing-Listen. Bei den Computer-related-Nutzern zeigte sich auch hier wieder die hohe Zahl von 90%. Die restlichen Teilnehmer lagen „nur“ bei ungefähr 79%.

Die nächste Frage bezog sich darauf, ob der Teilnehmer selbst bzw. dessen Arbeitgeber eine private SDB unterhält. Nur bei 20% der Teilnehmer wurde zum Zeitpunkt der Umfrage eine private SDB eingesetzt. Bei 68% der Teilnehmer war dies nicht der Fall. Allerdings ist hier die Aufgliederung interessant. Die Gruppe Computer-related nutzte zu 30% private SDBs. Allerdings waren es im Bereich Software 57%, während es im Bereich IS/MIS/DP/Internet nur 15% waren. Die restlichen Teilnehmer nutzten private SDBs nur zu 15%. Hier ist zu vermuten, daß die Nutzer, die im Bereich Software tätig sind, für die zu entwickelnde Software SDBs unterhalten, die der internen Fehlerbeseitigung dienen. Nur so wäre der deutliche Unterschied zwischen den zwei Computer-related-Gruppen zu erklären, denn bei einer eigenen, allgemein gehaltenen SDB gäbe es für die Unterschiede in der Antworten keinen plausiblen Grund.

Schließlich wurde zum Thema der Nutzung von SDBs noch gefragt, ob die Teilnehmer mehr als eine Quelle einer der Kategorien SDB, Mailing-Liste oder Newsgroup nutzen. Es ergab sich, daß zum Zeitpunkt der Umfrage insgesamt 72% aller Teilnehmer mehr als eine Quelle in einem der oben angesprochenen Bereiche nutzten. Insgesamt lagen alle Gruppen ungefähr bei 70%. Nur die Computer-related-Nutzer aus dem Bereich Software stachen hier mit 86% heraus.

Frage 2.2 befaßte sich mit dem Thema, ob die Teilnehmer eine SDB aus beruflichen Gründen nutzen. Von den Teilnehmern beantworteten diese Frage 57% mit Ja, 38% mit Nein. Interessant, wenn auch nicht unbedingt überraschend, war wiederum die Aufteilung zwischen den Gruppen. Die Gruppe der Forscher und der „Anderen“ lag hier jeweils bei ca. 48%. Dagegen lag der Anteil der Teilnehmer des Bereichs Computer-related bei 75%, wobei jedoch hier der Bereich Software nur mit 57% beteiligt war, während der Bereich IS/MIS/DP/Internet mit 85% beteiligt war.

Die nächste Frage befaßte sich mit der aktiven Teilnahme der Teilnehmer an wenigstens einer SDB. Nur 27% aller Teilnehmer beantworteten diese Frage mit Ja, 66% dagegen mit Nein. Allerdings teilte sich hier wieder das Bild zwischen den Gruppen: Aus dem Bereich Computer-related beteiligten sich 45% an den Diskussionen und Beiträgen mindestens einer SDB, während es im Bereich Forschung nur 19% waren und bei den anderen Teilnehmern sogar nur 14%.

Auf die Frage, ob sich die Teilnehmer an mehr als einer SDB beteiligen, antworteten nur noch 18% mit Ja, 65% dagegen mit Nein. Die Veränderung ist insbesondere auf einen, gegenüber Frage 2.3 um zehn Prozentpunkte höheren Anteil von Personen zurückzuführen, die keine Antwort gaben. Ein Grund für diesen Anstieg ist nicht ersichtlich.

Die nächste Frage befaßte sich mit der bevorzugten Regelung für lesenden Zugriff auf die SDB. Es konnte zwischen den vier, in Abschnitt 3.1.2 erläuterten Alternativen gewählt werden. Es ergab sich ein Mehrheit von 58% für einen anonymen Lesezugriff. Für einen individualisierten Zugriff sprachen sich 13% aus, 18% für personalisierten Zugriff, bei dem nur die SDB-Administration den „echten“ Namen des Nutzers kennt, und 8% für einen personalisierten Zugriff, bei dem jeder den wirklichen Namen kennt. Interessant ist hier die Verteilung in den einzelnen Gruppen. Während bei den Gruppen Computer-Related und Education / Research 60% bzw. 65% für einen anonymen lesenden Zugriff stimmten, waren dies bei den restlichen Teilnehmern nur ungefähr 35%. Dafür war dort der Anteil der Teilnehmer, die für einen individualisierten Zugriff stimmten, mit 21% mit Abstand am höchsten.

Die analoge Frage bezüglich des Schreibzugriffs auf die SDB ergab ein anderes Bild. Nur 13% aller Teilnehmer sprachen sich für einen anonymen Schreibzugriff aus. Dagegen sprachen sich 27% für einen individualisierten Zugriff aus. Für die beiden personalisierten Zugriffsvariaten sprachen sich 29% bzw. 28% aus. Es muß festgestellt werden, daß insgesamt alle Optionen ähnlich präferiert wurden. Allerdings ist auffällig, daß die Gruppe Education / Research den personalisierten Schreibzugriff, bei dem die „SDB Administration“ den tatsächlichen Namen des Nutzers kennt, bevorzugte, während die restlichen Teilnehmer den personalisierten Zugriff, bei dem jeder die Identität der Teilnehmer kennt, bevorzugten. Insgesamt sind die Unterschiede aber nicht gravierend.

Ergebnis Abschnitt 3: Veröffentlichungspolitik Abschnitt 3 des Fragebogens enthielt Fragen bezüglich der Veröffentlichungspolitik von Informationen zu Schwachstellen.

Die erste Frage des dritten Abschnitts betraf die bevorzugte Veröffentlichungspolitik. Zur Auswahl standen die drei, in Abschnitt 3.1.3 erläuterten Veröffentlichungspolitiken. 62% aller Teilnehmer der Umfrage sprachen sich für die mehrstufige Veröffentlichungspolitik aus. Für eine zweistufige Veröffentlichungspolitik sprachen sich 13% und für eine unmittelbare Veröffentlichung 23% aus. Zwischen den einzelnen Gruppen gab es keine größeren Unterschiede.

In Bezug auf die ein- oder mehrstufige Veröffentlichungspolitik wurde dann nach einer angemessenen „Grace Period“ gefragt. Bedauerlicherweise haben auch Teilnehmer, die bei Frage 3.1 eine unmittelbare Veröffentlichung der Informationen einer Schwachstelle gefordert haben, bei dieser Frage einen Wert ausgewählt, obwohl dies eigentlich keinen Sinn ergab. Nichtsdestotrotz ist zu erkennen, daß sich eine Mehrheit für eine „Grace Period“ von ein bis zwei Wochen aussprach. 40% stimmten für eine Periode von einer Woche, 17% für zwei Wochen. Für eine Dauer von eins, drei oder fünf Tagen sprachen sich insgesamt 17% der Teilnehmer aus. 15% enthielten sich der Stimme. Für eine Untersuchung der einzelnen Gruppen reichte bei der hohen Anzahl von vierzehn auswählbarer Möglichkeiten die vorhandene Teilnehmerzahl leider nicht aus.

Die nächste Frage bezog sich auf eine variable „Grace Period“ in Abhängigkeit des Risikos, das von der Schwachstelle ausgeht. 67% der Teilnehmer sprachen sich dabei für eine „Grace Period“ aus, die vom Risiko, das von der Schwachstelle ausgeht, abhängt, 23% dagegen. Ein besonders hoher Anteil von Befürwortern für dieses Vorgehen gab es bei der Gruppe der Forscher mit 77%, während die computerbezogene Gruppe dies nur zu 55% befürwortete.

Frage 3.2 betraf das Thema, ob die Teilnehmer Zugang zum größtmöglichen Informationsumfang bezüglich einer Schwachstelle haben wollen, inklusive detaillierter Exploit-Informationen und eventuell sogar Exploit-Skripten. 88% der Teilnehmer sprachen sich für eine möglichst umfassende Information über Schwachstellen aus. Den höchsten Anteil gab es in der Gruppe Computer-related mit 95%, während der Bereich Forschung sich mit 88% dafür aussprach und die übrigen Teilnehmer „nur“ mit 79%.

Ergebnis Abschnitt 4: Qualitätssicherung Abschnitt 4 des Fragebogens enthielt Fragen bezüglich der Möglichkeiten, die Qualität der Daten in der SDB möglichst hoch zu halten. Die Fragen beziehen sich dementsprechend auf Abschnitt 3.1.4.

Die erste Frage zielte darauf ab, ob die Teilnehmer bereit wären, eine SDB mit einem steuernden Moderator zu nutzen. 78% der Teilnehmer antworteten darauf mit Ja. Dabei lag die Zustimmung bei der Gruppe der Computer-related-Nutzer und bei der Gruppe der Forscher bei 85%. Die übrigen Nutzer sprachen sich allerdings nur zu 57% für ein solches Modell aus.

Die darauffolgende Frage befaßte sich mit der Bereitschaft zur Nutzung einer SDB mit einem freigebenden Moderator. Für die Nutzung einer SDB, bei der Beiträge nur nach einer Überprüfung durch den Moderator freigegeben werden, erklärten sich immerhin noch 67% der Teilnehmer bereit. Aus der Gruppe der Forscher erklärten sich auch hier immer noch 85% bereit, eine solche SDB zu nutzen. Allerdings sank die Zustimmung bei den Computer-related-Nutzern auf 55% und die übrigen Teilnehmer erklärten sich nur noch zu 50% bereit, eine solche SDB zu nutzen.

Die Frage 4.3 betraf die potentielle Befürchtung der Teilnehmer, daß es bei einer moderierten SDB zu Zensur kommen könnte. Es zeigte sich ein zwiespältiges Bild. 50% der Teilnehmer würden sich Sorgen bezüglich einer Zensur machen, wenn es einen Moderator gäbe, der Beiträge zurückhalten könnte. 42% teilten solche Befürchtungen nicht. Unterschiede gab es insbesondere zwischen den Forschern und den restlichen Teilnehmern. Bei den Forschern befürchteten nur 42% eine Zensur, während bei allen anderen Teilnehmer 56% eine Zensur fürchteten.

Die nächste Frage befaßte sich mit dem Thema wie „Vertrauen“ in die Schwachstellenmeldungen der SDB geschaffen werden könnte. Dabei standen die drei, bereits vorgestellten Bewertungsmodelle zur Auswahl. Insgesamt ergab sich hier keine allgemeine Präferenz. Ein relativ eindeutiges Bild ergab sich bei den Computer-related-Nutzern, die sich zu 55% für eine Lösung mittels eines *Web of trust* aussprachen. Bei der Gruppe Education / Research ergab sich schon kein eindeutiges Bild mehr: 42% sprachen sich für eine gewählte Gruppe von Freiwilligen aus, die eine Bewertung von Beiträgen durchführt. 35% sprachen sich für ein Web of Trust aus und 23% für eine Bewertung durch die Betreiber der SDB. Bei den anderen Teilnehmern ergab sich ein noch unklareres Bild: 37% sprachen sich für eine gewähltes Bewertungteam aus, ansonsten wurden die anderen Optionen identisch bewertet, insbesondere fällt auch auf, daß 21% der „anderen“ Teilnehmer keine Aussage machten.

Frage 4.5 betraf den Punkt, welche Beiträge bewertet werden sollten, nur die Originalnachrichten oder auch einige der unmittelbaren Kommentare dazu. Es ergab sich ein eindeutiges Bild. 81% aller Teilnehmer sprachen sich dafür aus, sowohl die originale Meldung als auch eine näher zu bestimmende Anzahl von Kommentaren zu diesem Beitrag zu bewerten. Besondere Auffälligkeiten zwischen den Gruppen waren nicht zu beobachten.

Die darauffolgende Frage befaßte sich damit, ob die Teilnehmer der Umfrage der Auffassung wären, daß es sinnvoll sei, für die Bewertung von Beiträgen regelmäßig qualifizierte Freiwillige zu wählen? Auch auf diese Frage ergab sich ein relativ eindeutiges Bild. 77% sprachen sich dafür aus, die Mitglieder des Bewertungsteams regelmäßig zu wählen. Auffällig ist, daß die Gruppen Computer-related und Education / Reseach sich zu jeweils 80% für die regelmäßige Wahl aussprachen, während sich von den restlichen Teilnehmern nur 64% dafür aussprachen.

Auf die letzte Frage des vierten Abschnitt sollten die Teilnehmer antworten, ob sie bereit wären, sich freiwillig für die Bewertungtätigkeit zur Verfügung zu stellen. Immerhin 28% der Teilnehmer erklärten sich bereit, Zeit für diese wichtige Arbeit zur Verfügung zu stellen. 60% erklärten, daß sie dies nicht tun könnten bzw. wollten. Auffällig ist, daß sich die Teilnehmer aus der Gruppe der Forscher mit über 38% bereit erklärten, diese Tätigkeit auszuüben, während es aus den übrigen beiden Gruppen nur jeweils ca. 20% waren. Dies könnte mit der unterschiedlichen zeitlichen Flexibilität der einzelnen Gruppen zusammenhängen. Es muß allerdings auch angemerkt werden, daß einzelne Teilnehmer die Nichtbereitschaft zu dieser Tätigkeit damit begründeten, daß sie sich nicht als qualifiziert genug einschätzten.

Ergebnis Abschnitt 5: Finanzierung Abschnitt 5 des Fragebogens enthielt Fragen dazu, wie die Finanzierung der SDB sichergestellt werden könnte und zwar insbesondere so, daß die Nutzer keine Abhängigkeiten zwischen der SDB-Administration und den Geldgebern befürchten.

Die erste Frage befaßte sich mit der Bereitschaft der Teilnehmer als Individuen eine monatliche Nutzungsgebühr an die SDB zu zahlen. Es ergab sich das relativ eindeutige Bild, daß private Nutzer kaum bereit waren, für die Nutzung einer SDB zu bezahlen (75%). Insbesondere der Anteil von Nutzern der Gruppe Computer-related war mit nur 5% für die Zahlung einer Nutzungsgebühr durch Individuen sehr niedrig. Dieser Anteil war bei den Forschern mit 30% deutlich höher und bei den anderen Teilnehmern immerhin noch 21%.

Die konsequenterweise nachfolgende Frage befaßte sich mit der Höhe des Betrags, den die Teilnehmer bereit wären zu zahlen. Entsprechend des Ergebnisses der letzten Frage ergab sich das deutliche Bild, daß die meisten Teilnehmer der Umfrage nicht bereit waren, für die Nutzung der SDB zu zahlen. Die dreizehn Teilnehmer, die bereit waren, für die Nutzung einen Kostenbeitrag zu übernehmen, waren bereit, zwischen einem und 10 Euro / US-$ pro Monat zu zahlen.

Frage 5.3 bezog sich auf die eventuelle Bereitschaft eines Arbeitgebers der Teilnehmer, für die Nutzung der SDB eine monatlichen Preis zu zahlen. 15% der Teilnehmer äußerten, daß die Frage auf sie nicht anwendbar wäre. 60% bejahten die Frage und nur 12% gaben eine negative Antwort. Es fällt auf, daß aus der Gruppe der Forscher nur 46% auf diese Frage mit Ja antworteten, während es bei den restlichen zwei Gruppen jeweils 70% waren. Dieser unterschiedliche Anteil ist vermutlich auf die schwierige Finanzsituation von Hochschulen und Forschungseinrichtungen zurückzuführen.

Die nächste Frage betraf die Akzeptanz einer finanziellen Unterstützung der SDB durch einige große Unternehmen. Von den Teilnehmern erklärten 63%, daß sie oder ihr Arbeitgeber (vermutlich) keine Probleme mit einigen großen Partnern aus dem kommerziellen Umfeld hätten. Allerdings äußerten immerhin 27% Bedenken. Insbesondere die Teilnehmer der Gruppe Computer-related äußerten sich mit einem relativ hohen Anteil von 70% überraschend positiv zu einer Unterstützung durch große Unternehmen.

Als nächstes wurde nach der Akzeptanz einer Finanzierung der SDB durch eine oder mehrere Regierungen bzw. Regierungseinrichtungen, beispielsweise durch Subventionen, gefragt. Von Teilnehmer äußerten 70%, daß es keine Bedenken bezüglich einer Finanzierung durch Regierungen gäbe. Fast schon erstaunlich ist die höhere Zustimmung zu einer finanziellen Unterstützung durch Regierungen gegenüber der Unterstützung durch Unternehmen. Auch hier fielen die Nutzer der Gruppe Computer-related mit einer großen Zustimmung von 80% auf. Offensichlich fürchteten die Teilnehmer eine Beeinflußung der SDB-Betreiber durch Unternehmen eher als eine entsprechende Beeinflußung durch staatliche Stellen. Ein erstaunliches Ergebnis.

Die darauffolgende Frage befaßte sich mit der Akzeptanz eines „Pay per Query“-Systems durch die Nutzer, beispielsweise durch ein Micro-Payment-System. Für ein Pay per Query-System ergab sich eine breite Ablehnung: 67% der Teilnehmer lehnten ein solches System ab, wobei es unter den verschiedenen Gruppen keine auffälligen Abweichungen gab.

Die nächste Frage betraf den Bereich „Bonussystem“. Es wurde gefragt, ob ein Bonussystem, daß für gute Beiträge die Nutzungskosten des Teilnehmers senkt, für die Melder solcher Beiträge attraktiv wäre? 57% der Teilnehmer äußerten sich dahingehend, daß ein Bonussystem die Attraktivität der SDB steigern könnte. 33% äußerten sich dazu negativ. Dabei gab es zwischen den verschiedenen Gruppen nur geringe Abweichungen.

Der letzte Punkt des fünften Abschnitts befaßte sich mit der Frage, ob die Teilnehmer sich vorstellen könnten, daß die simple Nutzung der SDB kostenlos wäre und sie die SDB

durch in Anspruchnahme von Value Added Services (VAS) finanzieren würden? 68% der Teilnehmer beantworteten die Frage mit Ja. Den geringsten Zuspruch fand die Finanzierung durch Value Added Services bei der Gruppe der Computer-related-Nutzer. Aus dieser Gruppe sprachen sich nur 60% für diesen Finanzierungsweg aus. Bei den Forschern waren es immerhin 69% und bei den übrigen Teilnehmer sogar 79%.

Ergebnis Abschnitt 6: „Abschließende Fragen“ Abschnitt 6 des Fragebogens enthielt einige ergänzende Fragen an die Teilnehmer der Umfrage.

Die erste Frage des letzten Abschnitts der Umfrage befaßte sich mit dem Wunsch innerhalb der SDB auch Off-Topic-Diskussionen zu führen. Eine große Mehrheit von 77% sprach sich für die Möglichkeit von Off-Topic-Diskussionen aus. Zwischen den Teilnehmergruppen gab es keine deutlichen Unterschiede.

Die darauffolgende Frage betraf die Bereitschaft der Teilnehmer eine neue SDB, die die von ihnen gewünschten Eigenschaften hätte, auch tatsächlich zu nutzen. Immerhin 95% der Teilnehmer erklärten, daß sie sich vorstellen könnten, *die* SDB, für die sie sich ausgesprochen haben, zu nutzen. Dies zeigt zumindest, daß, wenn ein geeigneter Kompromiß gefunden wird, der möglichst viele potentielle Nutzer anspricht und außerdem eine hohe Qualität der Information sichergestellt werden kann, tatsächlich eine Möglichkeit besteht, auf dem breiten Feld der existierenden SDBs noch Nutzer zu gewinnen.

Die dritte Frage bezog sich auf das Interesse der Teilnehmer an weiteren, spezialisierteren Umfragen zu diesem Thema teilzunehmen. Erfreuliche 63% der Teilnehmer erklärten sich bereit, an weiteren Umfragen teilzunehmen. Erstaunlicherweise erklärten sich 75% der Computer-related-Nutzer bereit, an weiteren Umfragen teilzunehmen, während die Nutzer aus dem Bereich Education / Research dies nur zu 62% erklärten und die übrigen Teilnehmer nur zu 50%.

Die letzte Frage des Fragebogens betraf die Bereitsschaft an der Entwicklung des SDB-Projekts teilzunehmen. Immerhin 37% der Teilnehmer der Umfrage erklärten sich dazu bereit, wobei die Bereitschaft bei den Computer-Related- und den Education / Research-Nutzern mit ca. 41% am höchsten war. Bei den übrigen Teilnehmern erklärten sich nur 21% zur Teilnahme bereit.

3.1.11 Zwei geeignete Geschäftsmodelle

Aufgrund der Vorüberlegungen und der Umfrage können zwei denkbare Geschäftsmodelle empfohlen werden, abhängig von dem Aufwand an Personal und Finanzen, den der SDB-Betreiber betreiben will oder kann.

Das Geschäftsmodell mit geringem Aufwand sollte auf dem Open Data-Modell basieren. Dabei sollte versucht werden, einen möglichst starken Gemeinschaftseffekt zu erreichen, so daß die notwendigen Entwicklungsarbeiten auf eine breite Gruppe verteilt werden. Dabei muß darauf gehofft werden, daß es mittelfristig nicht zu einer starken Aufspaltung des Projekts kommt. Bezüglich des Zugriffs sollte zumindest für die Originalinstanz versucht werden, einen individualisierten oder personalisierten schreibenden Zugriff zu implementieren.

Der lesende Zugriff kann anonym erfolgen. Dabei ist es selbstverständlich, daß Personendaten an Dritte nicht zusammen mit den Schwachstellendaten übermittelt werden können. Informationen zu Schwachstellen sollten unmittelbar veröffentlicht werden. Bezüglich Moderation und Bewertung sollte versucht werden, Freiwillige mit einer ausgewiesenen Expertiese im Bereich IT-Sicherheit zu gewinnen. Dabei sollte, wegen des geringeren Aufwands, auf einen steuernden Moderator gesetzt werden. Das Bewertungsteam sollte in einer Anlaufphase bestimmt und später im günstigsten Fall von den Nutzern gewählt werden. Für die trotz allem notwendige Finanzierung sollte versucht werden, Spender und Sponsoren zu gewinnen. Auch Werbung könnte dafür herangezogen werden. Dabei muß vor allem für die Anschaffung der notwendigen Infrastruktur gesorgt werden, sofern dafür anfänglich nicht bereits vorhandene Systeme genutzt werden können.

Für ein eher kommerzielles Umfeld sollte auf das föderierte Organisationsmodell zurückgegriffen werden. Anfänglich werden dabei vermutlich nur eine zentrale öffentliche Instanz und verschieden private Instanzen vorhanden sein. Die zentrale SDB kann später durch Spezialisierungen aufgeteilt werden. Bei lesendem Zugriff sollte Anonymität gewahrt bleiben können, schreibender Zugriff sollte wahlweise individualisiert oder personalisiert möglich sein. Auch in diesem Ansatz sollte die unmittelbare Veröffentlichungspolitik verfolgt werden. Bei der Moderation kann, je nach Aufwand, der betrieben werden soll, zwischen der steuernden und der freigebenden Variante gewählt werden. Bei der Bewertung sollte versucht werden, ein geeignetes Team von Experten zu gewinnen, das in der Lage ist, die Bewertung durchzuführen. Auf ein Bonussystem sollte verzichtet werden. Zur Finanzierung sollte versucht werden, Sponsoren und staatliche Mittel zu gewinnen. Außerdem könnten zusätzlich VAS angeboten werden. Auch Einnahmen durch Werbung sind denkbar. Bezüglich der rechtlichen Organisation sollte zumindest anfänglich versucht werden, den Ansatz des Vereins mit Födermitgliedern umzusetzen.

3.2 „Noch eine weitere SDB?“

Sowohl während der Erstellung dieser Arbeit, als auch im Rahmen der Umfrage stellte sich die Frage, wie wahrscheinlich es wäre, daß „nur noch eine weitere“ Schwachstellendatenbank aufgebaut würde. Diese Frage ist sicherlich selbst dann zulässig, wenn davon ausgegangen wird, daß für Forschungszwecke eine lokale SDB benötigt wird, um einen schnellen, unkomplizierten und effizienten Zugriff auf die Daten zu erhalten. Ansonsten könnte auch mit, gegenüber der Neuerfassung der Schwachstelleninformationen, verhältnismäßig wenig Aufwand versucht werden, eine der bestehenden und frei zugänglichen SDBs zu kopieren und dann lokal zu nutzen.

Die Verbesserungen der SEC://HOUSE-SDB gegenüber bestehenden SDBs werden dabei in die Richtung einer stärkeren Strukturierung der Daten gehen müssen, wie sie zwar die SDB von Krsul bietet, jedoch nicht die öffentlich zugänglichen SDBs. Erst eine solche SDB erlaubt aufwendige Recherchen und eine eindeutige und widerspruchsfreie Darstellung der zu erfassenden Daten bzw. Informationen. Weiterhin würde eine starke Strukturierung der Daten es erlauben, Werkzeuge für Sicherheitsanalysen direkt über die SDB mit den für die Analyse notwendigen Informationen zu speisen. Dies ist wiederum eine Voraussetzung, um Data Mining effizient betreiben zu können.

Zusätzlich wird es, je nach tatsächlich gewähltem Geschäftsmodell, jedem Nutzer möglich sein, die SDB zu kopieren und auf diese Art und Weise lokal zu nutzen, wodurch er die Vorteile des lokalen Vorliegens einer SDB genießt. Auch ist vorgesehen, in regelmäßigen Abständen inkrementelle Snapshots der SDB anzubieten, die die Nutzer dann zu ihrer lokalen Kopie der SEC://HOUSE-SDB hinzufügen können und so in der Lage sind, mit relativ geringem Aufwand stets eine aktuelle Version der SDB zu nutzen.

Weitere Vorteile der SEC://HOUSE-SDB liegen in der Transparenz des Datenschemas. Das Schema ist öffentlich und frei zugänglich, so daß auch für Dritte ein gezielter Zugriff auf die strukturierten Daten möglich ist. Betrachtet man beispielsweise die BUGTRAQ-SDB, so ist nicht ersichtlich, ob die Informationen, die dort präsentiert werden, in dieser nicht stark strukturierten Form in der Datenbank gespeichert sind, oder ob dieses Format ausschließlich für die Ausgabe genutzt wird. Es ist zum Beispiel nicht klar, ob bei der Auflistung betroffener Systeme diese Systeme tatsächlich in textueller Form in der Datenbank gespeichert sind oder ob hier eine strukturierte Datenbank zugrunde liegt und nur die Ausgabe in dieser weniger strukturierten Form wiedergegeben wird. Prinzipiell wäre allerdings auch die Kenntnis des Datenbankschemas von BUGTRAQ nicht hilfreich, da es keine Möglichkeit gibt, direkt auf die Datenbank zuzugreifen.

Ebenfalls vorteilhaft ist, daß, bei entsprechend gewähltem Geschäftsmodell, auch die Informationen selbst transparent vorliegen, d.h. jeder Nutzer sieht alle Daten und es werden nicht noch zusätzliche Informationen gespeichert, die nur einem ausgewählten Personenkreis zugänglich sind.

Zusätzlich sind Zugriffsmöglichkeiten über Standardschnittstellen, wie beispielsweise über Treiber für die Java Database Connectivity oder Open Database Connectivity (JDBC/ODBC) vorgesehen.

Darüber hinaus sollte ein geeignetes Geschäftsmodell dazu führen, daß die SEC://HOUSE-SDB vor der Bedeutungslosigkeit bewahrt werden kann, da ein entsprechend ausgelegtes Geschäftsmodell das Interesse an einer Nutzung und Mitwirkung an der SDB deutlich steigern sollte.

3.3 Bewertung bekannter SDBs

In diesem Abschnitt soll eine kurze Bewertung von zwei etablierten SDBs vorgenommen werden. Dabei wurde Krsuls Datenbank als stark strukturierte SDB und die X-Force Datenbank als strukturierte, aber ausschließlich textorientierte SDB gewählt.

3.3.1 Krsuls SDB

Nach der Analyse der SDB von Krsul, die in Abschnitt 2.1.1 behandelt wurde, drängte sich die Überlegungen auf, ob es nicht möglich wäre, einfach die Struktur der SDB zu kopieren. Einer der Hauptnachteile von Krsuls SDB ist jedoch, daß keine Datenbank verwandt wurde, um die SDB zu implementieren [Krs98, S. 50]. Somit wäre zumindest eine „Konvertierung“ von Krsuls SDB in ein relationales Datenbankschema anzustreben, um die

Vorteile von Relationalen Datenbank Managementsystemen (RDBMS), insbesondere bei der Zugriffsgeschwindigkeit, den Recherchemöglichkeiten, der Transaktionsunterstützung, die bei gleichzeitigen Zugriffen auf die SDB wünschenswert ist, und nicht zuletzt der Konsistenzsicherung, nutzen zu können. Da Krsuls Schema nicht die erste Normalform für relationale Schemata[8] erfüllt, muß diese Konvertierung die Umsetzung der zahlreichen, nicht atomaren Felder in „vernünftig“ recherchierbare, atomare Felder unter Verwendung zusätzlicher Klassen bzw. Entitäten beinhalten.

Darüber hinaus zeigt das Schema Krsuls Schwächen bezüglich der Erfassung betroffener Betriebssysteme und Software. Es existiert keine zentrale Erfassung von Betriebssystemen und Software und unterschiedlicher Softwareversionen. Betroffene Systeme werden grundsätzlich von neuem erfaßt und nicht als Verweise auf ein „Softwareversions-Repository“ realisiert, was wiederum Recherchen erheblich erschwert, da diese bezüglich eines Herstellers, einer Software oder auch einer Version einer Software nicht effizient realisiert werden können.

Auch sieht Krsul keine Möglichkeit vor, Vorfälle zu erfassen. Da ein Teil der Forschung im Rahmen von SEC://HOUSE die Analyse von Systemen, die von Vorfällen betroffen waren, beinhalten soll, ist jedoch eine Erfassung von Vorfällen wünschenswert und notwendig. Die aus der Analyse gewonnenen Erkenntnisse sollen dann direkt in die Untersuchung von Systemen auf das Vorhandensein von Schwachstellen einfließen.

Trotz der Kritik existieren Verwandtschaften mit Krsuls SDB. So wurden die Kriterien, die der Einordnung von Schwachstellen dienen und die von Krsul im Rahmen seiner Dissertation entwickelt wurden, übernommen. Dies liegt darin begründet, daß Krsuls Klassifizierungen geeignet erscheinen, strukturierte Informationen über Schwachstellen zu erfassen, anstatt ausschließlich textbasierte Informationen zu Schwachstellen zu speichern. Daher wurde versucht, die Vorteile von Krsuls SDB zu übernehmen und Nachteile zu beseitigen.

