Schwachstellen in Sicherheitssoftware und deren Auswirkungen

Diplomarbeit

eingereicht an der Fakultät

ELEKTROTECHNIK

der Hochschule für Technik und Wirtschaft Dresden

zur Erlangung des ersten akademischen Grades

DIPLOMINGENIEUR (FH)

vorgelegt

von

Robert Kossatz

geb. am 27.10.1981

Inhaltsverzeichnis

Abkürzungsverzeichnis

CD-ROM
Compact Disc Read-Only Memory.

CPE
Common Platform Enumeration.

CSV
Comma Separated Values.

CVE
Common Vulnerabilities and Exposures.

CVSS
Common Vulnerability Scoring System.

CWE
Common Weakness Enumeration.

DBMS
Datenbankmanagementsystem.

DTD
Document Type Definition.

GPL
General Public License.

HTML
Hypertext Markup Language.

IEEE
Institute of Electrical and Electronics Engineers.

IT
Information Technology, Informationstechnik.

NIST

National Institute of Standards and Technology.

NVD

National Vulnerability Database.

OOXML

Office Open XML.

OSVDB

Open Source Vulnerability Database.

SCAP

Security Content Automation Protocol.

SQL

Structured Query Language.

URL

Uniform Resource Locator.

XML

Extensible Markup Language.

XSL

Extensible Stylesheet Language.

XSLT

Extensible Stylesheet Language for Transformations.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Danksagung

Hiermit möchte ich mich bei allen bedanken, die mich im Rahmen meiner Diplomarbeit unterstützt haben. Besonderer Dank geht dabei an Matthias für die stetigen fachlichen Ratschläge.

Desweiteren möchte ich Heiko und meinem Bruder Marcus für die Hilfe besten Dank aussprechen. Besondere Anerkennung gilt meiner Freundin Tabea, die mich in allen Phasen dieser Arbeit zu unterstützen versucht hat, und nicht zuletzt danke ich meinen Eltern für ihre unermüdende moralische Unterstützung, mit der auf meiner Seite nichts unmöglich erscheint.

Kapitel 1

Einleitung

1.1 Motivation

Ein entscheidender Bestandteil des menschlichen Lebens ist der Wunsch nach Sicherheit. So sind Airbags und Sicherheitsgurte im Fahrzeug oder das Abschließen einer Autoversicherung Maßnahmen, die im Alltag fest integriert und akzeptiert sind. Doch welche Rolle spielt dieser angestrebte gefahrenfreie Zustand in der Informationstechnik?

Das Internet eröffnet den Menschen viele Möglichkeiten, bringt jedoch auch Gefahren mit sich. Um diesen entgegenzuwirken, stellt der Markt verschiedene Anwendungen bereit. Doch kann diese so genannte Sicherheitssoftware auch fehlerfrei sein? Oder in welchem Maße können sie dem Anwender den Wunsch nach Sicherheit erfüllen?

Die vorliegende Arbeit beschäftigt sich mit diesen Fragen, zeigt Wege, sich über Schwachstellen in der Informationstechnik zu informieren und widmet sich einer Möglichkeit, diese auszuwerten.

1.2 Aufbau und Kapitelübersicht

Das zweite Kapitel liefert Grundlagen der Arbeit, indem es sich mit den zentralen Begriffen *Schwachstelle*, *Sicherheit*, *Sicherheitssoftware* und *Datenbank*, sowie deren Bedeutung im Umfeld der Informationstechnik auseinandersetzt. Die Recherche zu Schwachstellen unter Verwendung geeigneter Quellen beschreibt der dritte Abschnitt. Kapitel vier erläutert anschließend detailliert die Art und Weise des Imports der Daten aus der gewählten Quelle, und die Auswertung und Beurteilung schließen diese Arbeit ab.

Kapitel 2

Grundlagen

2.1 Schwachstellen

2.1.1 Typen

Im Allgemeinen stellen *Schwachstellen* eine Eigenschaft von Objekten dar, die zu sicherheitsrelevanten Ereignissen führen können. Der Autor Klaus Schmidt teilt sie in [Sch06] allgemein in die drei Bereiche auf, in denen sie auftreten können. Der erste Typ ist die *technische* Schwachstelle, die auf einer Schwäche in der produkttechnischen Realisierung oder im Zusammenspiel von verschiedenen Komponenten basieren. Ein einfaches Beispiel aus dem Bereich Software ist die Verwendung eines zu geringen Speicherbereiches, was zu einem Überlauf an Daten und damit einem Absturz der Anwendung führen kann.

Liegt einer Schwachstelle ein Fehler in der Ablauforganisation wie eine unterdimensionierte Klimatisierung im Serverraum eines Rechenzentrums zu Grunde, dann ist dies eine *organisatorische* Schwachstelle. Für den dritten Typ ist der Mensch ausschlaggebend. Dabei können beispielsweise psychologische Schwächen oder das Verlassen des Arbeitsplatzes, ohne den Rechner dabei zu sperren, zu einer *personellen* Schwachstelle führen.

2.1.2 Sicherheitsziele

In der Informationstechnik sind drei Sicherheits- oder Schutzziele von großer Bedeutung. Die *Vertraulichkeit* ist dabei eine Eigenschaft von Daten, die bedeutet, dass nur diejenigen Personen und Systeme Kenntnisse vom Inhalt der Daten erhalten, die dazu befugt sind. Die Vertraulichkeit bezieht sich dabei nicht nur auf sensible Nutzdaten wie Datenbanken oder Dokumente, sondern auch auf Systemkonfigurationen oder Passwörter. Die *Verfügbarkeit* umfasst ebenfalls einen wichtigen

Aspekt der IT-Sicherheit. Dabei müssen die Daten zugreifbar sein, wann immer sie an einem bestimmten Ort benötigt werden. Maßnahmen für den Erhalt der geforderten Verfügbarkeit sind auf die Umgebung abgestimmt. So sind beispielsweise für einen Rechner, der die zentrale Produktion eines Unternehmens steuert, umfangreichere Sicherheitsmaßnahmen zum Erhalt der Verfügbarkeit benötigt als für einen Rechner eines Mitarbeiters, der lediglich eine Verbindung ins Internet herstellen kann. *Integrität* beschreibt den Schutz vor Verlust und vorsätzlicher Veränderung von Daten. Sie ist dann gewahrt, wenn der Inhalt von Daten auf der Sende- und Empfangsseite identisch ist, also nicht verändert wurde. [SR05]

2.1.3 Definition

Die Innovationszyklen in der Branche der Informationstechnik wurden im Laufe der Zeit immer kürzer und Systeme in einem immer schneller werdenden Takt entwickelt oder erweitert. Dieser Trend bringt einen gravierende Zunahme für die Bedeutung der Sicherheitsproblematik mit sich, denn trotz vieler Richtlinien für den sicheren Entwurf von Systemen steigt die Anzahl entdeckter Schwachstellen weiter stetig an.

Schwachstellen können sowohl an Komponenten der Software wie auch Hardware auftreten. Diese Arbeit beschäftigt sich jedoch ausschließlich mit dem Software-Bereich, in dem die Bezeichnung *Sicherheitslücke* synonym verwendet werden kann. Ein weiteres Wort, das in diesem Zusammenhang – vorrangig von Herstellern aus dem Bereich Sicherheitssoftware – gern verwendet wird, ist die englische „Vulnerability“. Korrekt übersetzt bedeutet dieser Begriff auch *Verletzlichkeit* oder *Verwundbarkeit* und der Autor beschreibt in [SRM07] die Abgrenzung der Verletzlichkeit von einer Sicherheitslücke:

> „Von einer Sicherheitslücke wird jedoch nur dann gesprochen, wenn eine Schutzmaßnahme fehlerhaft ist.“

Eine exakte und allgemein gültige Definition für Schwachstellen oder Sicherheitslücken im Umfeld der IT[1] zu finden ist nicht möglich, und so definiert das Unternehmen „Microsoft Corporation“ den Begriff *Sicherheitsschwachstelle* in [Micb] wie folgt:

> „Sicherheitsschwachstellen sind Lücken in Softwareprogrammen, über die ein Angreifer die Integrität, Verfügbarkeit oder Vertraulichkeit dieser Software gefährden kann. Im schlimmsten Fall erhält ein Angreifer aufgrund einer Sicherheitsschwachstelle die Möglichkeit, schädlichen Code auf gefährdeten Systemen auszuführen.“

Eine exakte Aussage über die Menge existierender Schwachstellen zu treffen ist unmöglich. Das liegt darin begründet, dass es keine allumfassende Quelle gibt, die Informationen über sämtliche entdeckte Fälle enthält. Ein weiteres Problem beim Betrachten des Umfangs von Sicherheitslücken

[1] Information Technology, Informationstechnik

besteht darin, dass ein großer Teil von den Softwareherstellern selbst entdeckt, aber nie veröffentlicht werden. Auf wichtige verfügbare Quellen wird im Kapitel 3.2 „Ausgewählte Informationsquellen“ näher eingegangen.

2.1.4 Veröffentlichungen

Anbieter von Software haben verschiedene Möglichkeiten, sich über Sicherheitslücken in ihren Produkten zu informieren und gegebenenfalls darauf zu reagieren. In der Praxis ist das jedoch keineswegs an der Tagesordnung, denn nur wenige Entwickler nutzen die gegebenen Möglichkeiten, um nach Meldungen über eigene Produkte zu suchen. So kommt es vor, dass selbst auf hoch kritische Schwachstellen monatelang keine Reaktion erfolgt. Bei der Untersuchung einzelner Anwendungen zeigt sich, dass öffentlich bekannte Sicherheitslücken auch in den Folgeversionen der Anwendung wieder auftreten.

Der Idealfall für den Umgang mit einer Schwachstelle durch den Hersteller ist das Beheben der Ursache in der nächsten Programmversion oder auch das Veröffentlichen eines Patches für die aktuelle Version. In dem Bericht „Die Lage der IT-Sicherheit in Deutschland 2009“ [Bunb] schreibt das Bundesamt für Sicherheit in der Informationstechnik jedoch:

> „Für rund die Hälfte der neu gemeldeten Schwachstellen wurde von den Herstellern der Produkte kein Update zur Behebung des Sicherheitsproblems bereitgestellt.“

Der Software-Anwender auf der anderen Seite hat nicht immer ausreichend Möglichkeiten, sich direkt bei dem Hersteller über Schwachstellen zu informieren. Obwohl für die meisten Anwendungen ein detaillierter Verlauf von Veränderungen in den einzelnen Programmversionen verfügbar ist, sind eventuell behobene Schwachstellen dort nicht immer detailliert beschrieben. Das Bekanntwerden einer Schwachstelle wird von dem Hersteller als Imageverlust der entsprechenden Anwendung betrachtet und diesen versucht er nach Möglichkeit zu unterbinden.

Weiterhin sind sich viele Anwender gar nicht der Tatsache bewusst, dass auch für ihre Anwendungen auf dem Rechner Updates verfügbar – und unter Sicherheitsaspekten oft sogar erforderlich – sind. Da wenig versierte Anwender mit der manuellen Installation von Updates überfordert sind, stellen Anwendungen ohne automatischen Update-Mechanismus eine besondere Gefahr dar.

Eine spezielle Rolle im Umgang mit Schwachstellen spielt derjenige, der eine entdeckt. Für diesen Fall gibt es drei Möglichkeiten, die den Umgang mit dem Fund einer Schwachstelle beschreiben. Eine allgemein angewandte Methode ist die *Responsible Disclosure*[2]. Dabei wird nach dem Fund einer Schwachstelle als erstes der Hersteller informiert. Erst nach einer angemessenen Frist werden dann Einzelheiten zur Schwachstelle an den entsprechenden Stellen veröffentlicht. Der Hersteller

[2] englisch für „verantwortungsbewusstes Aufdecken“

der betreffenden Software erhält damit die Möglichkeit, das Problem selbst durch ein Update oder Patch zu beheben und zusammen mit der Meldung über die Schwachstelle zu veröffentlichen.

In diesem Zusammenhang ist nach [HR] auch noch *Full Disclosure* erwähnenswert. Sie beinhaltet das sofortige Veröffentlichen detaillierter Informationen über eine Schwachstelle. Eine Anleitung, wie diese ausgenutzt werden kann, wird dazu auch herausgegeben. Damit hat jeder Angreifer die Möglichkeit, eine Schwachstelle auszunutzen, bevor der Hersteller eine Gegenmaßnahme treffen kann. Auch wenn Sicherheitsspezialisten diese Strategie zum Teil für unverzichtbar halten, existieren auch viele Stimmen gegen Full Disclosure. Die Hacker-Gruppierung „Anti-Sec“ begründet die Ablehung so:

> „Die vollständige Offenlegung mit Veröffentlichung von Exploits diene nur dazu, Schreckenszenarien zu malen, um Leute zum Kauf einer Firewall, Antiviren-Software und der Beauftragung von Auditing-Dienstleitstungen zu bringen [Heib].“

Im Gegensatz dazu sieht *No Disclosure* keine Bekanntmachung von Schwachstellen vor. Diese Strategie geht davon aus, dass die Verantwortung für Softwaresicherheit allein beim Hersteller liegt. Obwohl alle drei genannten Strategien Vor- und Nachteile besitzen, sprechen sich die meisten Unternehmen für Responsible Disclosure aus. So schreibt das Unternehmen „Google Inc.“ in [Goo] zu diesem Thema:

> „Ein derart verantwortungsbewusstes Verhalten ist wichtig für das Ökosystem Internet. Unternehmen wie Google haben so die Möglichkeit, die Sicherheit für Nutzer zu gewährleisten, indem sie Sicherheitslücken und -probleme beheben, bevor Leute mit schlechten Absichten davon erfahren. Wir möchten jeden, der Interesse hat, Sicherheitslücken zu suchen und zu melden, auffordern, sich an die Umgangsformen und Verhaltensregeln der Responsible Disclosure zu halten.“

2.1.5 Common Vulnerability Scoring System

2.1.5.1 Einleitung

Wird eine neue Schwachstelle entdeckt, ist es wichtig, diese möglichst genau definieren zu können. Die Merkmale und Auswirkungen in Worten zu erklären, ist dabei unerlässlich. Derjenige, der die Schwachstelle zu dokumentieren versucht, stößt bei der Beschreibung des Schweregrads jedoch sehr schnell an eine Grenze. Qualitative Aussagen wie „wichtig“ oder „kritisch“ sind zwar für jeden verständlich, sie beantworten jedoch nicht die Frage, wie dieser Schweregrad genau definiert ist oder ob eine „kritische“ Schwachstelle bei Hersteller A genauso eingestuft wird wie bei Hersteller B.

Es besteht daher die Notwendigkeit, eine Schwachstelle quantitativ beschreiben zu können. Bestrebungen dieser Art gibt es schon seit Längerem, jedoch galt bis ins Jahr 2005 laut [Heid]:

> „Bisherigen proprietären Rating-Systemen zur Risikoeinschätzung von Sicherheitslücken ist gemeinsam, dass sich keines davon als überlegen erwiesen hat oder sich durchgesetzt hätte – jeder Sicherheitsdienstleister kocht noch immer sein eigenes Süppchen."

Im Jahr 2004 starteten mehrere namhafte Hersteller gemeinsam unter der Aufsicht des US-amerikanischen *National Infrastructure Advisory Council*[3] die Entwicklung eines einheitlichen Systems mit dem Ziel, Schwachstellen vergleichbar zu machen [Sei]. Im Frühjahr 2005 fingen dann rund 30 Hersteller an, dieses System mit der Bezeichnung *Common Vulnerability Scoring System* zu testen. Es nutzt dabei einheitliche und plattformübergreifende Maßstäbe zur Beschreibung des Risikopotenzials aufgedeckter Schwachstellen. CVSS[4] verwendet dafür Zahlenwerte, so genannte *Scores*, die sich an einer Skala von 1 bis 10 orientieren.

Mehrere Konzerne stellten CVSS 2005 auf einer Konferenz vor und sahen es als Voraussetzung für den Erfolg des Systems an, dass möglichst alle unabhängigen Softwarehersteller und -händler CVSS bald in ihren Produkten unterstützen. Daraufhin machte sich auch das *Forum of Incident Response and Security Teams*, das die weltweite Entwicklungsarbeit koordinierte, für eine breitere Unterstützung von CVSS stark [FIR].

Dies Ziel wurde binnen kurzer Zeit erreicht, denn immer mehr Unternehmen setzten fortan CVSS ein. Dazu kamen auch Branchengrößen wie die „Microsoft Corporation", die sich nach der Veröffentlichung von CVSS zunächst kritisch dazu äußerten und an dem eigenen Bewertungssystem festhalten wollten. Mit CVSS entstand also die erste freie und einheitliche Bewertungsmethodik, die stringente Bewertungen ermöglicht. In die CVSS-Bewertung fließen nicht nur externe Faktoren ein, sondern auch vom Administrator festgelegte Kriterien. Die Beschreibung erfolgt in drei Abschnitten.

2.1.5.2 Base Score

Die Basisbewertung *Base Score* erklärt, wie gefährlich die Schwachstelle ist. Dabei wird definiert, ob sie lokal oder über das Netzwerk ausgenutzt werden kann („Access Vector"), wie komplex sich ein möglicher Angriff darstellt („Access Complexity") und ob der Angreifer eine Authentifizierung vornehmen muss („Authentication"). Anhand dieser drei Merkmale ermittelt CVSS den so genannten *Exploitability Subscore*. Zur Basisbewertung werden zusätzlich noch die Auswirkungen auf die drei in [Micb] definierten Sicherheitsziele Integrität, Verfügbarkeit und Vertraulichkeit berücksichtigt und im *Impact Subscore* zusammengefasst.

Aus den beiden beschriebenen Subscores berechnet sich der Base Score anhand folgender Gleichung:
$Base\ Score = (0{,}6 \cdot Impact\ Subscore + 0{,}4 \cdot Exploitability\ Subscore - 1{,}5) \cdot f(Impact\ Subscore)$
Dabei ist $f(Impact\ Subscore) = 1{,}176$, sobald eines der drei Sicherheitsziele betroffen ist. Ist dies

[3] englisch für „Beirat für die nationale Infrastruktur"
[4] Common Vulnerability Scoring System

für alle drei Punkte nicht zutreffend, gilt $f(Impact\ Subscore) = 0$ und damit auch $Base\ Score = 0$. Die genauen Berechnungsgrundlage für alle Scores sind öffentlich verfügbar und das NIST stellt sie unter [Natc] zur Verfügung.

Die Score-Werte geben auf einen Blick eine gute Übersicht darüber, wie stark die Auswirkungen der Schwachstelle sind. Für die detaillierte Betrachtung sind sie jedoch nicht ausreichend. Speziell für die maschinelle Auswertung stellt das System den so genannten *CVSS Base Vector* zur Verfügung. Dieser Vektor ist eine 26-stellige Zeichenkette in der Form

```
(AV:[L,A,N]/AC:[H,M,L]/Au:[N,S,M]/C:[N,P,C]/I:[N,P,C]/A:[N,P,C])
```

Dieser Vektor beschreibt alle sechs Merkmale des Base Score, die durch einen Schrägstrich voneinander getrennt sind. In den eckigen Klammern sind jeweils die drei möglichen Werte angegeben. Ein Base Vector kann beispielsweise

```
(AV:N/AC:M/Au:M/C:C/I:N/A:P)
```

sein. Das erste Element `AV` steht für den „Access Vector“, gefolgt von einem Doppelpunkt und einem der drei Buchstaben `L` für „Local Access“ (lokalen Zugriff), `A` für „Adjacent Network“ (benachbartes Netzwerk) oder `N` für „Network“. Die Bedeutung aller Vektor-Elemente sind auf der Website des NIST[5] unter „Common Vulnerability Scoring System Version 2 Calculator“ [Nata] dokumentiert.

2.1.5.3 Environmental Score

Die zweite CVSS-Bewertung ist der unternehmensspezifische *Environmental Score*. Er hat das Ziel, die Umgebung des eigenen Unternehmens in die Bewertung einfließen zu lassen. Dabei gibt „Collateral Damage Potential“ den maximalen potentiellen Verlust für das Unternehmen durch die Schwachstelle an. Dieser beinhaltet Produktivitäts- oder Umsatzverlust, Diebstahl oder die Gefahr für Leib und Leben. Das Merkmal „Target Distribution“ beschreibt den prozentualen Anteil der durch die Schwachstelle verwundbaren Systeme.

Die Integrität, Verfügbarkeit und Vertraulichkeit werden auch beim Environmental Score berücksichtigt. Hier wird jedoch nicht bestimmt, wie sich die Schwachstelle auswirkt, sondern wie groß die Anforderungen des Unternehmens an die Sicherheitsziele sind.

2.1.5.4 Temporal Score

Das Kriterium *Temporal Score* beschreibt die momentane Bedrohung durch die Schwachstelle. Im Gegensatz zum Base Score, der einmalig festgelegt wird, gilt der Temporal Score nur für den Moment und kann sich jederzeit verändern. Die drei Merkmale sind die Verfügbarkeit eines Exploits, die

[5] National Institute of Standards and Technology

momentane Möglichkeit des Patchens zur Umgehung beziehungsweise Behebung und der Grad der Sicherheit darüber, ob die Existenz der Schwachstelle überhaupt bestätigt werden kann.

Der bereits erwähnte CVSS Calculator bietet dem Anwender die Möglichkeit, nach dem Festlegen der CVSS-Merkmale die entsprechenden Scores zu berechnen. In Abbildung 2.1 sind die ermittelten Werte für eine definierte Kombination von Merkmalen dargestellt. Ein Klick auf „concise“[6] in der Kopfzeile der Webseite öffnet sich eine erweiterte Ansicht, die dem Anwender nach Eingabe der Merkmale auch der dazu gehörigen *CVSS v2 Vector* anzeigt. Dieser setzt sich aus der Basisbeschreibung sowie den drei Temporal- und fünf Environmental-Merkmalen zusammen. Der Vektor für das Beispiel in der Abbildung lautet

```
(AV:N/AC:M/Au:M/C:C/I:N/A:P/E:F/RL:T/RC:ND/CDP:L/TD:M/CR:M/IR:ND/AR:L)
```

und beschreibt die Schwachstelle gemäß CVSS vollständig.

2.1.6 Schwachstellen im Betriebssystem Microsoft Windows

Mircosoft Windows ist weltweit das am meisten eingesetzte Betriebssystem. Die genaue Verbreitung geben die verschiedenen Quellen sehr unterschiedlich an. Der durchschnittliche Marktanteil der aktuellen Version *Windows Vista* liegt demnach bei 10-15 Prozent, während der Vorgänger „Windows XP“ immer noch auf rund 70 Prozent aller Computer installiert ist.

Windows wurde ursprünglich zu dem Zweck entwickelt, für Laien bedienbar zu sein und auf möglichst viele Dokumente – auch gemeinsam im Netzwerk – einfach zugreifen zu können. Sicherheitsaspekte wie die Verwaltung von Zugriffs- und Benutzerrechten wurde erst in den jüngsten Versionen integriert. Im Oktober 2003 hat Microsoft den so genannten „Patch Day“[7] eingeführt. Seitdem werden Software-Aktualisierungen nicht mehr sofort nach Fertigstellung veröffentlicht, sondern gesammelt an jeden zweiten Dienstag eines Monats.

In seinem zweimal jährlich erscheinenden Bericht „Microsoft Security Intelligence Report“ beschreibt der Hersteller die aktuelle Entwicklungen sicherheitsrelevanter Themen. In der sechsten Ausgabe des Berichts zeigt der Hersteller die Entwicklung der industrieweit gemeldeten Veröffentlichungen von Sicherheitsschwachstellen in den Jahren 2003 bis 2008. In Abbildung 2.2 ist erkennbar, dass die Angriffsmöglichkeiten seit 2007 rückläufig sind.

[6] englisch für „kurzgefasst“
[7] englisch für „Patch-Tag“

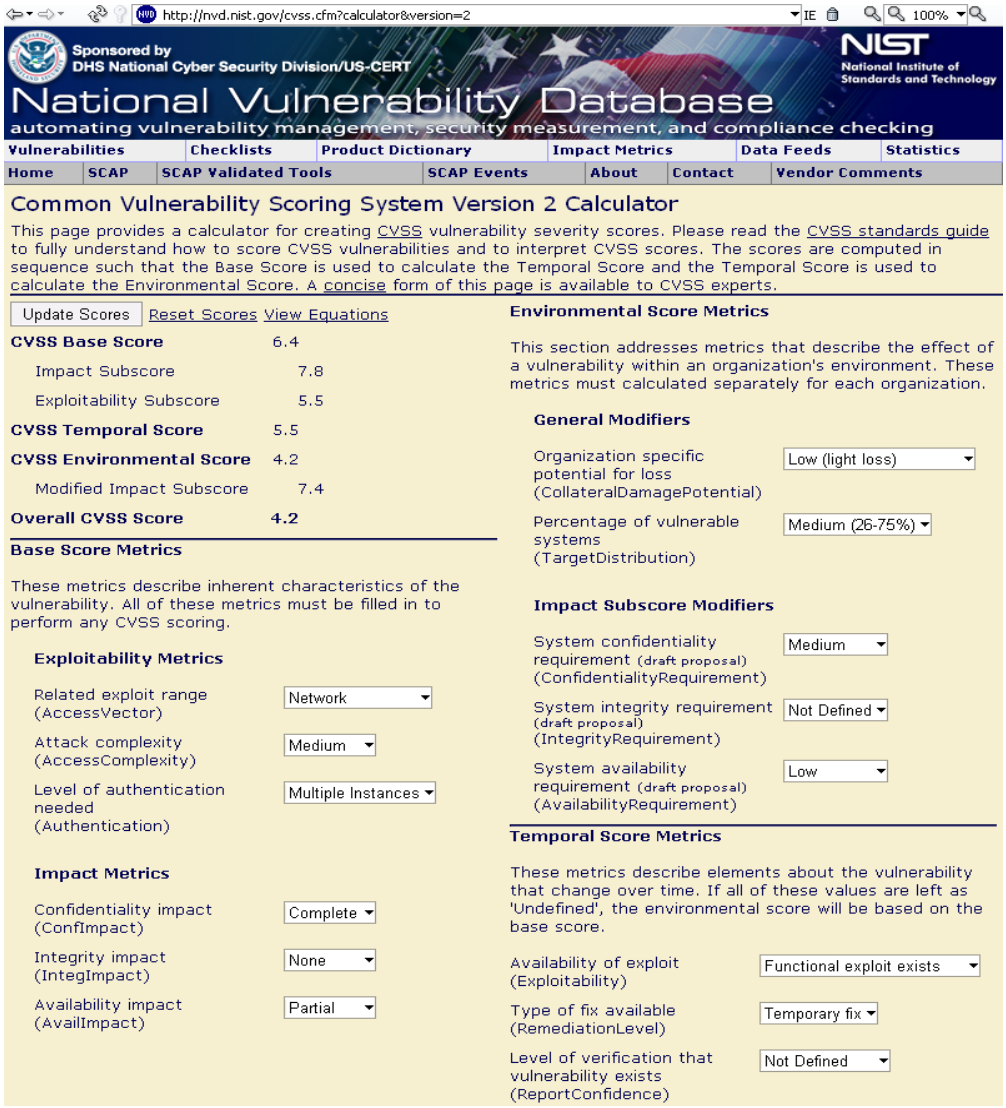


Abbildung 2.1: Bewertung von Merkmalen einer Schwachstelle durch den „Common Vulnerability Scoring System Version 2 Calculator“ – Stand: 6. September 2009 [Nata]

2.2 Sicherheitssoftware

2.2.1 Definitionen

2.2.1.1 Sicherheit

Sicherheit beschreibt einen Zustand, in dem man vor jeglicher Gefahr geschützt ist. Da es jedoch keinen absolut gefahren- beziehungsweise risikofreien Zustand gibt, kann es eine hundertprozentige Sicherheit ebenfalls nicht geben. Sie sollte daher immer als ein relativer Zustand der Gefahrenfreiheit angesehen werden, also auf bestimmte Rahmenbedingungen oder einen bestimmten Zeitraum bezogen. Daher darf der Anwender einer Sicherheitssoftware niemals hundertprozentigen Schutz vor Bedrohungen erwarten, sondern sie als eine ergänzende Maßnahme betrachten, die dennoch einen

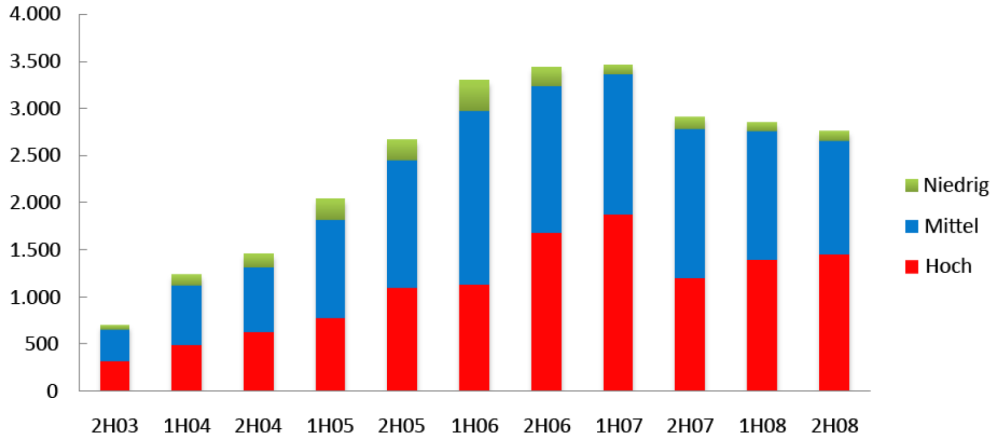


Abbildung 2.2: Industrieweit gemeldete Veröffentlichungen von Sicherheitsschwachstellen nach Schweregrad pro Halbjahr [Micb]

verantwortungsvollen Umgang erfordert.

2.2.1.2 Computervirus

Eine sehr gute Definition des Begriffs *Computervirus* gibt Claudia Eckert in [Eck07]:

> „Ein Computervirus ist eine Befehlsfolge, die ein Wirtsprogramm zur Ausführung benötigt. Viren sind zur Reproduktion fähig. Dazu wird bei der Ausführung des Virus eine Kopie (Reproduktion) oder eine modifizierte Version des Virus in einen Speicherbereich, der diese Befehlssequenz noch nicht enthält, geschrieben (Infektion). Zusätzlich zur Fähigkeit zur Reproduktion enthalten Viren in der Regel einen Schadensteil. Dieser kann unbedingt oder bedingt durch einen Auslöser aktiviert werden.“

Autoren von Computerviren handeln aus verschiedenen Beweggründen. Das Handbuch der Informations- und Kommunikationssicherheit [KW00] nennt im Abschnitt „Gefahren und Angriffe“ die sechs Motive Spieltrieb, Unzufriedenheit, Geltungsbedürfnis, Geldgier, Zerstörungswut und Machtausübung.

2.2.2 Einschränkung

Die Mehrheit der Anwender denken bei dem Beriff *Sicherheitssoftware* an Anwendungen zum Aufspüren, Erkennen und Beseitigen von Computerviren, also eine *Antiviren-Software*. Tatsächlich beschreibt der Begriff jedoch eine Produktgruppe von Programmen wie Personal Firewalls, Spam- oder Contentfilter, aber auch Software zur Datensicherung. Sicherheitssoftware hat dabei allgemein das Bestreben, die in Abschnitt 2.1.3 beschrieben Sicherheitsziele Integrität, Verfügbarkeit und Vertraulichkeit zu gewährleisten.

Antiviren-Programme werden häufig auch „Antivirus-Programme“ oder einfach nur „Virenscanner“ genannt. Viele Hersteller bieten sie zusammen mit anderer Sicherheitssoftware aus dem selben Unternehmen in einer gemeinsamen Oberfläche als so genannte *Suite* an. Die weit verbreitete Kombination aus Antiviren-Software, Personal Firewall und Spamfilter erhalten Bezeichnung wie „Internet Security Suite“. Diese Arbeit bezieht sich mit dem Begriff Sicherheitssoftware jedoch nur auf reine Antiviren-Programme. Die Beschränkung ist erforderlich, da ein Vergleich von Produkten mit völlig verschiedenen Anwendungsbereichen schwer zu realisieren ist.

Weiterhin ist zu beachten, dass als eingesetztes Betriebssystem „Microsoft Windows“ zu Grunde gelegt wird. Obwohl Antiviren-Software auch für andere Plattformen verfügbar ist, stellen diese Programme angesichts der weiten Verbreitung von Windows eine Randerscheinung dar und werden an dieser Stelle vernachlässigt.

2.2.3 Geschichtliche Entwicklung

Bis Ende der 1990er Jahre hat der Anwender ein Antiviren-Programm daran gemessen, wie viele Viren dem einzelnen Produkt bekannt sind. Da es jedoch kein zentrales Depot für Viren gab, war eine Angabe über die Anzahl der erkennbaren Viren irreführend. Unterschiedliche Auffassungen gab es darüber, ob der Scanner nur nach neu entdeckten Viren suchen, oder auch alte mit einbeziehen soll.

Auch heute ist es noch so, dass ein Virus unter verschiedenen Namen bekannt ist. Im Anhang A.2 zeigt die Abbildung vom 26. August 2009 das Ergebnis nach dem Scannen einer infizierten Datei durch 21 verschiedene Antiviren-Programme [Jot]. Nur zehn von ihnen erkannten den Virus, aber erstaunlicherweise tauchte dieser nur in zwei Fällen unter der selben Bezeichnung auf. Einzelne Hersteller zeigen mittlerweile Betrebungen zum Einführen einheitliche Standards. So haben sich die sechs Unternehmen McAfee Inc., Symantec Corporation, AVG Technologies, Sophos Plc, Microsoft Corporation und Trend Micro Inc. gemeinsam mit IEEE[8]-Mitgliedern im August 2009 zur *Industry Connections Security Group* [Heia, Ins] zusammengeschlossen.

Im Jahr 1993 begann der Wissenschaftslektor Joe Wells mit dem weltweiten Sammeln von Viren und erstellte eine zentrale Sammelstelle für Viren und nannte sie *WildList* [Wil]. Diese Liste unterscheidet zwischen dem Virustyp *in the wild* und *in the zoo*. Zur ersten Gruppe gehören sämtliche Viren, die bereits im freien Umlauf Systeme infiziert haben. Die zweite beinhaltet auch die so genannten *Laborviren*, die bis dato nicht aktiv infektiös vorkamen. Wells schuf dabei auch Namenskonventionen für Viren, um das Pflegen einer effizienten und durchsuchbaren Datenbank zu ermöglichen. Seine Virenbibliothek lässt er monatlich aktualisieren und stellt sie seriösen Antiviren-Forschern zur Verfügung.

[8] Institute of Electrical and Electronics Engineers

Kurz nach dem Schaffen der WildList entwickelte das Unternehmen *National Computer Security Association*[9] [ICS] ein kommerzielles Antiviren-Test- und Zertifizierungsverfahren. Dabei gründete der Virenexperte Dr. Richard Ford ein Viren-Testlabor, in dem die Hersteller ihre Produkte dahingehend testen lassen konnten, ob alle in der WildList enthaltenen Viren erkannt werden können. In den ersten Versuchen schafften es die meisten der getesteten Produkte zwar nicht, mehr als 80 Prozent der gelisteten Viren auszufiltern, aber dennoch wurde damit eine Standard-Testmethode geschaffen und die Hersteller bemühten sich fortan sehr, die Tests erfolgreich zu bestehen. Nach Verbesserungen in ihren Produkten gelang es immer mehr Herstellern, eine Zertifizierung für ihre Produkte zu erhalten um damit offiziell zu belegen, dass ihre Produkte zuverlässig funktionieren.

Eine der ersten professionellen Antiviren-Software, die auch heute noch häufig genutzt wird, ist *AntiVir*. Im Jahr 1988 veröffentlichte das zwei Jahre zuvor gegründete Unternehmen „H+BEDV Datentechnik“ die erste offizielle Version von AntiVir. Die *Scan-Engine* erhielt damals den Namen „Luke Filewalker“ und trägt ihn bis heute noch. Seit März 2006 heißt das Unternehmen „Avira GmbH“.

In den 1990er Jahren befanden sich bereits 19 verschiedene Antiviren-Software auf dem Markt. Darunter waren die auch heute noch bekannten Produkte „Norton AntiVirus“ und „McAfee VirusScan“. Bevor das Internet der breiten Bevölkerung zur Verfügung stand, verbreiteten sich Viren hauptsächlich über das Kopieren infizierter Dateien von einem auf den anderen Rechner – sehr häufig über magnetische Datenträger wie Disketten. Antiviren-Programme beschränkten sich in dieser Zeit auf die Überprüfung aller ausführbaren Dateien und Boot-Sektoren der jeweiligen Datenträger.

Mit der Verbreitung von Anwendungen, die Makros einsetzen, kamen auch immer mehr spezielle Makro-Viren in Umlauf, die sich versteckt in Dokumenten einbetten konnten. Später sorgte das einfache Anzeigen von E-Mails mit schadhaftem HTML[10]-Code in Programmen wie „Microsoft Outlook“ beziehungsweise „Microsoft Outlook Express“ für ganz neue Angriffsmöglichkeiten. Viren verbreiteten sich fortan nicht mehr nur über eine bestimmte Gruppe von Datei-Typen und die Antiviren-Programme waren nun ganz neuen Anforderungen ausgesetzt.

2.2.4 Die aktuelle Situation

Das Bundesamt für Sicherheit in der Informationstechnik schreibt in seinem Lagebericht zur IT-Sicherheit in Deutschland 2009 [Bunb]:

> „Wurden vor zwei Jahren die meisten Schadprogramme per E-Mail verschickt, erfolgt die Verbreitung inzwischen in großer Zahl über präparierte Webseiten [...] Bislang arbeitet Schutzsoftware hauptsächlich signaturbasiert, das heißt, ein Virenschutzprogramm

[9] 1997 nannte sich das Unternehmen in „ICSA Labs“ um
[10] Hypertext Markup Language

erkennt nur bereits bekannte Schadprogramme. Diese Technik ist an ihre Grenzen gestoßen, so dass intensiv an Technologien gearbeitet wird, die neue Schadprogramme auch an ihren Eigenschaften bzw. ihrem Verhalten erkennen können. Das Problem dabei: Verhaltensbasierte Erkennungsverfahren führen immer zu einer höheren Anzahl von unberechtigten Alarmen."

Zu den so genannten „Onlinern", also den Nutzern des Internets, zählen laut [Ini] 2009 bereits zwei von drei Bürger. Die Abbildung 2.3 aus der Studie veranschaulicht die Entwicklung der letzten neun Jahre.

	'01	'02	'03	'04	'05	'06	'07	'08	'09
IN	37,0	41,7	50,1	52,7	55,1	58,2	60,2	65,1	69,1
BB					22,1	27,7	35,9	42,6	46,2
SB					26,5	24,8	18,7	15,9	16,3
Sonstiges / k. A.					6,5	5,7	5,6	6,6	6,6

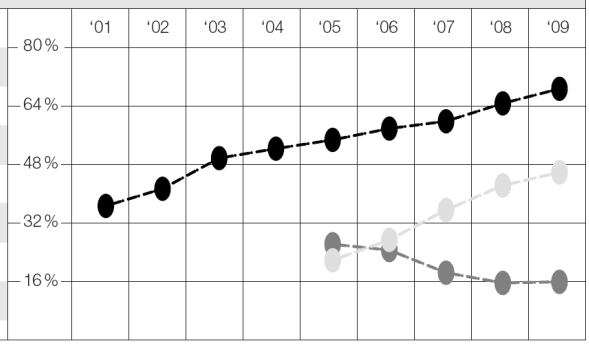


Abbildung 2.3: Internetnutzung der deutschen Bevölkerung ab 14 Jahren von 2001 bis 2009 [Ini]

Mit zunehmender Nutzung des Internets durch die breite Masse der Bevölkerung spielt das Streben nach Sicherheit in der IT eine immer wichtigere Rolle. Die Möglichkeiten der Bedrohung für die Anwender haben sich im Laufe der letzten Jahre vervielfacht und auf der anderen Seite nimmt das Angebot an Software, die den Anwender vor diesen Bedrohungen schützen will, in ähnlicher Weise zu. Die Thematik der Sicherheitssoftware ist also aktuell wie nie.

2.2.5 Methoden zur Virenerkennung

Ein Virus ist eine eigene Anwendung und ein Virenscanner muss daher untersuchen, wie dieser funktioniert. Um den Aufwand des Analysierens jeder Programmzeile zu vermeiden, sucht der Scanner nach Hinweisen, die auf einen Virus schließen lassen.

Ein Antiviren-Programm versucht auf verschiedenen Wegen Infektionen zu erkennen, das Computersystem vor ihr zu schützen und gegebenenfalls den Virus zu entfernen. Dabei existieren drei grundlegende Methoden. Die erste ist das Erkennen bekannter Viren anhand der Virensignaturen, die das Programm in seiner Datenbank gespeichert hat. Eine Signatur besteht aus einer kleinen Sequenz von Bytes aus dem Programmcode des betreffenden Virus. Ein Problem bei dieser Methode ist, dass die erkannte Sequenz, die zu einem Virus gehört, auch zufällig Bestandteil des regulären Programmcodes sein kann. Eine Garantie, dass die Datei tatsächlich infiziert ist, kann die Antivi-

ren-Software nicht geben. Beim Finden einer bekannten Sequenz hat der Anwender je nach Konfiguration verschiedene Aktionen zur Auswahl. Er kann einen Versuch zum Desinfizieren der Datei unternehmen, sie in einen geschützten Bereich – „Quarantäne“ genannt – verschieben oder komplett löschen. Einige Anwendungen bieten weiterhin die Option an, die betroffene Datei an die herstellereigenen Laboratorien zu senden. Dort können Spezialisten überprüfen, ob es sich tatsächlich um einen Virusbefall handelt oder nicht. Da ständig neue Viren oder neue Varianten bekannter Viren programmiert und in Umlauf gebracht werden, stellen Hersteller von Antivirus-Software regelmäßig Aktualisierungen der Virensignaturen ihrer Software zur Verfügung. Der Anwender ist angehalten, so häufig wie möglich davon Gebrauch machen, da die Software mit dieser Methode sonst neue Viren nicht entdecken kann. Ein Problem bei dieser Methode ist, dass Antiviren-Programme damit nur Viren abfangen können, die sie bereits kennen.

Eine andere Methode ist der „Integritäts-Check“. Dabei überwacht die Software Änderungen an Dateien und Programmen beziehungsweise bereits den Versuch, eine Änderung vorzunehmen. Viele Antiviren-Programme speichern Informationen über die vorhandenen Dateien anhand von Prüfsummen. Da ein Virus bei der Infektion den Inhalt der betreffenden Datei verändert, ergibt sich dabei auch eine neue Prüfsumme, die beim Scannen mit der gespeicherten verglichen wird. Findet die Antiviren-Software Veränderungen, dann gibt es einen konkreten Verdacht auf eine Infektion der betreffenden Datei. Ein Nachteil dieser Methode ist, dass sie sich eigentlich nur zum Schutz von Dateien eignet, die unverändert bleiben. Bei Dokumenten, an denen der Anwender jederzeit Änderungen vornehmen kann, ändert sich auch ständig die Prüfsumme, die das Antiviren-Programm permanent vergleichen müsste.

Mit den beiden beschriebenen Methoden wird es immer schwerer, der rasanten Viren-Entwicklung standzuhalten. Daher bedienen sich die meisten Anwendungen auch der so genannten „heuristischen Suche“. Bei dieser Methode sucht die Antiviren-Software mit Hilfe regelbasierter Logik nach verdächtigen Aktivitäten, wobei typische Merkmale dazu dienen, einen neuen Virus zu erkennen. Die heuristischen Algorithmen stellen Vermutungen darüber an, wie ein Virus aussehen könnte. Dabei wird die Struktur eines Programms, seine Attribute und sein Verhalten analysiert. Falls die betreffende Anwendung nach der Analyse wie ein Virus aussieht und sich wie ein Virus verhält, geht der Scanner auch davon aus, dass es sich dabei auch um einen Virus handelt. Die Reaktionen auf diese Annahme können von Produkt zu Produkt sehr unterschiedlich sein. Ein großer Vorteil dieser Methode liegt darin, daß die Scan-Engine Viren schon vor der Infizierung erkennen und abwehren kann. Es ist jedoch unmöglich, ganz sicher zu erraten, was ein neuer Virus ist. Daher treten bei der heuristischen Suche verhältnismäßig viele „false positive“-Ergebnisse auf.

Zusammenfassend lässt sich sagen, dass die Virenerkennung nie eine exakte Wissenschaft sein kann und es daher unmöglich ist, eine Antiviren-Software zu entwickeln, die hundertprozentig fehlerfrei arbeitet.

2.2.6 Anfälligkeiten

Komplexe Produkte wie Computerprogramme können je nach Funktionsumfang sehr viel Programmcode enthalten. Ein großer Teil davon dient der Vermeidung oder Behandlung von Fehlern, wie beispielsweise triviales Überprüfen von Umgebungsbedingungen oder das Bestätigen von Korrekturmaßnahmen. Sobald ein Programm ein nicht erwartetes oder nicht erwartbares Verhalten zeigt, gibt es keine vorgesehene Fehlervermeidung oder -behandlung und es besitzt einen Programmfehler. Tritt dagegen eine Situation auf, in der der Anwender eine spezielle Fehlermeldung erhält, dann ist das im Normalfall keine Fehlersituation, da die Situation im Programmablauf vorhersehbar war.

Die Komplexität der Anwendungen steigt stetig an, da Software im Allgemeinen immer neuen Anforderungen genügen muss und je umfangreicher ein Programm ist, desto höher ist auch die Wahrscheinlichkeit, dass es einen oder mehrere Fehler enthält. Sobald dabei die Integrität, Verfügbarkeit oder Vertraulichkeit des Programms gefährdet wird, besitzt das Programm eine Schwachstelle. Der Anbieter von Sicherheitssoftware „Kaspersky Lab“ schreibt dazu:

> „Es muss auch gesagt werden, dass theoretisch alle Computersysteme Schwachstellen haben – der Unterschied, ob sie praktisch ausgenutzt werden können oder nicht, liegt in den Kosten eines Angriffs [Kas].“

Bei der Suche nach Gründen, die zum Entstehen von Sicherheitslücken führen können, zeigt sich, dass ein großer Teil versehentlich bei der Entwicklung von Software entsteht. So können fehlende oder eingeschränkte Tests von Beta-Versionen oder ein allgemeiner Mangel an Qualitätskontrolle eine wichtige Rolle spielen. Die Analyse des Programmcodes hinsichtlich des Aufspürens von möglichen Sicherheitslücken ist für viele Hersteller sehr arbeits- und daher auch kostenintensiv. Häufig sehen die Hersteller die Mehrkosten für das Durchforsten als nicht gerechtfertigt an.

Im Bereich der Sicherheitssoftware führen Programmierfehler zu besonders sicherheitskritischen zu Schwachstellen. Als häufigste Ursache dafür gilt eine der Kernfunktionen der Antiviren-Programme, dem Zerlegen der Daten in analysierbare Einzelteile, also das Parsen.

Die Zeitschrift *Computerwoche* [Fri07] schreibt im November 2007 in dem Artikel „Virenscanner öffnen Hackern die Türen“:

> „Antiviren-Lösungen ermöglichen genau das, wogegen sie eigentlich schützen sollen: das Einschleusen und Ausführen von Schadcode [...] Die Virenscanner helfen sogar dabei, den Code auszuführen. ‚Von den Schwachstellen war jede am Markt befindliche Scan-Engine gleich mehrfach betroffen‘, verdeutlicht Thierry Zoller, Security Engineer.“

Mittlerweise existieren bis zu 3 000 verschiedene Formate und jede Antiviren-Software ist bestrebt, davon so viel wie möglich zu erkennen, um möglichst viele Schädlinge ausfindig machen zu können. Die dabei entstehende Fehleranfälligkeit steigt mit der Unterstützung weiterer Dateiformate ent-

sprechend an. Generell kann gesagt werden: Je mehr Dateiformate ein Scanner unterstützt, desto größer ist die Wahrscheinlichkeit, dass dieser fehleranfällig ist.

Die Anbieter von Antiviren-Software haben weiterhin einen hohen Druck durch die Konkurrenz und die Kunden. Dieser erwartet einerseits, dass die Software einen neuen Schädling innerhalb von Stunden nach dem ersten Auftreten erkennt, andererseits soll der Virenscanner möglichst unbemerkt im Hintergrund arbeiten. Der Zeitdruck zwingt die Programmierer dazu, von jeder Möglichkeit Gebrauch zu machen, die den Entwicklungsprozess abkürzt. Thierry Zoller beschreibt die Situation so:

> „Die Aniviren-Industrie und deren Entwickler stehen unter enormem Zeitdruck: Hier geht es darum, wer am schnellsten neue Gefahren erkennt - was die Qualität des Codes nicht unbedingt steigert [Fri07]."