3.3.2 X-Force SDB

Wie bereits Abschnitt 2.1.6 aufzeigte, ist ein erster Nachteil der X-Force-SDB, daß es zwei Kategorien von Einträgen gibt, die einen deutlich unterschiedlichen Informationsgehalt aufweisen. Dadurch ist es häufig zwingend notwendig, weitere Quellen zu überprüfen um ausreichend Informationen über eine Schwachstelle zu erhalten. Weiterhin sind die Informationen der ISS Advisories vollkommen textbasiert, was Recherchen und Analysen erheblich erschwert. Dies zeigt sich auch zusätzlich darin, daß Recherchen innerhalb der SDB nur über zwei Varianten möglich sind: einerseits ist es möglich, nach Schwachstellen für ein bestimmtes Betriebssystem zu suchen, wobei hier keine Versionsinformationen enthalten sind, und andererseits kann man textbasiert nach Schlüsselwörtern suchen. Damit sind die Recherchemöglichkeiten jedoch bereits erschöpft.

Durch die mangelnde Strukturierung, zumindest ist eine solche von außen nicht erkenn- oder nutzbar, ist es kaum möglich detaillierte Analysen durchzuführen, die über Text Mining hinausgehen. Da zusätzlich viele Einträge nur ein absolutes Minimum an Informationen enthalten und dann zusätzlich auf externe Quellen verweisen, ist es nahezu unmöglich über eine einheitliche Informationsbasis Analysen durchzuführen.

[8] Vgl. dazu [HS95, S. 94 ff., S.195 ff.]

3.3.3 Vergleich der SDBs

Tabelle 3.4 zeigt einen Vergleich der Krsul-, der X-Force- und der SEC://HOUSE-SDB anhand der in diesem Abschnitt angesprochenen Kriterien. Unter *Strukturierte Speicherung von Schwachstellen* ist dabei zu verstehen, daß das Datenschema der SDB komplexe Recherchen, beispielsweise nach Kombinationen verschiedener Daten von Schwachstellen, ermöglicht. Das Kriterium *Homogene Schwachstelleneinträge* bezieht sich darauf, daß alle Einträge einen vergleichbaren Informationsgehalt haben.

Kriterium	**Krsul**	**X-Force**	**SEC://HOUSE**
Nutzung eines DBMS	nein	vermutlich ja	ja
Erfassung von Vorfällen	nein	nein	ja
Strukturierte Speicherung von Schwachstellen	teilweise	nein	ja
Strukturierte Einordnung von Schwachstellen	ja	nein	ja
Homogene Schwachstelleneinträge	ja	nein	ja

Tabelle 3.4: Vergleich einiger Eigenschaften von SDBs

3.4 Notwendige Zugangswege zur SDB

Durch Ergebnisse der Umfrage unterstützt, sollten möglichst viele verschiedene Zugangswege zur SDB angeboten werden:

- Administrator-Schnittstelle
- Anwender-Schnittstelle
- WWW-Schnittstelle
- Systemnaher Zugang zur Datenbank
- e-Mail-Listen & Newsgroups
- Lokaler direkter Zugang zur Datenbank

3.4.1 Administrator-Schnittstelle

Das Administrator-Tool dient, wie der Name vermuten läßt, als Werkzeug für einen Administrator der SDB. Dementsprechend erlaubt es den Zugriff auf alle Informationen der SDB.

Folgende Funktionen sollten vom Admin-Tool unterstützt werden:

- Login / Authentifizierung

- Anlegen / Lesen / Editieren / Löschen von Vorfällen
- Anlegen / Lesen / Editieren / Löschen von Vulnerability-Meldungen (inklusive Gegenmaßnahmen, Exploits)
- Bearbeiten von „beliebigen“ Daten in der Datenbank (z.B. Hilfsdaten für den SDB-Betrieb)
- Anlegen / Lesen / Editieren / Sperren von Nutzeraccounts / -bewertungen
- Anlegen / Lesen / Editieren / Löschen von Diskussionsbeiträgen
- Freigabe von Warnungen
- Recherchefunktionen

Dabei sollte auf Sicherheit Wert gelegt werden, so sollte beispielsweise Kommunikation zwischen Administrator und den SDB-Systemen nur verschlüsselt, beispielsweise über SSL, durchgeführt werden.

3.4.2 Anwender-Schnittstelle

Das Anwender-Werkzeug könnte als im Funktionsumfang verringerte Version des Administrator-Tools gestaltet werden, das nur die dem „normalen“ Anwender erlaubten Funktionen ermöglicht:

- Login / Authentifizierung
- Anlegen / Lesen von Vorfällen
- Anlegen / Lesen von Vulnerability-Meldungen (inklusive Gegenmaßnahmen, Exploits)
- Anlegen / Lesen von Diskussionsbeiträgen
- Recherchefunktionen

Auch das Anwender-Tool sollte nur verschlüsselte Kommunikation verwenden.

3.4.3 WWW-Schnittstelle

Die WWW-Schnittstelle sollte im Idealfall die gleiche Funktionalität bieten wie das Anwender-Tool. Dabei wäre die WWW-Schnittstelle dem Anwender-Tool vorzuziehen, da das WWW im Zweifelsfall die am häufigsten vorhandene Zugangsmöglichkeit ist, jedoch stellt die Implementierung der WWW-Schnittstelle zusätzlichen Aufwand gegenüber dem Anwender-Tool dar.

- Login / Authentifizierung

- Anlegen von Accounts
- Anlegen / Lesen von Vorfällen
- Anlegen / Lesen von Vulnerability-Meldungen
- Anlegen / Lesen von Diskussionsbeiträgen
- Recherche

Auch die WWW-Schnittstelle sollte nur verschlüsselte Kommunikation über SSL einsetzen.

3.4.4 Systemnaher Zugang zur Datenbank

Da u.U. auch Dritte komplexe Anfragen an die Datenbank stellen möchten und andererseits dafür nicht unbedingt einen direkter Datenbank-Zugang, beispielsweise über eine ODBC- oder JDBC-Schnittstelle zur Verfügung gestellt werden soll, könnte eine wohldokumentierte Datenbankzugriffsschnittstelle implementiert werden, die Dritten einen relativ direkten Zugriff auf die Informationen der SDB erlaubt. In dieser Mittelschicht könnten dann bei Zugriffen bestimmte Plausibilitäts- und Sicherheitskontrollen durchgeführt werden. Auf diese Weise wäre es u.U. sogar denkbar, schreibenden Zugriff zur SDB über diese Schnittstelle zu ermöglichen. Eine weitere Anwendungsmöglichkeit für diese Schnittstelle könnte der SDB-Zugriff durch Tools von Drittherstellern sein, beispielsweise Analysewerkzeuge für Systeme oder auch Intrusion Detection Systeme.

3.4.5 e-Mail-Listen & Newsgroups

Parallel zu den Beiträgen und Diskussionen zu Meldungen in der SDB könnten verschiedene Mailing-Listen eingerichtet werden. Denkbar wäre dabei beispielsweise eine Mailing-Liste, über die die freigegebenen Schwachstellenmeldungen verteilt werden. Weiterhin könnte eine Mailing-Liste eingerichtet werden, die neue, noch unbestätigte Schwachstellenmeldungen verteilt, so daß Abonnenten der Mailing-Liste bei Interesse auf die SDB direkt zugreifen könnten. Weiterhin könnten auch alle Diskussionen in der SDB an eine Mailing-Liste gesandt werden und denkbar wäre u.U. auch ein Gateway von der Mailing-Liste in die SDB, so daß Nutzer auch über die Mailing-Liste an den Diskussionen in der SDB teilnehmen könnten.

Der gleiche Funktionsumfang wie für die Mailing-Listen könnte parallel oder alternativ auch mittels einer oder mehrerer Newsgroups implementiert werden.

3.4.6 Lokaler direkter Zugang zur Datenbank

Für den Betreiber der SDB ist es selbstverständlich sinnvoll, ungehindert und mit maximaler Performance auf die SDB zuzugreifen. Daher kann der Betreiber, der per Definition vertrauenswürdig ist, über die direkten Datenbankschnittstellen auf die SDB zugreifen.

3.5 Initiale Füllung und halbautomatische Erweiterung der SDB

Um möglichst schnell eine, wenn auch nur schwach strukturierte Datenbasis zu erhalten, sollte erwogen werden, Importfilter für mindestens eine der großen, frei zugänglichen Schwachstellendatenbank zu entwickeln, der es zumindest halbautomatisch ermöglichen würde, einen Grunddatenbestand in die SDB zu übertragen. Entsprechende Personalkapazitäten vorausgesetzt könnten dann die übernommenen Daten während der Übernahme oder aber nachträglich in stark strukturierte Daten überführt werden, die dann relativ schnell eingesetzt werden könnten. Dabei sollte aber der Aufwand für die Bearbeitung der Datensätze nicht unterschätzt werden. Die Recherche der genauen Details zu der Schwachstelle kann u.U. sehr aufwendig sein.

Eine, sowohl für die initiale Füllung, als auch für die fortlaufende Aktualisierung der SDB, denkbare Systemarchitektur zeigt Abbildung 3.2. Zusätzlich zeigt die Abbildung, wie die Nutzung der SDB für Forschungszwecke langfristig zu einem Feedback führen soll, das sich wiederum in einer gesteigerten Qualität zukünftiger Systeme niederschlagen soll.

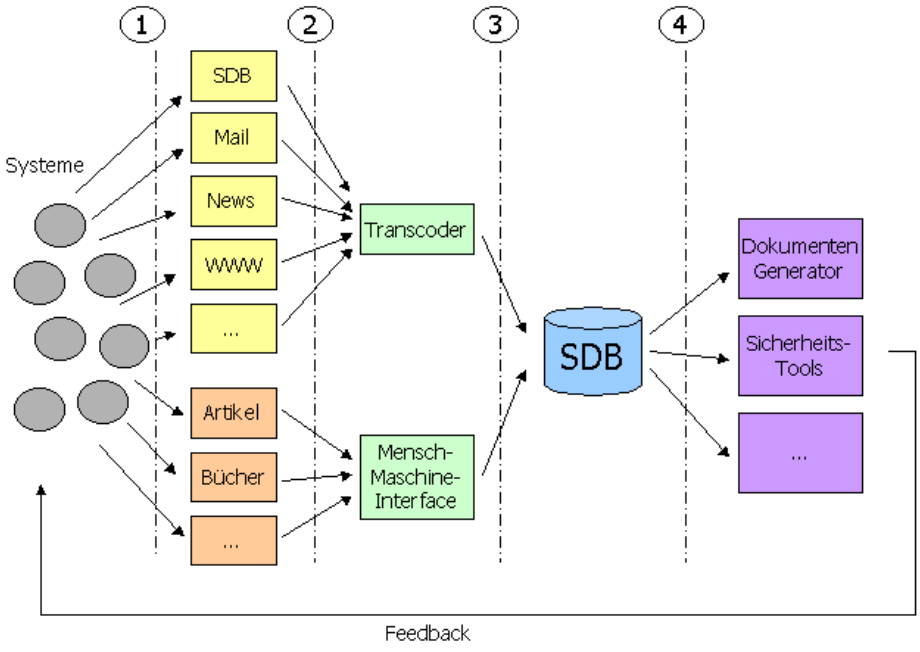

Abbildung 3.2: Übernahme von Schwachstelleninformationen und Feedback

Die Abbildung zeigt vier Schnittstellen zwischen einzelnen Komponenten des Gesamtsystems. **Schnittstelle 1** befindet sich zwischen den in Betrieb befindlichen Systemen und potentiellen Quellen von Schwachstelleninformationen. Bei den Quellen muß man zwischen elektronischen Medien wie online zugänglichen Schwachstellendatenbanken, eMail, Newsgroups, dem World Wide Web sowie anderen denkbaren Quellen und nicht elektronischen

Quellen, wie Büchern oder Fachartikeln unterscheiden. Schnittstelle 1 wird heute i.d.R. durch Menschen erfüllt, die Schwachstellen in Systemen finden und auf irgendeine Art und Weise veröffentlichen. In Zukunft wird es dafür hoffentlich auch elektronische Systeme geben, die Schwachstellen in Software aufdecken können, beispielsweise durch Analyse des Quellcodes von Anwendungen.

Auf die potentiellen Quellen folgt **Schnittstelle 2**. Diese Schnittstelle stellt die Erfassung von unbearbeiteten Schwachstelleninformationen aus den zur Verfügung stehenden Quellen dar. Im Fall der elektronischen Quellen ist dies weniger aufwendig, da man die entsprechenden Informationen auch auf elektronischem Weg mittels entsprechend zu implementierender Dienste sammeln kann. Dabei kann man Mailinglisten und Newsgroups abonnieren und die entsprechenden Beiträge sammeln und man kann beispielsweise wohl bekannte Webangebote nach neuen Meldungen durchsuchen lassen und diese Meldungen sammeln. Im Fall der nicht elektronischen Quellen wiederum ist ein Mensch notwendig, der die Informationen liest und sammelt, um sie dem nächsten Verarbeitungsschritt zuzuführen. Dieser besteht im Fall der elektronischen Medien aus zu entwickelnden *Transcodern*, die versuchen, die Informationen, die aus den elektronischen Quellen gesammelt wurden, in einer bestmöglichen Art und Weise aufzubereiten, um dann eine Weiterbearbeitung im Rahmen der nächsten Schnittstelle zu ermöglichen.[9] Einer dieser Transcoder könnte beispielsweise die Diskussionsfäden innerhalb einer Mailingliste oder Newsgroup analysieren und mit Hilfe der Informationen über Reihenfolge bzw. Anordnung von Beiträgen und Zitaten in Beiträgen versuchen, möglichst ein Dokument zu generieren, das an den entsprechenden Stellen innerhalb des Dokuments die Beiträge zu dieser Stelle aufführt. Auf diesem Gebiet ist jedoch noch viel Foschungsarbeit zu leisten. Auch könnte mittels spezieller *Relevanzfilter* versucht werden, „unwichtige“ Beiträge von vornherein herauszufiltern und so die Datenmenge zu reduzieren und die Informationsdichte zu erhöhen.

Im Fall der nicht elektronischen Quellen muß hier eine Mensch-Maschine-Schnittstelle zur Erfassung, insbesondere der Elektronifizierung, der gesammelten Informationsquellen dienen, d.h. es ist eine manuelle Bearbeitung notwendig.

Die nachfolgende **Schnittstelle 3** dient zur Bearbeitung der gesammelten Informationen. Dies ist notwendigerweise ein manueller Bearbeitungsschritt, in dem die aus den verschiednen Quellen gesammelten und aufbereiteten Informationen in ein für die SDB geeignetes Format gebracht werden. Dies beinhaltet eine Analyse der gegebenen Informationen und eventuell zusätzliche Recherchen bezüglich einer Schwachstelle. Auch muß geprüft werden, ob eine Schwachstelle möglicherweise schon erfaßt ist und nur noch um weitere Informationen ergänzt werden muß. Sind die Informationen zu einer Schwachstelle möglichst vollständig ermittelt, wird die Schwachstelle in der SDB gespeichert.

Schnittstelle 4 stellt alle denkbaren Schnittstellen zur Nutzung der Daten bzw. Informationen aus der SDB dar. Das können direkte Datenbankschnittstellen, Web-Schnittstellen oder auch speziell definierte Abfrageschnittstellen sein. Sie ermöglichen es insbesondere auf elektronischem Weg auf die Daten der SDB zuzugreifen und diese zu Forschungszwecken, beispielsweise mittels Data-Mining, zu analysieren. Aus diesen Abfragen bzw. Analysen ergeben sich dann verschiedene denkbare Ergebnisdarstellungen, wie beispielsweise Dokumente oder Berichte. Gleichzeitig könnte die SDB jedoch auch als Quelle für Sicherheits-Analyse-Tools dienen. Beispielsweise wäre es denkbar, daß man das sogenannte „Security-

[9] Vgl. dazu auch Abschnitt 1.3.2

Auditing-Tool“ Nessus dahingehend erweitert, daß es die Informationen zu Schwachstellen, deren Vorhandensein es überprüfen könnte respektive sollte, direkt aus der SDB abzufragen. Dadurch erreichte man, daß Nessus quasi jederzeit einen minutengenauen Stand der aktuellen Schwachstellen nutzen könnte, ohne daß die Informationen über neue Schwachstellen Nessus manuell zur Verfügung gestellt werden müßten, wie dies heute der Fall ist.

Im Idealfall sollten nun die Resultate der Abfragen und Analysen der Daten in der SDB im Rahmen eines Feedbackprozesses zu dem Ergebnis führen, daß bestehende Systeme sicherer werden und neue Systeme von Beginn an sicher sind, indem beispielsweise sich ständig wiederholende Fehler bei der Entwicklung von Software nicht regelmäßig wiederholt werden.

3.6 Notwendige Informationen zu Schwachstellen

Dieser Abschnitt soll einen kurzen Überblick über die in der SDB zu erfassenden Daten und über die zu erwartenden Anwendungsfälle bei der Nutzung der SDB geben.

3.6.1 Grobübersicht über die zu erfassenden Daten

In der SDB müssen folgende Entitäten erfaßbar sein:

- *Incident* (Vorfall)

  Incident erfaßt Daten zu sicherheitsrelevanten Vorfällen, die bei Nutzern aufgetreten sind

- *Vulnerability* (Schwachstelle)

  Vulnerability erfaßt die Daten zu Schwachstellen.

- *Countermeasure* (Gegenmaßnahme)

  Countermeasure erfaßt Gegenmaßnahmen zu Schwachstellen. Dabei kann es sich sowohl um Workarounds, als auch um Patches oder neue Versionen handeln.

- *Exploit*

  Exploit erfaßt Exploits von Schwachstellen. Dabei kann es sich sowohl um verbale Beschreibungen, als auch um Exploit-Skripten handeln.

- *Vendor* (Hersteller)

  Vendor erfaßt den Hersteller einer Software. Dabei sind beispielsweise Kontaktadressen wichtig. Vendor wird dann für zu erfassende Software verwendet.

- *Software*

  Software erfaßt grundlegende Informationen zu Software, die unabhängig von der Version der Software ist.

- *Software-Version*

  Software-Version erfaßt einzelnen Versionen einer Software.

- *Comment* (Kommentare)

  Comment dient zur Speicherung von Kommentaren und Anmerkungen zu Incident-, Exploit, Countermeasure und Vulnerability-Meldungen.

- *User*

  User dient zur Speicherung von Benutzerinformationen und Paßwörtern.

- *(Snapshot)*

  Snapshot ist eine Erweiterungmöglichkeit für die SDB, die später für die Speicherung von sogenannten System-Snapshots genutzt werden soll. Die Idee hinter den System-Snapshots ist eine automatische Analyse eines IT-Systems mittels eines speziellen Werkzeugs. Dabei sollen möglichst umfassende Informationen über die in dem System eingesetzte Software und die dazugehörigen Software-Versionen gesammelt werden. Dieser Snapshot soll eine schnelle Analyse von Systemen bezüglich der Quelle eines Vorfalls ermöglichen. Außerdem könnte ein solches Snapshot-Werkzeug später für die Schwachstellenanalyse von vermeintlich sicheren IT-Systemen eingesetzt werden.

Zu den Details der einzelnen Entitäten sei auf das folgende Kapitel verwiesen, in dem u.a. das Schema der SDB präsentiert wird.

3.6.2 Anwendungsfälle bei der Arbeit mit der SDB

Bei der Arbeit eines Nutzers oder Administrators mit der SDB sind verschieden Tätigkeitsabfolgen zu erwarten. Der Administrator kann dabei die gleichen Möglichkeiten nutzen wie der „normale“ Nutzer und kann zusätzlich darüber hinausgehende Funktionalitäten nutzen.

- Anwender

  1. Meldung eines Vorfalls (und eventuell eines System-Snapshots)
     Erfolgt ein Einbruch in ein IT-System, werden bei diesem Anwendungsfall die relevanten Daten bezüglich des Vorfalls gemeldet. Es kann sein, daß hierfür zuerst die Informationen zur betroffenen Software erfaßt werden müssen, siehe dazu Punkt 6.
  2. Meldung einer Schwachstelle zu einem oder mehreren Vorfällen
     Werden Informationen bezüglich der Schwachstelle bekannt, auf die ein Vorfall zurückzuführen ist, so werden zu diesen Vorfällen die entsprechenden Schwachstelleninformationen gespeichert. Es kann sein, daß hierfür zuerst die Informationen zur betroffenen Software erfaßt werden müssen, siehe dazu Punkt 6.
  3. Meldung einer Schwachstelle ohne zuvor erfaßten Vorfall
     Im Falle der Übernahme von Schwachstelleninformationen und im Falle der direkten Meldung einer Schwachstelle, der kein bekannter und gemeldeter Vorfall vorausging, wird ein Schwachstelle ohne zugehörige Vorfälle erfaßt. Es kann sein, daß hierfür zuerst die Informationen zur betroffenen Software erfaßt werden müssen, siehe dazu Punkt 6.

4. Meldung einer Gegenmaßnahme zu einer Schwachstelle
   Zu einer bereits erfaßten Schwachstelle kann eine Gegenmaßnahme eingegeben werden.
5. Meldung eines Exploits zu einer Schwachstelle
   Zu einer bereits erfaßten Schwachstelle wird eine Exploitmöglichkeit eingegeben.
6. Anlegen einer neuen Software, Software-Version oder eines neuen Vendors im Rahmen der Meldung einer Schwachstelle oder eines Vorfalls
7. Anlegen eines neuen Nutzers
8. Authentifizierung gegenüber der SDB
   Um Schreibzugriff auf die SDB zu erhalten, ist es notwendig sich gegenüber der SDB zu authentifizieren.
9. Eingeben eines Kommentars zu einem Vorfall / einer Schwachstelle / einer Gegenmaßnahme / einem Exploit bzw. eines Kommentars zu einem Kommentar zu einem Vorfall / einer Schwachstelle / einer Gegenmaßnahme / einem Exploit
   Zu einem einmal erfaßten Vorfall, einer einmal erfaßten Schwachstelle, einer einmal erfaßten Gegenmaßnahme oder einem einmal erfaßten Exploit können Kommentare, Anmerkungen und Hinweise eingegeben werden, aus denen dann die freigegebene Vorfall- oder Schwachstellenmeldung erzeugt wird.
10. Recherche über die Informationen der SDB
    Recherchen über Software, Schwachstellen und vergleichbare Themen sollen das Auffinden von bestimmten Schwachstellen ermöglichen.

- Administrator (zusätzliche Funktionalität)

1. Bearbeiten / Löschen eines Vorfalls
   Über diese Funktion kann ein Administrator Änderungen an Vorfallmeldungen vornehmen oder Einträge, die keine Vorfälle sind, löschen.
2. Bearbeiten / Löschen / Freigeben / Bewerten einer Schwachstelle
   In diesem Anwendungsfall kann der Administrator Änderungen an Schwachstellenmeldungen vornehmen und falsche oder doppelte Schwachstellenmeldungen löschen. Auch kann der Administrator zutreffende und in genügend großem Umfang erfaßte Schwachstellenmeldungen freigeben. Sofern unterstützt, kann er die Schwachstellenmeldung bewerten.
3. Bearbeiten / Löschen / Freigeben / Bewerten einer Gegenmaßnahme
4. Bearbeiten / Löschen / Freigeben / Bewerten eines Exploits
5. Anlegen / Bearbeiten / Löschen / Freigeben von Software, Software-Versionen oder Herstellern
   Der Administrator sollte im Gegensatz zum „normalen“ Nutzer in der Lage sein, Software, Software-Versionen und Hersteller ohne Zusammenhang mit einer Meldung zu erfassen.
6. Bearbeiten / Sperren / Löschen / Bewerten eines Nutzers
   Sollte ein Nutzer Mißbrauch bei der Nutzung der SDB betreiben, so kann sein Zugang vorübergehend gesperrt oder gar gelöscht werden. Ein böswilliger Nutzer wäre in diesem Fall zumindest gezwungen einen neuen individualisierten Account anzulegen.

7. Bearbeiten / Löschen / Sperren eines Kommentars zu einem Vorfall
8. Bearbeiten / Löschen / Sperren eines Kommentars zu einer Schwachstelle
9. Bearbeiten / Löschen / Sperren eines Kommentars zu einer Gegenmaßnahme
10. Bearbeiten / Löschen / Sperren eines Kommentars zu einem Exploit
11. Eingeben / Bearbeiten / Löschen von Hilfsdaten für die SDB
    Hilfsdaten dienen in der SDB als Vorgaben für bestimmte Feldeinträge oder können beispielsweise Entscheidungsbäume, die als Hilfe bei der Selektion eines zutreffenden Eintrags dienen, repräsentieren. Der Administrator sollte in der Lage sein, diese Daten zu modifizieren.

Kapitel 4

Die SEC://HOUSE-Schwachstellendatenbank

Diese Kapitel befaßt sich mit verschiedenen Aspekten der Implementierung der SEC://HOUSE-SDB. Abschnitt 4.1 erläutert das Datenbankschema der SDB. Danach wird in Abschnitt 4.2 eine geeignete Architektur der Administrator-Schnittstelle vorgestellt und schließlich präsentiert Abschnitt 4.3 den Prototypen der Administrator-Schnittstelle, der im Rahmen dieser Diplomarbeit implementiert wurde.

4.1 Datenbankschema

Dieser Abschnitt geht auf das Schema der SEC://HOUSE-Schwachstellendatenbank ein. Abbildung 4.1 zeigt das vollständige Schema der SDB. Das Schema wurde dabei in Englisch definiert. Man erkennt sofort die in Abschnitt 3.6 aufgeführten Klassen.

Abbildung 4.2 zeigt einen Ausschnitt aus dem Gesamtschema. Diser Ausschnitt befaßt sich mit den zentralen Klassen `Vulnerability`, `Vendor`, `Software`, `SW_Version`, `Operating_System` und `OS_Version`.

`Vulnerability` repräsentiert dabei die Daten, die zu einer Schwachstelle erfaßt werden. Für eine ausführlichere Erläuterung der einzelnen Elemente der Klasse wird auf Abschnitt A.2 im Anhang dieser Arbeit verwiesen.

`Vendor` wiederum speichert Daten zu Herstellern bzw. im Falle von Open Source Projekten, wie beispielsweise Apache[1] oder Samba[2], Betreuern (die sogenannten *Benevolent Dictators*) von Software. Einer `Software` bzw. einem `Operating_System` wird dann ein solcher `Vendor` zugewiesen. `Software` speichert die Daten zu einer Software, die kein Betriebssystem ist. `Operating_System` ist eine Spezialisierung von `Software`, bei der, im Gegensatz zu `Software`, der „Type of Software“ (*TypeOfSW*) als „Betriebssystem“ festgelegt ist.

[1] Vgl. dazu `http://www.apache.org`
[2] Vgl. dazu `http://www.samba.org`

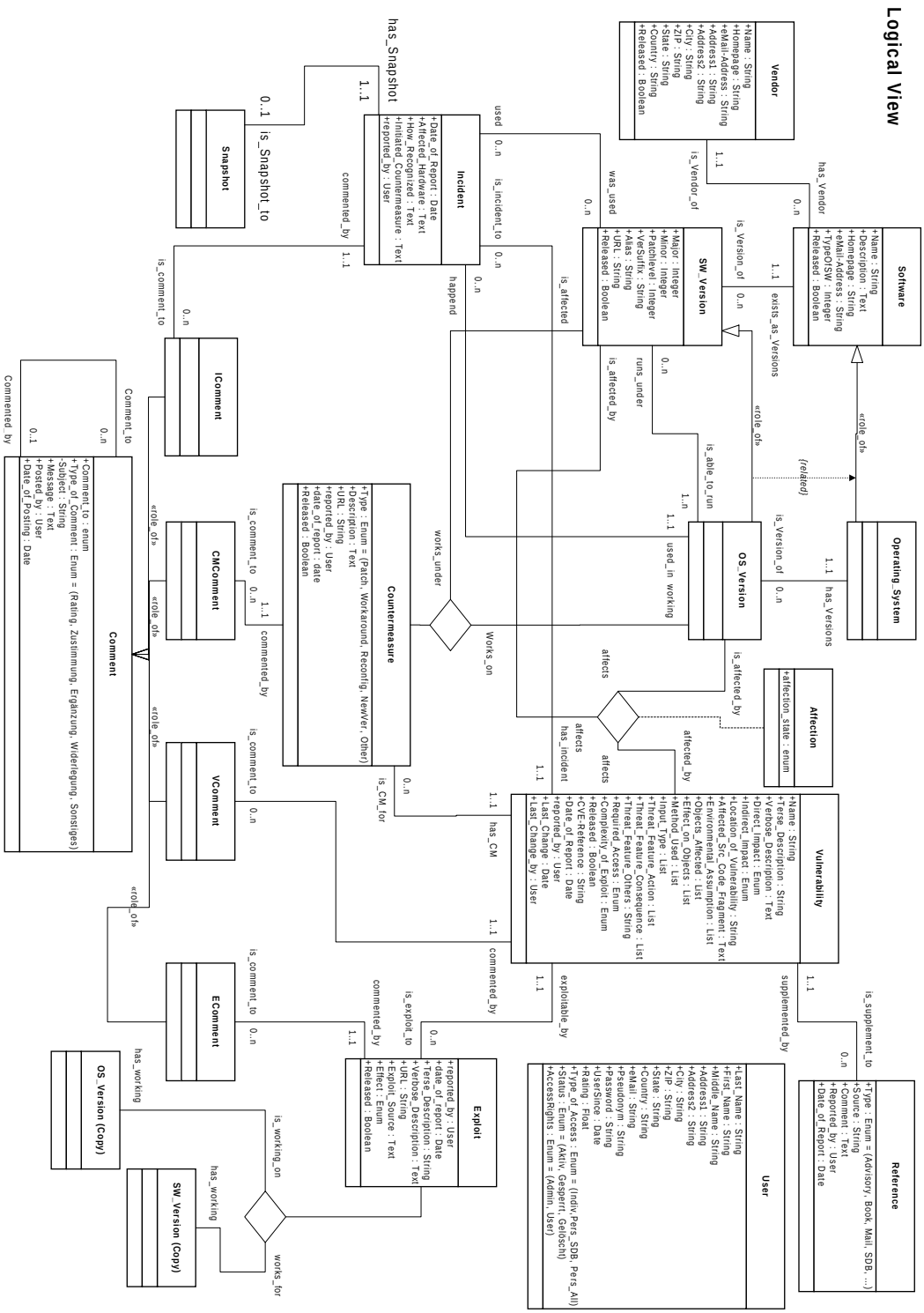

Abbildung 4.1: UML-Schema der SDB

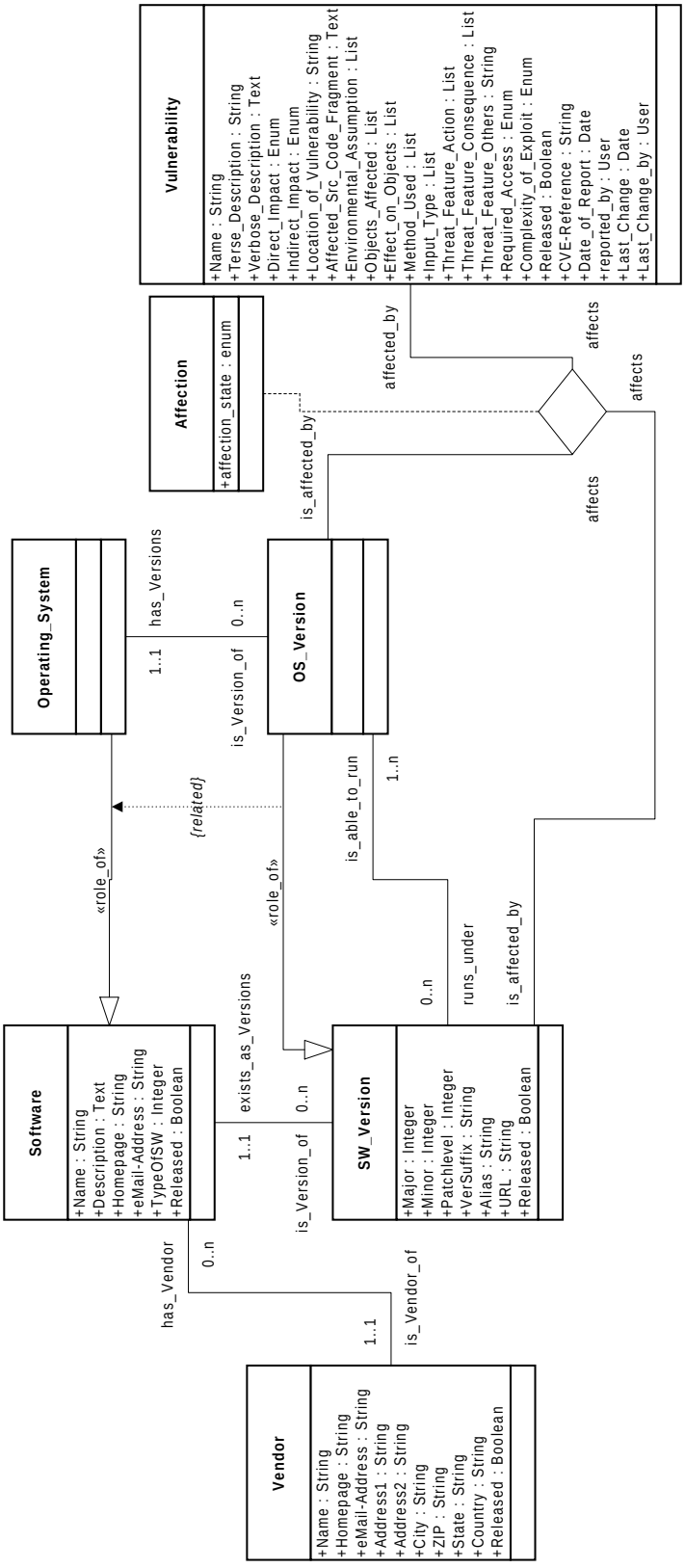

Abbildung 4.2: UML-Schema Vulnerability & Software

Einer `Software` sind beliebig viele Einträge in `SW_Version`, d.h. Softwareversionen, zugeordnet. Entsprechendes gilt für `Operating_System` und `OS_Version`, wobei `OS_Version` nur eine Spezialisierung von `SW_Version` darstellt.

Einer Schwachstelle sind nun eine Reihe von Kombinationen von Softwareversionen und Betriebssystemversionen zugeordnet. Sie werden durch das Attribut *affection_state*, das erfaßt, ob die entsprechende Kombination von der zugeordneten Schwachstelle betroffen, nicht betroffen oder möglicherweise betroffen ist, ergänzt. Die entsprechende Beziehung wird durch die ternäre Beziehung zwischen `Vulnerability`, `SW_Version` und `OS_Version` ausgedrückt.

Abbildung 4.3 ergänzt das Schema aus Abbildung 4.2 um den Bereich `Incident` und `Snapshot`. `Incident` erfaßt Vorfälle, die „bemerkt“ wurden. Dabei ist es möglich, daß ein Vorfall keiner Schwachstelle zugewiesen ist, was dann sinnvoll ist, wenn die Schwachstelle, die zu einem Vorfall geführt hat, nicht bekannt ist.

`Incident` enthält ausschließlich textuelle Angaben. Die Informationen über das auf dem System eingesetzte Betriebssystem und die auf dem System eingesetzte Software werden im Rahmen der entsprechenden Beziehungen gespeichert. Weitere, detailliertere Informationen zu einem Vorfall respektive zu einem System, auf dem sich ein Vorfall ereignet hat, sollen in der noch nicht genauer spezifizierten Klasse `Snapshot` gespeichert werden. Ziel ist es, eine Software zu entwickeln, die möglichst exakt analysiert, um was für ein System es sich handelt und welche Software in welchen Versionen dort eingesetzt wird. Dabei muß zusätzlich eine „Anonymisierung“ der Daten vorgenommen werden, so daß beispielsweise über Rechnernamen oder IP-Adressen keine Rückschlüsse auf das untersuchte System möglich sind. Ansonsten könnten die Analyseinformationen in den falschen Händen als Ausgangspunkt für Angriffe auf das entsprechende System genutzt werden. Die entsprechenden Analyseprogramme existieren zur Zeit noch nicht und sollen im Rahmen kommender Forschungsarbeiten entwickelt werden.

Abbildung 4.4 wiederum ergänzt das Schema aus Abbildung 4.2 um die Klasse `Countermeasure` (Gegenmaßnahme). Diese erfaßt Möglichkeiten, die Gefährdung durch eine Schwachstelle abzuschwächen oder zu beseitigen. Zu jeder Gegenmaßnahme wird gespeichert, auf welchen der von der Schwachstelle betroffenen Systemen, d.h. Softwareversion-Betriebssystemversion-Kombinationen, die Gegenmaßnahme eingesetzt werden kann. Dies wird durch die ternäre Beziehung zwischen `Countermeasure`, `SW_Version` und `OS_Version` ausgedrückt.

Die Klasse `Exploit`, dargestellt in Abbildung 4.5, erfaßt bekannte Möglichkeiten zur „Ausnutzung“ einer Schwachstelle, um ein System anzugreifen respektive auf das Vorhandensein der Schwachstelle zu testen. Dabei wird zu jedem Exploit erfaßt, auf welchen der von der Schwachstelle betroffenen Systemen der Exploit verwendet werden kann. Um vor einem Test die Auswirkungen des Tests auf das laufende System abschätzen zu können, wird im Feld *Effect* erfaßt, welchen Einfluß der Test auf das laufende System hat.

An dieser Stelle sei noch einmal auf Abbildung 4.1 verwiesen. Dort sind einige, bisher noch nicht erläuterte Klassen zu sehen. Dies ist zum einem die Klasse `Reference`. Diese speichert zu jeder Schwachstelle Referenzen auf Quellen, die Informationen zu der Schwachstelle enthalten. Darunter sollten sich beispielsweise auch die Quellen befinden, die zur Erstellung des Schwachstellen-Eintrags in der SDB gedient haben.

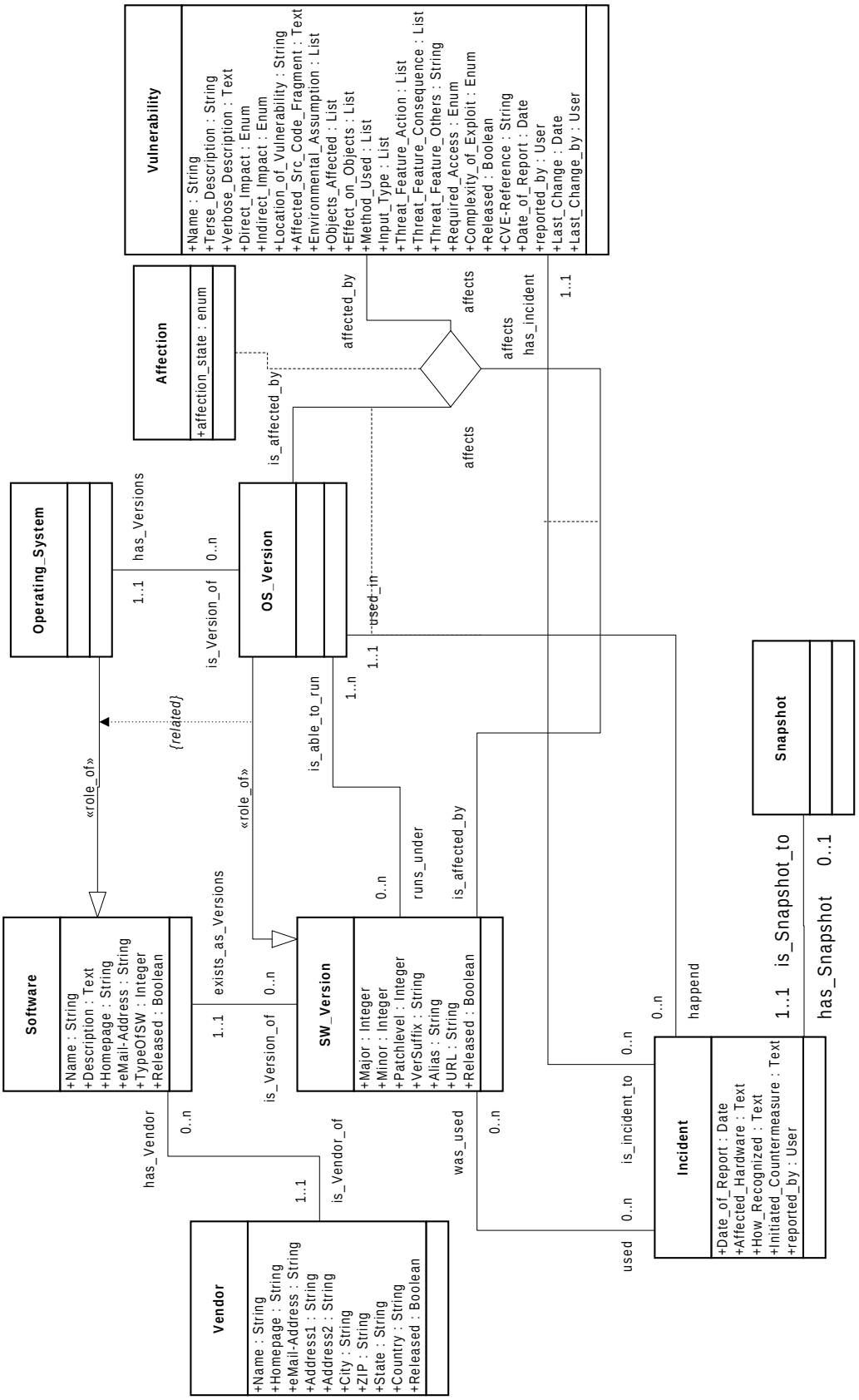

Abbildung 4.3: UML-Schema Incident

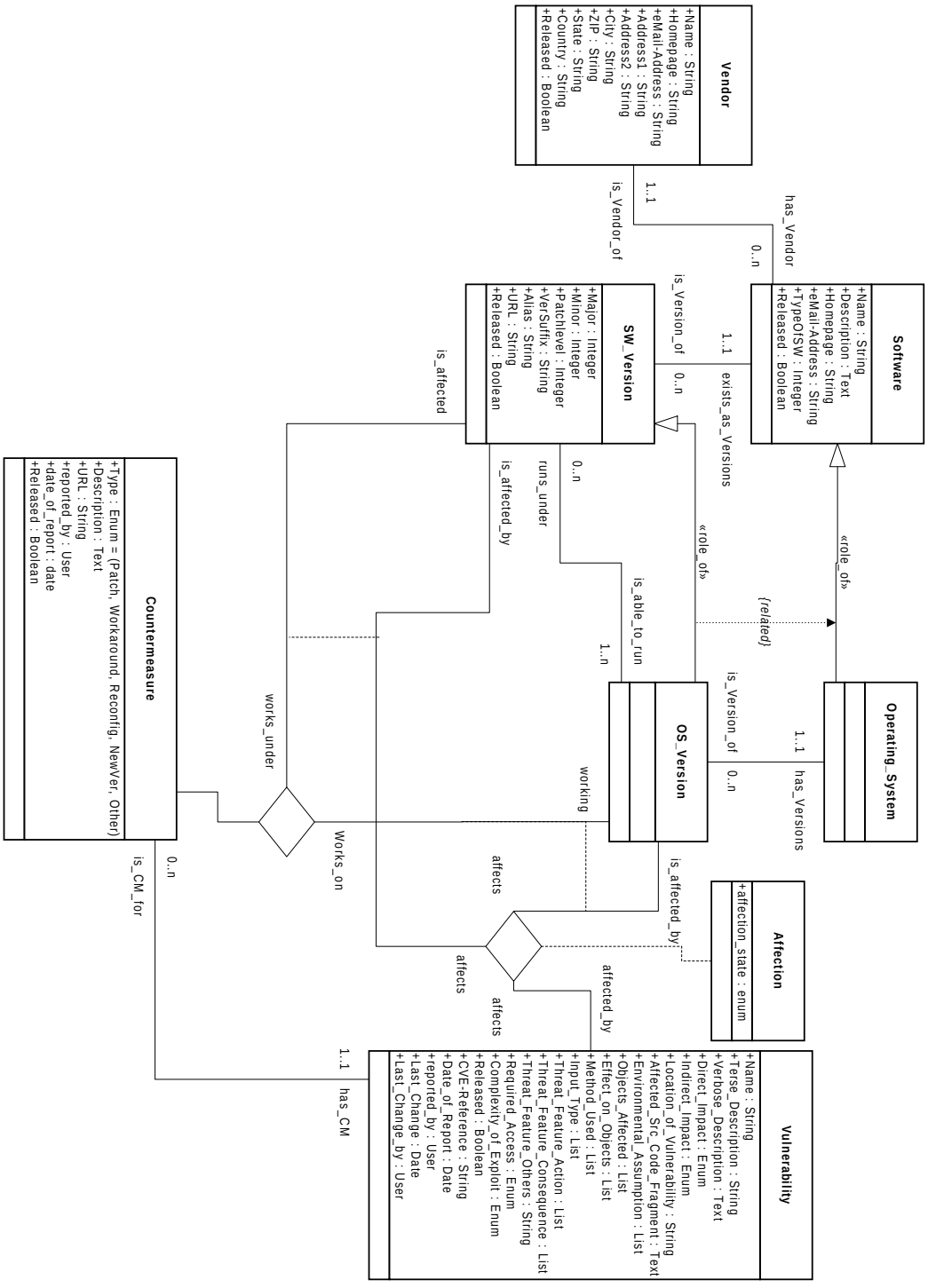

Abbildung 4.4: UML-Schema Countermeasure

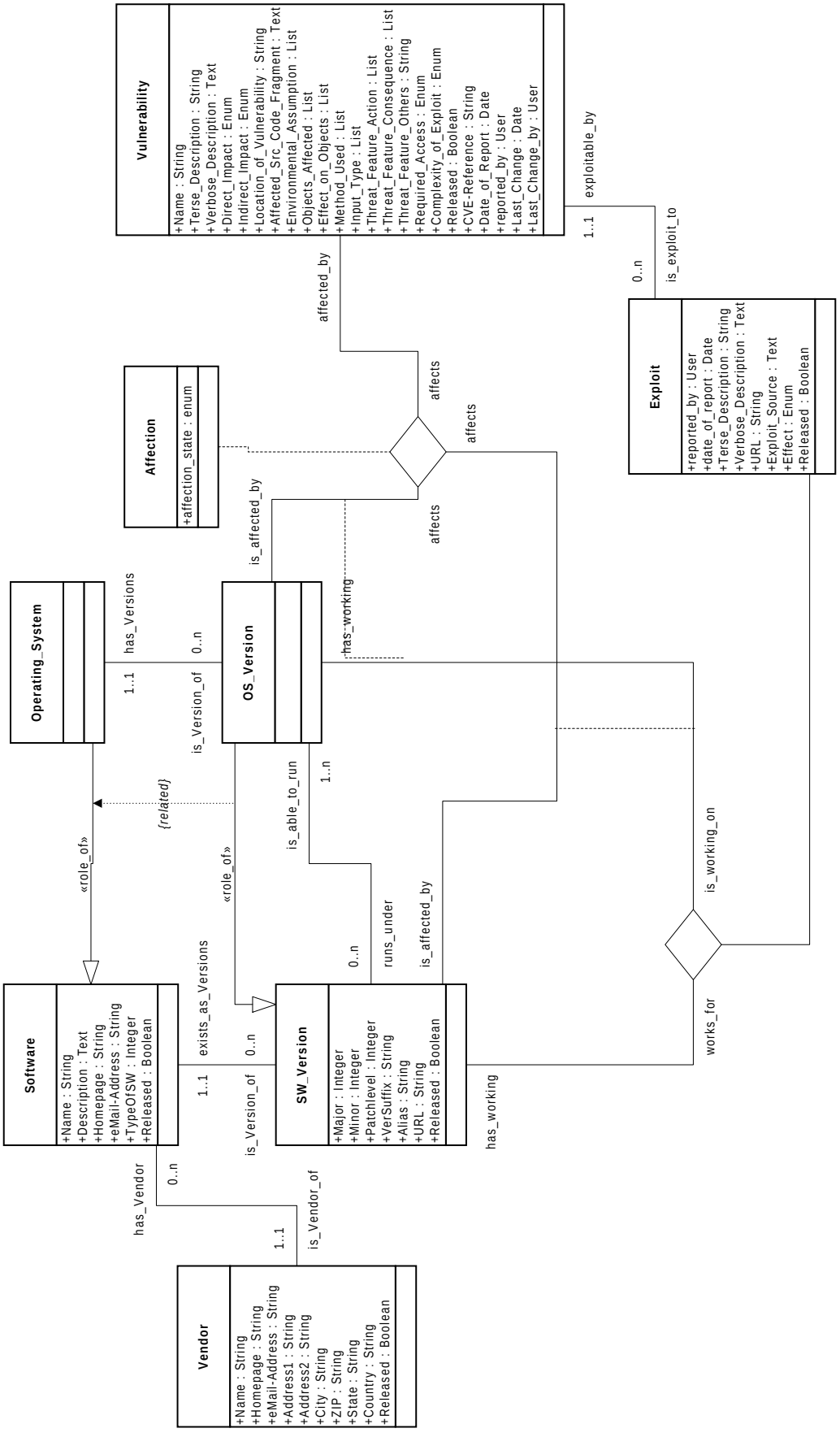

Abbildung 4.5: UML-Schema Exploit

Als nächstes ist noch die Klasse `User` zu sehen. `User` erfaßt Benutzerdaten, darunter insbesondere das Pseudonym, das der Nutzer in der SDB verwendet und sein Zugangspaßwort. Auch wird hier erfaßt, welchen Zugangsmodus der Nutzer verwendet,[3] ob dieses Nutzerkonto aktiv, temporär gesperrt oder gar gelöscht ist und ob es sich bei dem Nutzer um einen „normalen“ Nutzer oder einen SDB-Administrator handelt.

Schließlich sind in Abbildung 4.1 noch die Klassen `Comment`, `VComment`, `IComment`, `CMComment` und `EComment` dargestellt. `Comment` dient dazu Diskussionen innerhalb der SDB über Schwachstellen, Vorfälle, Gegenmaßnahmen und Exploits zu ermöglichen. Dabei stellen `VComment`, `IComment`, `CMComment` und `EComment` nur Spezialisierungen der Klasse `Comment` dar, die dazu dienen, Diskussionen zu Schwachstellen, Vorfällen, Gegenmaßnahmen und Exploits zu unterscheiden. Über `Comment` sind Diskussionen ähnlich wie in Mailing-Listen oder Newsgroups möglich, mit Ausnahme der Tatsache, daß sogenannte „Crosspostings“, also Diskussionsbeiträge, die in mehreren der vier Bereiche gleichzeitig erscheinen, nicht möglich sind. Dies erscheint in Verbindung mit der gegebenen Thematik allerdings auch nicht sinnvoll.

4.2 Anzustrebende Architektur der Administrator-Schnittstelle

Beim Betrieb der SDB und ihrer Schnittstellen und speziell beim Betrieb der Administrator-Schnittstelle (im folgenden kurz: Admin-Tool) ist es unumgänglich auf Sicherheit zu achten. Daher ist es u.a. notwendig, die Kommunikation zwischen SDB und Nutzer zu verschlüsseln, so daß niemand die Möglichkeit hat, Zugangspaßwörter oder auch nur die Meldungen einzelner Teilnehmer abzufangen und mitzulesen. Dies ist insbesondere aus den in Abschnitt 3.1.2 erläuterten Gründen notwendig.

Weiterhin empfiehlt es sich, das Admin-Tool, aber auch andere SEC://HOUSE-Anwendungen, in Java zu entwickeln, da dadurch eine größtmögliche Portabilität der Anwendung auf viele verschiedene Plattformen erreicht wird. Zusätzlich ermöglich die Verwendung von Java auch die Verwendung von CORBA. Dies ist zwar in Verbindung mit dem Admin-Tool von nachrangiger Bedeutung, jedoch können Anwendungen, die CORBA verwenden, mit geringem Aufwand an Java angebunden werden.

Es müssen zwei Punkte sichergestellt werden. Zum einen muß ein Nutzer die Möglichkeit haben sicherzustellen, daß eine Software, die er heruntergeladen hat, um auf SEC://HOUSE zuzugreifen, integer ist. Nur so kann er die Gewißheit haben, daß es sich um die Originalsoftware der SEC://HOUSE-SDB handelt und er kein „trojanisches Pferd“ oder ähnliches verwendet. Zum anderen muß sichergestellt sein, daß, wenn der Anwender die Software nutzt, sein Zugriff nicht von Dritten belauscht werden kann. Wie dies ermöglicht werden kann, zeigt Abbildung 4.6 exemplarisch für das Admin-Tool.

Um in Besitz einer integren Version des Admin-Tools zu gelangen, lädt ein Administrator die Software von der Homepage der SEC://HOUSE-Datenbank herunter. Dabei wird über die Secure Socket Layer (SSL), eine genormte Verschlüsselungsschnittstelle für Netzwerkkommunikation, eine gesicherte Verbindung aufgebaut. Dadurch ist es möglich, daß ein Nutzer mit hoher Wahrscheinlichkeit sicher sein kann, mit dem Server zu kommunizieren,

[3] Vgl. Abschnitt 3.1.2

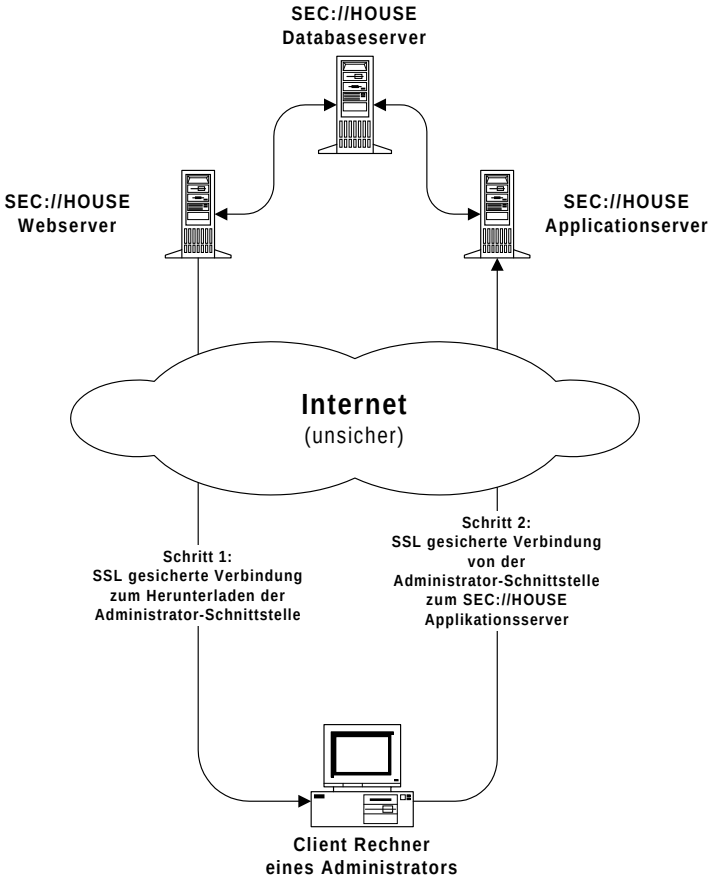

Abbildung 4.6: Wahrung von Integrität und Vertraulichkeit

mit dem er kommunizieren will. Über diese Verbindung kann nun der Nutzer das Admin-Tool herunterladen und er kann sicher sein, daß es sich um eine unveränderte Originalversion des Tools handelt.

Wird zusätzlich für den Programmcode des Tools eine Prüfsumme angegeben, die es dem Anwender erlaubt, die Authentizität des Programms zu überprüfen, so steigt die Sicherheit eine integre Version zu nutzen weiter. Für die Berechnung einer solchen Prüfsumme eignen sich genormte Verfahren, wie beispielsweise der Secure Hash Algorithm (SHA).

Nutzer sollten auf SEC://HOUSE selbst über eine Drei-Schichten-Architektur aus Anwendung, Applikations-Server und Datenbank-Server zugreifen.

Ein Applikations-Server wird aus verschiedenen Gründen anstatt einer direkten Verbindung zur Datenbank über die Java Datenbankschnittstelle JDBC vorgeschlagen. Moderne Datenbanken erlauben zwar eine relativ feingranulare Vergabe von Zugriffsrechten bis auf die Ebene einzelner Attribute hinab. Auch ist es je nach DBMS möglich, auf SQL-Views Updates oder Inserts auszuführen, ohne daß dabei nicht durch den View erfaßte Datensätze eingefügt oder geändert werden könnten. Allerdings kann eine solche Kontrolle auf der Ebene des DBMS dazu führen, daß u.U. ein View pro Benutzer und Tabelle angelegt werden muß. Soll beispielsweise ein Benutzer aus einer Tabelle nur Datensätze sehen, die er auch

angelegt hat, so kann es nötig sein, für diesen Benutzer einen eigenen View anzulegen. Reichen solche Beschränkungen immer noch nicht aus, so ist es weiterhin möglich, die entsprechende Stored Procedure Language des DBMS zu verwenden, um weitere Kontrollen durchzuführen. Jedoch kann es auch in diesem Fall möglich sein, daß solche Stored Procedures pro Nutzer definiert werden müssen. Dies ließe sich zwar weitgehend automatisieren, stellt, insbesondere bei einer großen Nutzerzahl, jedoch u.U. einen hohen Aufwand dar, der bei Verwendung eines Applikations-Servers eher geringer ausfallen dürfte, da hier eine dem Zweck angepaßtere Rechtevergabe möglich ist. Allerdings entstehen mit der Verwendung eines Applikation-Servers auch neue Probleme in der Art, daß dieser in geeignet Art und Weise gegen Angriffe gehärtet werden muß.

Außerdem nutzen verschiedene DBMS, wie beispielsweise Informix Foundation 2000, zur Nutzerauthentifizierung die entsprechenden Mechanismen des Betriebssystems, d.h. für jeden Nutzer, der sich direkt an der SDB anmelden können soll, muß auf Betriebssystemebene ein Benutzerkonto angelegt werden, was bei einer hohen Zahl von Nutzern, die eigentlich gar keinen Betriebssystemzugriff benötigen, schnell aufwendig werden kann. Entkoppelt man die Nutzerauthentifizierung von der Ebene des DBMS, so erhält man damit eine flexiblere und gleichzeitig portablere Lösung, auch für den Fall, daß einmal ein anderes DBMS eingesetzt werden soll.

Zusätzlich spricht für die Applikations-Server-Variante, daß viele heute verfügbare JDBC-Treiber unverschlüsselte Verbindungen zur Datenbank nutzen, was wiederum der Sicherheitsprämisse widerspricht. Und schließlich ist es bei der Verwendung eines Applikations-Servers nicht notwendig, daß der Anwender den JDBC-Treiber lokal vorliegen hat, was Probleme mit dem Copyright verursachen könnte, oder den JDBC-Treiber bei jeder Verwendung des Programms über das Internet herunterladen muß, was bei Größen im Megabyte-Bereich sehr aufwendig sein kann.

Nutzt nun ein Administrator das Admin-Tool, so sollte dies wiederum eine sichere Kommunikationsverbindung zum Applikations-Server von SEC://HOUSE herstellen.

Die sichere Verbindung könnte wiederum durch die Verwendung von SSL realisiert werden. Unterstellt man weiterhin, daß erstens Java 1.2 oder eine neuere Version eingesetzt wird und zweitens die Kommunikation zwischen Admin-Tool und Applikationsserver über die Java-eigene Schnittstelle zum entfernten Methodenaufruf *Remote Method Invocation* (RMI) realisiert wird, so ergibt sich durch die *Java Secure Socket Extension* (JSSE) [SM00], eine spezielle, von Sun Microsystems entwickelte, Programmierschnittstelle für Java, eine relativ einfache Möglichkeit, die RMI-Kommunikation durch eine SSL-Verbindung zu „tunneln“ und somit für Dritte unlesbar zu machen. Die Verbindung wird dabei durch X.509-Zertifikate gesichert. Dabei würde der Applikations-Server den privaten Schlüssel sicher „verwahren“ und in das Admin-Tool oder in eine andere Software könnte der öffentliche Schlüssel integriert werden. Da, entsprechend obiger Beschreibung, garantiert werden kann, daß das Admin-Tool unverändert vorliegt, kann es sich auch nur mit dem „richtigen“ Applikations-Server in Verbindung setzen, da nur dann die entsprechende Authentifizierung funktioniert.