2.3 Integration von Antiviren-Software

2.3.1 Einleitung

Das Thema IT-Sicherheit nimmt eine immer wichtigere Rolle im Alltag der Menschen ein. Nicht nur Unternehmen, sondern auch immer mehr Privatanwender versuchen daher durch technische und organisatorische Maßnahmen ihre Werte zu schützen. Über die durch Schwachstellen entstehenden Schäden schreibt der Autor von [SR05] in seinem Buch:

> „Populärste Beispiele sind die in immer kürzeren Abständen bekannt werdenden neuen Computerviren (insgesamt gibt es mittlerweile über 80.000!). Deren rasante Verbreitung ist nur durch unzureichende Sicherheitsmaßnahmen im Privat- und Geschäftsbereich zu erklären. Jährlich entstehen durch Schadsoftware weltweit immer noch Schäden in zweistelliger Milliardenhöhe, wovon in sehr hohem Maße auch mittelständische Unternehmen betroffen sind."

Der folgende Abschnitt vergleicht die unterschiedlichen Umstände bei der Einbindung einer Antiviren-Software in das Betriebssystem Microsoft Windows für einen einzelnen privat genutzten Computer und einen, der in das Netzwerk eines Unternehmens eingebunden ist.

2.3.2 Privatanwender

2.3.2.1 Einleitung

Der Anwender, der eine Antiviren-Software auf seinem Computer einsetzen will, muss sich zwischen einer Vielzahl von Produkten entscheiden. Redaktionen von Computerzeitschriften oder Organisa-

tionen, die sich auf das Testen von Produkten spezialisiert haben, untersuchen regelmäßig die auf dem Markt befindlichen Programme. Die Ergebnisse können jedoch nicht repräsentativ sein, da jedem Test andere Kriterien und Prüfverfahren zu Grunde liegen. Während einige Anwendungen mit zahlreichen Nachweisen über gutes Abschneiden in Software-Tests etikettiert sind, halten sich andere Hersteller dabei zurück und versuchen den Kunden durch technische Details zu überzeugen.

2.3.2.2 Installation und Konfiguration

Hat sich der Anwender für eine Antiviren-Software entschieden, legt er bereits bei der Installation fest, in welcher Weise der Computer zukünftig geschützt wird. Ein wichtiges Merkmal sind die Standard-Einstellungen, die der Hersteller seinem Produkt vergibt. Viele Anwender entscheiden sich für die einfache Installationsart, weil sie sich nicht mit Details beschäftigen möchten. Andere glauben, die Software würde die richtigen Einstellungen für den Computer selbst erkennen und verzichten daher – oft unbewusst – auf die Möglichkeiten, wichtige Einstellungen vorzunehmen.

Das Verhalten der Anwendung im Falle eines Virenfunds ist ein entscheidender Punkt. Immer mehr Hersteller sind dazu übergegangen, auftretende Probleme standardmäßig ohne Hinweise selbständig zu beheben. Unter Umständen ist es daher möglich, dass der Benutzer gar nichts von diesem Problem erfährt. Bei anderen Produkten ist es jedoch umgekehrt, so dass sie bei jeder Möglichkeit der Gefahr eine entsprechende Meldung herausgeben. Das kann eine Desensibilisierung der Anwender zur Folge haben, so dass mögliche ernsthafte Probleme dann unter Umständen ignoriert werden. Bereits bei der Installation oder direkt im Anschluss daran sollte der Benutzer selbst festlegen, über welche Ereignisse er informiert werden möchte und welche Aktionen die Antiviren-Software selbst durchführen darf.

Eine wichtige Regel für jeden Windows-Anwender ist, ein Benutzerkonto mit eingeschränkten Rechten zu verwenden. Wer das Betriebssystem mit vollständigen Administrator-Rechten nutzt, nimmt eine große Angriffsfläche in Kauf. Der Grund dafür ist, dass nicht nur der Nutzer, sondern auch potentielle Schadsoftware dadurch vollen Zugriff auf Systemdateien erhält. Leider ist das Betriebssystem selbst nach einer Standard-Installation so konfiguriert, dass der Benutzer diese Administrator-Rechte ungefragt erhält[11]. Der Zweck dieser Konfiguration besteht darin, den Anwendern die Installation von Anwendungen zu vereinfachen und – ohne sich Gedanken über Benutzer- und Zugriffsrechte Gedanken machen zu müssen – einen schnellen Zugriff auf das gesamte System zu ermöglichen. In der aktuellen Version „Windows Vista“ hat der Hersteller sein Produkt an dieser Stelle erweitert. Um die Sicherheit zu erhöhen, stellt dieses Betriebssystem dem Benutzer bei Versuchen, eine Anwendung zu installieren oder auf Systemdateien zuzugreifen, Sicherheitsfragen.

[11] gilt für das Betriebssystem „Windows XP“

2.3.2.3 Sicherheitsbewusstsein der Anwender

Annähernd 70 Prozent der Deutschen nutzen laut [Ini] im Jahr 2009 das Internet. Dabei sind ihnen Themen wie das Versenden und Empfangen privater E-Mails, sowie das Einkaufen und Abwickeln von Bankgeschäften über das Internet sehr wichtig, wie Abbildung 2.4 zeigt [AGO].

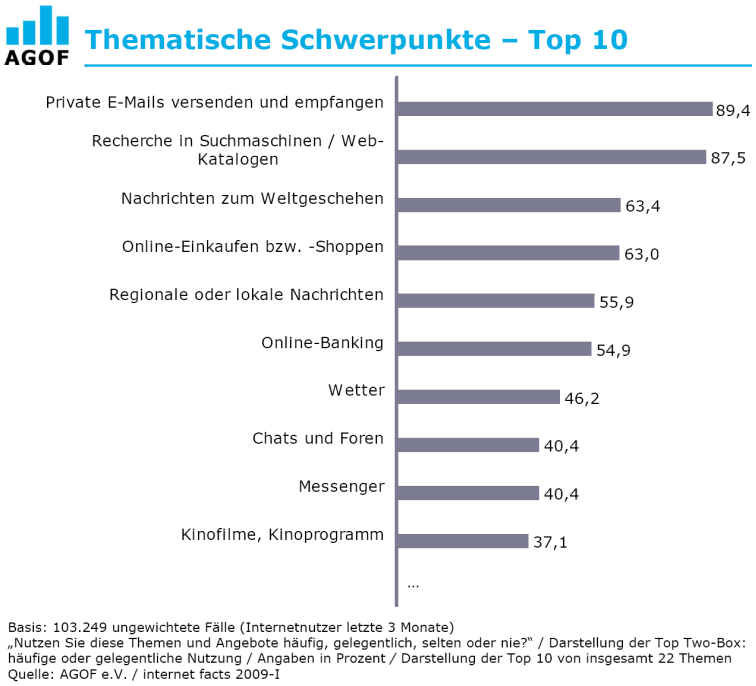


Abbildung 2.4: Die zehn wichtigsten thematischen Schwerpunkte der Internetnutzung im ersten Quartal 2009 [AGO]

Viele dieser thematischen Schwerpunkte erfordern eine gute Sicherheitsausstattung, um vertraulichen Informationen zu schützen. Personal Firewalls und Antiviren-Software sind heute weitgehend gängige Schutzmaßnahmen.

> „Immerhin sind fast vier Millionen Deutsche bereits Opfer von Computerkriminalität geworden und erlitten einen finanziellen Schaden, beispielsweise durch Viren, bei Online-Auktionen oder beim Online-Banking [Bunb]."

Auch wenn das Bundesamt für Sicherheit in der Informationstechnik von einem positiven Trend beim Sicherheitsbewusstsein der Deutschen spricht, unterschätzen viele Anwender die Gefahr. Auch wenn sie vom Benutzer oft nicht bemerkt werden, existieren Schwachstellen bei viel genutzten Programmen wie Webbrowser, Office-Anwendungen, E-Mail-Client oder sogar Bildbetrachtungs-Software. Für viele dieser Anwendungen stellen die Hersteller Sicherheits-Updates zur Verfügung, doch nicht immer informiert die Software den Benutzer darüber, dass eine Aktualisierung bereitsteht.

Eine Möglichkeit, sich über die Aktualität der verwendeten Software auf dem eigenen Rechner zu informieren, bietet die für Privatanwender kostenlose Anwendung „Secunia Personal Software In-

spector“ [Secb]. Sie durchsucht die Datenträger und überprüft, ob der Rechner durch möglicherweise veraltete Programmversionen potentielle Schwachstellen aufweist. Ein Resultat dieser Untersuchung ist im Anhang A.3 dargestellt.

Durch die statistischen Auswertungsmöglichkeiten von „Secunia Personal Software Inspector“[12] stellte Secunia am 7. Februar 2008 fest, dass auf 81 Prozent aller Computer durch das Verwenden mindestens einer veralteten Anwendung Schwachstellen aufweist. Besonders stark betroffen waren dabei die Programme „Adobe Reader 8“ auf 61 Prozent der Rechner und „Apple QuickTime 7“, was knapp die Hälfte der Rechner eingesetzt haben [Seca].

2.3.2.4 Scan-Typen

Die klassische Variante ist das Scannen, das der Benutzer selbst veranlasst oder durch einen festgelegten Zeitplan ausgelöst wird. Alle Hersteller von Antiviren-Software empfehlen den so genannten *On-Demand*[13]-Scan in festgelegten Zeitabständen durchzuführen beziehungsweise durchführen zu lassen. Der andere Scan-Typ ist der *On-Access*[14]- oder auch Echtzeit-Scan, der als Hintergrund-Dienst im Arbeitsspeicher geladen ist und sämtlichen lesenden und schreibenden Daten- und Netzwerkverkehr überprüft. Trotz aktivem On-Access-Scanner sollten regelmäßig geplante Überprüfungen der Datenträger ausgeführt werden, um möglicherweise mit aktuellen Virensignaturen einen bisher unerkannt gebliebenen Schädling entdecken zu können.

2.3.2.5 Schwierigkeiten

Neben den möglichen Schwachstellen, die beim Parsen unbekanter Dateiformate entstehen können, bringt der Einsatz von Antiviren-Programmen noch weitere Schwierigkeiten mit sich. So können „false positive“-Ergebnisse je nach Konfiguration der Antiviren-Software den Zugriff auf die angeblich infizierte Datei verhindern, ohne dass der Anwender die Möglichkeit hat, dies zu verhindern.

Für besondere Schwierigkeiten sorgt in einigen Fällen der „On-Access-Scanner“ der Antiviren-Software. Ein generelles Problem dabei ist, dass er zwar bestimmte Dateiformate von der Überprüfung ausschließen kann, die permanente Überwachung von Aktivitäten wirkt sich aber dennoch negativ auf das Arbeiten am Computer aus. Je nach Antiviren-Programm und Leistungsfähigkeit der Hardware kann das von minimalen Geschwindigkeits-Einbußen bis hin zum temporären Blockieren des gesamten Betriebssystems reichen. Die durch das Scannen der Daten entstehene Verzögerung kann aber bei Anwendungen mit zeitkritischen Funktionen so groß werden, dass es zu Störungen oder Fehlern bei der Ausführung kommt. Weitere Komplikationen kann der Scanner hervorrufen,

[12] der Hersteller sichert in den „Privacy Statements“ zu, die ermittelten Daten anonym auszuwerten
[13] englisch für „auf Nachfrage“
[14] englisch für „beim Zugriff“

wenn ein permanenter Zugriff auf eine bestimmte Datenmenge besteht. Hat ein Rechner beispielsweise eine dauerhafte Verbindung zu einer Datenbank aufgebaut, versuchen einige Scanner auch die ganze Zeit, diese zu überprüfen. Das kann im Extremfall zu Zeitüberschreitungen beim Zugriff bis hin zu Beschädigungen an der Datenbank führen. Ein weiteres Problem ist möglich, wenn der On-Access-Scanner Zugriff auf einen Mailserver hat. Liegt dort in einer E-Mail eine Virusinfektion vor, dann kann das Entfernen der betreffenden E-Mail zu Funktionsstörungen führen, wenn diese vom Mailserver nicht mehr gefunden wird.

2.3.3 Computer in einem Unternehmen

2.3.3.1 Szenario eines ungeschützten Unternehmens

Einleitend lässt sich die Notwendigkeit von Sicherheitssoftware in einem Unternehmen an folgendem fiktiven Szenario erklären: In einem Unternehmen, dass weder Türen noch Fenster verschließen kann, hat jeder Passant die Möglichkeit, jederzeit alle Räume des Unternehmens frei zu betreten. Die gesamte Firmenpost ist offen zugänglich und alle Gegenstände können entwendet und bei Bedarf auch zerstört werden. Herumliegende Briefe können mit falschem Namen unterschrieben werden und Ausweise der Mitarbeiter sind ebenfalls für jeden nutzbar. Weiterhin werden Produktionsanlagen der Firma sabotiert und können nicht mehr verwendet werden.

Die eben beschriebene Situation würde, auf die Informationstechnik übertragen, so lauten: Das Firmen-Netzwerk ist direkt mit dem Internet verbunden und die Passwörter für den Zugang der einzelnen Benutzerkonten sind auf einem Zettel neben dem Monitor des jeweiligen Rechners vermerkt. Innerhalb des Netzwerks haben alle Benutzer vollen Zugriff auf die nicht durch Zugriffsrechte geschützten Unternehmensdaten. Diese kann also jeder lesen, verändern oder auch löschen, um beispielsweise die Website des Unternehmens zu verändern. Die Übertragung aller E-Mails in dem Unternehmen erfolgt unverschlüsselt und jeder Interessierte kann sie mitlesen.

Dieses Szenario ist ein Extrembeispiel für die Vernachlässigung von Sicherheitsmaßnahmen und verdeutlicht, wie wichtig die Sicherheit in der Informationstechnik für ein Unternehmen ist.

2.3.3.2 Notwendigkeit von Sicherheitssoftware

Seit einigen Jahren wird der Wert eines Unternehmens nicht mehr allein anhand des Substanz- oder Eigenwerts bemessen, sondern muss sich der Herausforderung des Einsatzes neuer Technologien stellen, um wettbewerbsfähig zu sein und effizient arbeiten zu können. Durch hohe Innovationskraft und intelligenten Einsatz moderner Informationstechnik kann der Unternehmenswert deutlich ansteigen, auf der anderen Seite bedeutet die Kompromittierung oder der Verlust von Informationen und Wissen einen großen Schaden für jedes Unternehmen. Dieser kann finanziell sein, aber auch das

Ansehen der Firma negativ beeinträchtigen, wenn Kunden oder Mitbewerber von diesem Schaden erfahren. Ein einfaches Beispiel dafür ist der Ausfall der Firmen-Website.

Der Schutz von Unternehmensinformationen endet heute nicht mehr an der Pforte und die Erfolgschancen werden in Zukunft immer stärker davon abhängig sein, ob ein sicherer Umgang mit Informationen gewährleistet und eine schnelle und sichere Beschaffung beziehungsweise Nutzung dieser Informationen möglich ist.

2.3.3.3 Besondere Gefährdung durch Antiviren-Software

Sicherheitssoftware ist in einem Unternehmensnetzwerk oft an zentralen Stellen platziert, an denen sensible zu schützende Daten verarbeitet beziehungsweise gespeichert werden. Auftretende Schwachstellen in einer Antiviren-Software kann für das Unternehmen daher sehr großen Schaden anrichten, da sie in der Regel mit vielen Zugriffsrechten ausgestattet sind und damit „attraktive“ Angriffsziele darstellen.

Antiviren-Software muss jeweils gezielt für den entsprechenden Einsatz ausgewählt sein. So existieren Produkte mit Stärken beim E-Mail-Schutz, wohingegen sich andere durch wenig Beeinträchtigung der Arbeitsgeschwindigkeit auszeichnen. In größeren Unternehmen sind Antiviren-Software verschiedener Hersteller zur Verbesserung der Erkennungsrate in Reihe geschaltet. Da jede einzelne Software jedoch von sich aus bereits potentiell Schwachstellen aufweist, gefährdet das jedoch die Sicherheit, anstatt sie zu erhöhen. Der Sicherheits-Forscher Thierry Zoller beschreibt die Situation von mehreren Virenscannern so:

> „Unternehmen gehen davon aus, sich umfassend zu schützen, vergrößern aber in Wirklichkeit mit jeder zusätzlichen AV-Engine die Angriffsfläche [Fri07].“

Ein besonderes Augenmerk gilt portablen Rechnern, die beispielsweise Außendienstmitarbeitern zur Verfügung gestellt werden. Mit diesen Rechnern wählen sich die Benutzer unter Umständen außerhalb des Unternehmensnetzwerks in das Internet ein und können sich dabei infizieren. Bei der Rückkehr müssen besondere Sicherheitsmaßnahmen für den Rechner gelten. Auch nicht zu vernachlässigen ist die Gefahr, die bei Diebstahl entstehen kann. Unternehmensinterne Daten können so von Unbefugten eingesehen werden.

2.3.3.4 Konfiguration der Antiviren-Software

Da sich die verschiedenen Antiviren-Programme in den technischen Einzelheiten stark voneinander unterscheiden, ist besonders in einem Unternehmen die Auswahl der eingesetzten Sicherheitssoftware und deren Konfigurationvon großer Tragweite.

Ein Unternehmen, das bei der Konfiguration der Sicherheitssoftware ein Kompromiss zwischen Geschwindigkeit und Sicherheit finden will, muss dabei immer die Sicherheit im Vordergrund stellen. Eine wichtige Entscheidung liegt auch darin, in welcher Weise die Software eingebunden wird. Ab einer bestimmten Anzahl von Rechnern im Unternehmensnetzwerk ist die lokale Installation auf jedem einzelnen Rechner nicht mehr zweckmäßig, denn der Einsatz eines Antiviren-Servers bringt viele Vorteile mit sich. So ist der Aktualisierungsprozess für die Virensignaturen jeweils nur einmal nötig. Die Software stellt die aktuellen Signaturen dann jedem Rechner im Netzwerk sofort zentral zur Verfügung. Das Administrieren eines Antiviren-Servers ist komfortabel und einfach, da alle Einstellungen nur an einer Stelle vorgenommen werden. Diese Variante bringt jedoch auch Gefahren mit sich, da ein guter Virenschutz eine permanente Pflege und Beobachtung erfordert. Jede Entscheidung ist von großer Tragweite und Fehlkonfigurationen gefährden nicht nur einen einzelnen Rechner, sondern das gesamte Unternehmensnetzwerk. Die Aufgabe muss deshalb von speziell geschultem Personal durchgeführt werden.

Besonders in einem Unternehmen ist es wichtig, die einzelnen Benutzerkonten in ihren Rechten so stark wie möglich einzuschränken, so dass die Benutzer weder vorsätzlich noch fahrlässig die Möglichkeit haben, Sicherheitsmaßnahmen zu umgehen. Die Antiviren-Software muss bei jedem Benutzer permanent im Hintergrund aktiv und das Deaktivieren nicht möglich sein. Zusätzlich ist das Planen eines regelmäßigen On-Demand-Scans zwingend erforderlich.

2.4 Datenbanken

2.4.1 Einleitung

Um großen Mengen an Informationen effektiv auswerten zu können, bietet sich die Verwendung von Datenbanken an. Sie sind elektronische Archive logisch zusammenhängender Daten und zeichnen sich durch eine platzsparende Aufbewahrung großer Datenmengen und den schnellen Zugriff darauf aus. Die Datenbank gewinnt im Alltag immer mehr an Bedeutung und ist aus dem weltwirtschaftlichen Geschehen nicht mehr wegzudenken. Ohne Datenbanken wären Banken oder Versicherungsgesellschaften heute nicht mehr handlungsfähig und Suchmaschinen im Internet könnten ohne sie überhaupt nicht funktionieren.

Der Aufbau des einfachsten Datenbanktyps lässt sich leicht anhand des Modells einer Tabelle erklären. In den Spalten einer Datenbanktabelle befinden sich die verschiedenen Informationskategorien und jede Zeile stellt einen Datensatz (englisch „record“) der Tabelle dar. Die jeweiligen Schnittpunkte der Zeilen und Spalten, die Zellen, ergeben die Datenfelder. Sie stellen die kleinste Informationseinheit dar und enthalten die jeweils zugehörigen Informationen über das Datenbankelement.

Der eben beschriebene Aufbau gilt für eine Einzeltabellendatenbank. Aufgrund der stark einge-

schränkten Funktionalität ist sie eher für einfache Anwendungszwecke wie Adressverwaltungen geeignet. Ein wichtiger Punkt ist, dass bei diesem einfachen Datenbanktyp keine Möglichkeit besteht, Inkonsistenzen, also Widersprüchlichkeiten zwischen den Daten, auszugleichen. Daher werden die Grenzen der Einzeltabelle bei komplexeren Anwendungen schnell erreicht.

2.4.2 Relationales Datenbankmodell

Die meisten in der Praxis eingesetzten Datenbanksysteme sind für so genannte *relationale Datenbanken* konzipiert. Diese verwenden ebenfalls das Tabellenmodell, es können jedoch beliebig viele Einzeltabellen enthalten sein. Im relationalen Datenmodell spielt das Verknüpfen verschiedener Tabellen eine entscheidende Rolle. Dabei ist eine exakte Identifizierung und Adressierung eines jeden einzelnen Datensatzes notwendig, um eine eindeutige Verknüpfung zwischen zwei Tabellen herstellen zu können. Ein solches Attribut, welches einen Datensatz mit all seinen Feldwerten eindeutig identifiziert, heißt *Primärschlüssel*. Durch die verschiedenen Möglichkeiten des Verknüpfens mehrerer Einzeltabellen untereinander können die Daten konsistent gehalten werden, so dass Informationen, die an verschiedenen Stellen vorkommen, nur einmal aufgeführt werden müssen.

Relationale Datenbanksysteme dominieren die Datenbankwelt bereits seit mehreren Jahren, jedoch gibt es auch noch weitere Systeme, die von Bedeutung sein können. Das älteste Modell ist dabei das *hierarchische Datenbankmodell*, bei dem die Datensätze in einer verzweigten und baumartigen Struktur angeordnet werden. Jede Datenbank hat dabei einen Ursprung, die so genannte Wurzel, wobei jeder einzelne Datensatz direkt oder indirekt von dieser Wurzel abhängt. Ein gravierender Nachteil bei diesem Modell ist, dass Verknüpfungen zwischen unterschiedlichen Bäumen und verschiedenen Ebenen innerhalb eines Baumes nicht realisierbar sind.

Eine weiterere Art sind die *objektorientierten Datenbanksysteme*, die – wie die Programmiersprachen C++ oder Java – auf der Grundlage von Klassen und Objekten arbeiten. Bei diesem flexiblen Datenbanktyp können neben Texten und Zahlen auch Daten in Form von Bild-, Audio-, Videodaten gespeichert werden. Eine große Stärke objektorientierter Datenbanken ist die Möglichkeit, komplexe und nichtlineare Beziehungen zwischen gespeicherten Informationen abzubilden und Objekten bestimmte Eigenschaften und Daten eines anderen Objektes zu vererben. Das führt jedoch zu sehr umfassenden Strukturen, was bei großen Datenmengen eine niedrige Verarbeitungsgeschwindigkeit mit sich bringt. Die Vorteile der objektorientierten Datenbank, insbesondere das Konzept der Vererbung, wird immer häufiger mit dem relationalen System in so genannten *objektrelationalen Datenbanken* kombiniert.

2.4.3 Datenbanksysteme

Die Kombination aus einer Datenbank und der dazugehörigen Verwaltungssoftware, auch DBMS[15] genannt, heißt Datenbanksystem. Es organisiert das effiziente, widerspruchsfreie und dauerhafte Speichern großer Datenmengen in konsistenter Form und stellt sie in unterschiedlichen Darstellungsformen für Benutzer und Anwendungsprogramme bereit. Die Bezeichnung *Datenbankserver* ist eine synonyme Bezeichnung für ein Datenbanksystem. Da es einen Server gibt, kann es dementsprechend auch einen oder mehrere Clients geben. Die Abbildung 2.5 zeigt den Aufbau eines Datenbanksystems.