Das Admin-Tool kann nun über die gesicherte Verbindung Anfragen an den Applikations-Server stellen, die dieser, nach weitergehenden Überprüfungen, an den Datenbank-Server sendet und die Antwort an das Admin-Tool zurückleitet. Dies umfaßt natürlich auch die Authentifizierung des Anwenders gegenüber der SDB respektive dem Applikations-Server.

Während der Nutzung der SDB ist es weiterhin denkbar, die Zugriffe eines Anwenders über den Applikations-Server auf die SDB transparent zu protokollieren, um beispielsweise einen mißbräuchlichen Gebrauch der SDB erkennen zu können. Ein solches Vorgehen ist jedoch zumindest problematisch, da Anwender u.U. Probleme mit dem Logging der Anfragen haben dürften.[4]

Die in Abbildung 4.6 zusätzlich dargestellte Verbindung zwischen Datenbank-Server und Web-Server soll auf das Vorhandensein einer zusätzlichen Web-Schnittstelle hindeuten.

4.3 Prototyp der Administrator-Schnittstelle

Im Rahmen dieser Diplomarbeit wurde ein Prototyp für die Administrator-Schnittstelle entwickelt. Dieser Prototyp verwendet im Gegensatz zum Drei-Schichten-Ansatz, der in Kapitel 4.2 vorgestellt wurde, einen Zwei-Schichten-Ansatz, der das Admin-Tool und den Datenbank-Server umfaßt. Dieser Ansatz wurde ausschließlich deshalb gewählt, um den Umfang der Diplomarbeit zu begrenzen. Auch wurde der Funktionsumfang reduziert. So können mit dem Prototypen nur die Daten zu den folgenden Klassen erfaßt werden: `Vulnerability`, `Vendor`, `Software`, `SW_Version`, `Operating_System`, `OS_Version`, `Reference` und `Countermeasure`.

Der Prototyp wurde in Java 1.2 implementiert. Verwendet wurde für die Entwicklung das Werkzeugs Forte for Java 1.0 von Sun Microsystems. Als Datenbank kam Informix Foundation 2000 zum Einsatz. Das Excalibur Text Search Datablade Module (Release 1.30.UC2) dient zur Speicherung großer Textmengen. Der Einsatz des Text Datablades hat den Vorteil, daß bei späteren Analysen des SDB-Datenbestandes selbst auf den Textfeldern sehr komplexe Recherchen erfolgen können. Zum Zugriff auf die Datenbank wurde der Informix JDBC Treiber verwandt.

Aufgrund der Zwei-Schichten-Architektur sollte der Prototyp nur im lokalen Netz der SEC://HOUSE-SDB verwendet werden, da die Kommunikation über den JDBC-Treiber unverschlüsselt abläuft, so daß beispielsweise Pseudonym und Paßwort abgehört werden können.

4.3.1 Architektur des Prototypen

Für die Implementierung des Prototypen stand kein Java-Framework für Datenbank-Anwendungen zur Verfügung. Da auch kein solches Framework entwickelt wurde, weist der Prototyp eine verhältnismäßig monolithische Archtiktur auf, die die Steuerung von grafischen Elementen und den Datenbankzugriff in einer Klasse vereint. Daher ähnelt der Prototyp eher einem Programm einer prozeduralen als einer objektorientierten Sprache, wie Java. Objektorientierte Strukturen zeigen sich primär bezüglich der Erstellung der Benutzeroberfläche. Hierfür wurde die Java-Klassenbibliothek *Swing* eingesetzt.

[4] Vgl. auch Abschnitt 3.1.2

Die monolithische Struktur des Prototyps drückt sich primär dadurch aus, daß die zentrale Funktionalität des Admin-Tools in der Klasse `SecHouse` zusammengefaßt ist. Die Programmstruktur orientiert sich an der Struktur der Benutzeroberfläche. Dementsprechend repräsentieren die meisten Klassen, die außerdem noch zum Prototypen gehören, Dialogboxen oder andere unabhängig vom Hauptfenster darzustellende Benutzeroberflächenelemente. Für die ausführliche Dokumentation der Klassen, ihrer Variablen und Methoden sei darauf verwiesen, daß der Quellcode des Prototypen so dokumentiert ist, daß jederzeit mit Hilfe des JAVADOC-Programms, das zum Java Development Kit gehört, eine vollständige, HTML-basierte Onlinedokumentation generiert werden kann. Für den Quellcode und die ausführliche JAVADOC-Dokumentation des Admin-Tools sei auf die Abschnitte A.5 und A.6 des Anhangs verwiesen.

Abbildung 4.7 zeigt den groben Aufbau des Prototypen. Die zentrale Klasse des Admin-Tools ist die Klasse `SecHouse`, die von JFrame erbt. `SecHouse` benutzt das GUI-Element *jTabbedPaneMain*, daß eine Instanz der Klasse `JTabbedPane` ist, um ein Karteikartenregister anzuzeigen. Die einzelnen Karteikarten werden durch Instanzen der Klasse `JPanel` repräsentiert. Welche Instanz dabei welche Karteikarte repräsentiert, geht aus der Abbildung hervor.[5]

Weiterhin geht aus Abbildung 4.7 hervor, welche der zusätzlichen Klassen, in der Regel Nachfahren der Klasse `JDialog`, zu welchem funktionalen Teil des Prototypen gehören. Die Klasse `Login`, die von `JDialog` erbt, repräsentiert den Login-Dialog zur SEC://HOUSE-SDB.

Die Klasse `QueryBuilder`, die ebenfalls von `JDialog` erbt, ist für die Eingabe und Verarbeitung von Recherchen zuständig. Das Ergebnis einer Recherche wird mittels einer Instanz der Klasse `ResultViewer`, die von `JFrame` erbt, angezeigt.

Die Klasse `TreeWalker`, die von `JDialog` erbt, wird verwendet, um einen Nutzer durch einen Entscheidungsbaum navigieren zu lassen, wie es einerseits für die Klassifizierungsmerkmale auf der Karteikarte „Classification 1“, beispielsweise für Direct Impact, notwendig ist. Andererseits wird der `TreeWalker` für die Bestimmung des „Type of Software“ in der Klasse `SoftwareEditor` verwendet, die nachfolgend noch beschrieben wird.

Die Klasse `VendorEditor`, die wiederum von `JDialog` erbt, ermöglich es einen Vendor zu erzeugen oder dessen Daten zu bearbeiten. Dabei kann es sich um den Hersteller einer Software halten, weshalb er im *jPanelAffectedSys* genutzt wird, oder aber um einen Betriebssystemhersteller, weshalb `VendorEditor` auch in der später beschriebenen Klasse `OS4SWVerEditor` verwendet wird. Dasselbe gilt für die Klasse `SoftwareEditor`, die ebenfalls von der Klasse `JDialog` erbt. `SoftwareEditor` dient zur Erstellung einer neuen bzw. der Bearbeitung einer existierenden Software bzw. eines Betriebssystems. Ebenfalls sowohl bei *jPanelAffectedSys* als auch bei `OS4SWVerEditor` findet die Klasse `SWVersionEditor` Verwendung, die erneut von `JDialog` erbt. `SWVersionEditor` dient zur Erzeugung oder Bearbeitung von Software- und Betriebssystem-Versionen.

Die Klasse `OS4SWVerEditor`, die wiederum von `JDialog` erbt, dient dazu, zu einer Software-Version, die in die Karteikarte „Affected Systems“ ausgewählt wurde, die Liste der Betriebssystem-Versionen, auf denen diese Software-Version läuft, zu bearbeiten, d.h. Betriebssystem-Versionen hinzuzufügen oder zu löschen. Im Rahmen des Dialogs

[5] Vgl. dazu auch Abschnitt 4.3.3

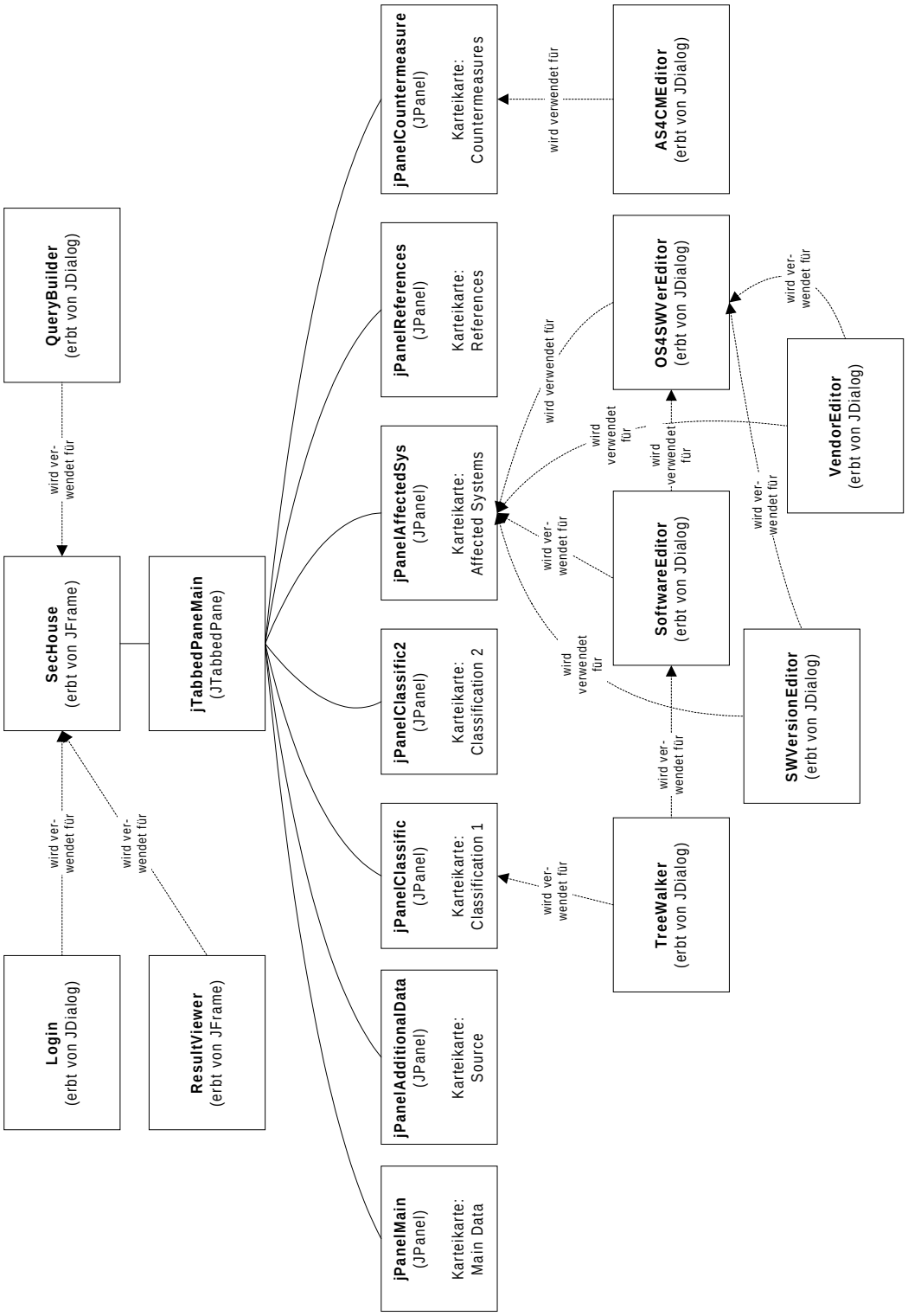

Abbildung 4.7: Grober Aufbau des Prototypen

`OS4SWVerEditor` ist es außerdem möglich neue Betriebssysteme zu definieren oder bestehende zu bearbeiten oder zu löschen.

Der `JDialog` `AS4CMEditor` dient dazu, zu einer Gegenmaßnahme, die Kombinationen aus Software-Version und Betriebssystem-Version auszuwählen, für die die Gegenmaßnahme eingesetzt werden kann.

Die Methoden, die zu bestimmten Karteikarten des Prototypen gehören, tragen durchgängig Bezeichnungen, die einen Rückschluß auf die Funktion der Methode, aber auch auf den thematischen Zusammenhang in dem die Methode steht, zuläßt. So gehören beispielsweise Methoden oder Variablen die ein „CM“ im Namen tragen zum Themenbreich Countermeasure, „Ref“ steht für Reference und AS für Affected Systems.

4.3.2 Erweiterung des Prototypen

Soll der Prototyp um die fehlenden Funktionen bezüglich `Exploit` oder `Incident` erweitert werden, so ist dies verhältnismäßig einfach zu erreichen. Für die entsprechende Funktion bietet es sich an, eine weitere Karteikarte zu erzeugen und die entsprechenden Benutzeroberflächen-Elemente für diese Funktion dort zu definieren.

Um eine weitere Karteikarte zu erzeugen, fügt man zu *jTabbedPaneMain* ein neues `JPanel` hinzu. Dies kann mittels des GUI-Builders von Forte for Java geschehen oder „von Hand“. Wird der Code von Hand hinzugefügt, so ist es empfehlenswert, den entsprechenden Code in eine separate Methode zu schreiben und diese im Constructor der Klasse `SecHouse` nach dem Aufruf der Methode *initComponents* aufzurufen.

Eine neu hinzugefügte Funktion ist weitgehend abgekoppelt von der restlichen Funktionalität des Prototypen. Einzig die folgenden Klassen bzw. Methoden sollten zusätzlich daraufhin überprüft werden, ob sie Erweiterungen für die zusätzliche Funktion benötigen:

- Klasse `SecHouse`
  - *Variable nbPanes*

    Gibt die Anzahl der Karteikarten an. Bei Erweiterung des Prototyps ist die Anzahl zu erhöhen. Im übrigen sollte die Reihenfolge der Karteikarten nicht verändert werden. Soll die Reihenfolge doch geändert werden, so sollte nach dem Auftreten der Variablen nbPanes gesucht werden. Probleme dürfte die verschiedenen Funktionen zur Deaktivierung von Karteikarten bereiten, da die Karteikartennummern nach Veränderung der Reihenfolge u.U. nicht mehr stimmen.
  - *Methode jMenuItemVulLoginActionPerformed*

    Sollen nach dem Login bestimmte Menüs zusätzlich aktiviert werden, so ist diese Funktionalität hier einzufügen.
  - *Methode activateNewVulnerability*

    Dient zur Initialisierung der Karteikarten, wenn eine neue Schwachstelle erzeugt werden soll, also beispielsweise direkt nach dem Login oder wenn New im *Menü Vulnerability* gewählt wurde.

  - *Methode activateDetails*

    Wird aufgerufen, wenn eine Schwachstelle geladen wurde und nun die Werte für die einzelnen Karteikarten Reference, Countermeasure et cetera geladen werden sollen.

  - *Methode loadAdditionalInitializationData*

    Müssen nach Aufbau der Verbindung zur Datenbank noch zusätzliche Initialisierungen, beispielsweise von Combo-Boxen durchgeführt werden, so kann dies in dieser Methode geschehen.

- Klasse `Helper`

  Die Klasse `Helper` enthält genau eine Methode, die nach dem erfolgreichen Login in die Datenbank aufgerufen wird und dann Daten aus der Datenbank liest und in der Klasse `StaticStorer` speichert. Sollen für die neue Funktion zusätzliche Daten geladen werden, so sollte dies in dieser Klasse geschehen.

- Klasse `StaticStorer`

  Die Klasse `StaticStorer` enthält verschiedene statische Daten, die nach dem ersten Login in die Datenbank geladen werden. Sollen zusätzliche Daten über die Klasse `Helper` geladen werden und in der Klasse `StaticStorer` gespeichert werden, so ist `StaticStorer` um die entsprechenden Variablen und Zugriffsmethoden zu erweitern.

4.3.3 Benutzung des Prototypen

Nach dem Start des Prototypen muß der Benutzer sich an der SEC://HOUSE-SDB anmelden. Dies geschieht über den *Menüpunkt Login* im *Menü Vulnerability*. Wurde die Anmeldung mittels Pseudonym und Paßwort erfolgreich durchgeführt, hat der Nutzer direkt die Möglichkeit eine neue Schwachstelle anzulegen. Abbildung 4.8 zeigt diesen Zustand.

Man erkennt, daß das Hauptfenster des Prototypen einen karteikartenähnlichen Aufbau mit sieben Reitern besitzt. Die erste Karteikarte **Main Data** enthält Hauptangaben zu einer Schwachstelle, darunter fällt die zu wählende Bezeichnung, eine kurze und eine ausführliche Beschreibung und die CVE-Bezeichnung der Schwachstelle. Außerdem kann angegeben werden, ob diese Schwachstelle bereits als offizielle Meldung der SDB freigegeben ist. Schließlich wird noch angezeigt, wann und von wem die Schwachstelle eingetragen und zuletzt verändert wurde.

Die zweite Karteikarte **Source** erlaubt es Angaben zu der Stelle des Quellcodes, der die Schwachstelle darstellt, und einen Teil des betroffenen Sourcecodes zu erfassen.

Die dritte Karteikarte **Classification 1**, die in Abbildung 4.9 dargestellt ist, enthält erste Kriterien für die Klassifikation von Schwachstellen. Dies sind *Direct Impact*, *Indirect Impact*, *Access Required* und *Complexity of Exploit*. Diese Kriterien wurden von Krsuls SDB übernommen. Für eine genaue Definition der Felder und der möglichen Werte sei noch einmal auf Abschnitt A.2 im Anhang dieser Arbeit verwiesen. Außerdem können hier weitere Bedrohungen, die im Rahmen der nächsten Karteikarte nicht erfaßt werden, eingetragen werden. Der Grund dafür, daß dieser Eingabepunkt vor die nächste Karteikarte gezogen wurde,

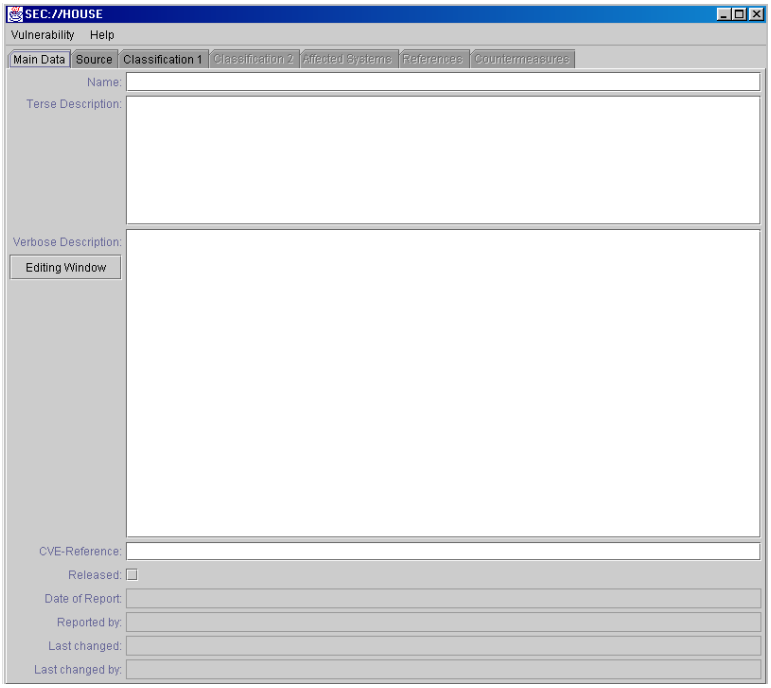

Abbildung 4.8: Der SEC://HOUSE-Prototyp nach dem erfolgreichen Login

lag primär darin, daß der Punkt auf dieser Karteikarte noch eingebaut werden konnte. Zusätzlich ist dies die letzte Karteikarte, die bei einem noch nicht gespeicherten Datensatz bearbeitet werden kann.

Die übrigen Karteikarten sind erst zugänglich, wenn eine Schwachstelle gespeichert wurde. Dies liegt primär darin begründet, daß erst dann der Datenbankschlüssel der Schwachstelle verfügbar ist, der benötigt wird, um die übrigen Daten in der Datenbank zu speichern. Gespeichert wird eine Schwachstelle bzw. Änderungen an den Daten einer Schwachstelle über den *Menüpunkt Save* im *Menü Vulnerability*.

Soll eines der Klassifikationskriterien geändert werden, erscheint ein Dialog, der die Spezifikation des Wertes ermöglicht. Die Einträge werden über ein „Frage-Antwort-Spiel“ innerhalb von Entscheidungsbäumen ermittelt. Den entsprechenden Dialog zeigt Abbildung 4.10.

Die nächste Karteikarte **Classification 2** enthält weitere Klassifikationsmöglichkeiten für die Schwachstelle. Über die *Combo-Box Topic* können mehrere Themengebiete gewählt werden, zu denen dann jeweils mehrere Aspekte angezeigt werden. Dies sind: *Environmental Assumption*, *Objects Affected*, *Effects on Objects*, *Method Used*, *Input Type*, *Threat Features: Action* und *Threat Features: Consequence*. Zu jedem dieser Aspekte kann einer der Werte Ja, Nein, Nicht zutreffend und Unbekannt ausgewählt werden. Auch diese Klassifikationskriterien sind von Krsul übernommen worden. Am unteren Rand des Bearbeitungsfensters wird, wenn eines der Merkmale angeklickt wird, eine Definition des Merkmals angezeigt. Abbildung 4.11 zeigt diese Karteikarte.

Die fünfte Karteikarte ist **Affected Systems** (Abbildung 4.12). Hier können die Kombinationen aus Software-Version und Betriebssystem-Version, die von der Schwachstelle betroffen, nicht betroffen oder eventuell betroffen sind, festgelegt und angezeigt werden.

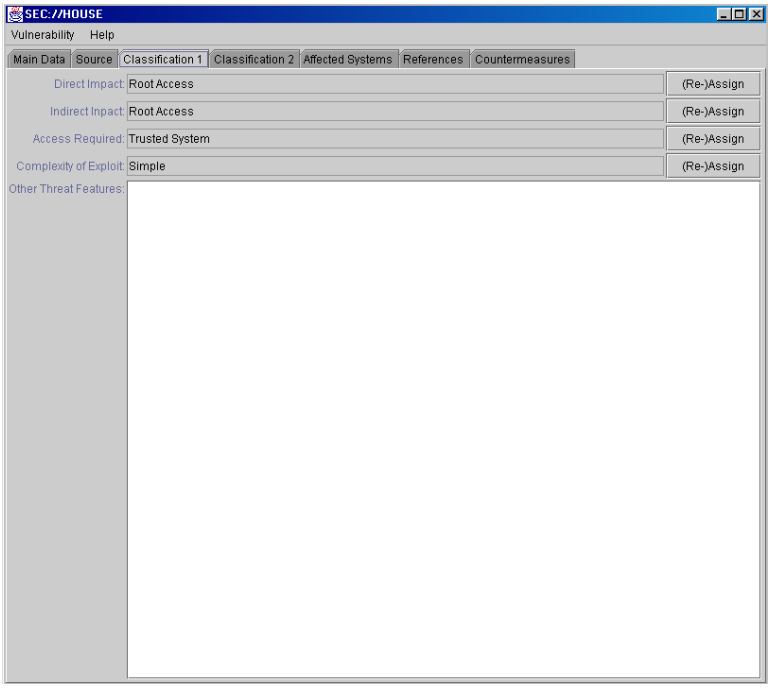

Abbildung 4.9: Karteikarte „Classification 1“

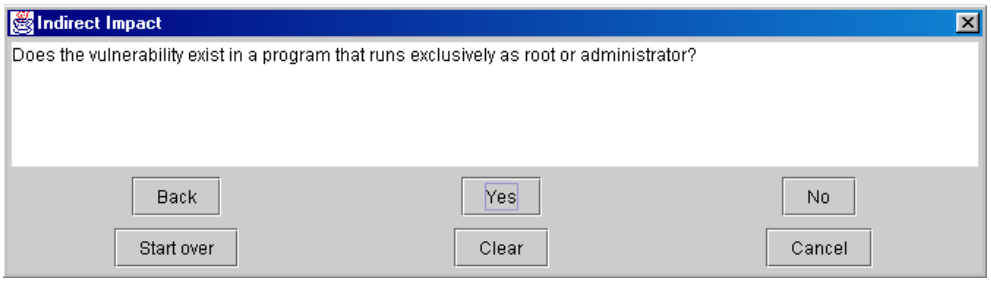

Abbildung 4.10: Auswahl eines Wertes für die Klassifikation aus einem Entscheidungsbaum

Am oberen Rand des Fensters werden die bereits festgelegten Kombinationen angezeigt. Durch Anklicken des entsprechenden Eintrags ist es möglich, den entsprechenden Datensatz zu laden. Außerdem kann ein Datensatz, nachdem er angewählt wurde, mit dem *Delete*-Knopf gelöscht werden.

Im mittleren Bereich des Fensters, der mit *Software* markiert ist, kann ein Hersteller, eine Software und eine entsprechende Version der Software ausgewählt werden, die von der Schwachstelle betroffen ist. Gleichzeitig können hier Hersteller, Software und Software-Versionen angelegt, bearbeitet oder auch gelöscht werden. Ist dies geschehen und wurden für die entsprechende Software-Version bereits Betriebssysteme und deren Versionen festgelegt, die von der entsprechenden Software-Version unterstützt werden, dann kann die entsprechende Betriebssystem-Version mit der Combo-Box für die Betriebssystem-Version ausgewählt werden.Sind noch keine oder noch nicht alle Betriebssystem-Versionen festgelegt worden, so kann dies über Auswahl des Knopfes *Edit list of Operating Systems this software runs on* geschehen.

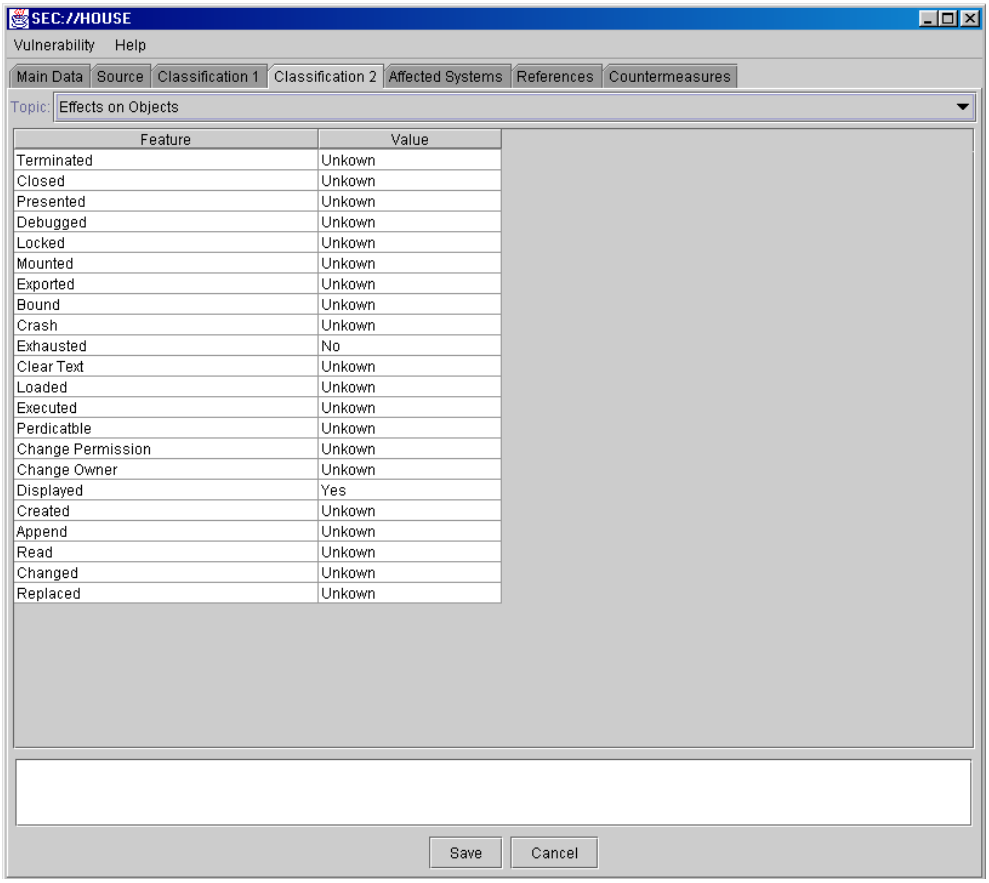

Abbildung 4.11: Karteikarte „Classification 2“

Wurde eine Betriebssystem-Version gewählt, so kann nun noch angegeben werden, ob die gewählte Kombination betroffen, nicht betroffen oder möglicherweise von der Schwachstelle betroffen sein könnte. Mittels *Add*-Knopf wird die neue Kombination zu der Schwachstelle gespeichert. Wird der Knopf zum Editieren der Betriebssystem-Versionen, auf denen eine Software-Version läuft, betätigt, so erscheint ein entsprechendes Dialogfenster (Abbildung 4.13). Die Funktionsweise ist dabei nahezu identisch mit der, für die Festlegung von betroffener Software.

Die Karteikarte **References** ermöglicht es, Referenzen, auf die man als ergänzende oder alternative Quelle zugreifen kann, anzugeben. Darunter können eMails, Webseiten oder beispielsweise auch Bücher fallen.

Die letzte Karteikarte **Countermeasures** erlaubt es, Gegenmaßnahmen für Schwachstellen zu erfassen. Dies können Hinweise auf Patches, neue Programmversionen oder auch Work-arounds sein. Für die Gegenmaßnahmen kann man wiederum angeben, auf welchen Software-Betriebsystem-Kombinationen die Gegenmaßnahme eingesetzt werden kann. Abbildung 4.14 zeigt den entsprechenden Eingabebildschirm.

Möchte der Nutzer statt eine neue Schwachstelle zu erfassen, nach einer bereits existierenden recherchieren, um diese beispielsweise zu editieren, so muß er im *Vulnerability-Menü* den *Menüpunkt Search* wählen. Es erscheint der Recherche-Dialog in dem der Nutzer die entsprechenden Recherche-Kriterien eingeben kann. Dies zeigt Abbildung 4.15. Nachdem

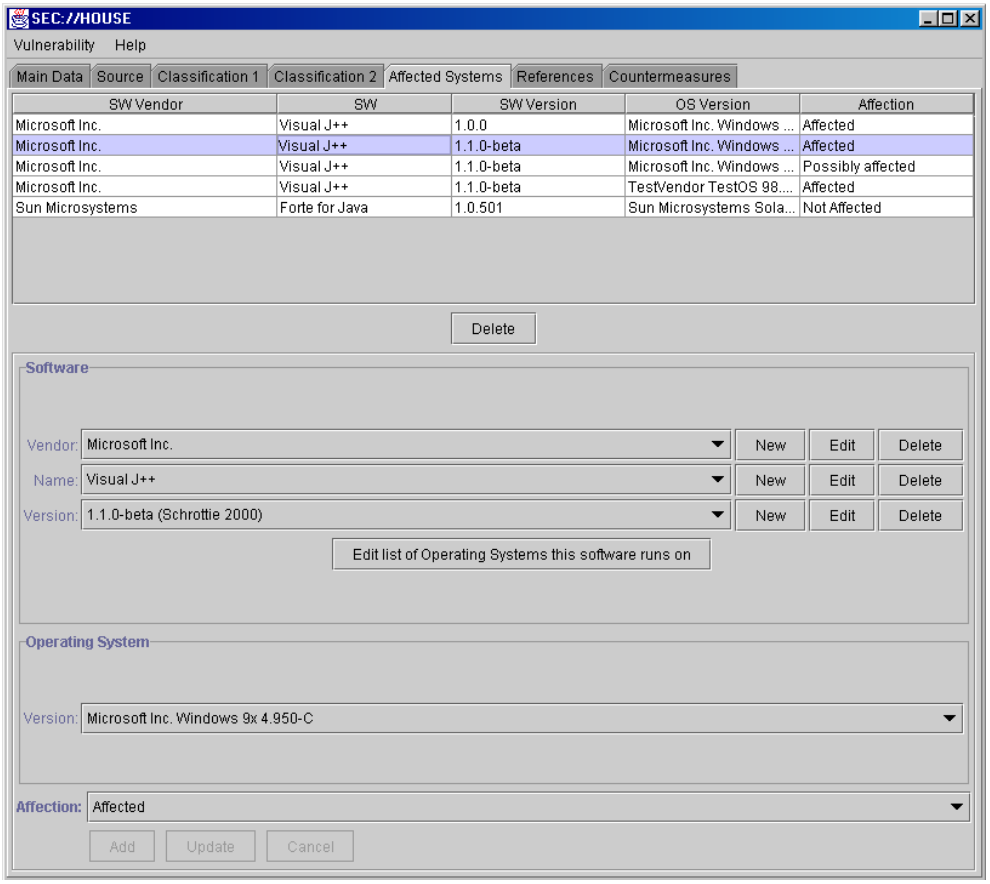

Abbildung 4.12: Karteikarte „Affected Systems“

die Recherche ausgeführt wurde, wird das Ergebnis in einem separaten Fenster angezeigt. Der Nutzer kann dann einen der gefundenen Datensätze anklicken, woraufhin dieser in das Hauptfenster des Admin-Tools geladen wird.

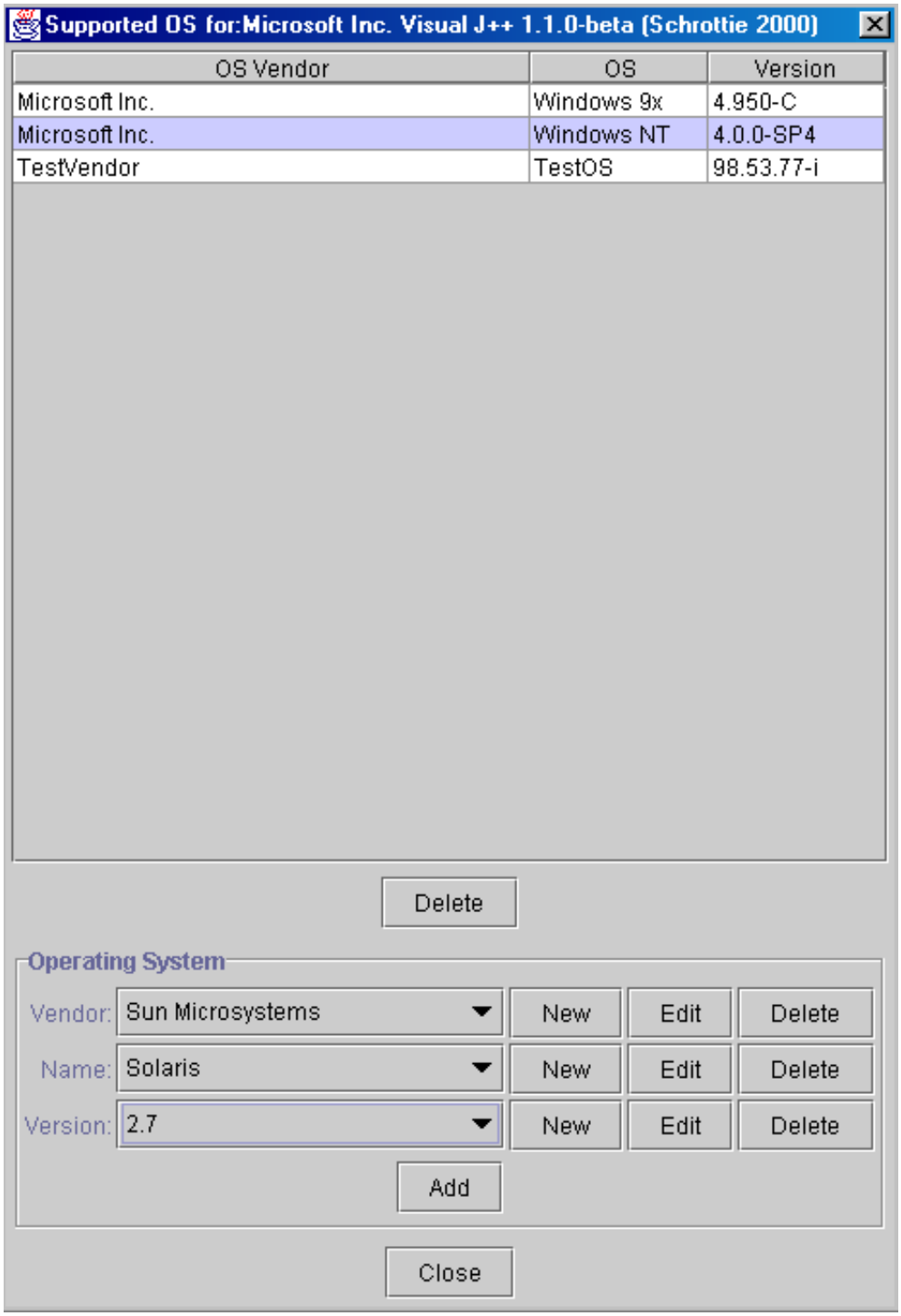

Abbildung 4.13: Dialogbox für Betriebssystem-Versionen

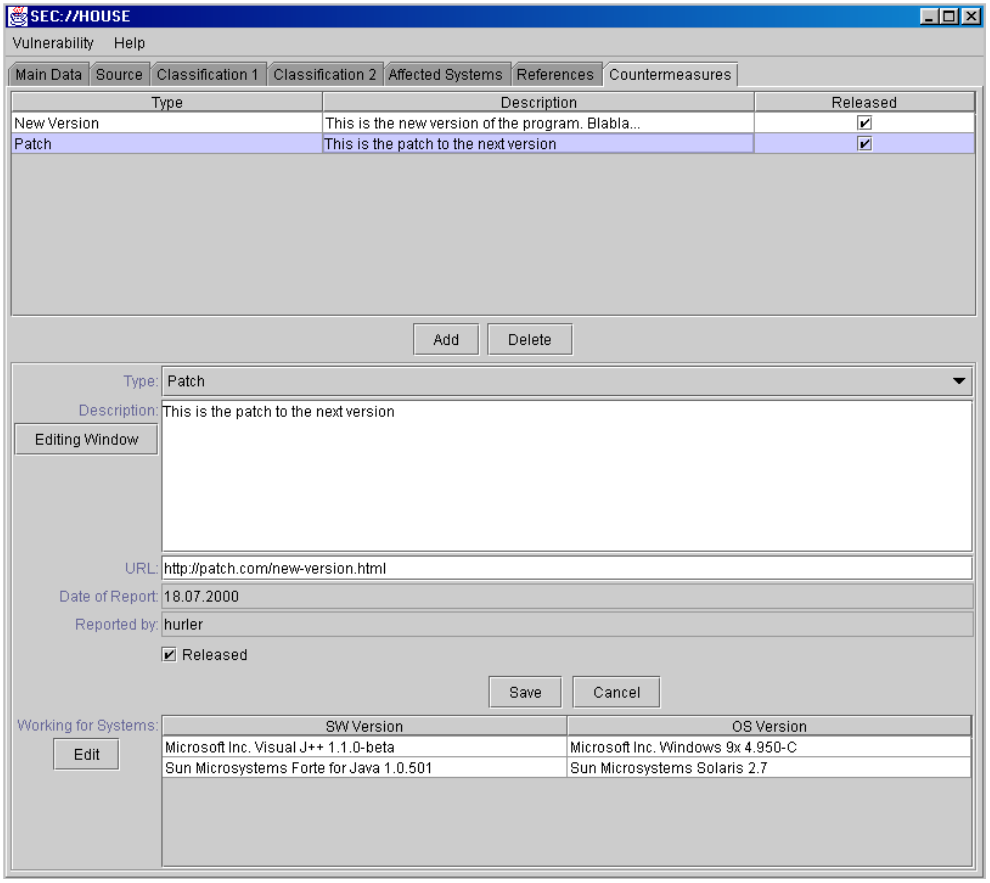

Abbildung 4.14: Karteikarte „Countermeasures“

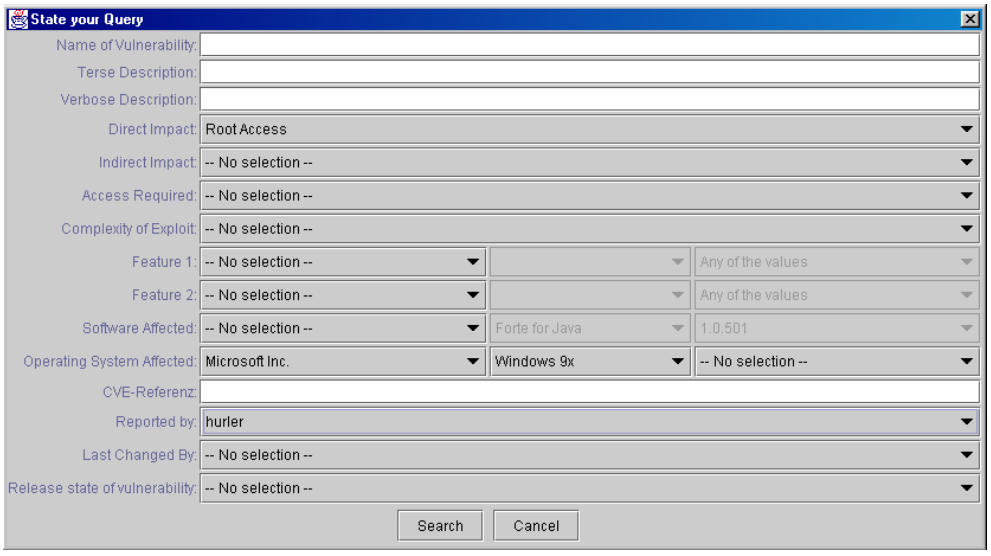

Abbildung 4.15: Recherchefenster des SEC://HOUSE-Prototypen

Kapitel 5

Die Zukunft von SEC://HOUSE

Schwachstellen stellen eine allgegenwärtige Bedrohung für die Sicherheit von Softwaresystemen dar. Es wurde dargestellt, daß viele verschiedene charakteristische Informationsquellen zu Schwachstellen existieren, die einen stark unterschiedlichen Strukturierungsgrad aufweisen.

Das Problem besteht darin, daß es keine öffentlich zugängliche, stark strukturierte und gut dokumentierte Schwachstellendatenbank gibt, die sich sowohl für den praktischen Einsatz in Unternehmen als auch für Forschungszwecke eignet.

Zwar stellt Krsuls SDB eine stark strukturierte Schwachstellendatenbank dar, die, abgesehen von einigen Schwächen, das notwendige Ziel einer systematischen Klassifizierung von Schwachstellen verfolgt. Jedoch liegt Krsuls SDB nur in Textform vor und ist nicht öffentlich zugänglich.

Schwachstellendatenbanken wie X-Force oder Bugtraq wiederum sind zwar öffentlich zugänglich, enthalten jedoch primär Informationen in Form von Text und erlauben dem Nutzer nur sehr einfache Recherchen. Dies war einer der Hauptgründe, die SEC://HOUSE-SDB zu entwickeln. Einerseits wird diese SDB auf Basis eines öffentlichen und gut dokumentierten Geschäftsmodells betrieben werden, was für mehr Transparenz bei den Nutzern und Teilnehmern der SDB sorgt. Andererseits wurden auch die Interessen von potentiellen Nutzern der SDB berücksichtigt. Und zusätzlich ist die SDB selbst gut dokumentiert, so daß auch Dritte sie für praktische oder forschungsbezogene Tätigkeiten einsetzen können.

Damit die SDB auch direkt eingesetzt werden kann, wurde der Prototyp einer Administrator-Schnittstelle entwickelt, der die Eingabe und Bearbeitung von strukturierten Schwachstellendaten ermöglicht.

Diese Diplomarbeit stellt jedoch erst den Anfang des SEC://HOUSE-Projekts dar. Es folgen weitere Schritte, bis die Schwachstellendatenbank ihren vollen Nutzen entfalten kann. Die nächste wichtige Aufgabe stellt die initiale Füllung der SDB dar. Dazu wird derzeit untersucht, inwieweit dieser Vorgang unter Nutzung aller zur Verfügung stehenden Quellen automatisiert werden kann, wie auch schon in Abschnitt 3.5 dargelegt wurde.

Weiterhin wird an der möglichst vollständigen Implementierung der in Abschnitt 3.4 erläuterten Zugangsmöglichkeiten zur SDB gearbeitet. Dies umfaßt außerdem die Erweiterung

des Prototypen der Administrator-Schnittstelle auf die vollständige, im Datenschema enthaltene Funktionalität, d.h. es muß unter anderem die Funktionalität für die Exploits, Incidents, Userverwaltung und Kommentare ergänzt werden. Möglicherweise wird auch eine Reimplementierung der Schnittstelle stattfinden, die auf der in Abschnitt 4.2 vorgeschlagenen Architektur basiert.

Einen anderen Forschungsschwerpunkt stellt die Entwicklung eines Systemanalysewerkzeugs dar, das, wie bereits in Abschnitt 4.1 erläutert, ein System, das von einem Vorfall betroffen war, analysiert und die entsprechenden Informationen über verwendete Software-Versionen und Konfigurationen in der SDB speichert. Gleichzeitig werden Analysewerkzeuge entwickelt, die bestehende Systeme auf das (potentielle) Vorhandensein schon bekannter Schwachstellen untersuchen und dabei ihre Informationen über Schwachstellen direkt aus der SDB beziehen. Ein Ziel ist dabei Intrusion Detection Systeme und Sicherheits-Auditing-Werkzeuge, wie beispielsweise Nessus, direkt an die SDB zu koppeln.

Außerdem wird nach geeigneten Data-Mining-Verfahren und -Werkzeuge geforscht, um Analysen auf dem Datenbestand der SEC://HOUSE-SDB durchführen zu können. In diesem Zusammenhang werden auch geeigneten Bedrohungs- und Risikomaße zu definieren sein, die eine Einordnung von Schwachstellen und Systemen anhand der Bedrohung, die von ihnen ausgeht, bzw. den Bedrohungen, denen sie ausgesetzt sind, erlauben.

Einen weiteren Schwerpunkt stellt die Untersuchung der rechtlichen Aspekte dar, die mit dem Betrieb der SDB zusammenhängen und auf die im Abschnitt 3.1.7 bereits kurz eingegangen wurde. Dies ist notwendig, um beispielsweise mögliche Schadensersatzforderungen von „Opfern“ von Angriffen ausschließen zu können.

Die möglichen Auswirkungen eines negativen Einsatzes von Informationen, die in der Schwachstellendatenbank gespeichert werden, definieren ein weiteres Forschungsgebiet. Werden solche Informationen systematisch und gezielt von politischen Gruppen oder staatlichen Organisationen eingesetzt, spricht man dabei heute i.d.R. von *Information Warfare*. Aber natürlich ist auch der mögliche Mißbrauch „im Kleinen“ durch Hacker und die sogenannten Script Kiddies zu untersuchen.

Schließlich ist es wichtig, das langfristige Ziel der Forschungsarbeiten von SEC://HOUSE nicht aus den Augen zu verlieren. Dieses Ziel läßt sich am besten mit den folgenden Fragen skizzieren:

- Wie ist es möglich, aus den sich ständig wiederholenden sicherheitsrelevanten Fehlern bei der Entwicklung von Software zu lernen und diese Fehler zu vermeiden?
- Wie ist es möglich aus der heutigen reaktiven Verhaltensweise der Nutzer von IT-Systemen zu einer proaktiven Verhaltensweise zu kommen, die das Entstehen von Schwachstellen zu vermeiden sucht?

Dabei gilt ein besonderes Interesse den Fehlern beim Design von Software, da diese nur schwer und häufig gar nicht beseitigt werden können. Jedoch dürfen auch die Implementierungsfehler nicht außer acht gelassen werden, denn viel zu häufig stellt heute beispielsweise der profane „Buffer Overflow“ die Sicherheit von IT-Systemen in Frage. Langfristig

muß das Ziel in der Unterstützung aller Phases des Softwareentwicklungsprozesses bestehen. Dabei ist es notwendig, das Design von sicherer Software, beispielsweise durch entsprechende Werkzeuge oder Design Patterns, zu unterstützen. Ebenso muß die Implementierungsphase in die Betrachtung miteinbezogen werden. Denkbar ist hier einerseits die direkte Unterstützung der Entwickler im Rahmen von Anwendungsentwicklungsumgebungen oder auch die Zurverfügungstellung von speziellen Bibliotheken, die sichere Unterprogramme für fehleranfällige Routinen bereitstellen. Und schließlich muß untersucht werden, inwiefern die Phasen Test, Systemintegration und Konfiguration durch Methoden unterstützt werden können, die es ermöglichen Schwachstellen zu entdecken bzw. deren Auftreten zu vermeiden.

Die Verfügbarkeit von Methoden zur Erstellung sicherer Softwaresysteme in allen Bereichen der Softwareentwicklung und -nutzung ist eine grundlegende Voraussetzung für ein höheres Sicherheitsniveau. Damit dieses auch tatsächlich realisiert werden kann, ist es jedoch notwendig, den Einsatz der Methoden in die Richtlinien für die Softwareentwicklung aufzunehmen und deren Einhaltung durch die Entwickler auch sicherzustellen.

Anhang A

Ergänzendes Material

A.1 Detaillierte Analyse der Umfrage

A.1.1 Ergebnis Section 1: Business model of the VDB

Abschnitt 1 des Fragebogens enthielt Fragen bezüglich des Organisationsmodells der SDB.[1]

Frage 1.1

Would you be willing to use a strictly centralized VDB which contains all available information on vulnerabilities?

Zur Auswahl standen die Optionen *Yes* und *No*.

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.1. Die Antwort zeigt relativ eindeutig, daß der größte Teil der Teilnehmer keine Bedenken bezüglich der Nutzung einer stark zentralistischen SDB hätte. Es liegt jedoch ein deutlicher Unterschied zwischen den Antworten der Computer-related-Nutzer und der übrigen Nutzern vor. Während sich 95% der Computer-related-Nutzer zur Nutzung einer zentralisierten SDB bereit erklärten, sind es aus den anderen Gruppe nur ca. 63%.

Frage 1.2

Would you prefer a “federated style” VDB that is split in specialized instances by topic such as “operating system” but still offers some kind of coordination (e.g. regarding the database schema or some kind of publication policy)?

Zur Auswahl standen wiederum die Optionen *Yes* und *No*.

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.2. Wie man sieht, war die Zustimmung für das föderierte Modell noch größer, als für das zentralisierte. Hierbei fielen keine signifikanten Unterschiede zwischen den drei Gruppen der Computer-related-Nutzer, der „Forscher“ und der übrigen Befragten auf.

[1] Bedauerlicherweise wurde der Titel des Abschnitts falsch gewählt.

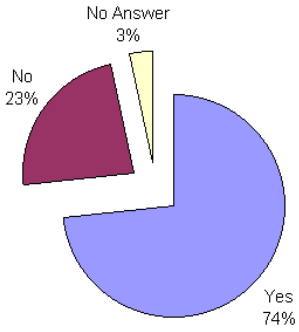

Abbildung A.1: Antworten zu Frage 1.1 „Zentralisierte SDB“

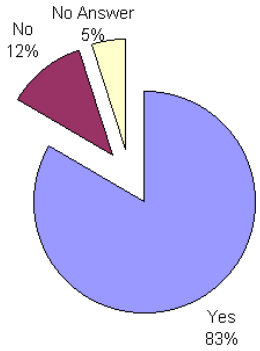

Abbildung A.2: Antworten zu Frage 1.2 „Föderierte SDB“

Frage 1.3

Do you think such a specialization would be useful / helpful?

Zur Auswahl standen auch hier die Optionen *Yes* und *No*. Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.3. Die Zustimmung von 90% für eine Spezialisierung der Datenbank war sehr groß. Zwischen den Teilnehmergruppen ergaben sich keine signifikanten Unterschiede.

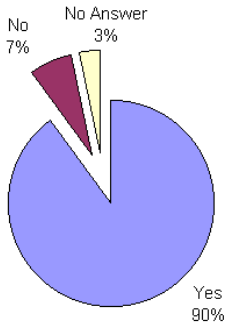

Abbildung A.3: Antworten zu Frage 1.3 „Spezialisierung“

Frage 1.4

Would you use an “open data” style (similar to open source) VDB? In brief “open data” means that the database including its content is available to anyone and that new data has to be provided by the community.

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.4. Offensichtlich fand der Gedanke ein offenes und frei verfügbares Modell zu implementieren große Zustimmung. Erstaunlicherweise war der Anteil unter den Befragten aus dem Bereich Computer-related am höchsten, denn hier stimmten 100% für diese Lösung, gegenüber 85% beim Bereich Education / Research und 79% bei den übrigen Befragten. Offensichtlich hat sich bei den Computer-related-Nutzern das „Open Source - GNU GPL“-Paradigma schon etabliert, während es selbst im Forschungs-Bereich eine höhere Skepsis gegenüber diesem Modell zu geben scheint.

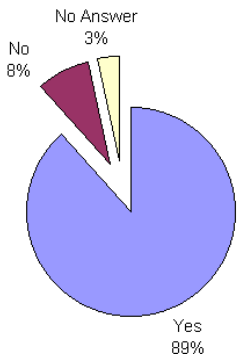

Abbildung A.4: Antworten zu Frage 1.4 „Open Data-SDB“

Frage 1.5

Do you think the consequence of an “open data” VDB would be a greater risk of abuse or (intentional) “pollution” of the information stored in the VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.5. Während 60% der Teilnehmer der Auffassung waren, daß mit dem *Open Data*-Ansatz kein erhöhtes Risiko für eine stärkere „Verschmutzung“ der Daten durch zufälliges oder vorsätzliches Einspeisen falscher Daten einhergeht, war es hier doch überraschend, daß von den Teilnehmern der Gruppe *Education / Research* 53,85% eine solche Möglichkeit bejahten, während aus der Gruppe *Computer-related* nur 25% diese Gefahr sahen. An dieser Stelle hätte man ein umgekehrtes Ergebnis erwarten können, wenn man ein verstärktes Mißtrauen kommerzieller IT-Nutzer gegenüber offenen Lösungen unterstellt hätte.

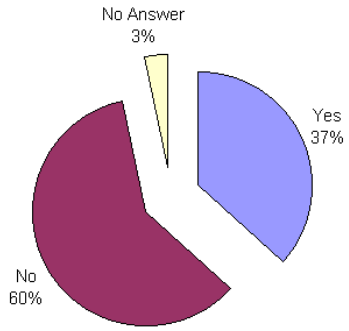

Abbildung A.5: Antworten zu Frage 1.5 „Verunreinigung der Daten bei Open Data“

Frage 1.6

Would you be willing to work voluntarily for the VDB on maintenance and enhancement tasks?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.6. Immerhin 40% der Teilnehmer sagten zu dieser Frage, daß sie bereit wären, an Arbeiten an der SDB mitzuwirken. Dabei gab es keine auffälligen Abweichungen zwischen den Branchen der Teilnehmer.

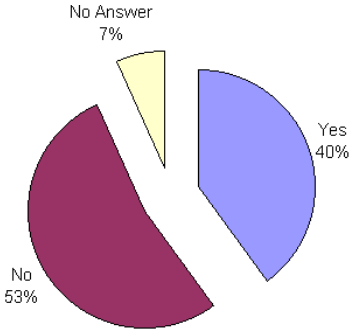

Abbildung A.6: Antworten zu Frage 1.6 „Beteiligung an Erweiterungsarbeiten“

Frage 1.7

Do you think anonymization of contributions in such a way, that there is no way to draw a conclusion back to the contributor, is essential for the acceptance of a public VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.7. Insgesamt ergibt sich bei dieser Frage kein eindeutiges Bild. 50% sagten, daß Anonymisierung der Daten nicht zwingend notwendig ist, 45% sprachen sich dagegen für eine Anonymisierung aus. Interessant ist allerdings, daß nur 25% der Teilnehmer aus dem *Computer-releated*-Bereich sich für eine Anonymisierung aussprachen, aus dem Bereich *Ausbildung / Forschung* dagegen, sprachen sich 61,5% dafür aus; bei den restlichen Teilnehmern waren es 42,9 %.

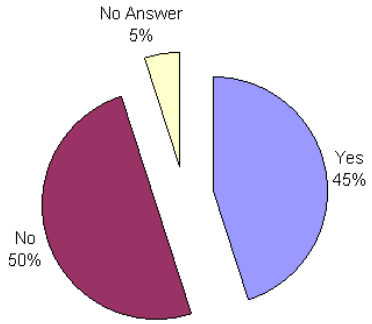

Abbildung A.7: Antworten zu Frage 1.7 „Anonymisierung“

Frage 1.8

In a database federation we can think of central databases for general use and additional instances for private use, e.g. vulnerability information about proprietary or private software. Do you think, this would make such a VDB more attractive?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.8. Insgesamt 69% aller Teilnehmer sprachen sich hier für die Unterstützung privater Instanzen aus. Allerdings ist auch hier ein Unterschied zwischen den „Forschern“ und den restlichen Teilnehmern zu bemerken. Die Forscher sprachen sich zu 81% dafür aus, während bei den restlichen Teilnehmern nur ca. 60% dafür stimmten.

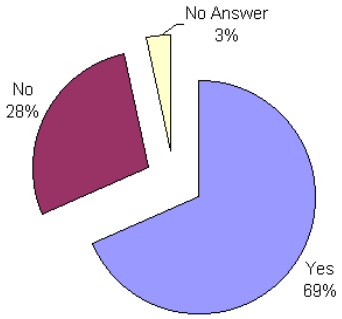

Abbildung A.8: Antworten zu Frage 1.8 „Private Instanzen“

Frage 1.9

Today’s knowledge on vulnerabilities is spread over countless independent sources of information, e.g. newsgroups, mailing lists and lots of existing vulnerability databases. In contrast to this “balkanized” environment think of a database federation with members that synchronize the vulnerability information among each other. Do you think, this would be a more efficient solution?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.9. Die Antwort auf diese Frage war einhellig. Von allen Teilnehmern beantworteten 92% die Frage mit Ja. Bei den einzelnen Gruppen gab es keine auffälligen Unterschiede.

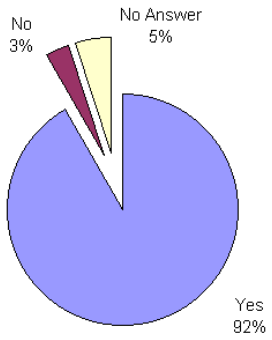

Abbildung A.9: Antworten zu Frage 1.9 „Effizienz einer Datenbankföderation“

Frage 1.10

Do you think a homogeneous database schema across VDBs could be helpful for the analysis of vulnerability information?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.10. Auch bei dieser Frage zeigt sich ein eindeutiges Bild: 88% der Teilnehmer bejahten die Frage. Der computerbezogene Bereich und der Forschungsbereich urteilten mit 95% bzw. 92,3% für die Vorteile eines homogenen Schemas, während die restlichen Teilnehmer darin nur zu 71% Vorteile erkennen konnten.

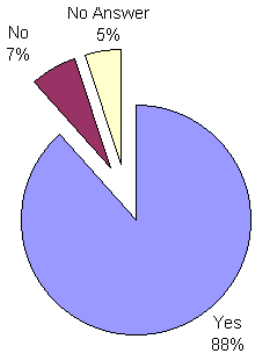

Abbildung A.10: Antworten zu Frage 1.10 „Homogenes Datenbankschema“

A.1.2 Ergebnis Section 2: Use of / Access to the VDB

Abschnitt 2 des Fragebogens enthielt Fragen bezüglich der Nutzung und des Zugriffs auf die SDB.

Frage 2.1

Do you currently use any sources to get more information about software vulnerabilities?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.11. 92% aller Teilnehmer nutzten zun Zeitpunkt der Umfrage Quellen über Schwachstellen. Teilnehmer der Gruppe Computer-related nutzten solche Quellen zu 95%, Forscher zu 92,3% und die restlichen Teilnehmer zu 85,7%.