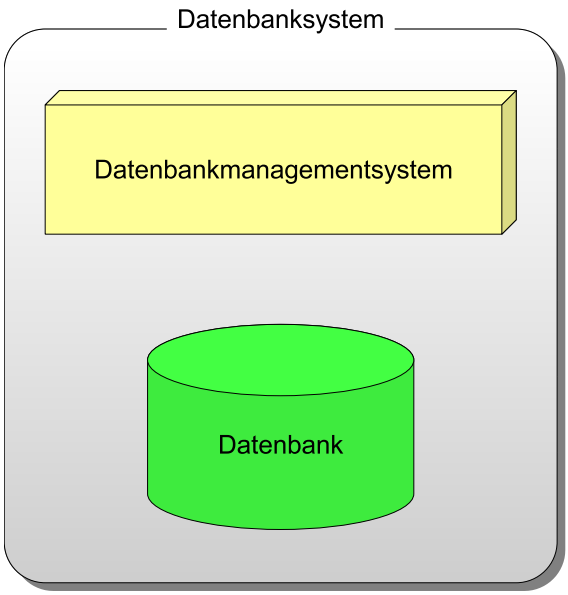


Abbildung 2.5: Schematische Darstellung zum Aufbau eines Datenbanksystems

Das Abfragen und Verwalten der Daten übernehmen spezielle Datenbanksprachen. Obwohl es eine Vielzahl von Sprachen gibt, hat sich im Zusammenhang mit relationalen Datenbanken SQL[16] als die am weitesten verbreitete Sprache durchgesetzt. Viele Systeme basieren auf dieser Sprache, so dass SQL im Bereich relationaler Datenbanksysteme schon seit mehreren Jahren als De-facto-Standard gilt.

Das DBMS organisiert intern die strukturierte Speicherung der Daten und kontrolliert alle lesenden und schreibenden Zugriffe auf die Datenbank. Zu den am weitesten verbreiteten relationalen Datenbankmanagementsystemen gehören „MySQL Server“, „Oracle Database“, „IBM DB2“ und „Microsoft SQL Server“. Diese vier Datenbanksysteme werden mit proprietärer Lizenz vertrieben. Eine Ausnahme stellt MySQL von „Sun Microsystems Inc.“ dar, welches unter proprietärer Lizenz und GPL[17] erhältlich ist.

[15] Datenbankmanagementsystem
[16] Structured Query Language
[17] General Public License

2.4.4 MySQL

2.4.4.1 Einleitung

Das schwedische Unternehmen „MySQL AB“ entwickelte am Anfang der 1990er Jahre das Datenbankmanagementsystem *MySQL Server* und brachte es 1994 auf dem Markt. Am 16. Januar 2008 erklärte das in Kalifornien ansässige Unternehmen Sun Microsystems Inc. die Absicht, MySQL Server mit einem Kaufwert von einer Milliarde Dollar zu übernehmen. In der Übernahmeverhandlungen beschlossen beide Seiten, dass die Datenbank auch nach der Übernahme seinen Open-Source-Status behalten soll. Für den Einsatz in kommerziellen Produkten bietet Sun fortan zusätzlich auch die kostenpflichtigen Variante *MySQL Enterprise* an.

Die Architektur von MySQL ist ein Client/Server-System, das heißt auf den MySQL Server können beliebig viele Clients – die durchaus zusammen mit dem Server auf dem selben Rechner installiert sein können – zugreifen beziehungsweise Anfragen an ihn richten. Die maximale Anzahl an Clients für den gleichzeitigen Zugriff auf den MySQL Server wird dabei durch technische Grenzen der Hardware bestimmt.

MySQL gilt mittlerweile weltweit als die populärste Open-Source-Datenbank und ist für verschiedene Plattformen wie „Microsoft Windows“, „Linux“, „Mac OS X“ und sogar „Symbian“ verfügbar. Auf der offiziellen Website [Suna] schreibt der Hersteller, dass bereits über 100 Millionen Anwender MySQL heruntergeladen oder anderweitig verteilt haben. Auch im kommerziellen Bereich wird das System heute von großen Unternehmen wie „Yahoo!“ und „T-Systems“ benutzt.

2.4.4.2 Installation von MySQL unter Microsoft Windows

Unter der Bezeichnung „MySQL“ wird in der Regel der Datenbankserver verstanden. Um diesen administrieren zu können, werden bei der Installation jedoch auch Client-Tools mitinstalliert. Die aktuelle Version stellt Sun auf `http://dev.mysql.com/downloads` [Sunb] zum Herunterladen bereit. Nach Auswahl der Open-Source-Variante *MySQL Community Server* und dem Betriebssystems Windows stehen nun drei verschiedene Versionen zur Auswahl. Für den grundlegenden Einsatz von MySQL ist die Variante „Windows Essentials (x86)“ in der aktuellen Version 5.1.35 ausreichend.

Nach dem Herunterladen des Installationsprogramms kann dieses gestartet werden und das Fenster „MySQL Server 5.1 – Setup Wizard“ erscheint. Falls die Installation nicht korrekt startet, sollte sichergestellt werden, dass der angemeldete Benutzer auch über die notwendigen Windows-Administratorrechte verfügt. Nach einem Klick auf *Next* stehen die Installationsarten zur Auswahl. Wie in Abbildung 2.6 dargestellt ist, kann die voreingestellte Auswahl *Typical* übernommen werden, denn sie ist für die einfache Anwendung angemessen.

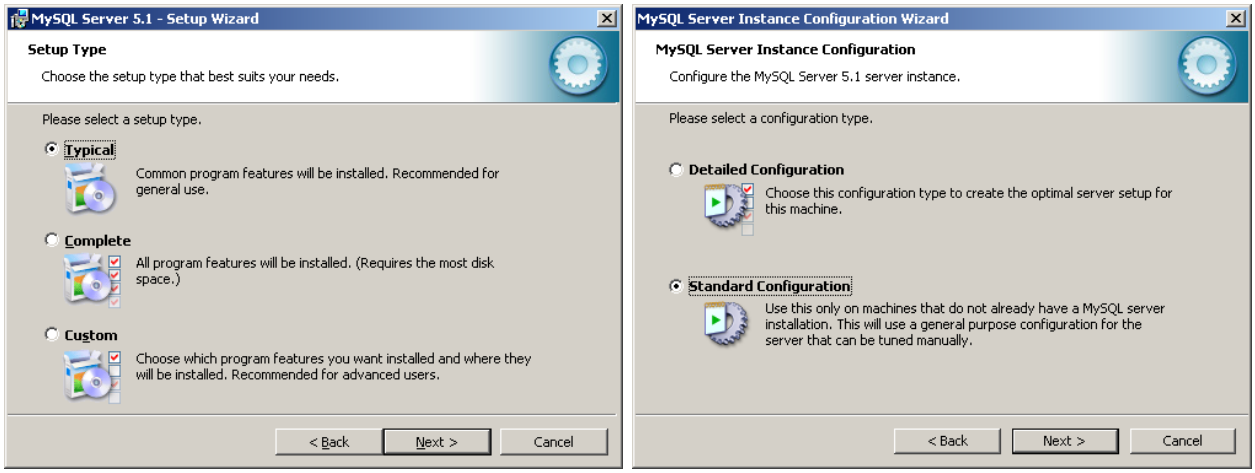


Abbildung 2.6: Auswahl der Installationsart von MySQL

Das darauf folgende Fenster zeigt an, in welches Verzeichnis die Installationsdateien und Benutzerdaten kopiert werden. Ein Klick auf *Install* startet den Installationsvorgang. Nachdem dieser abgeschlossen ist, bekommt der Benutzer zwei Seiten mit Hinweisen auf die kommerzielle *Enterprise* Version angezeigt. Das letzte Fenster bietet dann die Möglichkeit, den MySQL Server direkt nach der Installation zu konfigurieren und das installierte Produkt zu registrieren.

Da eine Konfiguration vor dem Einsatz von MySQL notwendig ist, kann diese nun direkt im Anschluss gestartet werden. Dabei ist für den grundlegenden Einsatz ausreichend, die *Standard Configuration* auszuwählen. In dem nächsten Schritt sollte – wie in Abbildung 2.7 dargestellt – der Punkt *Include Bin Directory in Windows PATH* ausgewählt werden, um den Installationspfad von MySQL in die PATH-Umgebung einzufügen. Dies ermöglicht zu jeder Zeit ein einfaches Aufrufen über die Kommandozeile von Windows.

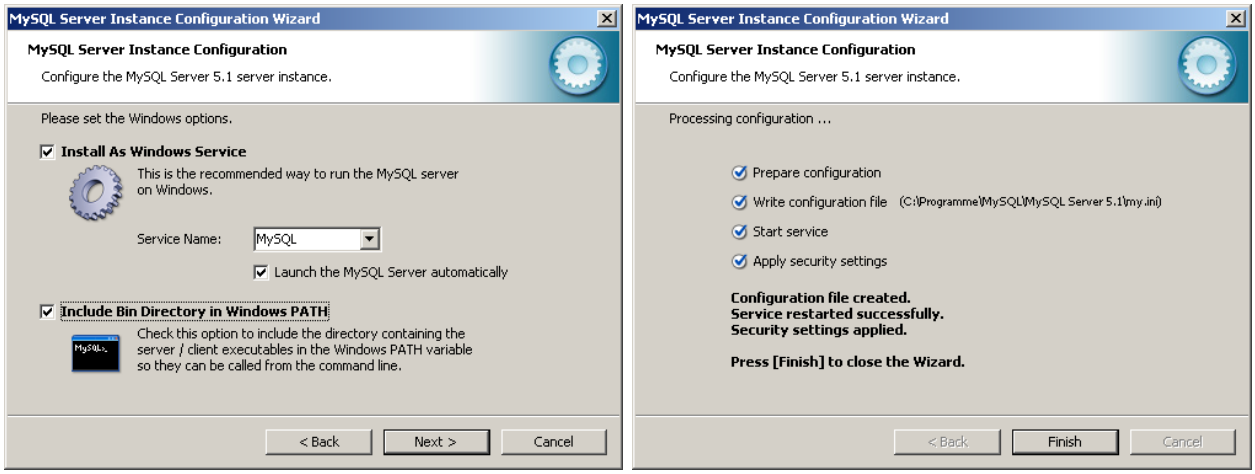


Abbildung 2.7: Einbinden von MySQL in Windows und Abschluss der Konfiguration

Es folgt ein letztes Fenster, in dem der Benutzer ein Passwort zum Zugriff auf MySQL festgelegt

kann, was aus Sicherheitsgründen auch empfehlenswert ist. Dann im abschließenden Schritt speichert die Konfiguration die Einstellungen, startet den Windows-Dienst *MySQL* und übernimmt die Sicherheitseinstellungen. War die Durchführung der einzelnen Schritte erfolgreich, erscheint das abgebildete Fenster über den Abschluss der Konfiguration. Ein Klick auf *Finish* beendet diese und MySQL ist nun einsatzbereit.

2.4.4.3 Das Kommandozeilenwerkzeug

Nach erfolgreicher Installation kann nun die Verbindung zum MySQL Server über das Kommandozeilenwerkzeug `mysql.exe` hergestellt werden. Im Windows-Startmenü befindet sich der entsprechende Eintrag unter „Programme“ ⇒ „MySQL“ ⇒ „MySQL Server 5.1“ ⇒ „MySQL Command Line Client“. Alternativ ist der Start des Clients auch durch das Ausführen von

```
mysql -uroot -p<Passwort>
```

über die Windows-Eingabeaufforderung möglich. Die Zeichenfolge „root“ hinter dem Parameter `-u` legt den Benutzernamen für die Anmeldung fest und `-p` übergibt das Passwort für den Benutzer. Neben diesen beiden Parametern existieren noch eine Reihe weiterer Optionen. Die Eingabe von `mysql --help` gibt eine vollständige Auflistung aller möglichen Parameter aus. Wenn während der Installation die entsprechende Option ausgewählt wurde, kann der Benutzer die ausführbare Datei `mysql.exe` im Unterverzeichnis `bin` der MySQL-Installation von jedem Verzeichnis aus aufrufen.

Eine häufige Fehlermeldung von MySQL beim Versuch, sich mit dem Server zu verbinden, lautet `ERROR 2003 (HY000): Can't connect to MySQL server on 'localhost' (10061)`. Für diesen Fall ist sicherzustellen, dass der Windows-Dienst – und damit der Datenbankserver – auch wirklich gestartet ist. Wenn die Verbindung zum Server erfolgreich hergestellt wurde, erscheint die in Abbildung 2.8 dargestellte Shell.

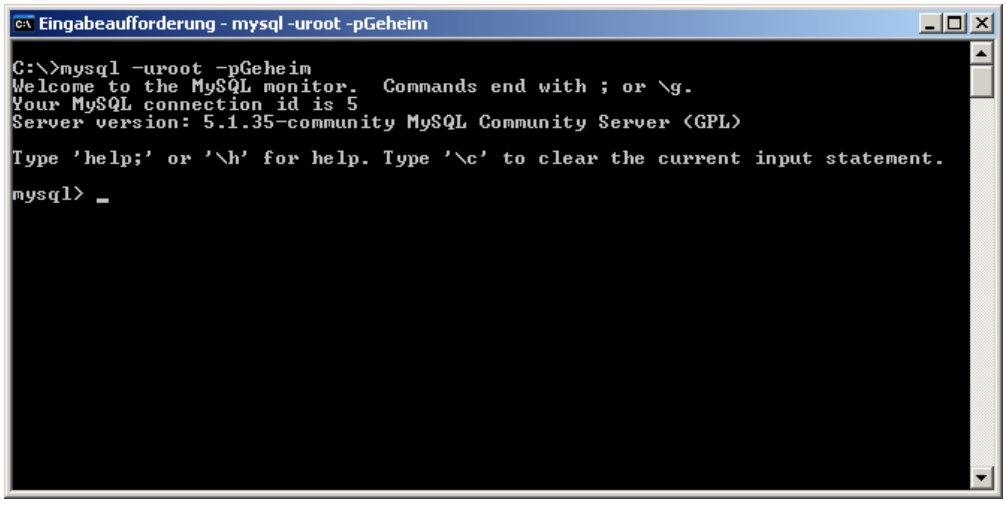


Abbildung 2.8: Darstellung der MySQL-Shell

Die MySQL-Shell stellt die Schnittstelle zwischen dem Benutzer und dem Datenbankserver dar und dient der Eingabe von Kommandos, die dann an den MySQL Server geschickt werden. Der Autor von [KK07] schreibt zur Shell:

> „Das wichtigste Tool, das im Standardumfang von MySQL mitgeliefert wird und auf allen Betriebssystemen verfügbar ist, ist das Kommandozeilenwerkzeug *mysql*. Dieser Client wird von MySQL AB als einfache SQL-Shell bezeichnet."

Um zu signalisieren, dass der Server bereit ist, Befehle anzunehmen, bekommt der Benutzer den Kommandozeilen-Prompt

```
mysql>
```

angezeigt. Bei der Eingabe von MySQL-Befehlen ist darauf zu achten, dass eine Anweisung im Client erst dann ausgeführt wird, wenn entweder ein Semikolon oder `\g` am Zeilenende angefügt ist. Durch das Weglassen des Abschlusszeichens ist es so möglich, einen langen Befehl übersichtlich über mehrere Zeilen einzugeben. Die zuletzt eingegebenen Befehle kann mit den Pfeiltasten oben und unten ausgewählt werden. Zum ordnungsgemäßen Beenden der Verbindung zum Server dienen die Kommandos `exit` oder `quit`. Eine Übersicht der wichtigsten MySQL-Befehle befindet sich in der Tabelle „Übersicht zu wichtigen MySQL-Befehlen" im Anhang A.4.

Kapitel 3

Recherche

3.1 Informationsquellen

Um Schwachstellen auswerten zu können, findet der Interessent auf der Suche nach einer geeigneten Quelle eine Vielzahl von Möglichkeiten. Darunter sind Mailinglisten, Artikel von Online-Nachrichtenportalen mit oder Internetangebote von Sicherheitsunternehmen. Eine geeignete Auswahl zu treffen kann bei der Fülle von Varianten schwer fallen. Schwachstellen sind aber nicht nur für den Anwender, sondern auch für immer mehr Software-Hersteller von Bedeutung, denn Informationen darüber können einen Wettbewerbsvorteil darstellen. Auf dem Software-Markt herrscht große Konkurrenz und die Hersteller zeigen wenig Anstrengungen, eine gemeinsame Quelle für Schwachstellen zu schaffen. Durch die vielen Angebote entsteht daher auch viel Redundanz in diesem Segment, was die Suche nach Informationen erschwert. Es ist weiterhin davon auszugehen, dass einige Hersteller aus dem Sicherheitsbereich selbst interne nicht-öffentliche Datenbanken führen, um eigene Interessen zu verfolgen.

Auf Seiten der private Herausgeber von sicherheitsbezogenen Internet-Angeboten sieht es ähnlich aus. Zwar besteht weniger Kokurrenzdruck untereinander, die Angebote sind jedoch so vielfältig, dass es schwer fällt, die „beste“ Informationsquelle zu bestimmen. Vielmehr muss der jeweilige Zweck berücksichtigt werden. Geht es darum, möglichst viele Informationen zu einer bestimmten Schwachstelle zu sammeln, dann ist es angebracht, mehrere Quellen zu besuchen. Sollen Schwachstellen systematisch gesucht und analysiert werden, sind spezielle Datenbanken am besten geeignet. Diese können öffentlich oder privat sein und ermöglichen das strukturierte Erfassen von Informationen aus verschiedenen Arten von Quellen. Darunter können sich Mailinglisten, Websites von Sicherheitsunternehmen, aber auch weitere Schwachstellen-Datenbanken befinden. Viele von ihnen stellen Schnittstellen nach außen bereit, um das maschinelle Auslesen zu vereinfachen beziehungsweise überhaupt erst zu ermöglichen.

Ein weiterer Punkt ist, dass öffentlich verfügbare Meldungen über Schwachstellen in der Regel mit der Absicht veröffentlicht werden, dass der Hersteller der betreffenden Software reagieren und sein Produkt verbessern kann. Ein Angreifer mit dem Ziel, Schaden anzurichten, würde seinen Fund vermutlich nicht in öffentlichen Bereichen des Internets verbreiten, obwohl er die detailliertesten Informationen von allen hat.

3.2 Ausgewählte Informationsquellen

3.2.1 National Vulnerability Database

Im August 2005 veröffentlichte die US-amerikanischen Bundesbehörde für Standardisierungsprozesse, das *National Institute of Standards and Technology*, in der Abteilung *Computer Security Division* eine Datenbank mit dem Namen *National Vulnerability Database*. Diese enthielt rund 12 000 Einträge über bisher bekannte Schwachstellen im Bereich Software. Im Laufe der Zeit kamen im Durchschnitt 10 bis 20 neue Einträge pro Tag dazu, so dass die Gesamtzahl bis heute auf fast 40 000 gestiegen ist. Die NVD[1] ist somit ein umfassendes Repositorium, das alle öffentlich verfügbaren Informationen der US-amerikanischen Regierung über bekannte Sicherheitslücken enthält.

Die NVD basiert auf dem Protokoll SCAP[2], einer Kombination von sechs offenen Standards. Die einzelnen Komponenten sind in der Tabelle 3.1 erläutert. Auf der offiziellen Website [Nate] hat der Besucher die Möglichkeit, sich anhand einer Vielzahl von Charakteristiken über entdeckte Schwachstellen zu informieren. So kann er beispielsweise nach dem Hersteller, dem Produktnamen oder einer bestimmten Version der betreffenden Anwendung suchen. Durch das Festlegen eines Zeitraums für das Datum der Veröffentlichung oder letzten Änderung eines Eintrags kann die NVD auch den zeitlichen Verlauf eines Produktes bezüglich Schwachstellen untersuchen.

Anbietern von proprietärer und Open Source Software gibt die National Vulnerability Database eine gute Möglichkeit, sich über Sicherheitslücken in ihren Produkten zu informieren und gegebenenfalls darauf zu reagieren. Veröffentlichungen in der NVD orientieren sich bezüglich der Bezeichnungen und des Formats an Industriestandards, um eine einheitliche Namenskonvention zu gewährleisten. Die meisten eingetragenen Schwachstellen werden durch Verweise auf externe Quellen belegt, was dazu beiträgt, eine offene und transparente Plattform zu schaffen. In der Regel werden die Angaben zu den Schwachstellen von Dritten herausgegeben und es kann daher nicht bei allen Informationen von einer Fehlerfreiheit ausgegangen werden. So kommt es häufig vor, dass ein bereits veröffentlichter Eintrag nach genauerem Analysen noch ergänzt wird oder Merkmale der Schwachstelle widerlegt werden können. Die NVD stellt jedoch nicht nur für die Hersteller, sondern auch für sicherheitsbewusste Anwender oder Interessenten eines bestimmten Produktes eine geeignete Informationsquelle dar,

[1] National Vulnerability Database
[2] Security Content Automation Protocol

SCAP Komponente	Beschreibung
Common Configuration Enumeration (CCE)	Nomenklatur und Verzeichnis für Themen, die die Sicherheit von Systemen betreffen
Common Platform Enumeration (CPE)	Nomenklatur und Verzeichnis für Produktnamen und –versionen
Common Vulnerabilities and Exposures (CVE)	Nomenklatur und Verzeichnis für sicherheitsbezogene Softwarefehler
Common Vulnerability Scoring System (CVSS)	Spezifikation für die Messung des Schweregrades von Schwachstellen
Extensible Configuration Checklist Description Format (XCCDF)	Sprache für die Spezifikation von Kontrolllisten
Open Vulnerability and Assessment Language (OVAL)	Sprache für die Spezifikation der von Kontrolllisten verwendeten Testverfahren

Tabelle 3.1: Komponenten des Protokolls SCAP [Natd]

denn alle verfügbaren Informationen sind von jedem Besucher der Website ohne Einschränkungen einsehbar.

Suche nach Schwachstellen über die Website der National Vulnerability Database

Um sich auf der Website Informationen zu einer Schwachstelle abzufragen, gelangt man mit einem Klick auf „Vulnerabilities“ auf die Suchseite [Natf]. Diese enthält das Eingabefeld „Keyword search“, in das der Benutzer ein Schlagwort eintragen und über die Schaltfläche „Search All“ anschließend die komplette Datenbank danach durchsuchen kann. Ist bei der Suche eine Beschränkung auf Einträge der letzten drei Monate oder drei Jahre erwünscht, stehen zusätzlich auch die Schaltflächen „Search last 3 months“ und „Search last 3 years“ zur Verfügung.

Im Folgenden führt der Benutzer eine einfache Anfrage zum Durchsuchen der NVD nach dem Begriff „Symantec Log Viewer“ durch. Die Suche liefert, wie in Abbildung 3.1 dargestellt, zwei Treffer mit den CVE[3]-Bezeichnungen „CVE-2009-1428“ und „CVE-2007-4380“. Unter jedem Treffer befindet sich eine Zusammenfassung (englisch „Summary“), die mit einem oder zwei Sätzen eine Beschreibung zu der Schwachstelle liefert. Weiterhin gibt der Punkt „Published“ das Veröffentlichungsdatum der Schwachstelle an und „CVSS Severity“ liefert den durch CVSS beschriebenen *Base Score*, der den allgemeinen Schweregrad der Schwachstelle auf einer Skala von 1 bis 10 widerspiegelt. Die zusätzlich in Klammern angegebene qualitative Beschreibung dieses Merkmals lautet „Low“, wenn der Base Score kleiner als 4,0 ist. Bis zu einem Wert von 6,9 bezeichnet die NVD den Schweregrad als „Medium“ und bei einem Base Score größer als 7,0 dann „High“.

Um sämtliche verfügbaren Informationen zu einer Schwachstelle abzurufen, zeigt ein Klick auf den

[3] Common Vulnerabilities and Exposures

Abbildung 3.1: Ergebnis der Suchanfrage nach dem Begriff „Symantec Log Viewer“ an die National Vulnerability Database – Stand: 18. August 2009 [Nate]

entsprechenden Titel die detaillierte Ansicht zu dem entsprechenden Eintrag an. Welche Einzelheiten zur Schwachstelle „CVE-2009-1428“ in der Datenbank hinterlegt sind, zeigt die Abbildung im Anhang A.5 an. Im Kopfteil ist das Datum der ersten Veröffentlichung, der letzten Änderung und die ursprüngliche Quelle vermerkt. Dann folgt der Abschnitt „Overview“, der – wie auch auf der Seite mit den Suchergebnissen – die Beschreibung der Schwachstelle anzeigt. Im nächsten Teil „Impact“ werden die Auswirkungen der Schwachstelle anhand der verfügbaren CVSS-Daten beschrieben. Genauere Informationen dazu sind im Abschnitt 2.1.5 zu finden. Es folgt die Passage „References to Advisories, Solutions, and Tools“ mit Namen und Links zu diversen externen Quellen. Je nachdem wie viele Anwendungen von der Schwachstelle betroffen sind, werden an dieser Stelle eine mehr oder weniger große Anzahl an Quellen genannt. Unter „Vulnerable software and versions“ sind dann sämtliche von der Schwachstelle betroffenen Anwendungen einschließlich der genauen Programmversionen gemäß CPE[4] aufgelistet. Im letzten Abschnitt „Technical Details“ findet eine Zuordnung der Schwachstelle zu einem bestimmten Typ statt. Bei dem gezeigten Beispiel handelt es sich um eine Schwachstelle der CWE[5]-Kategorie „Cross-Site Scripting“ [MIT]. Das NIST stellt unter [Natb] eine vollständige Übersicht über die 23 möglichen Kategorien zur Verfügung.