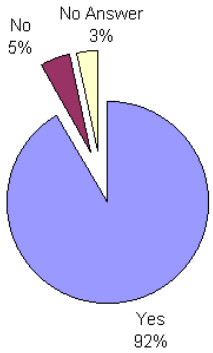

Abbildung A.11: Antworten zu Frage 2.1 „Nutzung von Quellen über Schwachstellen“

Frage 2.1.1

Do you use existing VDBs?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.12. Insgesamt 67% aller Teilnehmer nutzten SDBs. Allerdings lag der Anteil in der Gruppe Computer-related bei 85%, während bei den restlichen Teilnehmern der Anteil nur bei ca. 57,5% lag.

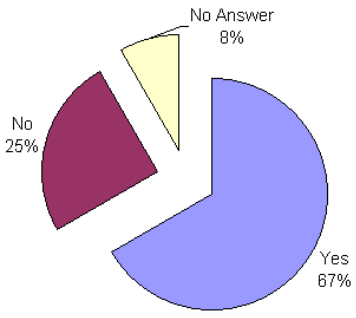

Abbildung A.12: Antworten zu Frage 2.1.1 „Nutzung von SDBs“

Frage 2.1.2

Do you use newsgroups?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.13. 69% der Teilnehmer nutzten zum Zeitpunkt der Umfrage Newsgroups als Quelle von Informationen zu Softwareschwachstellen. Teilnehmer der Gruppe Computer-related nutzten diese Quelle zu 70%, Forscher zu 61,5% und von den restlichen Teilnehmern nutzten 78,5% das USENET. Interessanterweise zeigt sich hier, entgegen den bisherigen Fällen, ein deutlicher Unterschied zwischen den beiden Teilgruppen der Gruppe Computer-related. Die Teilnehmer, die im Software-Bereich arbeiten, nutzten Newsgroups nur zu 42%, während die Teilnehmer der Gruppe IS/MIS/DP/Internet dieses Medium zu 84% nutzten.

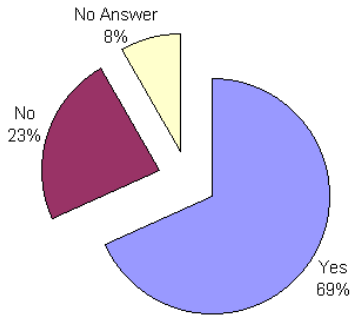

Abbildung A.13: Antworten zu Frage 2.1.2 „Nutzung von Newsgroups“

Frage 2.1.3

Do you use mailing lists like Bugtraq or NTBugtraq?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.14. Von allen Teilnehmern nutzten 83% Mailing-Listen. Bei den Computer-related-Nutzern zeigte sich auch hier wieder ein hoher Wert von 90%. Die restlichen Teilnehmer lagen „nur“ bei ungefähr 79%.

Frage 2.1.4

Do you or your employer maintain your own private VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.15. Nur bei 20% der Teilnehmer unterhielt der Arbeitgeber zum Zeitpunkt der Umfrage eine private SDB. Bei 68% der Teilnehmer war dies nicht der Fall. Allerdings ist hier die Aufgliederung interessant. Die Gruppe Computer-related nutzte zu 30% private SDBs. Allerdings waren es im Bereich Software 57%, während es im Bereich IS/MIS/DP/Internet nur 15% waren. Die restlichen Teilnehmer nutzten private SDBs nur zu 15%.

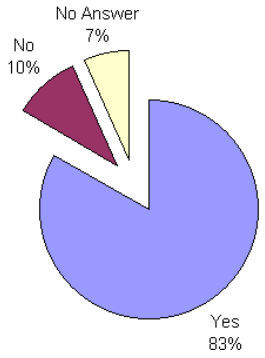

Abbildung A.14: Antworten zu Frage 2.1.3 „Nutzung von Mailing-Listen“

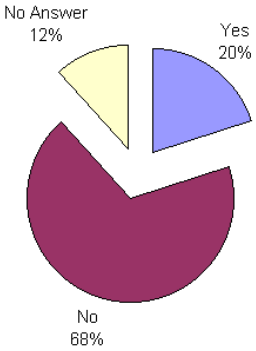

Abbildung A.15: Antworten zu Frage 2.1.4 „Nutzung einer privaten SDB“

Frage 2.1.5

More than one source in any category?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.16. Insgesamt nutzten 72% aller Teilnehmer mehr als eine Quelle in einem der oben angesprochenen Bereiche. Insgesamt lagen alle Gruppen ungefähr bei 70%. Nur die Computer-related-Nutzer aus dem Bereich Software stachen hier mit 86% heraus.

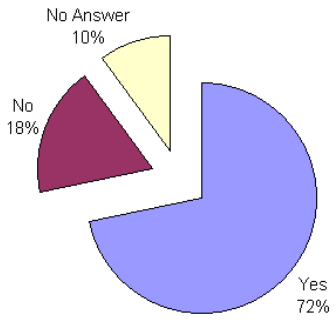

Abbildung A.16: Antworten zu Frage 2.1.5 „Nutzung mehr als einer Quelle eines Typs“

Frage 2.2

Are you a job-related user of a VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.17. 57% der Teilnehmer beantworteten diese Frage mit Ja, 38% mit Nein. Interessant, wenn auch nicht unbedingt überraschend, ist wiederum die Aufteilung zwischen den Gruppen. Die Gruppe der Forscher und der „Anderen“ lag hier jeweils bei ca. 48%. Dagegen lag der Anteil der Teilnehmer des Bereichs Computer-related bei 75%, wobei jedoch hier der Bereich Software nur mit 57% beteiligt war, während der Bereich IS/MIS/DP/Internet mit 85% beteiligt war.

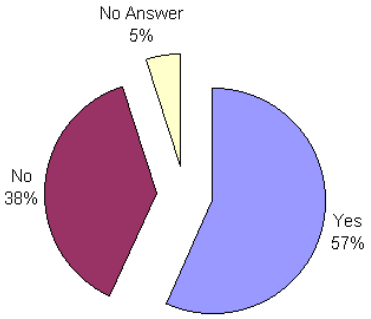

Abbildung A.17: Antworten zu Frage 2.2 „Berufliche Nutzung einer SDB“

Frage 2.3

Do you currently participate in at least one existing VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.18. Nur 27% beantworteten diese Frage mit Ja, 66% dagegen mit Nein. Allerdings teilt sich hier wieder das Bild zwischen den Gruppen: aus dem Bereich Computer-related beteiligten sich 45% an den Diskussionen und Beiträgen mindestens einer SDB, während es im Bereich Forschung nur 19% waren und bei den anderen Teilnehmern sogar nur 14%.

Frage 2.3.1

In more than one?

Mögliche Antworten: *Yes* und *No*

Auf diese Frage antworteten nur noch 18% mit Ja, 65% dagegen mit Nein. Die Veränderung ist insbesondere auf einen, gegenüber Frage 2.3 um zehn Prozentpunkte höheren Anteil von Personen zurückzuführen, die keine Antwort gaben. Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.19.

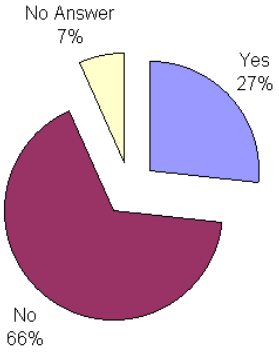

Abbildung A.18: Antworten zu Frage 2.3 „Mitarbeit an einer SDB“

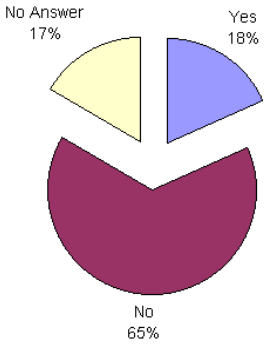

Abbildung A.19: Antworten zu Frage 2.3.1 „Mitarbeit an mehr als einer SDB“

Frage 2.4

What kind of access policy would you prefer for read access to the VDB?

Mögliche Antworten:

- Anonymous
- Individualized (via pseudonyms)
- Personalized (real name known only to VDB administration)
- Personalized (real name known to anyone)

Bei diese Frage ergab sich ein Mehrheit von 58% für einen anonymen Lesezugriff. Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.20.

Interessant ist hier die Verteilung in den einzelnen Gruppen. Während bei den Gruppen Computer-Related und Education / Research 60% bzw. 65% für einen anonymen lesenden Zugriff stimmten, waren dies bei den restlichen Teilnehmern nur ungefähr 35%. Dafür war dort der Anteil der Teilnehmer, die für einen individualisierten Zugriff stimmten, mit 21% mit Abstand am höchsten.

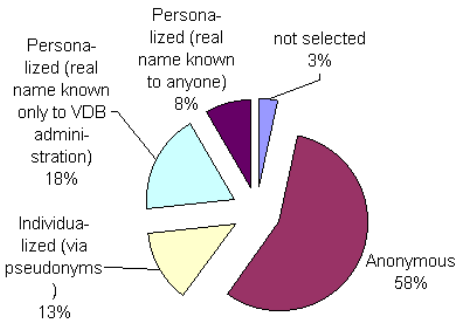

Abbildung A.20: Antworten zu Frage 2.4 „Richtlinien für lesenden Zugriff“

Frage 2.5

What kind of access policy would you prefer for write access to the VDB?

Mögliche Antworten:

- Anonymous
- Individualized (via pseudonyms)
- Personalized (real name known only to VDB administration)
- Personalized (real name known to anyone)

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.21. Man sieht, daß insgesamt alle Optionen ähnlich preferiert wurden. Allerdings ist auffällig, daß die Gruppe Education / Research den personalisierten Schreibzugriff, bei dem die „SDB Administration“ den tatsächlichen Namen des Nutzers kennt, bevorzugte, während die restlichen Teilnehmer den personalisierten Zugriff, bei dem jeder die Identität der Teilnehmer kennt, bevorzugten. Insgesamt waren die Unterschiede aber nicht gravierend.

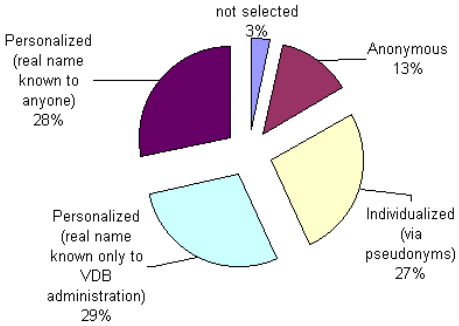

Abbildung A.21: Antworten zu Frage 2.5 „Richtlinien für schreibenden Zugriff“

A.1.3 Ergebnis Section 3: Publication Policy

Abschnitt 3 des Fragebogens enthielt Fragen bezüglich der Veröffentlichungspolitik von Informationen zu Schwachstellen.

Frage 3.1

In case of the discovery of a new vulnerability we have identified three publication policies:

1. Immediate publication of the vulnerability report together with all information (e.g. exact attack / vulnerability description, exploit scripts, etc) known about it.
2. After getting a report about a vulnerability, the manufacturer of the affected system will be contacted and is given a “grace period” to fix the problem. After that grace period the complete report gets published, whether a patch exists or not.
3. A multi level publication policy is used. Immediately after the first notification about a problem, a first report is published with all necessary information that is needed to temporarily fix the problem. At the same time the vendor of the affected system gets informed about the problem. After a “grace period” of some time the complete report is published, whether a patch exists or not.

Please consider that the policies 2 and 3 can’t be realized in a “newsgroup like” environment!

Which of these three general policies would you prefer?

Mögliche Antworten: *immediate*, *two level* und *multi level*.

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.22. 62% aller Teilnehmer der Umfrage sprachen sich für die mehrstufige Veröffentlichungspolitik aus. Zwischen den Gruppen gab es keine größeren Unterschiede.

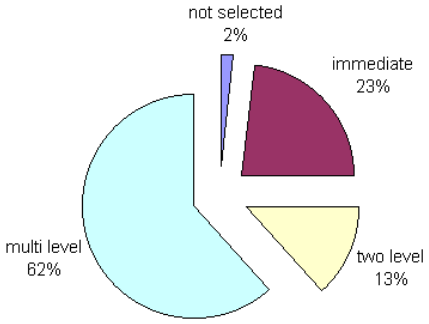

Abbildung A.22: Antworten zu Frage 3.1 „Veröffentlichungspolitik“

Frage 3.1.1

If you chose policy 2 or 3. What “grace period” would be appropriate?

Mögliche Antworten waren: *1,3 oder 5 days* und *1, 2, 3, 4, 5, 6, 8, 10, 12 und >12 weeks*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.23. Bedauerlicherweise wählten auch Teilnehmer, die bei Frage 3.1 eine unmittelbare Veröffentlichung der Informationen einer Schwachstelle gefordert haben, bei dieser Frage einen Wert aus, obwohl dies eigentlich keinen Sinn ergab. Nichtsdestotrotz kann man erkennen, daß eine Mehrheit für eine „Grace Period“ von ein bis zwei Wochen stimmte.

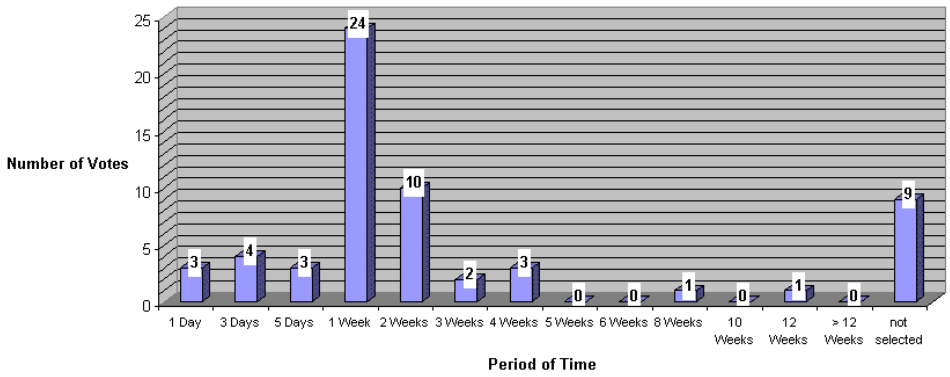

Abbildung A.23: Antworten zu Frage 3.1.1 „Grace Period“

Frage 3.1.2

Do you think that the grace period should depend on some kind of risk category, e.g. high risk, medium risk, low risk, no risk?

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.24. 67% der Teilnehmer sprachen sich dabei für eine „Grace Period“ aus, die vom Risiko, das von der Schwachstelle ausgeht, abhängt. Einen besonders hohen Anteil von Befürwortern für dieses Vorgehen gab es bei der Gruppe der Forscher mit 77%, während die computerbezogene Gruppe dies nur mit 55% befürwortete.

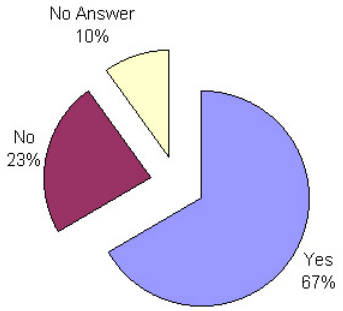

Abbildung A.24: Antworten zu Frage 3.1.2 „Grace Period in Abhängigkeit des Risikos“

Frage 3.2

Would you prefer to get the most extensive information possible about a vulnerability, including detailed exploit information and maybe exploit scripts?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.25. 88% der Teilnehmer sprachen sich für eine möglichst umfassende Information über Schwachstellen aus. Den höchsten Anteil gab es in der Gruppe Computer-related mit 95%, während der Bereich Forschung sich mit 88% dafür aussprach und die übrigen Teilnehmer „nur“ mit 79%.

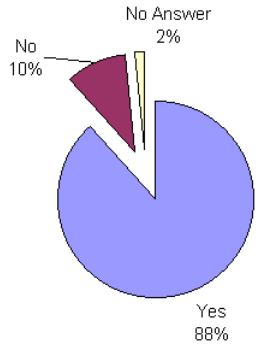

Abbildung A.25: Antworten zu Frage 3.2 „Größtmöglicher Umfang der Informationen“

A.1.4 Ergebnis Section 4: Quality Assurance

Abschnitt 4 des Fragebogens enthielt Fragen bezüglich der Möglichkeiten, die Qualität der Daten in der SDB möglichst hoch zu halten.

Frage 4.1

Think of a “privileged person or group of persons” steering discussions about contributions to the VDB. This “moderator” would have the ability to influence or stop the discussion at a certain point. Would you use this kind of a moderated VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.26. Von den Teilnehmern antworteten hier 78%, daß sie eine von einem Moderator gesteuerte SDB nutzen würden. Dabei lag die Zustimmung bei der Gruppe der Computer-related-Nutzer und bei der Gruppe der Forscher bei 85%. Die übrigen Nutzer sprachen sich allerdings nur zu 57% für ein solches Modell aus.

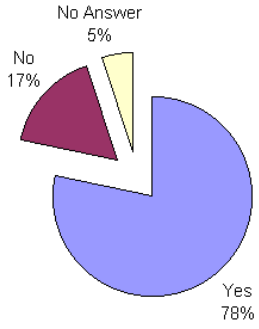

Abbildung A.26: Antworten zu Frage 4.1 „Mitlesender Moderator“

Frage 4.2

Think of a “privileged person or group of persons” steering the discussion about contributions to the VDB. This “moderator” would have the “right” to decide wether or not the contribution will be incorporated in the VDB. Would you use this kind of a moderated VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.27. Für die Nutzung einer SDB, bei der Beiträge nur nach einer „Überprüfung und Bewertung“ durch den Moderator freigegeben werden, erklärten sich immerhin noch 67% der Teilnehmer bereit. Aus der Gruppe der Forscher erklärten sich auch hier immer noch 85% bereit, eine solche SDB zu nutzen. Allerding sank die Zustimmung bei den Computer-related-Nutzern auf 55% und die übrigen Teilnehmer erklärten sich nur noch zu 50% bereit, eine solche SDB zu nutzen.

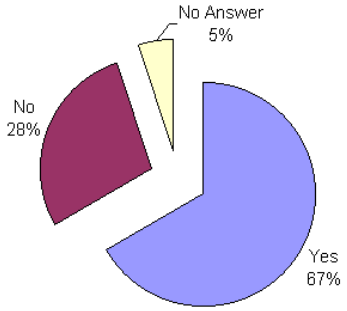

Abbildung A.27: Antworten zu Frage 4.2 „Freigebender Moderator“

Frage 4.3

Would you be concerned of censorship, if the VDB was moderated?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.28. Es zeigt sich ein zwiespältiges Bild. Von den Teilnehmern würden sich 50% Sorgen bezüglich einer Zensur machen, wenn es einen Moderator gäbe, der Beiträge zurückhalten könnte. 42% teilten solche Befürchtungen nicht. Unterschiede gab es insbesondere zwischen den Forschern und den restlichen Teilnehmern. Bei den Forschern befürchteten nur 42% eine Zensur, während bei allen anderen Teilnehmer 56% eine Zensur fürchteten.

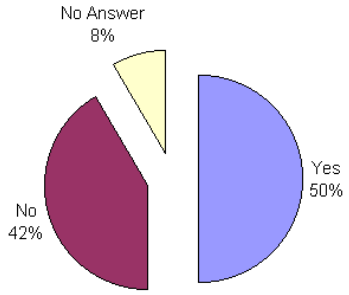

Abbildung A.28: Antworten zu Frage 4.3 „Befürchtungen bezüglich Zensur“

Frage 4.4

How could trust in vulnerability reports be established?

Mögliche Antworten:

- By a central rating committee selected by the VDB administration
- By a central rating committee selected by volunteers that are somehow elected
- By a web of trust like architecture where anyone can rate statements from other people

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.29. Insgesamt ergibt sich hier keine allgemeine Präferenz. Eine knappe Mehrheit ergab sich bei den Computer-related-Nutzern, die sich zu 55% für eine Lösung mittels eines *Web of trust* aussprachen. Bei der Gruppe Education / Research ergab sich schon kein klares Bild mehr: 42% sprachen sich für eine gewählte Gruppe von Freiwilligen aus, die eine Bewertung von Beiträgen durchführt. 35% sprachen sich für ein Web of Trust aus und 23% für eine Bewertung durch die Betreiber der SDB. Bei den anderen Teilnehmern ergab sich ein noch unklareres Bild. 37% sprachen sich für eine gewähltes Bewertungteam aus, ansonsten wurden die anderen Optionen identisch bewertet, insbesondere fällt auch auf, daß 21% der „anderen“ Teilnehmer keine Aussage machten.

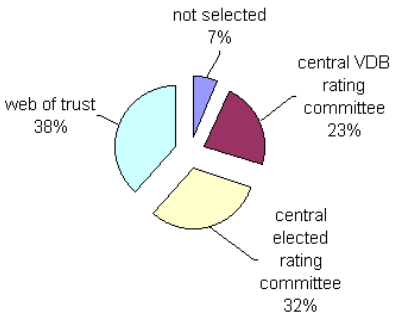

Abbildung A.29: Antworten zu Frage 4.4 „Vertrauensbildung“

Frage 4.5

What should be rated?

Mögliche Antworten:

- Only the original posting
- The original report and some of the immediate comments to it

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.30. Es ergibt sich ein eindeutiges Bild. 81% aller Teilnehmer sprachen sich dafür aus, sowohl die originale Meldung als auch eine näher zu bestimmende Anzahl von Kommentaren zu diesem Beitrag zu bewerten.

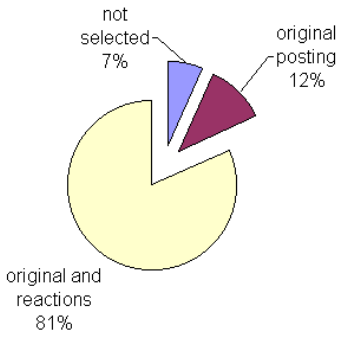

Abbildung A.30: Antworten zu Frage 4.5 „Bewertung von Beiträgen“

Frage 4.6

Do you think it would be a good idea to regularly elect qualified volunteers for some time to assess the contributions to the VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.31. Auch auf diese Frage ergab sich ein relativ eindeutiges Bild. 77% sprachen sich dafür aus, die Mitglieder des Bewertungsteams regelmäßig zu wählen. Auffällig ist, daß die Gruppen Computer-related und Education / Research sich zu jeweils 80% für die regelmäßige Wahl aussprachen, während sich von den restlichen Teilnehmern nur 64% dafür aussprachen.

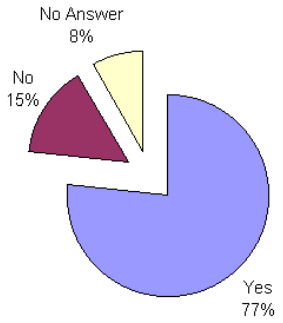

Abbildung A.31: Antworten zu Frage 4.6 „Regelmäßige Wahl eines Teams für Bewertungen“

Frage 4.7

Would you be willing to volunteer your time to do this job?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.32. Immerhin 28% der Teilnehmer erklärten sich bereit, Zeit für diese wichtige Arbeit zur Verfügung zu stellen. 60% erklärten, daß sie dies nicht tun könnten bzw. wollten. Auffällig ist, daß sich die Teilnehmer aus der Gruppe der Forscher mit über 38% bereit erklärten, diese Tätigkeit auszuüben, während es aus den übrigen beiden Gruppen nur jeweils ca. 20% waren.

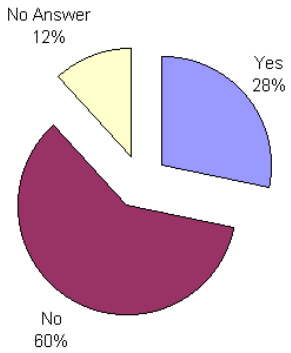

Abbildung A.32: Antworten zu Frage 4.7 „Bereitschaft am Bewertungsteam mitzuarbeiten“

A.1.5 Ergebnis Section 5: Financing

Abschnitt 5 des Fragebogens enthielt Fragen dazu, wie die Finanzierung der SDB sichergestellt werden könnte und zwar insbesondere so, daß die Nutzer keine Abhängigkeiten zwischen der SDB-Administration und den Geldgebern befürchten.

Frage 5.1

Would you, as an individual, be willing to pay a monthly fee for the usage of the VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.33. Es ergibt sich hier das relativ eindeutige Bild, daß private Nutzer kaum bereit waren, für die Nutzung einer SDB zu bezahlen (75%). Insbesondere der Anteil von Nutzern der Gruppe Computer-related war mit nur 5% für die Zahlung einer Nutzungsgebühr durch Individuen sehr niedrig. Dieser Anteil war bei den Forschern mit 30% deutlich höher und lag bei den anderen Teilnehmern immerhin noch bei 21%.

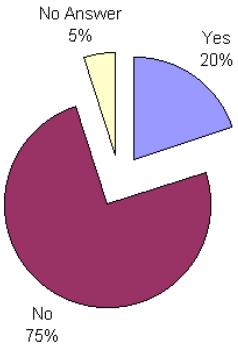

Abbildung A.33: Antworten zu Frage 5.1 „Monatlicher Beitrag pro Nutzer“

Frage 5.2

What amount of money would you be willing to pay per month?

Mögliche Antworten: *0, 1, 5, 10, 20, 30, 50, 100 und >100 Euro / US-$*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.34. Entsprechend des Ergebnisses aus Frage 5.1 ergibt sich das deutliche Bild, daß die meisten Teilnehmer der Umfrage nicht bereit waren, für die Nutzung der SDB zu zahlen. Die Teilnehmer, die bereit waren für die Nutzung einen Kostenbeitrag zu übernehmen, waren größtenteils bereit zwischen 1 und 10 Euro / US-$ pro Monat zu zahlen.

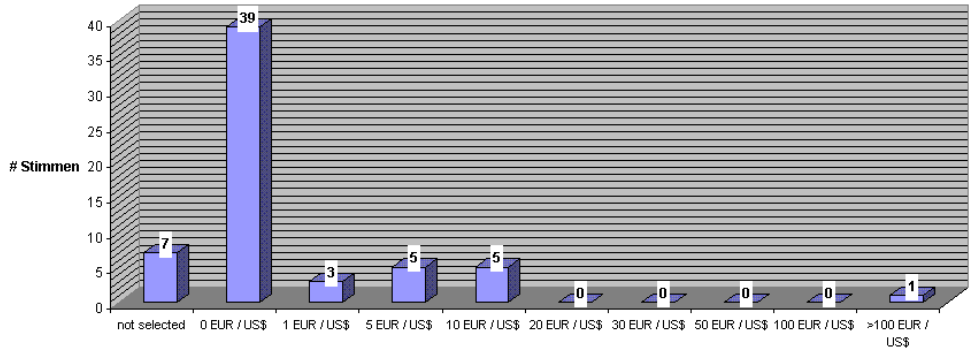

Abbildung A.34: Antworten zu Frage 5.2 „Höhe des monatlichen Beitrags“

Frage 5.3

If you are an employee of a company for which you would use the VDB, do you think your company would be willing to pay a monthly fee for using the VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.35. 15% der Teilnehmer äußerten, daß die Frage auf sie nicht anwendbar wäre. 60% bejahten die Frage und nur 12% gaben eine negative Antwort. Es fällt auf, daß aus der Gruppe der Forscher nur 46% auf diese Frage mit Ja antworteten, während es bei den restlichen zwei Gruppen jeweils 70% waren. Dieser unterschiedliche Anteil ist vermutlich auf die teilweise schwierige Finanzsituation von Hochschulen und Forschungseinrichtungen zurückzuführen.

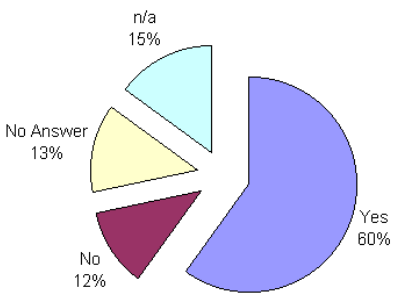

Abbildung A.35: Antworten zu Frage 5.3 „Beiträge von allen Unternehmen“

Frage 5.4

Would you or your employer accept a financial backing of the VDB by some large companies?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.36. Von den Teilnehmern erklärten 63%, daß sie oder ihr Arbeitgeber (vermutlich) keine Problem mit einigen großen Partner aus dem kommerziellen Umfeld hätten. Allerdings äußerten immerhin 27% Bedenken. Insbesondere die Teilnehmer der Gruppe Computer-related äußerten sich mit einem relativ hohen Anteil von 70% positiv zu einer Unterstützung durch große Unternehmen.

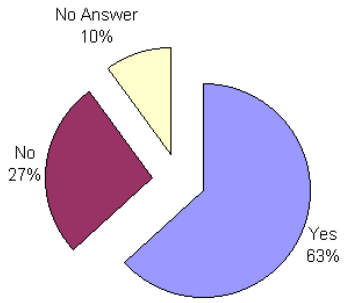

Abbildung A.36: Antworten zu Frage 5.4 „Akzeptanz von Sponsoren“

Frage 5.5

Would you or your employer accept a financial backing (e.g. subsidies) of the VDB by one or more governments?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.37. Von den Teilnehmern äußerten 70%, daß es keine Bedenken bezüglich einer Finanzierung durch Regierungen gäbe. Fast schon erstaunlich ist die höhere Zustimmung zu einer finanziellen Unterstützung durch Regierungen gegenüber der Unterstützung durch Unternehmen. Auch hier fallen die Nutzer der Gruppe Computer-related mit einer großen Zustimmung von 80% auf.

Frage 5.6

Would you or your employer accept a pay-per-query system for the usage of the VDB (e.g. using a micro payment system)?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.38. Für ein *Pay per Query*-System ergab sich eine breite Ablehnung: 67% der Teilnehmer lehnten ein solches System ab, wobei es unter den verschiedenen Gruppen keine auffälligen Abweichungen gab.

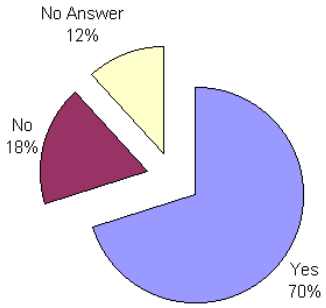

Abbildung A.37: Antworten zu Frage 5.5 „Akzeptanz staatlicher Beihilfen“

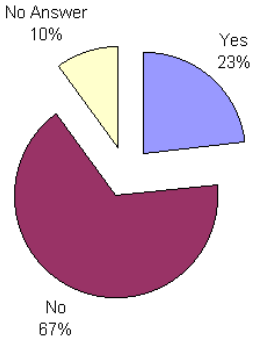

Abbildung A.38: Antworten zu Frage 5.6 „Akzeptanz Pay-per-Query“

Frage 5.7

Assume the existence of some kind of subscription system where a bonus system for “good” contribution could reduce the subscription fees. Do you think this would be attractive for contributors?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.39. 57% der Teilnehmer äußerten sich dahingehend, daß ein Bonussystem die Attraktivität der SDB steigern könnte. Dabei gab es zwischen den verschiedenen Gruppen nur geringe Abweichungen.

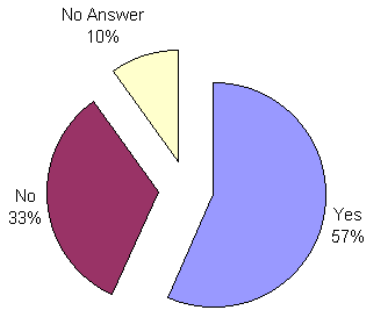

Abbildung A.39: Antworten zu Frage 5.7 „Attraktivität eines Bonussystems“

Frage 5.8

Assuming the normal access to the database would be free of charge, would you / your employer be willing to pay for “value added services” (e.g. security analysis of your systems) provided by the people running the VDB and thus financing the VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.40. Von den Teilnehmern beantworteten 68% die Frage mit Ja. Die geringste Zusprache fand die Finanzierung durch *Value Added Services* bei der Gruppe der Computer-related-Nutzer. Aus dieser Gruppe sprachen sich nur 60% für diesen Finanzierungsweg aus. Bei den Forschern waren es immerhin 69% und bei den übrigen Teilnehmer sogar 79%.

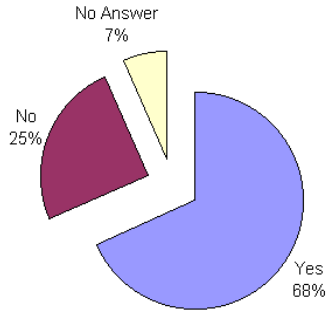

Abbildung A.40: Antworten zu Frage 5.8 „Finanzierung über Value Added Services“

A.1.6 Ergebnis Section 6: “Last few questions”

Abschnitt 6 des Fragebogens enthielt einige ergänzende Fragen an die Teilnehmer der Umfrage.

Frage 6.1

Would you like to have the ability to participate in security related off-topic discussions while using the VDB site?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.41. Eine große Mehrheit von 77% sprach sich für die Möglichkeit von *Off-Topic*-Diskussionen aus. Zwischen den Teilnehmergruppen gab es keine deutlichen Unterschiede.

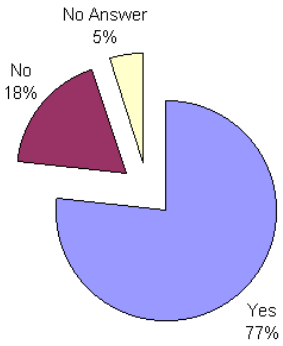

Abbildung A.41: Antworten zu Frage 6.1 „Möglichkeit für Off-Topic-Diskussionen“

Frage 6.2

Assume a new VDB would be launched which has all the features you requested above. Would you use this “new” VDB?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.42. Immerhin 95% der Teilnehmer erklärten, daß sie sich vorstellen könnten, *die* SDB, für die sie sich ausgesprochen haben, zu nutzen. Dies zeigt zumindest, daß man, wenn man einen geeigneten Kompromiß findet, der möglichst viele potentielle Nutzer anspricht, tatsächlich eine Möglichkeit hat, auf dem breiten Feld der existierenden SDBs noch Nutzer zu finden.

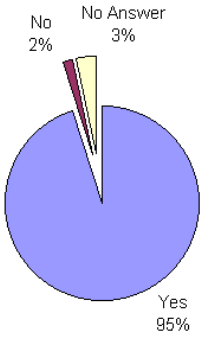

Abbildung A.42: Antworten zu Frage 6.2 „Bereitschaft zur Nutzung der neuen SDB“

Frage 6.3

Are you interested in taking part in further, more specialized surveys related to this topic?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.43. Von den Teilnehmern erklärten sich 63% bereit, an weiteren Umfragen teilzunehmen. Erstaunlicherweise erklärten sich 75% der Computer-related-Nutzer bereit, an weiteren Umfragen teilzunehmen, während die Nutzer aus dem Bereich Education / Research dies nur noch zu 62% erklärten und die übrigen Teilnehmer nur noch zu 50%.

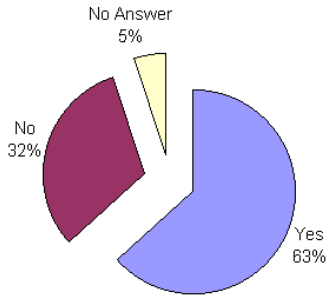

Abbildung A.43: Antworten zu Frage 6.3 „Teilnahme an weiteren Umfragen“

Frage 6.4

Would you contribute to the development of our VDB project?

Mögliche Antworten: *Yes* und *No*

Das Ergebnis der Antworten auf diese Frage zeigt Abbildung A.44. Erfreuliche 37% der Teilnehmer der Umfrage erklärten sich bereit an der Entwicklung der SDB teilzunehmen, wobei die Bereitschaft bei den Computer-Related- und den Education / Research-Nutzern mit ca. 41% am höchsten war. Bei den übrigen Teilnehmern erklärten sich nur 21% zur Teilnahme bereit.

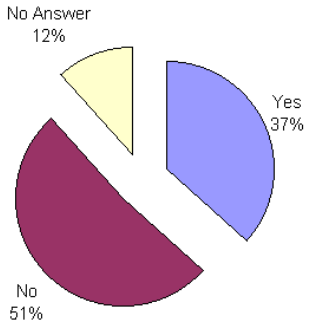

Abbildung A.44: Antworten zu Frage 6.4 „Teilnahme an der Entwicklung der SDB“

A.2 Definitionen der Felder

A.2.1 Klassen

A.2.1.1 Vendor

Hersteller einer Software

- Name

  Name des Herstellers

  Typ: String

  Pflichtfeld

- Homepage

  URL der Homepage des Herstellers

  Typ: String

- eMail-Address

  e-Mail-Adresse des Herstellers

  Typ: String

- Address1

  Adresse des Herstellers, erster Teil

  Typ: String

- Address2

  Adresse des Herstellers, zweiter Teil

  Typ: String

- City

  Stadt, in der der Hersteller seinen Sitz hat

  Typ: String

- ZIP

  Postleitzahl der Stadt, in der der Hersteller seinen Sitz hat

  Typ: String

- State

  Staat des Landes, in dem der Hersteller seinen Sitz hat

  Typ: String

- Country

  Land, in dem der Hersteller seinen Sitz hat

  Typ: String

- Released

  Ist der Hersteller-Eintrag offiziell freigegeben?

  Typ: Boolean

  Pflichtfeld

A.2.1.2 Software

Software speichert versionsunabhängige Informationen zu einem Software-Produkt.

- Name

  Name der Software

  Typ: String

  Pflichtfeld

- Description

  Beschreibung der Software

  Typ: Text

- Homepage

  URL, die zur Produkthomepage führt

  Typ: String

- eMail-Address

  e-Mail-Adresse, unter der Anfragen zu dieser Software gestellt werden können, möglichst die e-Mail-Adresse des technischen Supports

  Typ: String

- TypeOfSW

  Art der Software; um was für eine Software handelt es sich? Auswahl aus einem Entscheidungsbaum, der in der Hilfstabelle Tree gespeichert ist.[2] Abbildung A.45 zeigt den Aufbau des Entscheidungsbaums.

  Typ: Integer

  Pflichtfeld

- Released

  Ist dieser Software-Eintrag bereits offiziell freigegeben?

  Typ: Boolean

  Pflichtfeld

A.2.1.3 Operating_System

Operating_System ist eine Spezialisierung von Software. Bezüglich der Felder gibt es keine Unterschiede. Nur der Feldinhalt von TypeOfSW wird vorgegeben mit einem Wert, der die Softwarekategorie „Betriebssystem“ kennzeichnet. Der Schlüssel für diesen Eintrag ist in der Hilfstabelle Tree (siehe Abschnitt A.3) gespeichert.

[2] Vgl. dazu Abschnitt A.3

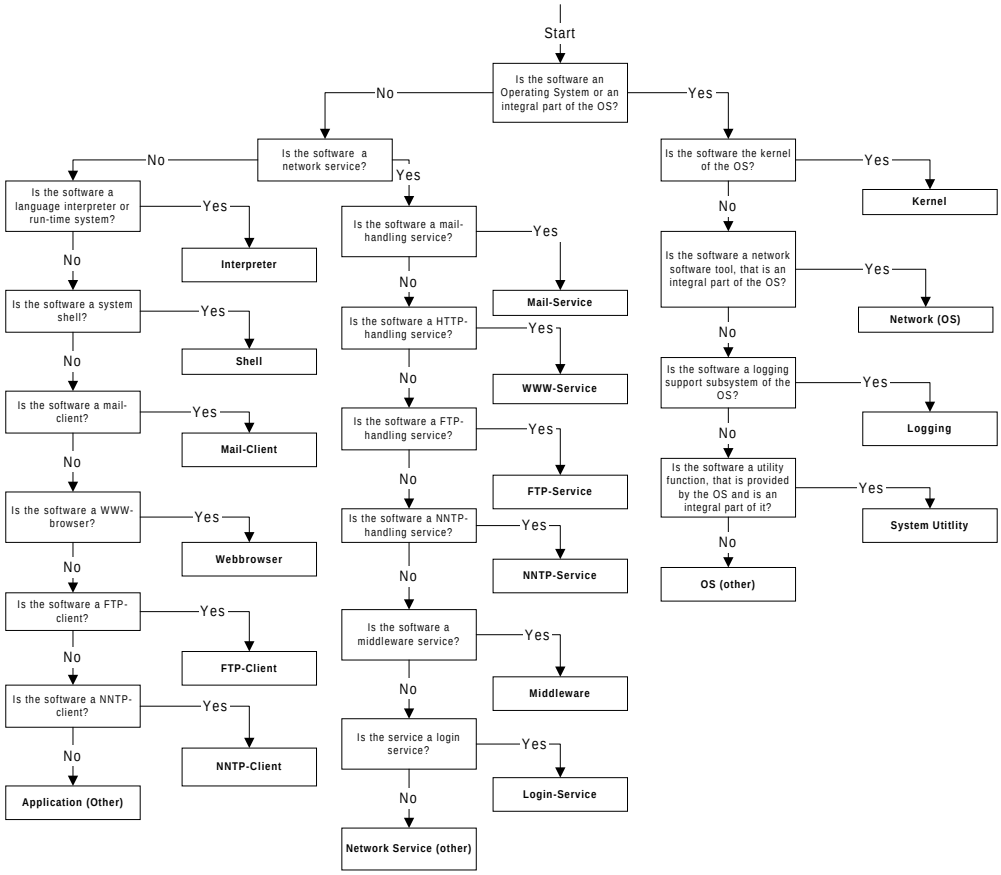

Abbildung A.45: Entscheidungsbaum: Type of Software

A.2.1.4 SW_Version

SW_Version speichert versionsabhängige Daten zu einem Software-Produkt.

- Major
  Speichert die Hauptversionsnummer einer Software-Version.
  Typ: Integer
  Pflichtfeld
- Minor
  Speichert die Unterversionsnummer einer Software-Version.
  Typ: Integer
  Pflichtfeld
- Patchlevel
  Speichert den Patchlevel einer Software-Version
  Typ: Integer

- VerSuffix

  Speichert Zusätze zur Versionsnummer. Beispielsweise wird manchmal bei internationalen Software-Versionen ein „i“ zur Versionsnummer hinzugefügt.

  Typ: String

- Alias

  Ermöglich die Angabe einer zweiten Bezeichnung für eine Software-Version. Beispielsweise könnte für eine bestimmte Software-Version eine „umgangssprachliche“ Bezeichnung existieren, die hier erfaßt werden kann, z.B. wird Microsoft Word 9.0.x als Microsoft Word 2000 bezeichnet. Auch ist es möglich, hier die original Versionsbezeichnung zu speichern, wenn sich diese nicht optimal in das Schema Major.Minor.Patchlevel-VerSuffix einordnen läßt.

  Typ: String

- URL

  URL beinhaltet eine URL auf eine spezielle Homepage für diese Version. Existiert eine solche Homepage nicht, kann hier auch eine Verweis auf eine Downloadmöglichkeit für die entsprechende Version gespeichert werden, beispielsweise ein FTP-Link.

  Typ: String

- Released

  Gibt an, ob dieser Software-Versions-Eintrag offiziell freigegeben wurde.

  Typ: Boolean

  Pflichtfeld

A.2.1.5 OS_Version

Spezialisierung von SW_Version, die bezüglich der Felder keine Unterschiede aufweist.

A.2.1.6 User

User dient der Verwaltung von Nutzern der SDB, ihren Daten, wie beispielsweise Pseudonym und Paßwort, und ihrer Zugangsrechte.

- Last_Name

  Nachname des Nutzers

  Typ: String

- First_Name

  Vorname des Nutzers

  Typ: String

- Middle_Name
  Zweiter Vorname des Nutzers
  Typ: String
- Address1
  Adresse des Nutzers, erster Teil
  Typ: String
- Address2
  Adresse des Nutzers, zweiter Teil
  Typ: String
- City
  Wohnort des Nutzers
  Typ: String
- ZIP
  Postleitzahl des Wohnorts des Nutzers
  Typ: String
- State
  Staat des Landes, in dem der Nutzer wohnt
  Typ: String
- Country
  Land, in dem der Nutzer wohnt
  Typ: String
- eMail
  e-Mail-Adresse des Nutzers
  Typ: String
- Pseudonym
  Pseudonym des Nutzers, unter dem er innerhalb der SDB auftritt
  Typ: String
  Pflichtfeld
- Password
  Paßwort des Nutzers
  Typ: String
  Pflichtfeld
- UserSince
  Registrierungsdatum des Nutzer
  Typ: Date
  Pflichtfeld

- Rating

  Bewertung des Nutzers

  Typ: Float

  Pflichtfeld

- Type_of_Access

  Gibt an, welchen Zugangsweg der Nutzer gewählt hat, also individualisiert oder personalisiert, wobei hier zwischen den Varianten unterschieden wird, ob nur die SDB-Administration die persönlichen Daten kennt oder alle Nutzer.

  Typ: Enum(Indiv,Pers_SDB,Pers_All)

  Pflichtfeld

- Status

  Gibt an, ob der Account aktiv, gesperrt oder gelöscht ist.

  Typ: Enum(Aktiv, Gesperrt, Gelöscht)

  Pflichtfeld

- AccessRights

  Gibt an, ob der Benutzer ein „normaler“ Nutzer oder ein Administrator ist.

  Typ: Enum(Admin, User)

  Pflichtfeld

A.2.1.7 Vulnerability

Vulnerability erfaßt die „Kerninformationen“ einer Schwachstelle.

- Name

  Bezeichnung der Schwachstelle

  Typ: String

  Pflichtfeld

- Terse_Description

  Kurze Beschreibung der Schwachstelle

  Typ: String

- Verbose_Description

  Ausführliche Beschreibung der Schwachstelle

  Typ: Text

- Direct_Impact

  Direkter Effekt, den die Ausnutzung der Schwachstelle hat (z.B. Root-Zugriff). Übernommen aus [Krs98]. Der Wert wird über einen Entscheidungsbaum bestimmt, vergleiche dazu Abbildung A.46. Der Entscheidungsbaum wird, implementierungstechnisch, in der Hilfstabelle Tree gespeichert und der Schlüssel des Blattes, das als Ergebnis der Bearbeitung des Entscheidungsbaums ermittelt wird, wird als Fremdschlüssel für den Eintrag verwendet.

  Typ: Enum

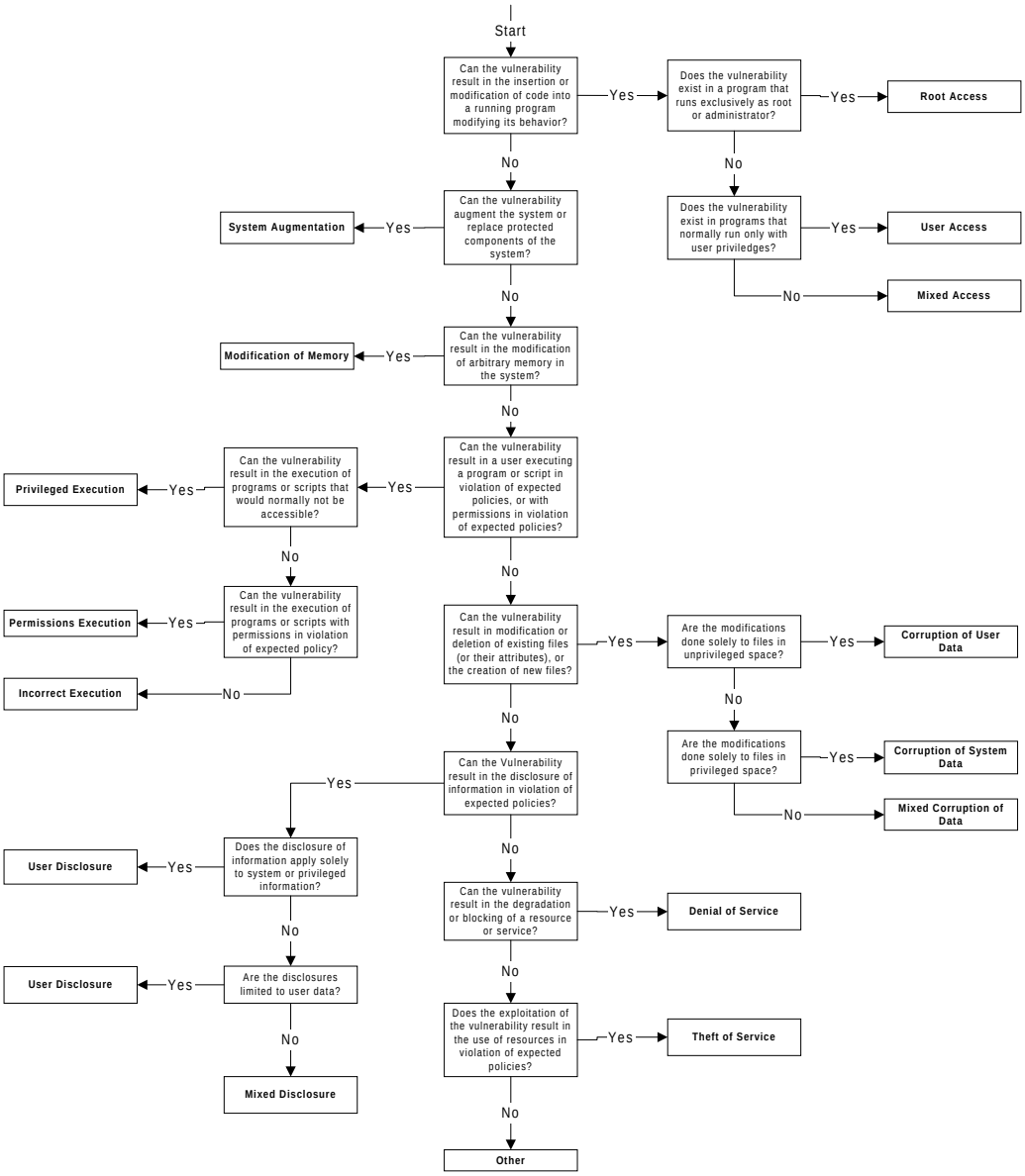

Abbildung A.46: Entscheidungsbaum: Direct Impact

- Indirect_Impact

Indirekter Effekt, den die Ausnutzung der Schwachstelle hat (z.B. External Root Access oder System Read). Übernommen aus [Krs98]. Der Wert wird über einen Entscheidungsbaum bestimmt, vergleiche dazu Abbildung A.47. Der Entscheidungsbaum wird, implementierungstechnisch, in der Hilfstabelle Tree gespeichert und der Schlüssel des Blattes, das als Ergebnis der Bearbeitung des Entscheidungsbaums ermittelt wird, wird als Fremdschlüssel für den Eintrag verwendet.

Typ: Enum

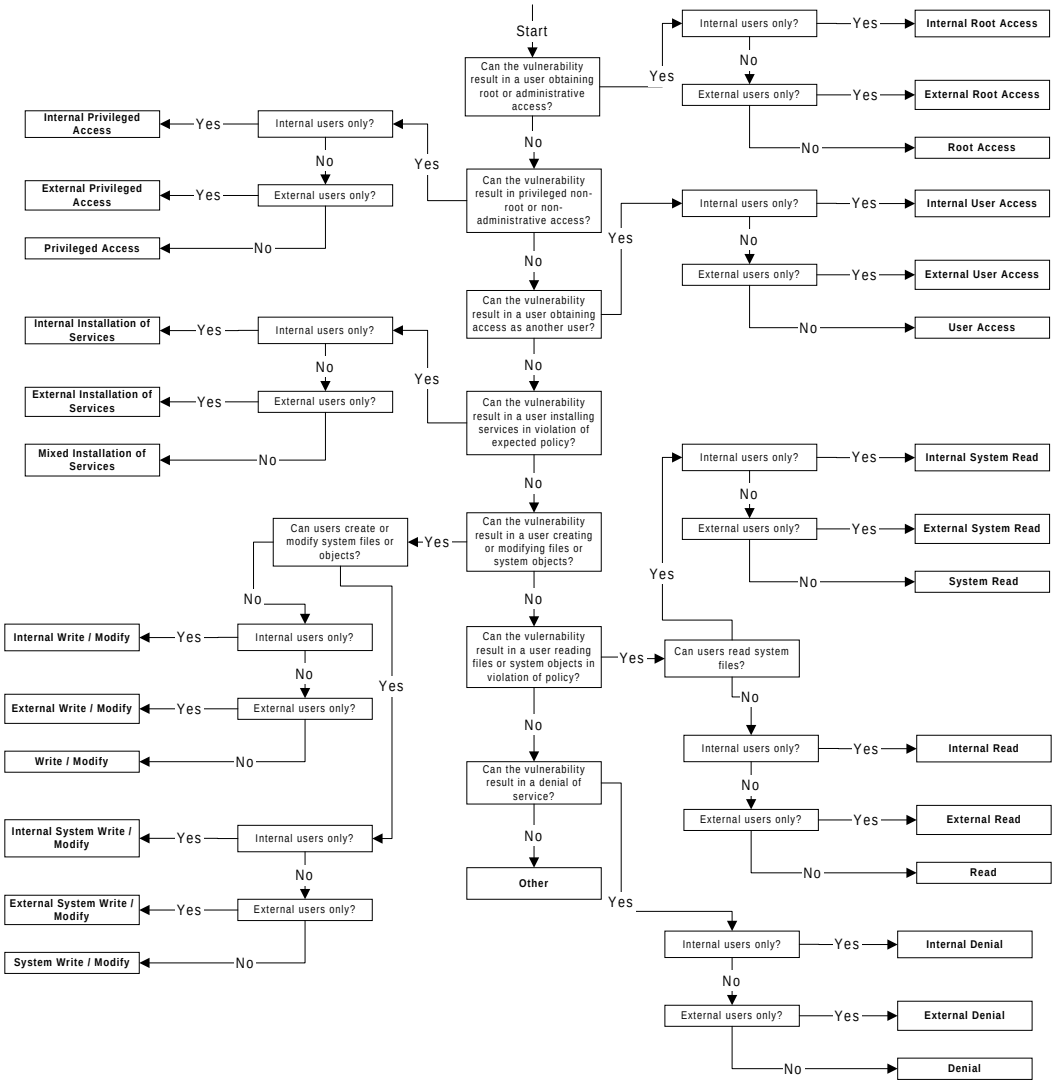

Abbildung A.47: Entscheidungsbaum: Indirect Impact

- Location_of_Vulnerability

  Location_of_Vulnerability dient dazu, den „Ursprung“ der Schwachstelle im Quellcode, soweit verfügbar, anzugeben. Das kann / können beispielsweise die Datei bzw. die Dateien, in denen die Ursache für die Schwachstelle liegt und die dazugehörigen Zeilennummern sein.

Typ: String

- Affected_Src_Code_Fragment

  Dient dazu, den Teil des Programmcodes, der ursächlich für die Schwachstelle verantwortlich ist, in der Datenbank zu speichern. Dabei können zusätzlich Kommentare eingefügt werden.

  Typ: Text

- Environmental_Assumption

  Environmental Assumptions beschreibt die Annahmen, die der Entwickler einer Software über die Laufzeitumgebung des Programms gemacht hat und die dann nicht gehalten haben, so daß es zu der Schwachstelle kam. Übernommen aus [Krs98]. Es gibt dabei eine Liste von möglichen Annahmen, die in der Hilfstabelle parameter gespeichert ist.[3] Zu jeder denkbaren Annahme kann angegeben werden, ob dies eine Annahme war, die der Entwickler traf (Yes) oder ob dies nicht der Fall ist (No), ob diese spezielle Annahme für die Schwachstelle nicht anwendbar ist (Does not apply) oder ob die Bewertung nicht bekannt ist (Unknown). Im folgenden die Liste der Annahmen und die (englischen) Erklärungstexte, die auch innerhalb des Programms angezeigt werden.

  Typ: List

  - *Directory permissions*: programmer assumes that a set of directories have a specific set of permissions (or a minimum set of permissions)
  - *No Core*: programmer assumes that a user cannot see the internals of the program as it is executing (i.e. no user readable core dumps in UNIX)
  - *Invariant Name*: programmer assumes that a name (i.e., a path) is strongly bound to a specific system object
  - *Invariant Object*: programmer assumes the invariance of an object during the execution of a program (i.e., the program assumes that no other subject can change the object while the program is running)
  - *Non existent Object*: programmer assumes that an object does not exist at the time of execution (i.e., a program assumes that a file with a specific name does not exist).
  - *Deletion of temporary item*: programmer assumes that a temporary item he created cannot be deleted by any other subject while the program is running
  - *Available Memory*: programmer assumes that sufficient memory for its execution will always exist
  - *Network Data*: programmer assumes that data from a network service will always be valid and bounded
  - *Environment Data*: programmer assumes that the data in environment variables is valid and bounded
  - *User Data*: programmer assumes that user-provided input is valid an bounded
  - *File Data*: programmer assumes that the input from a file is valid and bounded

[3] Vgl. dazu Abschnitt A.3

  - *Reassembly*: Programmer assumes that the re-assembly of a data object from fragments will not affect the essential properties of the original object
  - *Execution Path*: programmer assumes a specific execution path
  - *Object Attributes*: programmer assumes that certain attributes of certain objects have predefined values
  - *Persistent Storage*: programmer assumes that persistent store is immutable (i.e., assumes that a file it writes cannot be modified by any other subject in between program runs)
  - *Execution with Modified Data*: programmer assumes that the modification of program data (by external subjects) will not affect the semantics of the program
  - *Overwriting File*: programmer assumes that, while creating a file, any existing file that has the same name can be overwritten
  - *False Constraint*: programmer falsely assumes that a constraint or property holds in the system
  - *Insufficient Verification*: programmer falsely assumes that a set of operations are sufficient for the verification of the property of an object
  - *Binding between Name and Purpose*: programmer assumes that there is a strong binding between the name and purpose of an object
  - *Reserved Object*: programmer assumes that an object with a specific name will not be used by any other entity in the system by virtue of its name alone
  - *Trusted Network Object*: programmer assumes that a network object that claims an identity can be trusted

- Objects_Affected

  Beschreibung der Objekte, die direkt von der Schwachstelle betroffen sind. Übernommen aus [Krs98]. Pro Objekt ist die Angabe möglich, ob es betroffen ist (Yes), nicht betroffen ist (No), dieses Objekt im Zusammenhang mit der Schwachstelle respektive der Software nicht vorkommt (Does not apply) oder ob es unbekannt ist, ob dieses Objekt betroffen ist (Unknown). Im folgenden die Liste der Objekte und die (englischen) Erklärungstexte, die auch innerhalb des Programms angezeigt werden.

  Typ: List

  - *Command Prompt*: a command prompt presented to the user
  - *User Files*: user files in the system
  - *System Files*: system-related or administrative files in the system
  - *Public Files*: publicly available files in the system
  - *Directory*: directories in the system
  - *Partition*: file system partition
  - *Heap Data*: data in the heap of a running program
  - *Heap Code*: executable code in the heap of a running program
  - *Stack Data*: data in the stack of a running program
  - *Static Data*: data that is statically allocated in a running program
  - *Stack Return*: return address of a function in the stack of a running program

  - *Stack Code*: executable code in the stack of a running program
  - *Password*: password or access token - can also be a passphrase
  - *Shell Command*: shell command
  - *System Program*: system program
  - *User Program*: user installed or owned program
  - *System Info*: information regarding the system
  - *Outside files*: files outside a restricted space
  - *Classloader*: a classloader or object responsible for loading dynamic classes
  - *Library*: system function or service library
  - *A Network Connection*: network connections to arbitrary hosts
  - *Webpages*: WWW page
  - *Websession*: a WWW browsing session
  - *Names*: user names, domain names, work-group names, etc.
  - *Known Password*: well-known nonce encrypted with user password
  - *Object Attributes*: system-managed object attributes
  - *CPU*: CPU time
  - *OS*: Operating System
  - *eMail*: electronic mail
  - *Network Port*: network port
  - *Network Packets*: network packets
  - *System Names*: internal system names (in control of the system)
  - *Device*: a device in the system
  - *Address Mapping*: address mapping maintained by the system, i.e., an ARP cache

- Effect_on_Objects

  Effekt der Schwachstelle auf die betroffenen Objekte. Übernommen aus [Krs98]. Pro Effekt ist die Angabe möglich, ob er eintritt (Yes), nicht eintritt (No), dieser Effekt im Zusammenhang mit der Schwachstelle respektive der Software nicht vorkommen kann (Does not apply) oder ob es unbekannt ist, ob dieser Effekt eintritt (Unknown). Im folgenden die Liste der Effekte und die (englischen) Erklärungstexte, die auch innerhalb des Programms angezeigt werden.