Ist die CVE-Bezeichnung einer Schwachstelle bekannt, so ist es auch möglich, sich die entsprechen-

[4] Common Platform Enumeration
[5] Common Weakness Enumeration

de Webseite mit den Details direkt anzeigen zu lassen. Dafür muss die für den dargestellten Eintrag gültige URL[6] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-1428 [Natg] durch Einsetzen der entsprechenden CVE-Bezeichnung am Ende angepasst werden.

Erweiterte Suche über die Website der National Vulnerability Database

Neben der beschriebenen einfachen Suche nach Schlagwörtern bietet die Suchseite der National Vulnerability Database auch die Möglichkeit, anhand verschiedener Kriterien eine erweiterte Suche durchzuführen. Wird auf der Suchseite auf „Advanced Search“ geklickt, dann wird dem Benutzer das in Abbildung 3.2 gezeigte Fenster eingeblendet. Hier kann er die Datenbankeinträge anhand der CWE-Kategorie oder Angabe der betreffenden Hersteller und Produktnamen herausfiltern. Weiterhin ist es möglich, sich auf Einträge zu beschränken, die das NIST in einem bestimmten Zeitraum veröffentlicht oder verändert hat. Im Abschnittt „CVSS Version 2 Metrics“ kann der Benutzer zusätzlich detaillierte Suchparameter zu den Auswirkungen der Schwachstelle gemäß CVSS einstellen.

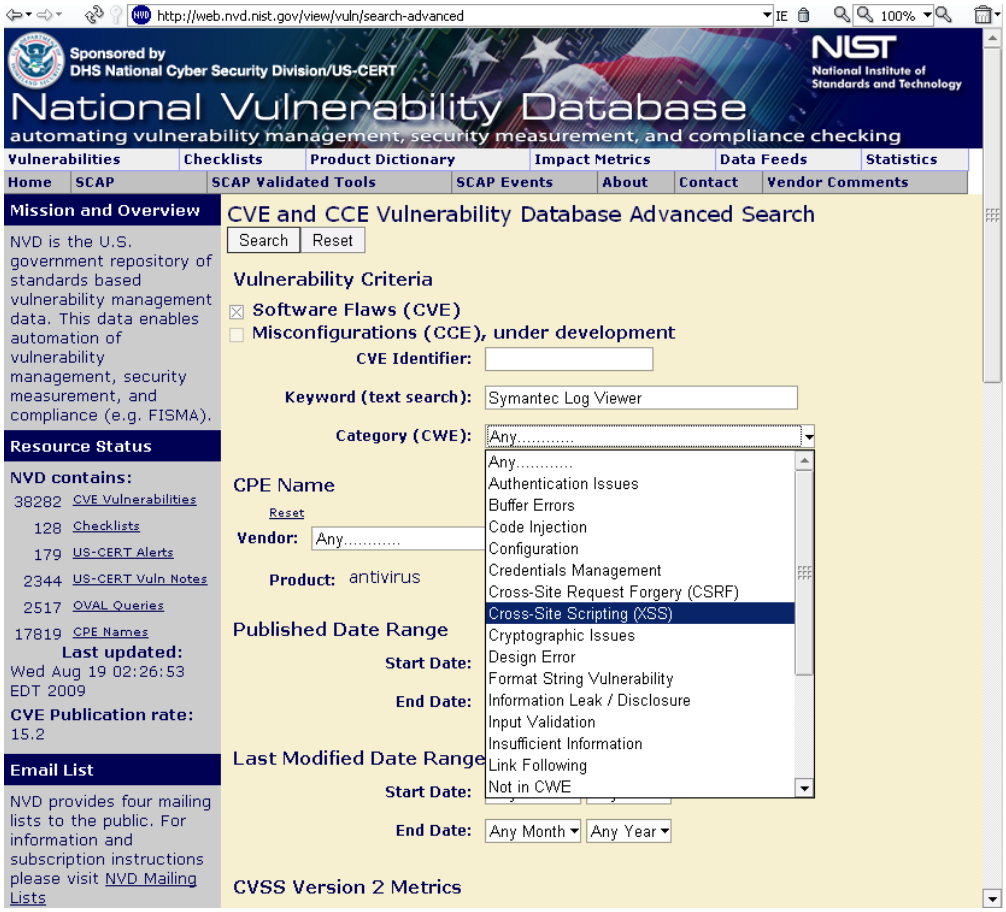


Abbildung 3.2: Erweiterte Suche nach Schwachstellen in der National Vulnerability Database – Stand: 18. August 2009 [Nate]

[6] Uniform Resource Locator

3.2.2 Open Source Vulnerability Database

Eine bislang recht unbekannte, aber dennoch umfassende Informationsquelle stellt die *Open Source Vulnerability Database* [Opec] dar. Das Projekt wurde 2002 gegründet und hat sich als Ziel gesetzt, eine zentrale und vor allem unabhängige Kollektion für Schwachstellen im Internet zu sein. Es wurde nach eigenen Angaben von und für die *Security Community*[7] geschaffen und soll die Zusammenarbeit zwischen den Unternehmen und Anwendern verbessern.

Die OSVDB[8] basiert darauf, dass prinzipiell jeder Benutzer selbständig Schwachstellen hinzufügen und bearbeiten kann. Da die Mehrheit der Anwender selbst Open-Source-Software einsetzt, liegt der Fokus der eingetragenen Schwachstellen auch in diesem Bereich. Da Sicherheitssoftware jedoch in der Regel unter proprietärer Lizenz steht, finden verhältnismäßig wenige Benutzer dieser Anwender den Weg zu diesem Open-Source-Projekt. Anhand der Schwachstelle „CVE-2009-1428“ aus dem vorherigen Abschnitt 3.2.1 kann der Umfang der verfügbaren Informationen verglichen werden. Der OSVDB-Eintrag, der die selbe Schwachstelle beschreibt, ist unter `http://osvdb.org/54132` [Opea] abrufbar und in Abbildung 3.3 dargestellt.

In der Darstellung ist erkennbar, dass der Eintrag in der OSVDB eine eigene Beschreibung der Schwachstelle besitzt und anhand von sieben Merkmalen klassifiziert wird. Weiterhin besteht ein Verweis auf die „CVE ID: 2009-1428“ und auch der im Anhang A.5 abgebildete NVD-Eintrag [Natg] ist verlinkt, weitere Informationen fehlen jedoch. So sind keine Hersteller- oder Produktnamen der betreffenden Anwendung vermerkt und es existiert auch keine quantitative Beschreibung der Auswirkung. Für die OSVDB spricht jedoch, dass die Schwachstelle am 28. April 2009, also einen Tag früher als in der NVD, eingetragen wurde.

Das Open-Source-Projekt legt neben aktuellen Einträgen großen Wert darauf, auch historische Schwachstellen zu katalogisieren. So reicht die älteste registrierte Schwachstelle bis ins Jahr 1965 zurück. Jedoch befinden sich auch für jeden offensichtlich fehlerhafte Eintäge in der Datenbank. So ist beispielsweise in dem Eintrag „57606“, der über `http://osvdb.org/57606` [Opeb] erreichbar ist, als Entdeckungsdatum der 14. Juli im Jahr 2099 eingetragen. Fehler dieser Art zeugen von mangelnder Kontrolle und sprechen gegen das Prinzip, jedem Benutzer das Eintragen und Verändern von Schwachstellen zu ermöglichen.

Eine große Stärke der OSVDB ist die Export-Möglichkeit der gesamten Datenbank im CSV[9]-, SQL- oder XML[10]-Format. Um den Datenbestand in einer dieser Formen herunterladen zu können, muss sich der Anwender zunächst auf der Website kostenlos ein Benutzerkonto anlegen. Dieser Schritt ist laut Website notwendig, um einen möglichen Missbrauch zu vermeiden. Um erweiterte Funktionen

[7] englisch für „Sicherheits-Gemeinschaft“
[8] Open Source Vulnerability Database
[9] Comma Separated Values
[10] Extensible Markup Language

Abbildung 3.3: Detaillierte Darstellung der Schwachstelle „54132“ auf der Website der Open Source Vulnerability Database – Stand: 7. September 2009 [Opea]

wie das unbegrenzte Beobachten gewünschter Produkte oder den Erhalt von Benachrichtigungen per E-Mail nutzen zu können, wird jeder registrierte Benutzer gebeten, das Projekt finanziell zu unterstützen.

3.2.3 Bugtraq

Es existieren eine Vielzahl von Mailinglisten, aber die wohl bekannteste ist *Bugtraq* [Secc]. Die von der Sicherheits-Community *Securityfocus* unterhaltene englischsprachige Liste gehört zu den bekanntesten, aber auch schnellsten öffentlichen Informationsquellen über Schwachstellen.

Bugtraq existiert seit 1993 und zeichnet sich heute durch einen großen Umfang sowie kurze Reaktionszeiten bei neuen Schwachstellen aus. Daher lassen einige Größen in der Software-Branche die

Liste von Mitarbeitern regelmäßig überwachen, um sich über mögliche Ereignisse zu den eigenen Produkten zu informieren. Bugtraq besitzt keine festen Vorgaben für die Meldung von Schwachstellen oder sicherheitsrelevanten Vorfällen. Die Mitglieder der Liste können die Inhalte dann bei Bedarf analysieren oder darüber öffentlich diskutieren.

Die bereits bekannte Beispiel-Schwachstelle kann der Anwender auch auf der Bugtraq-Website nach einem Klick auf „Vulnerabilities“ und Eingabe der CVE-Bezeichnung „CVE-2009-1428“ finden. Wie in Abbildung 3.4 gezeigt, bietet SecurityFocus eine Seite mit allgemeinen Informationen zur Schwachstelle, eine Diskussions-Seite, eine mit der Beschreibung des Exploits, Lösungsbeschreibungen und Referenzen. Die Seite ist unter `http://www.securityfocus.com/bid/34669` [Secd] erreichbar.

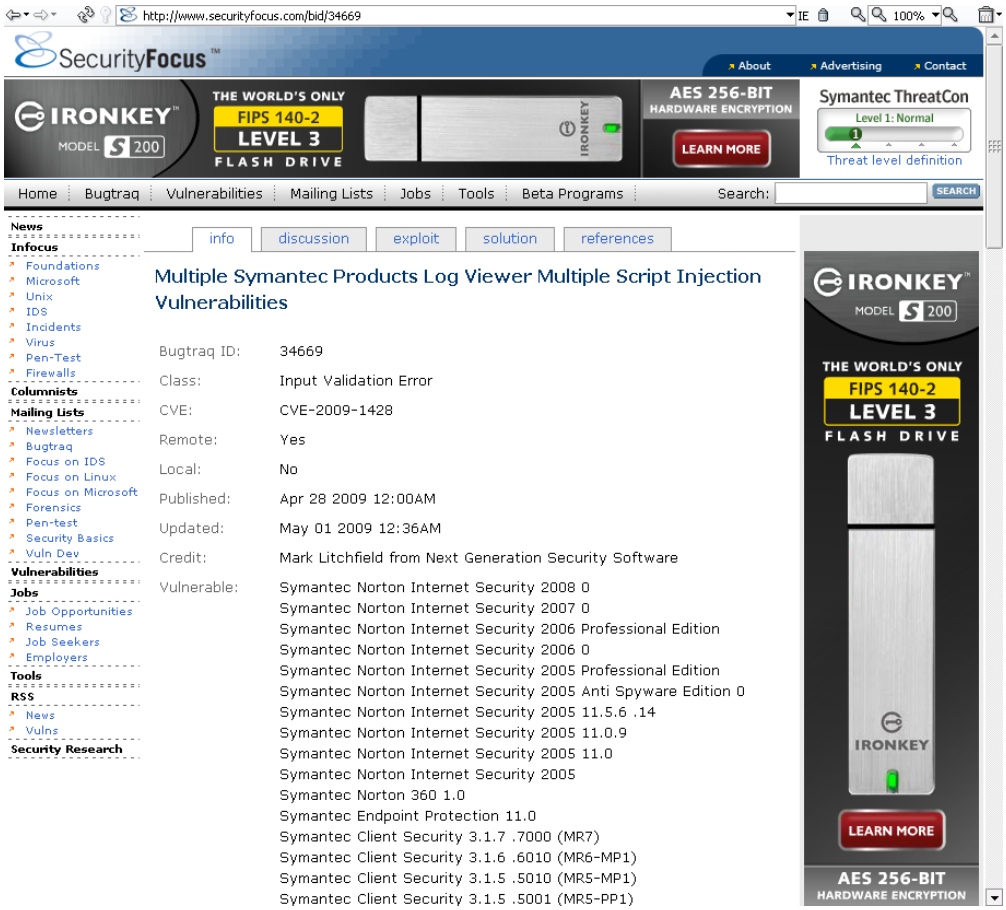


Abbildung 3.4: Detaillierte Darstellung der Schwachstelle mit der „Bugtraq ID: 34669“ auf der Website von SecurityFocus – Stand: 10. September 2009 [Secd]

3.2.4 Secdev

Der luxemburgische Sicherheits-Forscher Thierry Zoller erstellt in seinem privaten englischsprachigen Blog „Secdev“ unter `http://blog.zoller.lu` [Zol] Beiträge zum Thema Schwachstellen in Computeranwendungen. Dabei veröffentlicht er häufig Exploits und beschreibt diese sehr ausführ-

lich. Neue Einträge gibt es jedoch nur sehr unregelmäßig. Während Zoller im Mai 2009 mehr als 20 neue Blog-Einträge schrieb, waren es in den beiden darauf folgenden Monaten nur zwei bis drei. Dennoch kann sich jeder Interessierte durch E-Mail-Benachtitigungen oder Abonnieren des Newsfeeds über neue Einträge informieren. In zeitkritischen Anwendungsfällen ist das jedoch eher ungeeignet, weil das Benachrichtigungssystem eine E-Mail mit der Mitteilung über einen neuen Eintrag in einigen Fällen erst ein bis zwei Tage später verschickt.

Secdev bietet qualitativ hochwertige Informationen zu sicherheitsrelevanten Themen, so dass viele themenbezogene Artikel von bekannten Redaktionen wie „heise online“ [Heic] aus dem bekannten Heise Zeitschriften Verlag auf Zollers Einträge verweisen.

3.2.5 Weitere Quellen

Aus der Vielzahl von möglichen Quellen sind noch das Nachrichtenportal „Heise Online“ [Heic] erwähnenswert, das die interessierten Leser täglich mit zahlreichen Meldungen aus dem IT-Umfeld versorgt. In der speziellen Rubrik „heise security“ liegt der Fokus auf sicherheitsrelevanten Themen.

Auch das Bundesamt für Sicherheit in der Informationstechnik stellt eine seriöse deutschsprachige Quelle zur Verfügung:

> „Das Bürger-CERT informiert und warnt Bürger und kleine Unternehmen schnell und kompetent vor Viren, Würmern und Sicherheitslücken in Computeranwendungen – kostenfrei und absolut neutral. Unsere Experten analysieren für Sie rund um die Uhr die Sicherheitslage im Internet und verschicken bei Handlungsbedarf Warnmeldungen und Sicherheitshinweise per E-Mail. [Buna]“

In dem Archiv „Technische Warnungen“ kann sich der Leser über Gefahren im Internet informieren und sich auch per E-Mail auf dem neuesten Stand halten. Das Bürger-CERT ist eine gute deutsche Informationsquelle, die auch Empfehlungen zum Umgang mit den entsprechenden Schwachstellen zur Verfügung stellt.

Ein privates Archiv mit Spezialisierung auf *Full Disclosure* bietet John Cartwright mit finanzieller Unterstützung des dänischen Unternehmens „Secunia“ an [Car]. Wer sich dagegen speziell für Exploit-Code interessiert, der findet auf der Website „ExploitTree“ [Sece] eine gute Quelle.

3.3 Auswahl einer geeigneten Informationsquelle

Zur Auswahl einer geeigneten Informationsquelle findet zunächst ein Vergleich der verfügbaren Typen statt. Eine Mailingliste oder Blog bietet sehr gute Möglichkeiten, sich über Schwachstellen zu informieren. Besonders die Diskussionen oder Beschreibungen des Exploits sprechen für diese

Quellen. Ein Anwender, der speziell an aktuellen Themen interessiert ist, findet in Mailinglisten sehr schnell die neuesten Einträge.

Die in dieser Arbeit entscheidenden Auswahlkriterien sind jedoch nicht die Aktualität, sondern vielmehr der Datenumfang und Möglichkeiten, diese Informationen auszuwerten. Schwachstellen-Datenbanken sind dabei die Quellen, die den Anforderungen einer systematischen Untersuchung am besten gerecht werden.

Die in den Abschnitten 3.2.1 und 3.2.2 beschriebenen Datenbanken stellen jeweils eine große Anzahl an Einträgen bereit und ermöglichen es, diese extern auszuwerten. Aufgrund der unzureichenden Klassifizierung von Schwachstellen und der Vielzahl fehlerhafter Einträge ist die „Open Source Vulnerability Database“ jedoch momentan noch nicht sehr ausgereift und die am besten geeignete Quelle stellt daher die „National Vulnerability Database“ dar. Insbesondere das „Common Vulnerability Scoring System“ macht sie zu einer Informationsquelle von hoher Datenqualität.

3.4 Auswahl der Antiviren-Software

Aus der Vielzahl der aktuell auf dem Markt verfügbaren Antiviren-Software für das Betriebssystem „Microsoft Windows“ ist eine Auswahl der wichtigsten kostenlosen und kostenpflichtigen Produkte in den Tabellen im Anhang A.6 aufgeführt. Die Produktübersichten und Kopien der Hersteller-Websites sind auf der beiligenden CD im Verzeichnis `Antiviren-Software` gespeichert.

Kapitel 4

Importieren der Schwachstellen

4.1 Einleitung

Um die knapp 40 000 Datensätze der National Vulnerability Database praktisch auswerten zu können, ist es notwendig, den kompletten Datenbestand in eine dafür geeignete Form zu überführen. Es ist zwar möglich, durch erweiterte Suchanfragen anhand spezieller Charakteristiken alle gewünschten Datensätze zu finden und anzuzeigen, jedoch zeigt die Website `http://nvd.nist.gov` [Nate] dem Benutzer schnell die Grenzen der Suchmöglichkeiten auf. So ist es beispielsweise nicht realisierbar, mehrere Suchkriterien miteinander zu verknüpfen. Möchte der Benutzer also Schwachstellen finden, die mehrere Hersteller betreffen, so muss er mehrere Anfragen nacheinander stellen. Ein weiteres Problem ist, dass sämtliche Datenbankzugriffe auf den Servern des NIST durchgeführt werden. Abhängig von deren momentaner Auslastung kann die Dauer eines Suchauftrages von einer Sekunde bis zu mehr als einer Minute betragen.

Der Einsatz eines eigenen Datenbanksystems stellt eine Variante zum Umgehen dieser Probleme dar. Ein komfortables und für diesen Zweck gut geeignetes Datenbanksystem ist das in Abschnitt 2.4.4 beschriebene *MySQL*. Es stellt die Datenbasis für die Auswertung der Schwachstellen dar.

Den Datenbestand der NVD kann der Benutzer komplett und ohne Einschränkungen von der Website der NVD beziehen. Unter dem Menüpunkt „Data Feeds“ hat er die Möglichkeit, den gesamten Inhalt der Datenbank im XML-Format herunterzuladen. Eine wichtige Aufgabe besteht anschließend darin, die vorliegenden strukturierten XML-Daten so aufzubereiten, dass das Integrieren in die Datenbankstruktur möglich ist.

4.2 Das XML-Format

XML steht für *Extensible Markup Language* und bedeutet übersetzt „erweiterbare Auszeichnungssprache“. Sie hat sich schnell zur Standardsprache zum Darstellen hierarchisch strukturierter Daten entwickelt. Ein großer Teil des Erfolges liegt darin, dass XML plattform- und anwendungsunabhängige Beschreibungen der Daten einsetzt. Der Aufbau eines Dokuments ist nahe verwandt mit HTML, jedoch ist XML dabei nicht auf bestimmte Programmieranweisungen festgelegt und definiert die Regeln für die Struktur selbst. Das Erstellen von XML-Dokumenten ist mit jedem einfachen Text-Editor möglich, denn sie dürfen laut Definition keine Binärdaten enthalten. Damit ist XML nicht nur maschinell interpretierbar, sondern auch für den Menschen einfach zu lesen.

Für einen großen Kreis der Endanwender wurde XML bekannt, nachdem der Hersteller „Microsoft Corporation“ sein weltweit verbreitetes Softwarepaket „Microsoft Office 2003“ mit XML-Unterstützung auf den Markt brachte [Micc]. In der vier Jahre später veröffentlichten Folgeversion „Microsoft Office 2007“ ging der Hersteller noch weiter und setzte das XML-Dateiformat OOXML[1] sogar als neues Standard-Dateiformat ein. So erhielten Dokumente der Textverarbeitungssoftware „Microsoft Office Word“ beim Speichern fortan standardmäßig die bis dato unbekannte Dateiendung `.docx` als Ersatz zu dem seit Jahren bekannten proprietären `.doc`-Format.

XML-Feed in der einfachen Version 1.2		XML-Feed in der erweiterten Version 2.0	
Dateiname	Dateigröße	Dateiname	Dateigröße
nvdcve-2002.xml	10,6 MB	nvdcve-2.0-2002.xml	20,3 MB
nvdcve-2003.xml	2,9 MB	nvdcve-2.0-2003.xml	5,9 MB
nvdcve-2004.xml	6,0 MB	nvdcve-2.0-2004.xml	12,7 MB
nvdcve-2005.xml	9,7 MB	nvdcve-2.0-2005.xml	20,1 MB
nvdcve-2006.xml	16 MB	nvdcve-2.0-2006.xml	32,5 MB
nvdcve-2007.xml	14,4 MB	nvdcve-2.0-2007.xml	29,7 MB
nvdcve-2008.xml	14,7 MB	nvdcve-2.0-2008.xml	32,7 MB
nvdcve-2009.xml	4,5 MB	nvdcve-2.0-2009.xml	10,4 MB
nvdcve-modified.xml	3,9 MB	nvdcve-2.0-modified.xml	9,3 MB
nvdcve-recent.xml	1,4 MB	nvdcve-2.0-recent.xml	3,2 MB

Tabelle 4.1: XML-Feeds der National Vulnerability Database – Stand: 04. Juni 2009 [Nate]

Die auf der Website der NVD verfügbaren „Data Feeds“ im XML-Format liegen nach Veröffentlichungsdatum sortiert vor und stehen dem Besucher wie in Tabelle 4.1 gezeigt frei zum Herunterladen zur Verfügung. Das NIST bietet alle XML-Feeds in zwei verschiedenen Varianten an. Während die einfache Version 1.2 eine klassische DTD[2] zur Spezifikation des Dokuments enthält, stellt die Version 2.0 mit einem XML-Schema eine modernere Form der Inhaltsdefinition dar. Mit ihr lassen sich Datentypen und andere komplexe Strukturen definieren, die sich in einer DTD nur schwer oder

[1] Office Open XML
[2] Document Type Definition

gar nicht realisieren lassen. Für das angestrebte Einlesen in eine MySQL-Datenbank ist die Version mit der XML-DTD jedoch völlig ausreichend und vereinfacht sogar das Parsen der Daten. Beim Verwenden der Version 2.0 wäre es notwendig, das zur Verfügung gestellte „CVE XML 2.0 Schema“ zu berücksichtigen, obwohl sich die Menge der bereitgestellten Informationen nicht unterscheidet.

Die Datei `nvdcve-2002.xml` enthält sämtliche Schwachstellen der Jahre 1988 bis 2002 und die weiteren Dateien dann fortführend die Einträge bis zum Ende des angegebenen Jahres. Weiterhin stehen auch die Dateien `nvdcve-modified.xml` und `nvdcve-recent.xml` zur Verfügung. Diese beiden Feeds werden nicht nur täglich, sondern alle zwei Stunden aktualisiert und enthalten ausschließlich die letzten veränderten beziehungsweise veröffentlichten Datenbankeinträge.

Um einen Einblick ist die Struktur der XML-Feeds zu erhalten folgt ein Auszug der ersten und letzten sechs Zeilen der Datei `nvdcve-2008.xml`:

Listing 4.1: Auszug des XML-Feeds `nvdcve-2008.xml` der National Vulnerability Database

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <nvd xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://nvd.nist.gov/feeds/
       cve/1.2" nvd_xml_version="1.2" pub_date="2009-06-04" xsi:schemaLocation="http://nvd.nist
       .gov/feeds/cve/1.2 http://nvd.nist.gov/schema/nvdcve.xsd">
3      <entry CVSS_vector="(AV:N/AC:L/Au:N/C:N/I:N/A:P)" CVSS_base_score="5.0"
           CVSS_exploit_subscore="10.0" CVSS_impact_subscore="2.9" name="CVE-2008-0061" seq
           ="2008-0061" severity="Medium" type="CVE" published="2008-01-03" CVSS_version="2.0"
           CVSS_score="5.0" modified="2008-09-05">
4          <desc>
5              <descript source="cve">MaraDNS 1.0 before 1.0.41, 1.2 before 1.2.12.08, and 1.3
                   before 1.3.07.04 allows remote attackers to cause a denial of service via a
                   crafted DNS packet that prevents an authoritative name (CNAME) record from
                   resolving, aka "improper rotation of resource records."</descript>
6          </desc>

238325             <ref source="SECUNIA" url="http://secunia.com/advisories/31787">31787</ref>
238326             <ref source="CONFIRM" url="ftp://ftp.software.ibm.com/ps/products/db2/fixes/
                       english-us/aparlist/db2_v82/APARLIST.TXT">ftp://ftp.software.ibm.com/ps/
                       products/db2/fixes/english-us/aparlist/db2_v82/APARLIST.TXT</ref>
238327         </refs>
238328     </entry>
238329
238330 </nvd>
```

Die erste Zeile in der Datei stellt den Header des Dokuments dar, der die verwendete XML-Version und die eingesetzte Zeichenkodierung deklariert. In der nächsten Zeile wird dann das Element `<nvd>` beschrieben und erst in der letzten Zeile durch das `</nvd>`-Tag wieder geschlossen. Ein XML-Dokument muss definitionsgemäß genau ein Element auf der höchsten Ebene – das so genannte Wurzelelement – enthalten, damit es *wohlgeformt* ist. Wohlgeformtheit bedeutet, dass ein Dokument gemäß den Regeln der XML-Spezifikation aufgebaut ist. Sobald eine der Regeln nicht erfüllt ist, bricht der Parser die Verarbeitung ab und gibt eine Fehlermeldung aus.

Die fünf Attribute des Elements `<nvd>` dienen dessen Beschreibung und heißen `xmlns:xsi`, `xmlns`, `nvd_xml_version`, `pub_date` sowie `xsi:schemaLocation`. Sämtliche Informationen befinden sich in dem Dokument verschachtelt in Unterelementen des Wurzelelements. In der zweithöchsten Ebene deklariert jeweils ein Unterelement `<entry>` einen Eintrag der NVD. Zeile 3 des Listings zeigt, dass wichtige Eigenschaften der Schwachstellen wie der CVSS-Vektor, CVE-Name oder das Veröffentlichungsdatum als Attribute von `<entry>` beschrieben sind. Innerhalb dieses Elements existieren weitere Unterelemente wie `<desc>` mit der Beschreibung der Schwachstelle, `<refs>` mit den externen Verweisen und `<vuln_soft>`, das alle Namen und Versionen der von der Schwachstelle betroffenen Produkte laut CPE enthält. Die korrekte Verschachtelung der Elemente ist ebenfalls ein Kriterium für wohlgeformte XML-Dokumente.

4.3 Transformation der XML-Daten

4.3.1 XSLT

Damit die Datensätze der NVD in eine eigene MySQL-Datenbank eingelesen werden können, müssen die heruntergeladenen XML-Feeds in ein für die Datenbank verständliches Format übertragen werden. Diese Umwandlung kann ein XML-Parser – auch „Prozessor“ genannt – in Kombination mit der Transformationssprache XSLT[3] übernehmen. Doug Tidwell schreibt in [Tid02] dazu:

> „XSLT, die Extensible Stylesheet Language for Transformations, ist eine offizielle Empfehlung des World Wide Web Consortium (W3C). Sie bietet eine flexible, leistungsfähige Sprache, mit der sich XML-Dokumente in etwas anderes umwandeln lassen [...] Sie schreiben ein XSLT-Stylesheet, um die Regeln für die Umwandlung eines XML-Dokuments festzulegen, und der XSLT-Prozessor erledigt den Rest.“

Um ein XML-Dokument in eine vom Benutzer selbst festgelegte Darstellung mittels XSLT überführen zu können, müssen bestimmte Voraussetzungen erfüllt sein. Zum einen – so schreibt Günter Born in [Bor05] – werde ein XSLT-Prozessor für die Transformation, ein wohlgeformtes XML-Dokument und ein XSL-Stylesheet benötigt. Den Umgang mit XML-Dokumenten beherrscht jeder aktuelle Webbrowser. Der Benutzer kann diesen zum Transformieren verwenden und damit auf den Einsatz spezieller Software verzichten. Die zweite Voraussetzung, ein wohlgeformtes Dokument als Quelle zu verwenden, ist mit den Data Feeds der NVD ebenfalls erfüllt. Eine wichtige Aufgabe besteht jedoch noch im Anfertigen einer Stylesheet-Datei mit den Vorgaben, wie das XML-Dokument in das Ausgabeformat zu überführen ist.

[3] Extensible Stylesheet Language for Transformations

4.3.2 XSL-Stylesheet

XSL[4]-Stylesheets bestehen aus Vorlagen zum Beschreiben der Ausgabe und Regeln, nach denen die Ausgaben erfolgen sollen. Um ein Stylesheet erstellen zu können, muss genau bekannt sein, wie das XML-Dokument aufgebaut ist. Die im Abschnitt 3.2.1 beschriebene NVD-Schwachstelle dient dabei als Beispiel zum Analysieren der XML-Struktur. Die Abbildung im Anhang A.5 zeigt, wie die Informationen zu der Schwachstelle für den Besucher der NVD-Website aufbereitet und präsentiert werden. Die dabei zu Grunde liegenden Daten einschließlich der vollständigen XML-Struktur zeigt das Listing „XML-Beschreibung für die Schwachstelle ‚CVE-2009-1428'" im Anhang A.8.1.

Die Kerninformationen zu einer Schwachstelle beschreiben die Attribute des Elements `<nvd>` in der zweiten Zeile des Listings. Dazu muss in der Stylesheet-Anweisung

```
<xsl:template match="nvd">
    <xsl:for-each select="entry">
    ...
    </xsl:for-each>
</xsl:template>
```

über den Attributwert `"nvd"` das XML-Wurzelelement angegeben werden, auf das die Vorlage (englisch *template*) anzuwenden ist. Die XSLT-Anweisung `xsl:for-each` bewirkt, dass alle durch das `select`-Attribut beschriebenen `<entry>`-Unterelemente bearbeitet werden. Innerhalb jedes `<entry>` Elements kann der Wert eines bestimmten Attributs über die Anweisung

```
<xsl:value-of select="@name"/>
```

in die Ausgabe übertragen werden. In diesem Fall lautet der Attributwert „CVE-2009-1424". Die Anweisung `xsl:value-of` wird durch das „/" am Ende geschlossen. Eine weitere XSLT-Anweisung lautet `<xsl:apply-templates>`. Sie bewirkt, dass innerhalb einer definierten Vorlage eine andere Vorlage angewandt wird. Im konkreten Fall ist die Anweisung

```
<xsl:apply-templates select="desc"/>
```

notwendig, um die Beschreibung der Schwachstelle im Unterelement `<desc>` auszugeben.

Obwohl es eine Vielzahl von XSLT-Anweisungen gibt, lassen sich mit Hilfe der vier beschriebenen die Inhalte der XML Feeds in ein selbst bestimmtes Ausgabeformat überführen. Jedes XSL-Stylesheet muss selbst ein wohlgeformtes Dokument sein, also allen XML-Anforderungen genügen. Dazu gehört auch eine Deklaration in der ersten Zeile der Datei, die damit das Wurzelelement des Stylesheets darstellt. Die Anweisung definiert die eingesetzte XSLT-Version 2.0 und den XSL-Namensraum, der eine eindeutige Unterscheidung der einzelnen Elemente beschreibt:

[4] Extensible Stylesheet Language

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Das vollständige XSL-Stylesheet, das den Dateinamen `xsl-style.xsl` trägt, ist im Anhang A.8.2 als auskommentierter Quelltext zu finden. Es übergibt dem Parser sämtliche Informationen aus dem XML-Dokument, die für das spätere Analysieren der Schwachstellen von Bedeutung sein können. Zu diesen zählen neben den Attributen des Elements `<entry>` und der Schwachstellen-Beschreibung im Unterelement `<desc>` auch die bereits beschriebenen Elemente `<refs>` und `<vuln_soft>`.

4.4 Erstellen der MySQL-Anweisungen

Die endgültige Ausgabe der XML-Umwandlung soll eine Reihe vollständiger Anweisungen zum Einfügen der NVD-Datensätze in eine MySQL-Tabelle sein. Die Tabelle „Übersicht zu wichtigen MySQL-Befehlen“ im Anhang A.4 beschreibt die grundlegenden Möglichkeiten wie das Erstellen einer Tabelle oder das anschließende Einfügen von Daten in eine Tabelle. Die Schwachstellen der NVD sollen am Ende in einer Tabelle mit dem Namen *data* vorliegen. Dazu wird die Tabelle durch die Anweisung

```
CREATE TABLE data(
   id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(30),
   type VARCHAR(30),
   modified VARCHAR(30),
   published VARCHAR(30),
   description VARCHAR(4000),
   seq VARCHAR(30),
   severity VARCHAR(30),
   cvss_vector VARCHAR(30),
   cvss_base_score FLOAT,
   cvss_exploit_subscore FLOAT,
   cvss_impact_subscore FLOAT,
   cvss_version VARCHAR(30),
   cvss_score FLOAT
);
```

erstellt. MySQL legt in dieser Tabelle 14 Spalten an. Für jede Spalte muss ein Name und der Datentyp angegeben werden. Die Spaltennamen entsprechen dabei den Attributen des XML-Elements `<entry>`und die verwendeten Datentypen sind „VARCHAR“ für Zeichenketten mit variabler Länge und „FLOAT“ für Fließkommazahlen bei den CVSS-Werten. Der in Klammern angegebene Wert bei den „VARCHAR“-Spalten gibt die maximale Länge für die Zeichenkette an. Die Spalte erhält

zusätzlich noch die Angabe, dass sie den im Abschnitt „Relationales Datenbankmodell“ erwähnten Primärschlüssel enthält.

Für jede eingetragene Schwachstelle kann problemlos ein eigener Datensatz anlegen werden, jedoch existieren dazu in der Regel mehrere externe Quellen und Anwendungen. Zu der Schwachstelle „CVE-2009-1428“ sind beispielsweise 9 Quellen im XML-Element `<refs>` und sogar 27 Produkte in `<vuln_soft>` eingetragen. Um sämtliche Informationen in einer Datenbank zu organisieren, werden daher drei Tabellen benötigt: eine für die Informationen zu den Schwachstellen, eine Quellen- und eine Produkt-Tabelle. Sie erhalten in Anlehnung an die englischsprachige Quelle die Namen *data*, *refs* und *products*.

Die notwendige Verknüpfung der Datensätze zwischen diesen drei Tabellen stellt dabei der Primärschlüssel her. Die *data*-Tabelle benötigt daher zwingend eine Spalte für den Primärschlüssel, die den Namen „id“ erhält. Da jeder Primärschlüssel einzigartig sein muss, bietet es sich an, eine fortlaufende Nummerierung für die Spalte „id“ zu vergeben. Eine einfache Möglichkeit, diese zu erzeugen, stellt JavaScript zur Verfügung. Diese Skriptsprache wird im Header eines HTML-Dokuments wie folgt eingebunden:

Listing 4.2: Einbinden von JavaScript im Stylesheet `xsl-style.xsl`

```
6   <script language="Javascript">
7       <![CDATA[   var countID = 1;
8                   function zaehlen() {
9                       countID = countID+1;
10                  }
11      ]]>
12  </script>
```

Dieses kurze Javascript initialisiert eine Variable „countID“ mit dem Wert 1 und deklariert die Funktion „zaehlen“, die bei jedem Aufruf die Variable um eins erhöht. Da jede Schwachstelle, also jedes Element `<entry>`, eine eigene „id“ benötigt, ist die Verwendung der Funktion „zaehlen“ an die Transformationssprache XSLT gebunden. Aus diesem Grund wird das JavaScript im HTML-Abschnitt des XSL-Stylesheets `xsl-style.xsl` eingebunden. Zu beachten ist noch, dass das JavaScript in einer `CDATA`-Umgebung beschrieben wird. Diese ist notwendig, um Fehlinterpretationen zu vermeiden. In [Bor05] steht dazu:

> „XML-Elemente können jedoch Daten enthalten, die nicht für den XML-Prozessor bestimmt sind [...] Um zu verhindern, dass der XML-Prozessor diese Teile des XML-Dokuments auswertet und falsch unterpretiert, müssen Sie diese mit einer CDATA-Anweisung klammern.“

Die `CDATA`-Umgebung wird auch beim Einfügen der „id“ in die MySQL-Anweisung und beim Funktionsaufruf zum Erhöhen von „countID“ benötigt. Die Anweisung

Listing 4.3: Einfügen der Zählvariable im Stylesheet `xsl-style.xsl`

```
27          <script language="Javascript">
28              <![CDATA[document.write(countID);]]>
29          </script>,
```

setzt für das erste `<entry>`-Element die auf den Wert 1 gesetzte Variable ein. Nach dem Verarbeiten eines Elements ruft

Listing 4.4: Ausführen von JavaScript im Stylesheet `xsl-style.xsl`

```
77          <script language="Javascript">
78              <![CDATA[zaehlen();]]>
79          </script>
```

die Funktion „zaehlen“ auf, so dass „countID“ von 1 auf 2 erhöht wird. Anschließend verarbeitet der XSLT-Prozessor die nachfolgenden `<entry>`-Elemente.

Für das Beispiel der Schwachstelle „CVE-2009-1428“ lautet die vollständige MySQL-Anweisung

Listing 4.5: MySQL-Anweisung zum Einfügen der Schwachstelle „CVE-2009-1428“

```
INSERT INTO data (id, name, type, modified, published, description, seq, severity,
    cvss_vector, cvss_base_score, cvss_exploit_subscore, cvss_impact_subscore,
    cvss_version, cvss_score) VALUES
    (1166, "CVE-2009-1428", "CVE", "2009-05-14", "2009-04-29", "Multiple cross-site
    scripting (XSS) vulnerabilities in ccLgView.exe in the Symantec Log Viewer, as used
    in Symantec AntiVirus (SAV) before 10.1 MR8, Symantec Endpoint Protection (SEP) 11.0
    before 11.0 MR1, Norton 360 1.0, and Norton Internet Security 2005 through 2008, allow
    remote attackers to inject arbitrary web script or HTML via a crafted e-mail message,
    related to two parsing errors.", "2009-1428", "Medium",
    "(AV:N/AC:M/Au:N/C:N/I:P/A:N)", "4.3", "8.6", "2.9", "2.0", "4.3");
```

Die „id“ mit dem Wert 1166 deutet darauf hin, dass diese Schwachstelle als 1166. Element innerhalb der XML-Datei `nvdcve-2009.xml` beschrieben wird. Um die Verbindung mit den zugehörigen Datensätzen in den Quellen- und Produkt-Tabellen herstellen zu können, erhalten die Einträge in diesen Tabellen ebenfalls den Identifikationswert 1166. Um Verwechslungen mit der Bezeichnung „id“ in der *data*-Tabelle zu vermeiden, tragen die Spalten in den Tabellen *refs* und *products* jeweils die Bezeichnung „pkey_id“. Die Verknüpfung zwischen der Daten- und Produkttabelle spielt im Rahmen dieser Arbeit eine besondere Rolle, denn sie ist notwendig, um aus den kompletten Datenbestand die Schwachstellen zu ermitteln, die bestimmte Sicherheitssoftware betreffen. Die MySQL-Anweisungen für das Erzeugen der drei Tabellen sind im Anhang A.8.3 vollständig aufgeführt.

Mit dem vorliegenden Stylesheet können jetzt alle notwendigen MySQL-Anweisungen erstellt werden. Es wird dafür jedoch noch ein einfaches HTML-Dokument zum Aufruf der Transformation

benötigt. Dieses ist im Anhang A.8.4 als vollständiger Quelltext dokumentiert. Für die acht heruntergeladenen XML-Dateien `nvdcve-2002.xml` bis `nvdcve-2009.xml` wird jeweils ein eigenes HTML-Dokument benötigt. Diese erhalten dementsprechend die Dateinamen `xmltransformation02.html` bis `xmltransformation09.html` und sind außer im Wert der Variable „xml_path“, die auf die jeweilige XML-Datei verweist, identisch. Das HTML-Dokuments verwendet je nach eingesetztem Webbrowser einen speziellen XML-Aufruf für den „Microsoft Internet Explorer“ [Mica] oder für andere bekannte Produkte wie „Mozilla Firefox“ [Moz] oder „Opera“ [Oped].

Bevor das Parsen mit einem Webbrowser erfolgreich durchgeführt werden kann, ist eine kleine Änderung in den XML-Feeds notwendig. Im Wurzelelement

```
<nvd xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://nvd.nist.gov/feeds/cve/1.2"
   nvd_xml_version="1.2"
   pub_date="2009-06-04"
   xsi:schemaLocation="http://nvd.nist.gov/feeds/cve/1.2
   http://nvd.nist.gov/schema/nvdcve.xsd">
```

beschreibt das Attribut `xmlns` den verwendeten Namensraum. Da der angegebene Attributwert `http://nvd.nist.gov/feeds/cve/1.2` nicht gültig ist, muss das Attribut aus dem Element entfernt werden. Ohne diese Änderung schlägt das Parsen der XML-Dateien fehl, da die Deklaration eines gültigen Namensraums eine notwendige Bedingung ist. Wird diese Angabe entfernt, verwendet der Parser stattdessen einen Standard-Namensraum. Das zum Entfernen des Attributes nötige Bearbeiten von großen Dateien ist für viele Editor-Programme sehr zeitaufwendig oder auch gar nicht möglich. Ein kostenloser und leistungsfähiger Editor, der die 15 Megabyte große Datei `nvdcve-2006.xml` mit sehr wenig Ressourcen des Rechners sehr schnell bearbeitet, ist „WinVi“ [Mol].

Die drei genannten Webbrowser verwenden verschiedene XML-Parser, die zwar identische Ergebnisse liefern, jedoch große Unterschiede in der Verarbeitungsgeschwindigkeit zeigten. So reagierte der Internet Explorer nach dem Öffnen von `xmltransformation09.html`, also dem Verarbeiten der 4,5 Megabyte großen XML-Datei `nvdcve-2009.xml`, zunächst einige Minuten lang gar nicht. Erst nachdem mehr als eine halbe Stunde vergangen ist, reagierte die Anwendung wieder und stellte die erzeugten Anweisungen korrekt dar. Der Browser Opera konnte das HTML-Dokument ebenfalls erfolgreich öffnen und schrieb die Anweisungen Zeile für Zeile direkt in das Fenster. Somit war jederzeit ersichtlich, wie viel XML-Daten der Parser bereits verarbeitet hat. Der gesamte Vorgang dauerte aber dennoch 25 Minuten.

Der dritte getestete Webbrowser Firefox überraschte mit eine sehr hohen Geschwindigkeit beim Parsen. Nach nur 20 Sekunden Verarbeitungszeit zeigte er die kompletten MySQL-Anweisungen an. Nach Auswählen von „Datei“ ⇒ „Seite Speichern unter...“ kann der angezeigte Inhalt in einer Datei gespeichert werden. Als Dateityp ist muss dabei „Textdateien“ ausgewählt werden, da sich in

der Ausgabedatei keine zusätzlichen HTML-Anweisungen befinden dürfen. Die Ausgabe des durch `xmltransformation09.html` erzeugten Inhalts erfolgt dabei in die Datei `insert09.txt`.

4.5 Einfügen der Datensätze in MySQL

Die jetzt vorliegenden Textdateien `insert02.txt` bis `insert09.txt` enthalten Tausende von `INSERT` Anweisungen zum Einfügen von Datensätzen in eine MySQL-Tabelle. Doch bevor sie eingesetzt werden können, müssen die entsprechenden Datenbanken und Tabellen vorhanden sein. Eine neue Datenbank „nvd02“ für die Schwachstellen des Jahres 2002 wird durch

```
CREATE DATABASE nvd02;
USE nvd02;
```

erstellt und anschließend als aktive Datenbank ausgewählt. Nun müssen noch die drei Tabellen so wie im Anhang A.8.3 beschrieben angelegt werden. Die benötigten Anweisungen sind in der Datei `tabellen-anlegen.txt` festgehalten und können über die MySQL-Kommandozeile mit

```
SOURCE tabellen-anlegen.txt;
```

ausgeführt werden. Jetzt sind die Tabellen im passenden Format erstellt und der Datenimport kann durch Ausführen von

```
SOURCE insert02.txt;
```

gestartet werden. Je nach Leistungsfähigkeit des Datenbankservers kann das Einfügen der Datensätze etwa 20 bis 30 Minuten in Anspruch nehmen. Zur Vereinfachung des Imports aller Einträge von 2002 bis 2009 führt die Datei `nvd-import.txt` alle nötigen MySQL-Anweisungen nacheinander aus, so dass keine Benutzereingaben mehr erforderlich sind. Der Inhalt der Datei ist im Anhang A.8.5 „Vollständiger Datenimport in MySQL“ dokumentiert.

Zur Suche nach möglichen Fehlern hat MySQL alle Ein- und Ausgaben in der fast 120 Megabyte großen Datei `MySQL.log` dokumentiert. Dazu erfolgte der Start der Kommandozeile mit den zusätzlichen Parametern `-tee=MySQL.log -v`, welche die Ausgabedatei bestimmen und angeben, das ausführliche Format zu verwenden. Die Fehlersuche innerhalb der Datei ergab 369 Einträge, die unter Angabe der Fehlermeldung

```
ERROR 1265 (01000): Data truncated for column 'cvss_base_score' at row 1
```

nicht eingelesen werden konnten. Die Ursache dafür sind fehlende Angaben zu den CVSS Scores im XML-Feed. Die betreffenden 369 Schwachstellen sind jedoch im Element `<desc>` durch den zusätzlichen Hinweis „** REJECT ** DO NOT USE THIS CANDIDATE NUMBER.“ und anschließendem Verweis auf die korrekten CVE-Bezeichnungen für ungültig erklärt, so dass die Fehlermeldungen gerechtfertigt sind.

In allen Tabellen der acht Datenbanken befinden sich nun 36 772 Einträge zu Schwachstellen und mit Berücksichtigung der Quellen- und Produkt-Tabellen sind es insgesamt 551 939 Datensätze. Im Anhang A.7 befindet sich dazu eine detaillierte Übersicht zur Anzahl der Datensätze in den einzelnen Tabellen.

Kapitel 5

Auswertung der Schwachstellen

5.1 Vereinfachung der Datenbankstruktur

Die vorliegenden acht MySQL-Datenbanken „nvd02“ bis „nvd09“ stellen die Datenbasis für die Auswertung der Schwachstellen dar. Die Abfrage der Datensätze erfolgt über teilweise sehr komplexe `SELECT`-Anweisungen, die dem Anwender sehr viele Möglichkeiten geben. Obwohl MySQL die Möglichkeit bereitstellt, Tabellen aus verschiedenen Datenbanken in einer Abfrage zu bearbeiten, hat es sich als sehr nützlich herausgestellt, die acht Datenbanken zu vereinen. Das vereinfacht auf der einen Seite die Abfrage-Anweisungen, doch auf der anderen Seite zeigten sich datenbankübergreifende `SELECT`-Anweisungen bei so einem großen Datenumfang als sehr zeitintensiv.

Zum Vereinen der acht Datenbanken sind zwei Schritte notwendig. Zuerst müssen die Werte in der Spalte „id“ beziehungsweise „pkey_id“ in den Quellen- und Produkt-Tabellen aller Datenbanken so verändert werden, dass eindeutige Zuordnungen bestehen bleiben. In jeder Datenbank werden die Identifikationsnummern in der selben Weise vergeben. In der vereinten Datenbank gäbe es dann mehrere Datensätze mit dem selben Primärschlüssel, was nicht zulässig ist. Eine einfache Lösung zum Verändern der Identifikationsnummern ist das Addieren einer bestimmten Konstante in den einzelnen Datenbanken. Die entsprechende Jahreszahl der Datenbank eignet sich dafür sehr gut. Die Anweisungen für die Datenbank „nvd02“ zeigt der folgende Auszug der im Anhang A.8.6 gezeigten Datei `nvd-neue-id.txt`.

Listing 5.1: Verändern der Identifikationsnummern in den Tabellen der Datenbank „nvd02“