  Typ: List

  - *Replaced*: contents are completely replaced
  - *Changed*: can be written to or can be changed
  - *Read*: can be read
  - *Append*: information can be appended
  - *Created*: can be created
  - *Displayed*: can be displayed or revealed
  - *Change Owner*: ownership can be changed

  - *Change Permission*: permissions can be changed
  - *Predictable*: is predictable or can be guessed
  - *Executed*: can be executed in violation of expected policy
  - *Loaded*: can be dynamically loaded and linked
  - *Clear Text*: is transmitted or stored in clear text
  - *Exhausted*: is exhausted
  - *Crash*: crashes
  - *Bound*: can be bound to in violation of expected policy
  - *Exported*: can be exported for mounting
  - *Mounted*: is mounted or attached
  - *Locked*: can be locked
  - *Debugged*: can be debugged or attached to with a debugger
  - *Presented*: presented to the user in a console or terminal
  - *Closed*: can be closed
  - *Terminated*: can be terminated or killed

- Method_Used

  Methode oder Mechanismus, der benutzt wird, um die betroffenen Objekte zu beeinflussen. Übernommen aus [Krs98]. Pro Methode kann angegeben werden, ob sie verwandt wird (Yes), nicht verwandt wird (No), ob die Methode im Zusammenhang mit der Schwachstelle respektive der Software nicht anwendbar ist (Does not apply) oder ob es nicht bekannt ist, ob diese Methode angewandt wird (Unknown). Im folgenden die Liste der Methoden und die (englischen) Erklärungstexte, die auch innerhalb des Programms angezeigt werden.

  Typ: List

  - *Symbolic Link*: program follows symbolic link or late binding link
  - *Memcpy*: program uses strcpy, sprintf or bcopy to copy data to a stack buffer
  - *Config*: Configuration error
  - *Back Ticks*: back ticks in parameter or input string
  - *Special Characters*: special characters in input string
  - *DotDot*: uses ".." to climb up a directory tree past allowable bounds
  - *Verify Fail*: code verifier allows to catch security exception when creating an object loader
  - *Modify Name*: modifying compiled code to alter the name of objects
  - *Modify Environment*: modifying environment variables
  - *Inherit Privileges*: program inherits unnecessary privileges
  - *Capability*: system provides inappropriate capability
  - *Hidden Mount*: system provides hidden system mount point
  - *Syscall Disclosure*: system call discloses sensitive information

  - *Incorrect Implementation*: incorrect implementation given current environment (mistaken environmental assumption)
  - *Relative Paths*: program refers to relative paths
  - *Incorrect Protection*: system fails to implement the protection mechanisms correctly
  - *Proxy*: program uses a trusted intermediary or proxy to bypass protection mechanism
  - *Core Symlinks*: a program dumps a core file that follows symbolic links or late binding link
  - *Infinite Loop*: program uses an infinite and tight loop that consumes resources
  - *Critical Section*: program fails to protect or isolate a critical section
  - *Core Dump*: program dumps a core file that users can read

- Input_Type

  Quelle der Eingabe, die, wenn überhaupt, notwendig ist um das Objekt zu beeinflussen. Übernommen aus [Krs98]. Pro Eingabetypus kann angegeben werden, ob dieser verwandt wird (Yes), nicht verwandt wird (No), ob er im Zusammenhang mit der Schwachstelle respektive der Software nicht anwendbar ist (Does not apply) oder ob es nicht bekannt ist, ob dieser Eingabetypus angewandt wird (Unknown). Im folgenden die Liste der Eingabetypen und die (englischen) Erklärungstexte, die auch innerhalb des Programms angezeigt werden.

  Typ: List

  - *Environment*: environment variable
  - *Command*: user command line option
  - *Network Data*: network data
  - *Store*: persistent store
  - *Tempfile*: temporary file
  - *Configuration File*: configuration file
  - *Data File*: data file
  - *SysUserInfo*: system user information (name, phone number, ...)
  - *Parameter*: parameter to a system call
  - *Library Parameter*: parameter to a library call
  - *Floppy*: removable media

- Threat_Feature_Action

  Ermöglicht die Angabe welche Arten von bedrohenden Aktionen einem Angreifer durch die Ausnutzung der Schwachstelle ermöglicht werden. Dabei zielt die Angabe auf die direkte Auswirkung und nicht auf längerfristige, indirekte Auswirkungen ab und es wird Bezug zur einer geplanten Sicherheitsrichtlinie genommen. Übernommen aus [Krs98]. Pro bedrohender Aktion kann angegeben werden, ob diese möglich ist (Yes), nicht möglich ist (No), ob sie im Zusammenhang mit der Schwachstelle nicht anwendbar ist (Does not apply) oder ob nicht bekannt ist, ob diese Aktion

möglicht ist (Unknown). Im folgenden die Liste der Aktionen und die (englischen) Erklärungstexte, die auch innerhalb des Programms angezeigt werden.

Typ: List

  - Exploitation of vulnerability can result in a user *observing* objects, data, etc., in violation of expected policy
  - Exploitation of vulnerability can result in a user *destroying* objects, data, etc., in violation of expected policy
  - Exploitation of vulnerability can result in a user *modifying* objects, data, etc., in violation of expected policy
  - Exploitation of vulnerability can result in a user *creating* objects, data, etc., in violation of expected policy

- Threat_Feature_Consequence

  Ermöglicht die Angabe, welche bedrohlichen Konsequenzen im Bezug auf die geplante Sicherheitsrichtlinie die Ausnutzung der Schwachstelle durch einen Angreifer haben kann. Dabei zielt die Angabe auf die direkten Konsequenzen und nicht auf längerfristige, indirekte Konsequenzen ab. Übernommen aus [Krs98]. Pro Konsequenz kann angegeben werden, ob diese eintritt (Yes), nicht eintritt (No), ob sie im Zusammenhang mit der Schwachstelle nicht eintreten kann (Does not apply) oder ob nicht bekannt ist, ob diese Konsequenz eintreten kann (Unknown). Im folgenden die Liste der Konsequenzen und die (englischen) Erklärungstexte, die auch innerhalb des Programms angezeigt werden.

  Typ: List

  - The exploitation of the vulnerability can result in a change of *availability* of the system
  - The exploitation of the vulnerability can result in the *disclosure* of information in violation of expected policy
  - The exploitation of the vulnerability can result in the *misrepresentation* of information in violation of expected policy
  - The exploitation of the vulnerability can result in *repudiation* of information in violation of expected policy
  - The exploitation of the vulnerability can result in a *change of integrity* of the system in violation of expected policy
  - The exploitation of the vulnerability can result in the *loss of confidentiality* of information in violation of expected policy

- Threat_Feature_Others

  Threat_Feature_Others dient dazu zusätzliche Bedrohungen, die von Threat_Feature_Action und Threat_Feature_Consequence nicht erfaßt werden, anzugeben. Dabei ist auch hierbei darauf zu achten, daß nur direkte Bedrohungen betrachtet werden und keine indirekten, längerfristigen.

  Typ: String

- Required_Access

  Notwendiger Zugang, den ein Angreifer zum angegriffenen System haben muß, damit er die Schwachstelle ausnutzen kann. Übernommen aus [Krs98]. Der Wert wird über einen Entscheidungsbaum bestimmt, vergleiche dazu Abbildung A.48. Der Entscheidungsbaum wird implementierungstechnisch in der Hilfstabelle tree gespeichert, und der Schlüssel des Blattes, das als Ergebnis der Bearbeitung des Entscheidungsbaums ermittelt wird, wird als Fremdschlüssel für den Eintrag verwendet.

  Typ: Enum

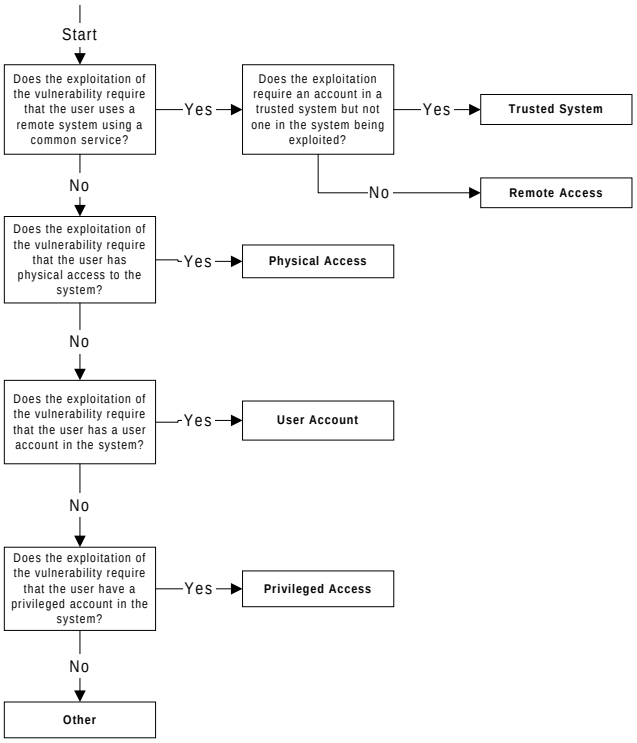

Abbildung A.48: Entscheidungsbaum: Required Access

- Complexity_of_Exploit

  Schwierigkeitsgrad der Ausnutzung der Schwachstelle. Übernommen aus [Krs98]. Der Wert wird über einen Entscheidungsbaum bestimmt, vergleiche dazu Abbildung A.49. Der Entscheidungsbaum wird implementierungstechnisch in der Hilfstabelle tree gespeichert, und der Schlüssel des Blattes, das als Ergebnis der Bearbeitung des Entscheidungsbaums ermittelt wird, wird als Fremdschlüssel für den Eintrag verwendet.

  Typ: Enum

- Released

  Gibt an, ob die Schwachstelle schon offiziell freigegeben wurde.

  Typ: Boolean

  Pflichtfeld

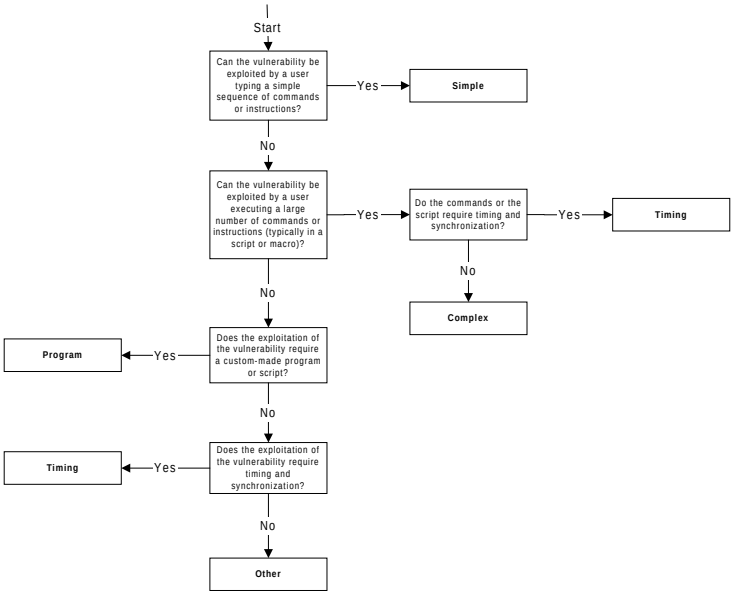

Abbildung A.49: Entscheidungsbaum: Complexity of Exploit

- CVE-Reference

  Speichert die CVE Kennung der Schwachstelle, soweit diese vergeben und bekannt ist.

  Typ: String

- Date_of_Report

  Speichert das Datum, an dem die Schwachstelle gemeldet wurde.

  Typ: Date

  Plichtfeld

- reported_by

  Speichert, welcher Nutzer die Schwachstelle gemeldet hat.

  Typ: User

- Last_Change

  Speichert, wann die letzte Änderung an dem Schwachstelleneintrag vorgenommen wurde.

  Typ: Date

- Last_Change_by

  Speichert, wer die letzte Änderung an dem Schwachstelleneintrag vorgenommen hat.

  Typ: User

A.2.1.8 Reference

Reference dient zur Erfassung von Quellen, die ebenfalls Informationen zu der Schwachstelle enthalten.

- Type

  Gibt den Typ der Quelle an.

  Typ: Enum(Advisory, Book, Mail, SDB, other)

  Pflichtfeld

- Source

  Gibt die zusätzliche Quelle an, beispielsweise Daten des Buches oder die URL der WWW-Seite.

  Typ: String

  Pflichtfeld

- Comment

  Kommentar zu der zusätzlichen Referenz.

  Typ: Text

- Reported_by

  Angabe, wer die Referenz gemeldet hat.

  Typ: User

  Pflichtfeld

- Date_of_Report

  Datum, an dem die Referenz gemeldet wurde.

  Typ: Date

  Pflichtfeld

A.2.1.9 Incident

Speichert Informationen zu einem Vorfall.

- Date_of_Report

  Speichert das Datum, an dem der Vorfall gemeldet wurde.

  Typ: Date

  Pflichtfeld

- Affected_Hardware

  Speichert in Textform, welche Hardware bei dem Vorfall betroffen war.

  Typ: Text

- How_Recognized

  Beschreibung, wie der Vorfall bemerkt wurde.

  Typ: Text

- Initiated_Countermeasure

  Beschreibt die Gegenmaßnahmen, die getroffen wurden, um eine Wiederholung des Vorfalls zu verhindern. Wenn möglich auch Angaben darüber, ob diese Maßnahmen erfolgreich waren.

  Typ: Text

- reported_by

  Speichert, wer den Vorfall gemeldet hat.

  Typ: User

  Pflichtfeld

A.2.1.10 Snapshot

Snapshot ist bisher eine „Dummy-Entität“. Sie soll später dazu dienen, mittels eines speziellen, noch zu entwickelnden Werkzeugs erzeugte Systeminformationen über ein System, das von einem Vorfall betroffen war, zu speichern. Dabei sollen speziell Konfigurationsdaten, aber auch Informationen über die verwendete Software und die verwendeten Software-Versionen ermittelt werden. Gleichzeitig müssen die Informationen anonymisiert werden, so daß sie nicht dazu verwendet werden können, neue Angriffe auf das von dem Vorfall betroffene System zu fahren.

A.2.1.11 Countermeasure

Countermeasure speichert Informationen über Gegenmaßnahmen zu einer Schwachstelle.

- Type

  Type gibt den Typ der Gegenmaßnahmen an, ob es sich beispielsweise um einen Patch für die betroffene(n) Software-Version(en) handelt oder ob die Gegenmaßnahme nur einen Workaround darstellt, bis eine Beseitigung der Ursache möglich ist.

  Typ: Enum(Patch, Workaround, Reconfiguration, New Version, Other)

  Pflichtfeld

- Description

  Ausführliche Beschreibung der Gegenmaßnahme

  Typ: Text

- URL

  URL in Bezug auf die Gegenmaßnahme, beispielsweise ein Verweis auf eine Seite, die den Download eines Patches ermöglicht.

  Typ: String

- reported_by
  Speichert, wer die Gegenmaßnahme gemeldet hat.
  Typ: User
  Pflichtfeld
- date_of_report
  Speichert, wann die Gegenmaßnahme gemeldet wurde.
  Typ: Date
  Pflichtfeld
- Released
  Gibt an, ob die Gegemaßnahme offiziell freigegeben wurde.
  Typ: Boolean
  Pflichtfeld

A.2.1.12 Exploit

Speichert Informationen zu sogenannten Exploits, d.h. zu Möglichkeiten eine Schwachstelle auszunutzen.

- reported_by
  Speichert, wer den Exploit gemeldet hat.
  Typ: User
  Pflichtfeld
- date_of_report
  Speichert, wann der Exploit gemeldet wurde.
  Typ: Date
  Pflichtfeld
- Terse_Description
  Kurze Beschreibung der Arbeitsweise des Exploits.
  Typ: String
- Verbose_Description
  Möglichst ausführliche Beschreibung der Arbeitsweise des Exploits.
  Typ: Text
- Effect
  Beschreibt den Effekt, den der Einsatz des Exploits hat. Die möglichen Effekte werden implementierungstechnisch in der Hilfstabelle parameter gespeichert. Mögliche Werte sind, daß eine Session eines Dienstes beendet wird, daß ein Dienst selbst beendet wird, daß das ganze System, auf dem ein Dienst läuft, herunterfährt oder auch gar kein Effekt.
  Typ: Enum

- URL

  Dient zur Speicherung eines Verweises auf den Quellcode.

  Typ: String

- Exploit_Source

  Dient zur Speicherung des Quellcodes des Exploits.

  Typ: Text

- Released

  Gibt an, ob der Exploit bereits offiziell freigegeben wurde.

  Typ: Boolean

  Pflichtfeld

A.2.1.13 Comment

Ermöglicht die Speicherung von Kommentaren und Diskussionen zu einzelnen Beiträgen in der SDB. Für die Bereiche Vulnerability, Incident, Countermeasure und Exploit werden spezialisierte Entitäten verwendet. Für freie oder auch off-topic Diskussionen kann direkt die Entität Comment verwendet werden.

- Comment_to

  Dient der Speicherung, wozu dieser Eintrag ein Kommentar ist.

  Typ: Enum

  Pflichtfeld

- Type_of_Comment

  Ermöglicht eine Einstufung, um was für einen Typ von Kommentar es sich handelt. Implementierungstechnisch werden die möglichen Werte aus der Hilfstabelle parameter eingelesen, in der die möglichen Werte gespeichert sind. Zum Zeitpunkt der Erstellung dieser Arbeit vorgesehene Werte: Rating, Consent, Supplement, Contradiction, Comment, Other.

  Typ: Enum

  Pflichtfeld

- Subject

  Betreff-Zeile der Meldung.

  Typ: String

- Message

  Der Kommentartext.

  Typ: Text

  Pflichtfeld

- Posted_by

  Angabe des Nutzers, der den Kommentar erstellt hat.

  Typ: User

  Pflichtfeld

- Date_of_Posting

  Datum der Erstellung des Kommentars.

  Typ: Date

  Pflichtfeld

A.2.1.14 IComment

Spezialisierung von Comment für Incidents. Implementierungstechnisch wird der Wert für das Comment-Feld comment_to vorgegeben.

A.2.1.15 CMComment

Spezialisierung von Comment für Countermeasure. Implementierungstechnisch wird der Wert für das Comment-Feld comment_to vorgegeben.

A.2.1.16 VComment

Spezialisierung von Comment für Vulnerability. Implementierungstechnisch wird der Wert für das Comment-Feld comment_to vorgegeben.

A.2.1.17 EComment

Spezialisierung von Comment für Exploit. Implementierungstechnisch wird der Wert für das Comment-Feld comment_to vorgegeben.

A.3 Data Definition Language für die SDB

Der Quellcode der Data Definition Language für die SQL-Datenbank Informix Foundation 2000 findet sich auf der dieser Diplomarbeit beiliegenden CD im Verzeichnis SQL.

A.4 Definition der statischen Daten der Tabellen parameter und tree

Für die SQL-Datenbank Informix Foundation 2000 findet sich der Quellcode der statischen Daten, die in den Hilfstabellen parameter und tree gespeichert sind, auf der dieser Diplomarbeit beiliegenden CD im Verzeichnis SQL.

A.5 Quellcode des Prototypen der Administrator-Schnittstelle

Der Quellcode des Admin-Tool-Prototypen findet sich auf der dieser Diplomarbeit beiliegenden CD im Verzeichnis Java.

A.6 Dokumentation des Quellcodes der Administrator-Schnittstelle

Die Javadoc-Dokumentation des Admin-Tool-Prototypen findet sich auf der, dieser Diplomarbeit beiliegenden, CD im Verzeichnis Javadoc.

A.7 Elektronische Quellen

Die elektronischen Quellen die im Rahmen dieser Diplomarbeit verwendet wurden, finden sich auf der dieser Diplomarbeit beiliegenden CD im Verzeichnis Elektronische Quellen.

A.8 Mailing-Listen zum Thema Sicherheit

Folgende Quellen empfehlen sich bei der Suche nach sicherheitsrelevanten Mailing-Listen. Kopien dieser Seiten finden sich auf der beiliegenden CD bei den elektronischen Quellen unter den bei den entsprechenden Seiten angegebenen Quellenbezeichnungen.

- `http://xforce.iss.net/maillists/`

  Übersicht über die Mailing-Listen von Internet Security Systems (ISS).

  Quelle: [ISS00c]

- `http://xforce.iss.net/maillists/otherlists.php`

  Übersicht über Mailing-Listen, zusammengestellt von Internet Security Systems (ISS).

  Quelle: [ISS00b]

- http://www.cert.dfn.de/resource/net_news/

  Übersicht über sicherheitsrelevante Mailing-Listen zusammengestellt vom DFN-CERT.

  Quelle: [DFN98]

A.9 Newsgroups zum Thema Sicherheit

Folgende Quellen empfehlen sich bei der Suche nach sicherheitsrelevanten Newsgroups. Kopien dieser Seiten finden sich auf der beiliegenden CD bei den elektronischen Quellen unter den bei den entsprechenden Seiten angegebenen Quellenbezeichnungen.

- http://www.cert.dfn.de/resource/net_news/

  Übersicht über sicherheitsrelevante Newsgroups zusammengestellt vom DFN-CERT.

  Quelle: [DFN97]

- http://www.alw.nih.gov/Security/security-newsgroups.html

  Übersicht über sicherheitsrelevante Newsgroups zusammengestellt vom Center for Information Technology, National Institutes of Health, Bethesda, MD, USA.

  Quelle: [Cen97]

- http://www.jjtc.com/Security/usenet.htm

  Übersicht über sicherheitsrelevante und auf die Privatssphäre bezogene Newsgroups von Johnson & Johnson Technology Consultants, LLC.

  Quelle: [JJTC00]

- http://www.dlxguard.com/secnews.htm

  Übersicht über sicherheitsrelevante Newsgroups zusammengestellt von DataLynx, Inc.

  Quelle: [Dat00]

- http://www.newsville.com/news/groups/comp.security.html

  Kommentierte Übersicht über die Newsgroups in der comp.security-Hierarchie des USENET. Zusammengestellt von Newsville.com, einem Service von Virtual Interactive Center. Ebenso ist unter http://www.newsville.com/news/groups/ eine kommentierte Liste von sehr vielen internationalen Newsgroups zu finden.

  Quelle: [New00]

Literaturverzeichnis

[AD99] D. Alessandri and M. Dacier. Vulda: A Vulnerability Database. Technical report, IBM Zurich, 1999.

[BS00a] Bruce Schneier. Computer Security: Will We Ever Learn? http://www.counterpane.com/crypto-gram-0005.html, May 2000.

[BS00b] Bruce Schneier. “Key Finding” Attacks and Publicity Attacks. http://www.counterpane.com/crypto-gram-0001.html, Januar 2000.

[BS00c] Bruce Schneier. Publicizing Vulnerabilities. http://www.counterpane.com/crypto-gram-0002.html, Februar 2000.

[CCC00] Chaos Computer Club. Chaos Computer Club e.V. Homepage. https://www.ccc.de, Mai 2000.

[Cen97] Center for Information Technology, National Institutes of Health, Bethesda, MD. Security Newsgroups. http://www.alw.nih.gov/Security/security-newsgroups.html, Dezember 1997.

[CER00a] CERT-AU. CERT Australia. http://www.auscert.org.au, 2000.

[CER00b] CERT/CC. CERT Advisory CA-2000-06 Multiple Buffer Overflows in Kerberos Authenticated Services. http://www.cert.org/advisories/CA-2000-06.html, Mai 2000.

[CER00c] CERT/CC. Cert advisory mailing list. http://www.cert.org/contact_cert/certmaillist.html, Mai 2000.

[CER00d] CERT/CC. Cert advisory search web interface. http://search.cert.org/, Mai 2000.

[CER00e] CERT/CC. CERT Coordination Center. http://www.cert.org, 2000.

[CER00f] CERT/CC. CERT Coordination Center Incident Reporting Form. http://www.cert.org/ftp/incident_reporting_form, Mai 2000.

[CER00g] CERT/CC. CERT Coordination Center Incident Reporting Guidelines. http://www.cert.org/tech_tips/incident_reporting.html, Mai 2000.

[CER00h] CERT/CC. CERT Coordination Center Vulnerability Reporting Form. http://www.cert.org/ftp/vul_reporting_form, Mai 2000.

[CER00i] CERT/CC. CERT/CC Incident Notes. http://www.cert.org/incident_notes/, Juli 2000.

[CER00j] CERT/CC. CERT/CC Vulnerability Notes. http://www.cert.org/vul_notes/, Juli 2000.

[CIA00] CIAC. Computer Incident Advisory Capability. http://ciac.llnl.gov/, 2000.

[Dat00] DataLynx, Inc. DataLynx Security NewsGroups Page. http://www.dlxguard.com/secnews.htm, Juli 2000.

[DFN97] DFN-CERT. Sicherheitsrelevante News-Groups. http://www.cert.dfn.de/resource/net_news/, 1997.

[DFN98] DFN-CERT. Sicherheitsrelevante Mailing-Listen. http://www.cert.dfn.de/resource/maillist/, 1998.

[DFN00] DFN-CERT. CERT des Deutschen Forschungs-Netzes. http://www.cert.dfn.de/, 2000.

[Fed00] FedCIRC. Federal Computer Incident Response Capability. http://www.fedcirc.gov/, 2000.

[FSF00a] Inc. Free Software Foundation. GNU Free Documentation License. http://www.gnu.org/copyleft/fdl.html, Juni 2000.

[FSF00b] Inc. Free Software Foundation. GNU General Public License. http://www.gnu.org/copyleft/gpl.html, Juni 2000.

[HD99] Jordan Hrycaj and Renaud Deraison. What is a security auditing tool? http://www.nessus.org/pres/workshop_12291999/2_1.html, Dezember 1999.

[HS95] Andreas Heuer and Gunter Saake. *Datenbanken: Konzepte und Sprachen*. International Thomson Publishing, Bonn, 1995.

[HS99] Michael Hurler and Thomas Schmitz. Sicherheit im Internet. http://www.informatik.tu-darmstadt.de/DVS1/staff/hurler/homepage/files/SicherheitImInternet.pdf, 1999.

[ISS00a] Internet Security Services. X-Force Database. http://xforce.iss.net/, 2000.

[ISS00b] Internet Security Systems. Aufstellung über IT-Security Mailing Listen. http://xforce.iss.net/maillists/otherlists.php, Mai 2000.

[ISS00c] Internet Security Systems. Aufstellung der ISS IT-Security Mailing Listen. http://xforce.iss.net/maillists/, Mai 2000.

[ISS00d] Internet Security Systems. Internet Security Systems Homepage. http://www.iss.net/, Mai 2000.

[JJTC00] LLC Johnson & Johnson Technology Consultants. Security and Privacy Newsgroups. http://www.jjtc.com/Security/usenet.htm, Juli 2000.

[Kni00] Eric Knight. Computer vulnerabilities. Technical report, Security Paradigm, 2000. http://www.securityparadigm.com/compvuln_draft.pdf.

[Krs98] Ivan Victor Krsul. *Software Vulnerability Analysis*. PhD thesis, Purdue University, USA, 1998.

[L0p00] L0phT. L0pht Heavy Industries. http://www.l0pht.com/, 2000.

[MC99] David E. Mann and Steven M. Christey. Towards a Common Enumeration of Vulnerabilities. *The MITRE Corporation*, 1999.

[MC00] Microsoft Corporation. http://www.microsoft.com/security/services/bulletin.asp, Mai 2000.

[MIT00a] MITRE. Common Vulnerabilities and Exposures (CVE). *http://cve.mitre.org*, 2000.

[MIT00b] MITRE. CVE-Compatible Sites and Databases. http://cve.mitre.org/About_CVE/About/othersites.html, Mai 2000.

[MS99] Pascal C. Meunier and Eugene H. Spafford. Final Report of the 2nd Workshop on Research with Security Vulnerability Databases. Technical report, CERIAS, Purdue University, USA, 1999.

[MSNS00] Microsoft Security Notification Service. Microsoft security bulletin (ms00-034) - patch available for “office 2000 ua control” vulnerability. Microsoft Security Mailing List, Message-ID: D1A11CCE78ADD111A35500805FD43F5867C2F4@RED-MSG-04, Mai 2000.

[Nes00] Nessus.org. Nessus. http://www.nessus.org, Juli 2000.

[New00] Virtual Interactive Center Newsville.com. comp.security newsgroups, Mai 2000.

[NTB00] NTBugtraq. NTBugtraq Homepage. http://www.ntbugtraq.com/, 2000.

[OEC97] OECD. Guidelines for the Security of Information Systems. http://www.oecd.org/dsti/sti/it/secur/prod/e_secur.htm, Juli 1997.

[Ope00a] OpenContent.org. OpenContent License 1.0. http://opencontent.org/opl.shtml, Juni 2000.

[Ope00b] OpenSource.Org. The Open Source Definition. http://www.opensource.org/osd.html, Juni 2000.

[Phr00] Phrack. Phrack Homepage. http://www.phrack.com/, 2000.

[Sec00a] INFILSEC Systems Security. INFILSEC. http://www.infilsec.com/vulnerabilities/, 2000.

[Sec00b] Securityfocus. Bugtraq. http://www.securityfocus.com/forums/bugtraq/faq.html, 2000.

[Sec00c] Securityfocus. Securityfocus Homepage. http://www.securityfocus.com, Mai 2000.

[SHHB00] Markus Schumacher, Christian Haul, Michael Hurler, and Alejandro Buchmann. Data-Mining von Schwachstellendatenbanken. http://www.ito.tu-darmstadt.de/publs/papers/dfncert2000.pdf, March 2000.

[SM00] Inc. Sun Microsystems. Java Secure Socket Extension. http://java.sun.com/products/jsse/, Juli 2000.

[SMR00] Markus Schumacher, Marie-Luise Moschgath, and Utz Roedig. Angewandte Informationssicherheit: Ein Hacker-Praktikum an Universitäten. *Informatik Spektrum*, 23, Juni 2000.

[TRU00] TRUSTED. Survey on Vulnerability Databases. *http://www.ito.tu-darmstadt.de/survey.html*, 2000.

[WC00a] WebTrends Corporation. WebTrends Homepage, Deutschland. http://www.WebTrends.de, Mai 2000.

[WC00b] WebTrends Corporation. WebTrends Homepage, USA. http://www.WebTrends.com, Mai 2000.

[WC00c] WebTrends Corporation. WebTrends Security Analyzer. http://www.webtrends.com/products/wsa/, Mai 2000.