```
1 USE nvd02;
2 UPDATE data SET id = (200200000 + id);
3 UPDATE refs SET id = (200200000 + id);
4 UPDATE refs SET pkey_id = (200200000 + pkey_id);
5 UPDATE products SET id = (200200000 + id);
6 UPDATE products SET pkey_id = (200200000 + pkey_id);
```

Dadurch verändert sich beispielsweise der Wert in der Spalte id des 5 000. Eintrags in der Tabelle von „5000“ auf „200205000“ und ist damit in der vereinten Datenbank einzigartig. Um alle entsprechenden Spalten der acht Datenbanken zu verändern, können die entsprechenden Anweisungen durch

```
SOURCE nvd-neue-id.txt;
```

ausgeführt werden. Nachdem alle Datensätze eindeutig zugeordnet werden können, erfolgt das eigentliche Vereinen der Datenbanken. Eine einfache Möglichkeit bietet die `INSERT`-Anweisung in Kombination mit `SELECT`. So werden alle Spalten der gesamten *data*-Tabelle in der Datenbank „nvd02“ durch

```
INSERT data SELECT * from nvd02.data;
```

ausgewählt und in die *data*-Tabelle der neuen vereinten Datenbank „nvd“ eingefügt. Die vollständigen Anweisungen für alle Tabellen stellt das Listing im Anhang A.8.7 dar. Die jetzt erstellte Datenbank „nvd“ ist die Grundlage zur Auswertung der Schwachstellen.

5.2 Abfragen der Informationen

Die Anweisung `SELECT` ist stellt die Grundlage für das Abfragen von Informationen der MySQL-Datebank dar. Bei der Berücksichtigung mehrerer Kriterien wie CVSS-Schweregrad entstehen sehr komplexe Anweisungen. Die folgenden sechs Diagramme zeigen Möglichkeiten, die Daten auszuwerten. Die zugehörigen Daten und die zu Grunde gelegten MySQL-Anweisungen befinden sich in dem beigefügten „Microsoft Excel“-Dokument `MySQL-Auswertung.xlsx`.

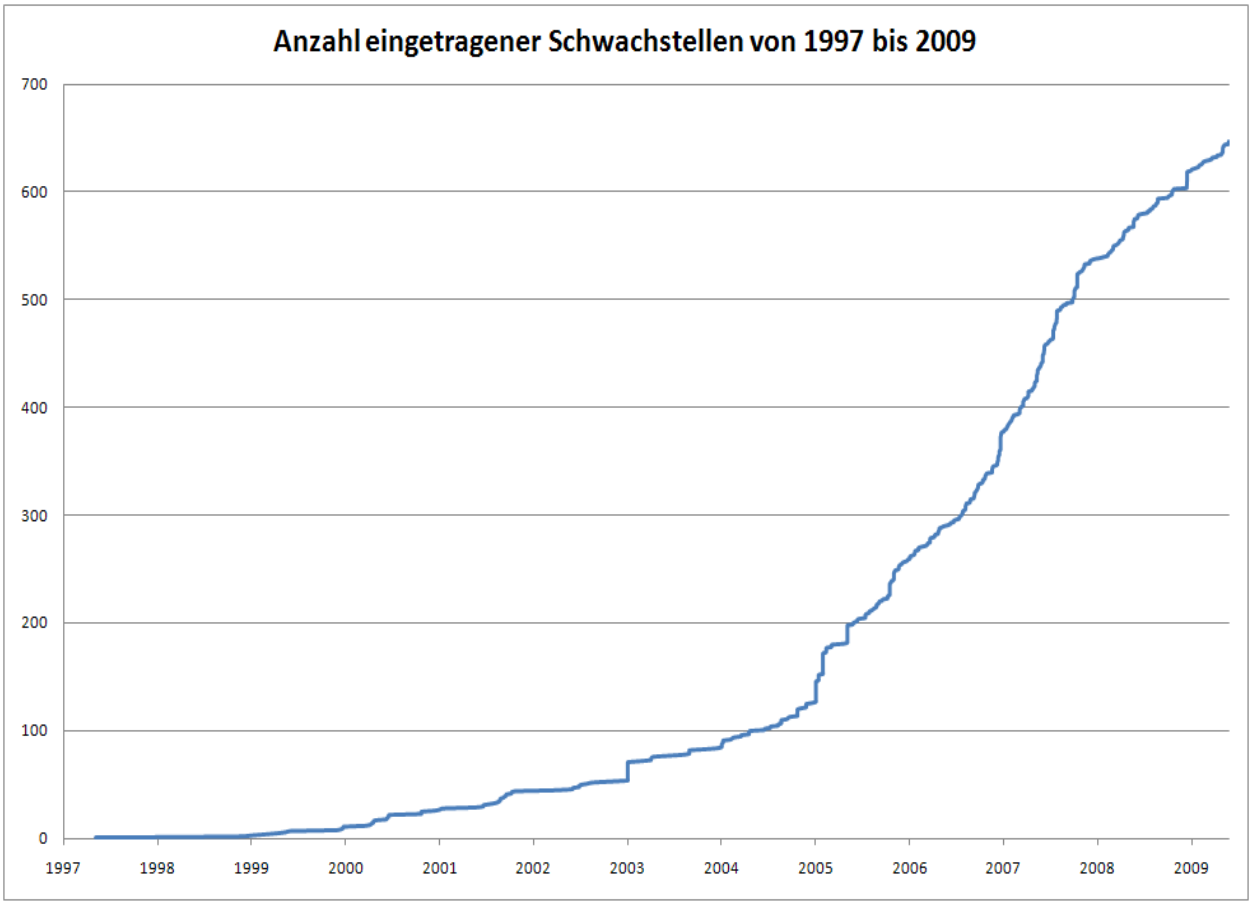


Abbildung 5.1: Anzahl eingetragener Schwachstellen von 1997 bis 2009

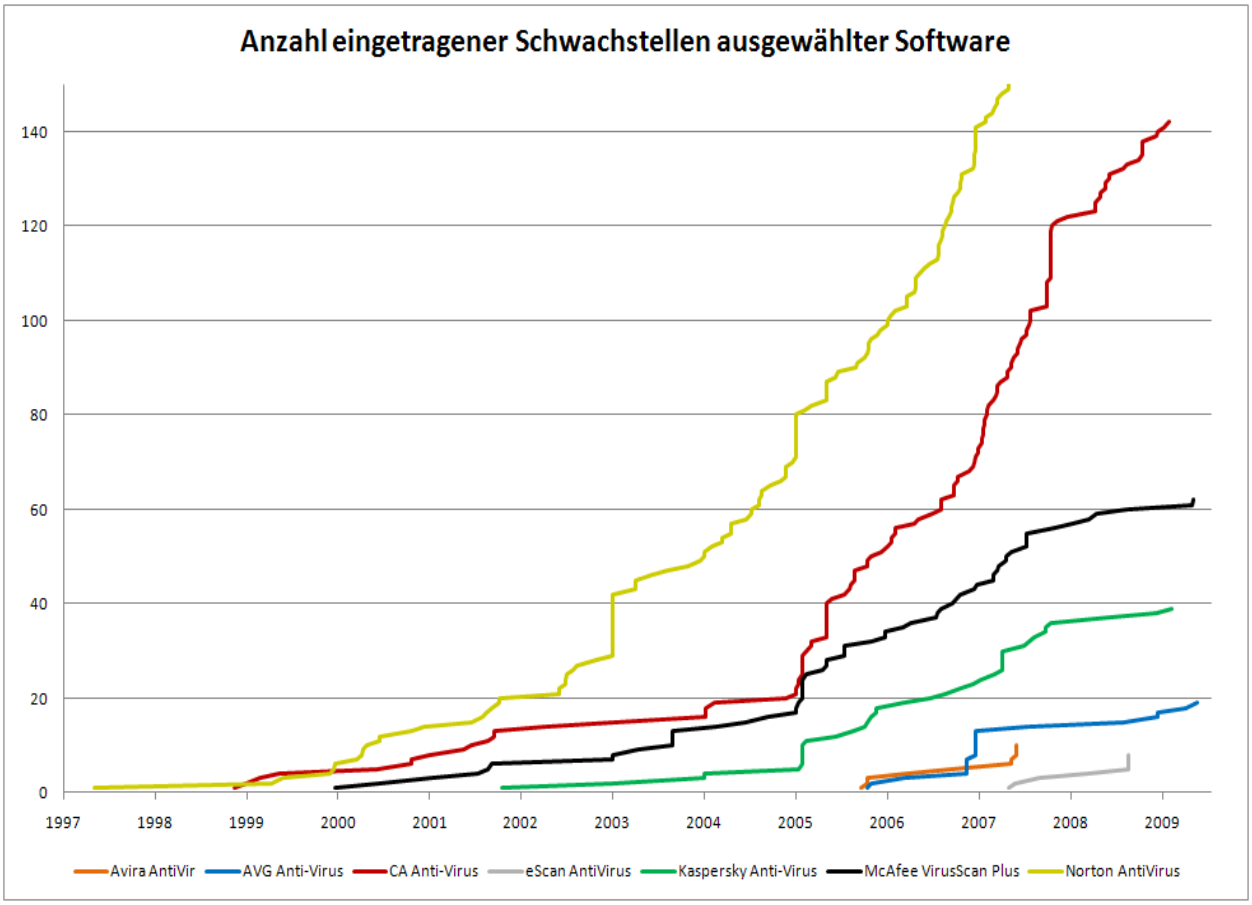


Abbildung 5.2: Anzahl eingetragener Schwachstellen ausgewählter Software

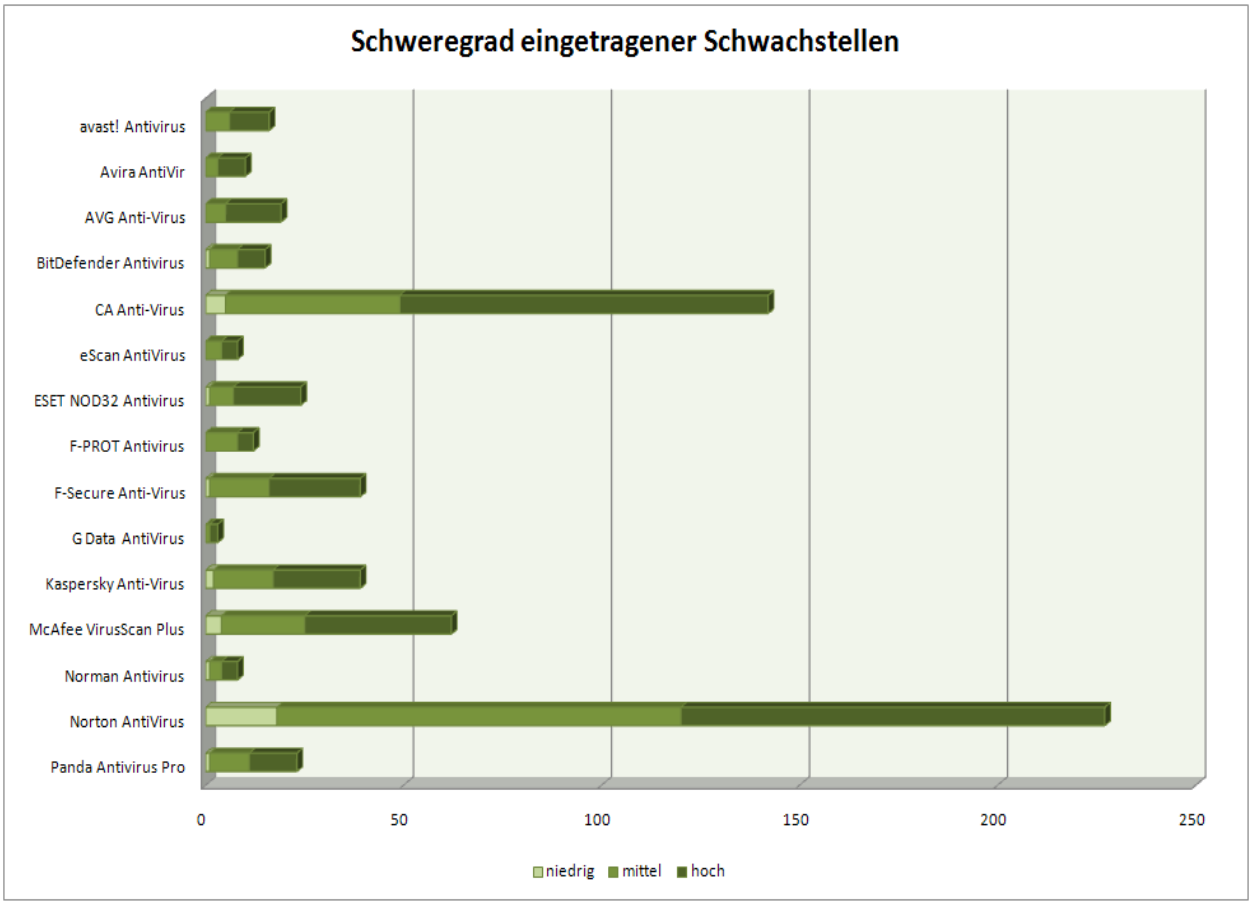


Abbildung 5.3: Schweregrad eingetragener Schwachstellen

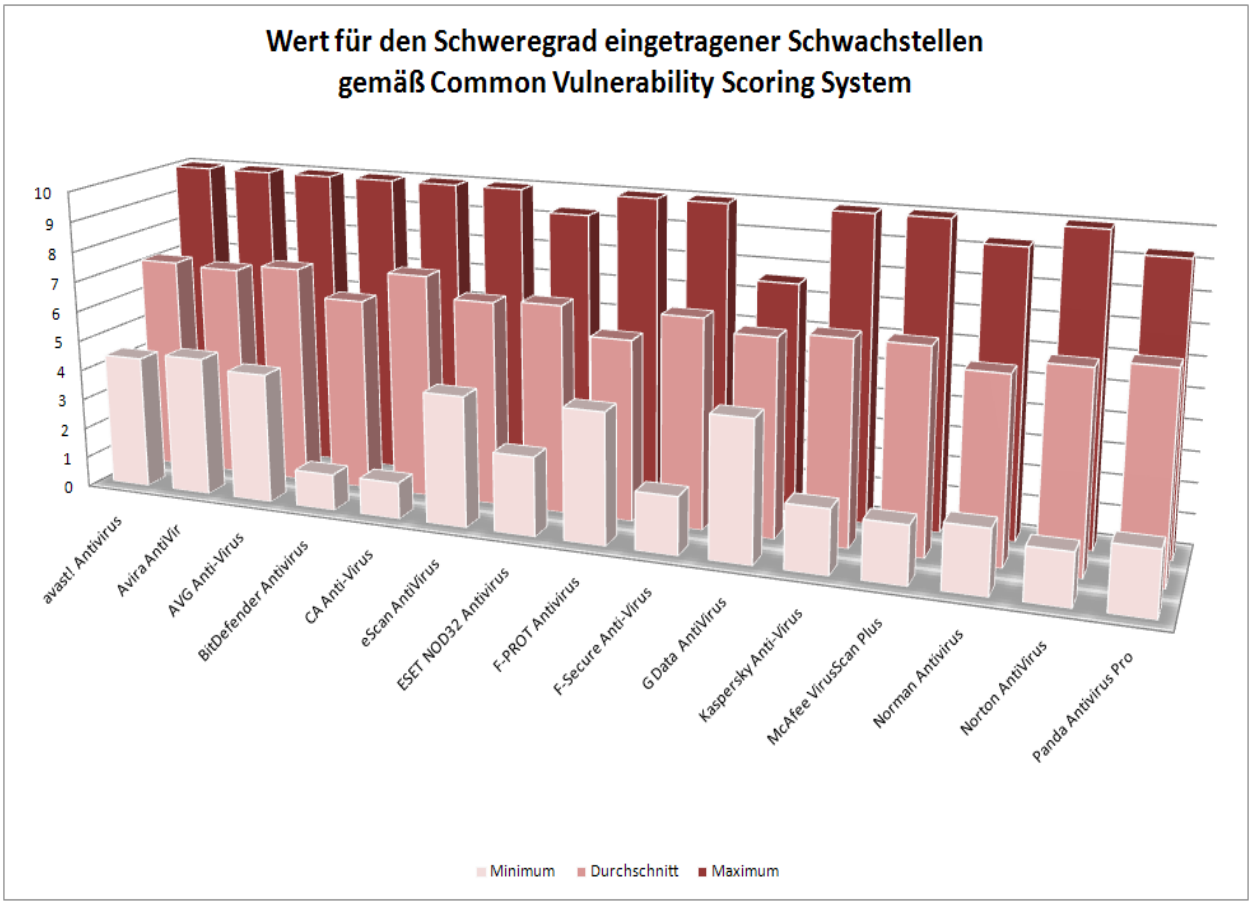


Abbildung 5.4: Wert für den Schweregrad eingetragener Schwachstellen gemäß Common Vulnerability Scoring System

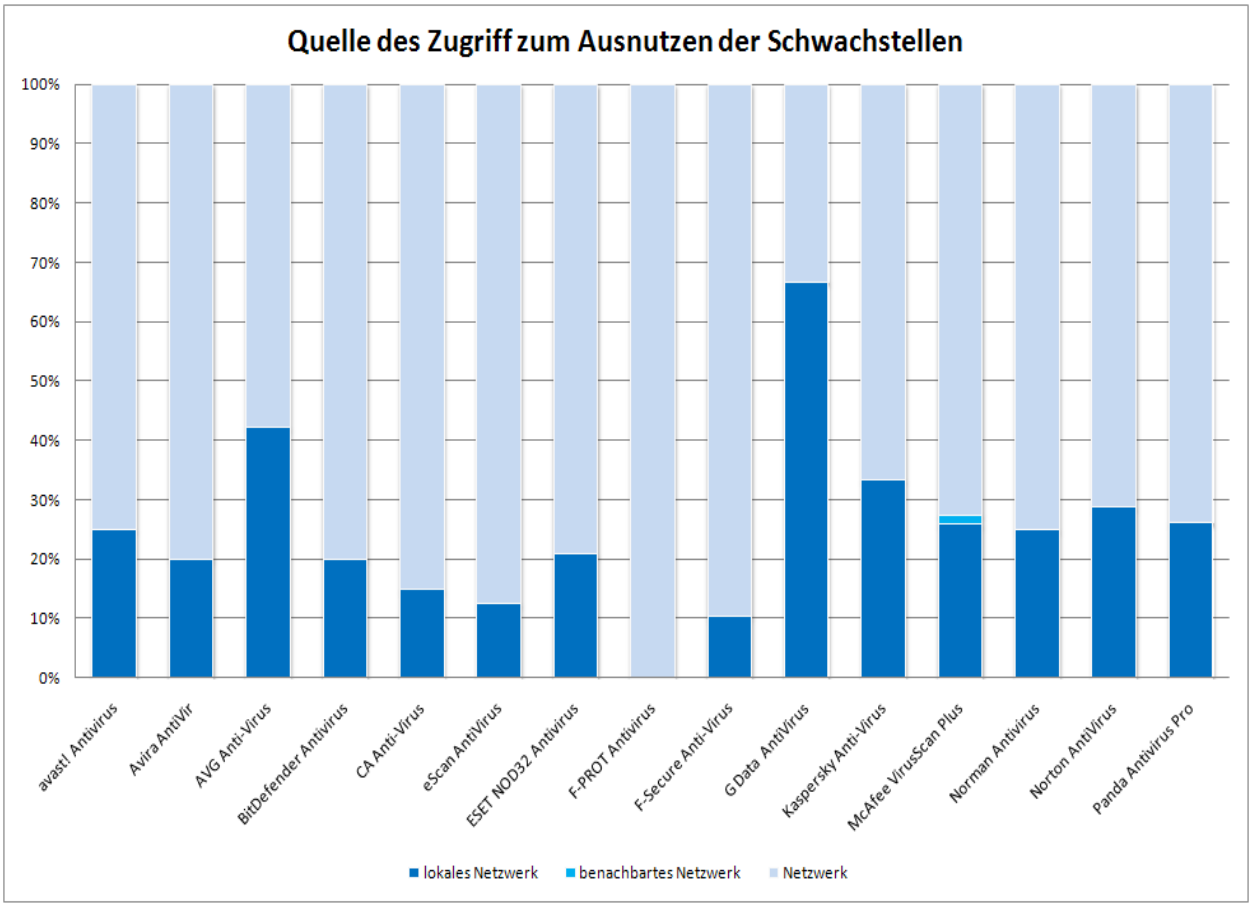


Abbildung 5.5: Quelle des Zugriff zum Ausnutzen der Schwachstellen

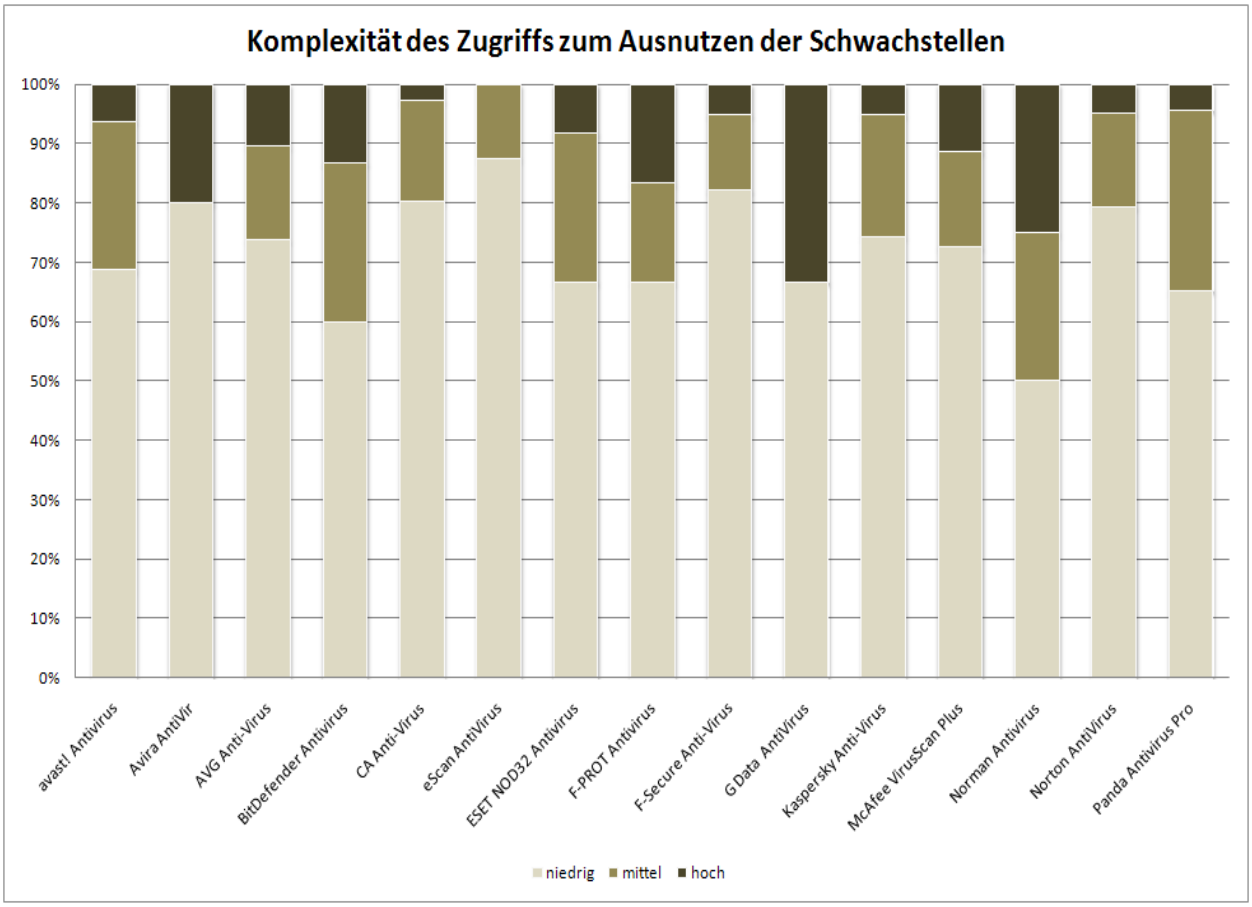


Abbildung 5.6: Komplexität des Zugriffs zum Ausnutzen der Schwachstellen

Kapitel 6

Schlussbetrachtungen

6.1 Ergebnisse der Arbeit

Die Kernaufgabe dieser Arbeit bestand darin, Schwachstellen in Sicherheitssoftware zu untersuchen. Dazu besteht am Ende dieser Arbeit eine eigene MySQL-Datenbank, die den Datenbestand der National Vulnerability Database als Grundlage hat. Diese Datenbank stellt eine Grundlage für umfassende Untersuchungen von Schwachstellen dar.

Die Qualität der Einträge in der gewählten Quelle ist sehr hoch, eine klare Einschätzung über die veränderte Verwundbarkeit eines System bei Verwendung einer entsprechenden Sicherheitssoftware vorzunehmen, ist mit den vorliegenden Daten jedoch schwer. Weiterhin ist zu beachten, dass bei der Analyse nur Einträge berücksichtig werden können, denen ein gemeldeter Vorfall zu Grunde liegt. Auch der durch das leistungsfähige Common Vulnerability Scoring System beschriebenen Schweregrad der Schwachstellen unterscheidet sich bei den 15 untersuchten Anwendungen nur wenig, so dass die Aussagekraft der Ergebnisse stark begrenzt ist.

Einer Software aufgrund zahlreicher Einträge in der Datenbank eine hohe Verwundbarkeit zu Grunde zu legen, ist nicht ohne Weiteres möglich. Dabei muss berücksicht werden, wie weit verbreitet diese Anwendung ist. Produkte, die nur von einem kleinen Anwenderkreis benutzt werden, sind ein weniger attraktives Angriffsziel als bekannte Größen der Sicherheitsindustrie. Leider existieren zur Verbreitung von Sicherheitssoftware keine verlässlichen Angaben, mit denen man die gefundenen Schwachstellen in Relation setzen kann.

Weiterhin ist es schwierig, die Menge der Einträge zu bewerten, da in der National Vulnerability Database keine Angaben hinterlegt sind, ob die Schwachstelle bereits geschlossen ist oder ob die Anwendung an diesem Punkt immer noch verwundbar ist.

6.2 Schwierigkeiten

Der Weg zur eigenen Datenbank bereitete zahlreiche Schwierigkeiten. So stellte sich die ursprüngliche Idee, einen vorgefertigten XML-Parser einzusetzen, als nicht realisierbar heraus. Die Suche nach weiteren Mitteln, die das Umwandeln der Formate ermöglichen oder unterstützen sollen, war zunächst erfolgreich, im konkreten Fall scheiterten sie jedoch bei der Anwendung auf die vorliegenden XML-Feeds.

Eines der wichtigsten Ziele war es, den Aufbau der eigenen Datenbank gut zu strukturieren. Ein großer Teil der Arbeit bestand deshalb darin, selbständig eine funktionierende Methode zur zielgerichteten Aufbereitung der Informationen zu erstellen. Im Laufe der Zeit stellte sich die Integration der Transformationssprache XSLT als unumgänglich heraus und erforderte das Einarbeiten ist die neue Thematik.

Durch den erhöhten Aufwand bei der Problematik des Importieren konnten in dieser Arbeit nicht alle Punkte der Aufgabenstellung gleichmäßig stark berücksichtigt werden. Dieses Vorgehen ist mit dem zuständigen Betreuer abgestimmt.

6.3 Ausblick

Die vorliegende Arbeit ist eine Grundlage für das umfassende Analysieren von Datenbank-Einträgen. Sie kann dazu verwendet werden, um im Rahmen weiterer Untersuchungen neue Erkenntnisse zu Schwachstellen zu sammeln. Auch eine Realisiserung von automatisiertem Einbinden über eine Skriptsprache ist denkbar.

Das National Institute of Standards and Technology kann seine Schwachstellen-Datenbank um die Möglichkeit eines direkten MySQL-Exports des Datenbestandes erweitern. Die Alternative, die Open Source Vulnerability Database, besitzt diese Möglichkeit bereits.

Literatur- und Quellenverzeichnis

[AGO] AGOF - Arbeitsgemeinschaft Online Forschung e.V.: *Graphiken zu dem Berichtsband 'internet facts 2009-I'.* http://www.agof.de/index.605.html, Abruf: 21. August 2009.

[Bor05] Born, Günter: *Jetzt lerne ich XML.* Markt und Technik, ISBN 3-8272-68540, 2005.

[Buna] Bundesamt für Sicherheit in der Informationstechnik: *Bürger-CERT - Ins Internet - mit Sicherheit!* https://www.buerger-cert.de/default.aspx, Abruf: 12. September 2009.

[Bunb] Bundesamt für Sicherheit in der Informationstechnik: *Die Lage der IT-Sicherheit in Deutschland 2009.* https://www.bsi.bund.de/cln_155/DE/Publikationen/Lageberichte/lageberichte_node.html, Abruf: 27. August 2009.

[Car] Cartwright, John: *The Full-Disclosure Archives.* http://lists.grok.org.uk/pipermail/full-disclosure, Abruf: 12. September 2009.

[Eck07] Eckert, Claudia: *IT-Sicherheit: Konzepte - Verfahren - Protokolle.* 5. Auflage. Oldenbourg Verlag, ISBN 978-3-486-58270-3, 2007.

[FIR] FIRST.org Inc.: *Common Vulnerability Scoring System (CVSS-SIG).* http://www.first.org/cvss, Abruf: 6. September 2009.

[Fri07] Friedmann, Katharina: *Virenscanner öffnen Hackern die Türen.* Computerwoche, Ausgabe 47/2007.

[Goo] Google Inc.: *Unternehmensbezogene Informationen – Sicherheit und Produktsicherheit bei Google.* http://www.google.de/corporate/security.html, Abruf: 5. September 2009.

[Heia] Heise Media UK Ltd.: *Industry Connections Security Group formed.* http://www.h-online.com/security/news/114044, Abruf: 9. September 2009.

[Heib] Heise Zeitschriften Verlag GmbH & Co. KG: *Hackergruppe sagt Sicherheitsindustrie den Kampf an.* http://www.heise.de/security/news/meldung/141960, Abruf: 9. September 2009.

[Heic] Heise Zeitschriften Verlag GmbH & Co. KG: *heise online*. http://www.heise.de, Abruf: 11. September 2009.

[Heid] Heise Zeitschriften Verlag GmbH & Co. KG: *Neues Rating-System für Sicherheitsfehler vorgestellt*. http://www.heise.de/security/news/meldung/64047, Abruf: 11. September 2009.

[HR] Hunziker, Stefan ; Riihs, Simon: *Risikoexposition bei Einsatz von Open-Source und proprietären Browsern*. http://www.ie.iwi.unibe.ch/publikationen/journals, Abruf: 5. September 2009.

[ICS] ICSA Labs: *ICSA Labs*. http://www.icsa.net, Abruf: 18. August 2009.

[Ini] Initiative D21 e.V.: *(N)ONLINER Atlas 2009*. http://www.initiatived21.de/wp-content/uploads/2009/06/NONLINER2009.pdf, Abruf: 21. August 2009.

[Ins] Institute of Electrical and Electronics Engineers Inc.: *Industry Connections Security Group (ICSG)*. http://standards.ieee.org/prod-serv/indconn/icsg/index.html, Abruf: 26. August 2009.

[Jot] Jotti: *Jottis Malwarescanner*. http://virusscan.jotti.org/de/scanresult/d6d7ff72804708ec7fb18cf706c05f0378872174, Abruf: 26. August 2009.

[Kas] Kaspersky Lab: *Software-Schwachstellen*. http://www.viruslist.com/de/hackers/info?chapter=153349525, Abruf: 27. August 2009.

[KK07] Kannengiesser, Caroline ; Kannengiesser, Matthias: *PHP 5 / MySQL 5*. 2., überarbeitete Ausgabe, Studienausgabe. Franzis Verlag, ISBN 978-3-7723-6919-3, 2007.

[KW00] Kersten, Heinrich ; Wolfenstetter, Klaus-Dieter: *Handbuch der Informations- und Kommunikationssicherheit. Gefahren. Standards. Szenarien.* Deutscher Wirtschaftsdienst, ISBN 978-3-87156-403-1, 2000.

[Mica] Microsoft Corporation: *Internet Explorer 8: Startseite*. http://www.microsoft.com/germany/windows/internet-explorer/default.aspx, Abruf: 7. September 2009.

[Micb] Microsoft Corporation: *Microsoft-Analyse zur IT-Sicherheit – Ausgabe 6 (Microsoft Security Intelligence Report)*. http://www.microsoft.com/sir, Abruf: 7. September 2009.

[Micc] Microsoft Corporation: *Office 2003 und XML*. http://msdn.microsoft.com/de-de/library/bb978980.aspx, Abruf: 7. September 2009.

[MIT] MITRE Corporation: *CWE-79: Failure to Preserve Web Page Structure ('Cross-site Scripting')*. http://cwe.mitre.org/data/definitions/79.html, Abruf: 21. August 2009.

[Mol] Molle, Raphael: *WinVi*. http://www.winvi.de/de, Abruf: 7. September 2009.

[Moz] Mozilla Foundation: *Webbrowser Firefox - Schneller, sicherer & anpassbar - Mozilla Europe*. http://www.mozilla-europe.org/de/firefox, Abruf: 7. September 2009.

[Nata] National Institute of Standards and Technology: *Common Vulnerability Scoring System Version 2 Calculator*. http://nvd.nist.gov/cvss.cfm?calculator&version=2, Abruf: 6. September 2009.

[Natb] National Institute of Standards and Technology: *CWE – Common Weakness Enumeration*. http://nvd.nist.gov/cwe.cfm#cwes, Abruf: 6. September 2009.

[Natc] National Institute of Standards and Technology: *DRAFT CVSS v2.10 Equations*. http://nvd.nist.gov/cvsseq2.htm, Abruf: 6. September 2009.

[Natd] National Institute of Standards and Technology: *Guide to Adopting and Using the Security Content Automation Protocol*. http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf, Abruf: 15. August 2009.

[Nate] National Institute of Standards and Technology: *National Vulnerability Database*. http://nvd.nist.gov, Abruf: 6. September 2009.

[Natf] National Institute of Standards and Technology: *Search CVE and CCE Vulnerability Database*. http://web.nvd.nist.gov/view/vuln/search, Abruf: 6. September 2009.

[Natg] National Institute of Standards and Technology: *Vulnerability Summary for CVE-2009-1428*. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-1428, Abruf: 6. September 2009.

[Opea] Open Source Vulnerability Database: *54132 : Symantec Log Viewer ccLgView.exe Email Filtering Statistics XSS*. http://osvdb.org/54132, Abruf: 7. September 2009.

[Opeb] Open Source Vulnerability Database: *57606 : Mozilla Firefox HTTP Response Location Header data: URI XSS*. http://osvdb.org/57606, Abruf: 7. September 2009.

[Opec] Open Source Vulnerability Database: *OSVDB: The Open Source Vulnerability Database*. http://osvdb.org, Abruf: 7. September 2009.

[Oped] Opera Software ASA: *Opera-Browser - Schneller und sicherer Internet-Zugriff - Kostenloser Download*. http://de.opera.com, Abruf: 7. September 2009.

[Sch06] Schmidt, Klaus: *Der IT Security Manager*. Hanser Fachbuchverlag, 978-3-446-40490-8, 2006.

[Seca] Secunia: *A rough 24 hours for Windows users - 81.01Secunia.com*. http://secunia.com/blog/20, Abruf: 13. September 2009.

[Secb] SECUNIA: *Secunia Personal Software Inspector (PSI)*. http://secunia.com/vulnerability_scanning/personal, Abruf: 13. September 2009.

[Secc] SECURITYFOCUS: *Bugtraq*. http://www.securityfocus.com/archive/1, Abruf: 13. September 2009.

[Secd] SECURITYFOCUS: *Multiple Symantec Products Log Viewer Multiple Script Injection Vulnerabilities*. http://www.securityfocus.com/bid/34669, Abruf: 13. September 2009.

[Sece] SECURITYFOREST: *Category:ExploitTree - SecurityForest*. http://www.securityforest.com/wiki/index.php/Category:ExploitTree, Abruf: 13. September 2009.

[Sei] SEILER, Martin: *CVSS: Einheitlicher Standard zur Einstufung von IT-Gefahren rückt näher*. http://www.tecchannel.de/432169, Abruf: 6. September 2009.

[SR05] SCHOOLMANN, Jürgen ; RIEGER, Holger: *Praxishandbuch IT-Sicherheit. Risiken, Prozesse, Standards*. Symposion Publishing, ISBN 3-936608-94-6, 2005.

[SRM07] SCHUMACHER, Markus ; RÖDIG, Utz ; MOSCHGATH, Marie-Luise: *Hacker Contest: Sicherheitsprobleme, Lösungen, Beispiele*. Springer Verlag, ISBN 3-540-41164-X, 2007.

[Suna] SUN MICROSYSTEMS INC.: *Über MySQL*. http://www.mysql.de/about, Abruf: 28. August 2009.

[Sunb] SUN MICROSYSTEMS INC.: *MySQL Downloads*. http://dev.mysql.com/downloads, Abruf: 28. August 2009.

[Tid02] TIDWELL, Doug: *XSLT. XML-Dokumente transformieren*. O'Reilly Verlag, ISBN 3-89721-292-7, 2002.

[Wil] WILDLIST ORGANIZATION INTERNATIONAL: *WildList*. http://www.wildlist.org/WildList, Abruf: 21. August 2009.

[Zol] ZOLLER, Thierry: *Secdev - Thierry Zoller*. http://blog.zoller.lu, Abruf: 10. September 2009.

Erklärung

Ich versichere an Eides statt, dass ich die beiliegende Diplomarbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe.
Ferner gestatte ich der Hochschule für Technik und Wirtschaft Dresden, die beiliegende Diplomarbeit unter Beachtung insbesondere urheber-, datenschutz- und wettbewerbsrechtlicher Vorschriften für Lehre und Forschung zu nutzen.
Mir ist bekannt, dass für die Weitergabe oder Veröffentlichung der Arbeit die Zustimmung der HTW Dresden sowie der an der Aufgabenstellung und Durchführung der Arbeit unmittelbar beteiligten Partnereinrichtungen erforderlich ist.

_____

Datum/Unterschrift

Glossar

Attribut

Ein *Attribut* ist ein Beschreibungsmerkmal, das einem Objekt wie Datei oder Datenfeld einer Datenbank zugewiesen werden kann. 51

Beta-Version

Die *Beta-Version* einer Anwendung ist eine bereits lauffähige, aber noch nicht für den produktiven Einsatz freigegebene Version eines Programms. Solche Fassungen werden häufig von freiwilligen „Beta-Testern“ oder auch Fachhändlern ausprobiert und dabei auf Fehleranfälligkeit überprüft. 24

Blog

Ein Blog ist die Kurzform von *Weblog* (englisch für „Netz-Logbuch“) ist eine Art webbasiertes und in der Regel öffentliches Tagebuch einer einzelnen Person. Die meisten Blogs bieten den Lesern die Möglichkeit, die einzelnen chronologisch sortierten Einträge zu kommentieren. 45

Boot-Sektor

Der erste Sektor auf einer Diskette oder einer Festplattenpartition heißt *Boot-Sektor*. In diesem Bereich sind Informationen über die Architektur des Datenträgers und optional ein Programm zum Starten des installierten Betriebssystems gespeichert. 21

Client

Ein *Client* (englisch für „Kunde“) nimmt Kontakt zu einem Server auf und fordert einen von ihm bereitgestellten Dienst an. Das kann auf Hardware- oder Software-Ebene geschehen. Ein einfaches Beispiel ist ein Webbrowser, der als Client Kontakt zu einem Webserver aufnimmt, um sich eine Webseite zuschicken zu lassen. 33

Comma Separated Values

Das Dateiformat *Comma Separated Values* (englisch für „durch Kommas getrennte Werte“) ermöglicht einen Austausch tabellarischer Daten. Die einzelnen Spalten einer Tabellenzeile sind einfach durch ein Komma oder Semikolon getrennt. 43

Contentfilter

Ein Contentfilter (englisch *content* = Inhalt) ist eine Software zum Verhindern, dass bestimmte Inhalte zum Anwender gelangen. Der Haupteinsatzbereich ist das Sperren illegaler, anstößiger oder jugendgefährdender Websites. 19

CPE

Common Platform Enumeration definiert eine Namensstruktur zur eindeutigen Identifikation von IT-Systemen wie Hardware oder Software. 41, 51

CVE

Common Vulnerabilities and Exposures (englisch für „Verbreitete Schwachstellen und Aufdeckungen“) ist ein Industriestandard, der eine Liste standardisierter Bezeichnungen zur eindeutigen Spezifizierung von Schwachstellen herausgibt und diese pflegt. Jeder CVE-Eintrag erhält einen eindeutigen Bezeichner, der sich aus der Jahreszahl und einer fortlaufenden Nummer zusammensetzt. 40, 51

CVSS

Das *Common Vulnerability Scoring System* (englisch für „Allgemeines Schwachstellenbewertungssystem“) ist eine Methodik zur Beschreibung und Bewertung von Schwachstellen. Es gibt an, wie kritisch sich die Schwachstelle auf bestimmte Bereiche auswirkt. 15, 40

CWE

Das Projekt *Common Weakness Enumeration* stellt einen Index über mögliche Arten von Software-Schwachstellen bereit, um deren Beschreibung zu vereinfachen. 41

Data Feed

Ein Data Feed (englisch *data* = Daten, *to feed* = füttern, zuführen) ist eine kompakte Datensammlung, die bei jeder Änderung aktualisiert wird und für den Benutzer jederzeit abrufbar ist. 48

De-facto-Standard

Als einen De-facto-Standard (lateinisch *de facto* = durch Fakten geschaffen), auch Quasi-Standard genannt, bezeichnet man eine Verfahrensweise oder Spezifikation, die durch den häufigen Gebrauch im Alltag als definierter Standard angesehen werden kann, obwohl diese nie offiziell als Industrienorm anerkannt wurde. Beispiele sind das Betriebssystem Microsoft Windows oder der Telekommunikationsdienst SMS. 33

Document Type Definition

Die *Document Type Definition*, auch Schema-Definition oder DOCTYPE genannt, ist eine formale Spezifikation über den Inhalt einer XML-Datei um Dokumente eines bestimmten Typs zu deklarieren. 49

Engine

Die *Engine* (englisch für „Motor“ oder „Antrieb“) bezeichnet einen eigenständigen Teil einer

Software, der für eine bestimmte Aufgabe entworfen wurde. 21

Exploit

Ein Exploit (englisch *to exploit* = ausnutzen) bezeichnet ein Programm, das eine Fehlfunktion einer Software ausnutzt um beispielsweise Privilegien oder Zugang zu fremden Systemen zu erhalten. 16, 45

false positive

Das Ergebnis eines Tests ist *false positive* (englisch für „falsch positiv"), wenn die notwendigen Kriterien für ein positives Ergebnis nicht erfüllt sind, jedoch fälschlicherweise als erfüllt erkannt werden. 23, 28

General Public License

Die *General Public License* ist eine Lizenzierung, die den privaten und kommerziellen Einsatz einer Software ohne jegliche Einschränkungen erlaubt. 33

Hacker

Ein *Hacker* ist eine Person, die sich unter Umgehung von Sicherheitsmechanismen unberechtigt Zugang zu einem anderen Computersystem verschafft. Oft geht es dabei um das Aufzeigen von Schwachstellen, ohne wirklich Schaden anzurichten. 14

Hardware

In der Computertechnik gehören zur *Hardware* alle materiellen Komponenten eines Computersystems wie beispielsweise Prozessor, Netzteil, Gehäuse oder Festplatte. 12

Header

Der Header (englisch *head* = Kopf) bezeichnet den Teil am Anfang einer Datei, der Informationen wie den Ursprung der Daten oder die Deklaration von Formaten beschreibt. 50

heuristisch

Ein heuristisches Verfahren (griechisch *heuriskein* = „entdecken") wendet bei wenig verfügbaren Informationen einfache Regeln an, um meist komplexe Probleme zu lösen. 23

HTML

Die *Hypertext Markup Language* (englisch für „Hypertext-Auszeichnungssprache") ist die Standardbeschreibungssprache für Dokumente im World Wide Web. 21, 49

IEEE

Das *Institute of Electrical and Electronics Engineers* ist ein internationaler Berufsverband von Ingenieuren aus den Bereichen Elektrotechnik und Informatik mit über 350 000 Mitgliedern aus 150 Ländern. 20

Integrität

Die Integrität (lateinisch *integer* = „unberührt") ist in der Informationstechnik ein Sicher-

heitsziel, das den Schutz vor Verlust und vorsätzlicher Veränderung von Daten beschreibt. Datenintegrität ist dann gewahrt, wenn der Inhalt der Daten auf der Sende- und Empfangsseite identisch ist, also nicht verändert wurde. 23

JavaScript

JavaScript ist eine objektorientierte Skriptsprache, die in HTML-Dokumente eingebettet werden kann. Sie ermöglicht eine dynamische und interaktive Gestaltung von Webseiten. 54

Kompromittierung

Ein System gilt als *kompromittiert*, wenn es aufgrund möglicher Manipulation oder ungewollter Weitergabe von Daten nicht mehr als vertrauenswürdig angesehen werden kann. 29

Konsistenz

In der Informatik sind Daten konsistent (lateinisch *con* = zusammen, *sistere* = halten), wenn die Wiederspruchsfreiheit innerhalb einer Datenbank gewährleistet ist. Dazu muss der Inhalt einer Datenbank bestimmte vordefinierte Konsistenzbedingungen erfüllen. 32

Mailingliste

Eine *Mailingliste* ist ein Dienst, der einer geschlossenen Gruppe von Mitgliedern den Austausch von Nachrichten in Form von elektronischen Rundschreiben ermöglicht. Jedes Mitglied erhält dabei alle Nachrichten innerhalb der Mailingliste. 38, 44

Makro

Ein Makro (griechisch *makros* = „groß“ oder „weit“) ist ein Programm, das eine vordefinierte Befehlssequenz oder Abfolge von Ationen enthält, um Routinearbeiten in Anwendungsprogrammen zu erleichtern. 21

Megabyte

Ein Byte ist ein Mengenbegriff für eine Zusammenstellung von 8 Bit und ein *Megabyte* bedeutet 1024 Kilobyte, was wiederum $1024 \cdot 1024$, also 1 048 576 Byte entspricht. 56

National Institute of Standards and Technology

Die US-amerikanischen Bundesbehörde *National Institute of Standards and Technology* ist für Standardisierungsprozesse zuständig und trug 1901 bis 1988 den Namen „The National Bureau of Standards“. 16, 41

National Vulnerability Database

Die *National Vulnerability Database* ist eine sehr umfangreiche Schwachstellen-Datenbank der US-amerikanischen Bundesbehörde „National Institute of Standards and Technology“. 39

Newsfeed

Ein Newsfeed (englisch *news* = Nachrichten, *to feed* = füttern, zuführen) übermittelt neue Einträge einer webbasierten Quelle, deren Inhalt sich der Anwender mit einem so genannten

„Feedreader“ anzeigen lassen kann. Newsfeeds werden beispielsweise von Nachrichtenportalen eingesetzt, um die neuesten Schlagzeilen schnell zu veröffentlichen. 46

OOXML

Office Open XML ist ein offener Standard für Dokumente mit dem Ziel, den Austausch zwischen verschiedenen Büroanwendungen zu ermöglichen. OOXML steht in Konkurrenz zu dem ebenfalls XML-basierten Standard ODF. 49

Open Source

Bei Software, die als *Open Source* (englisch für „quelloffen“) deklariert ist steht der Quelltext öffentlich zur Verfügung, so dass jeder Interessierte die Funktionsweise nachvollziehen und Veränderungen selbst vornehmen kann. 34, 43

Parser

Ein Parser (englisch *to parse* = analysieren) ist ein Software-Modul, das eine beliebige Eingabe, beispielsweise Dokumente oder Quellcode, syntaktisch analysiert, zerlegt und für die Weiterverarbeitung aufbereitet. Ein Parser für XML-Dokumente wird auch „XML-Prozessor“ genannt. 24, 28, 50, 51

Patch

Ein Patch (englisch *to patch* = flicken, ausbessern) steht für ein kleines Programm, das in einer bereits vorhandenen Anwendung Fehler ausbessert, Lücken schließt oder zusätliche Funktionen bereitstellt. Die Patches werden in der Regel von den Herstellern selbst kostenlos zur Verfügung gestellt. 13, 14

Personal Firewall

Eine *Personal Firewall* (englisch für „Private Brandwand“) ist eine Software, die den ein- und ausgehenden Netzwerkverkehr eines Computers zum Schutz vor Angriffen aus dem World Wide Web nach bestimmten Vorgaben überwacht und filtert. 19, 27

Prompt

Der Prompt (englisch *to prompt* = auffordern oder abfragen) bezeichnet die Markierung der Stelle, an der Befehle in einer Eingabekonsole eingegeben werden können. In der Windows-Eingabeaufforderung ist `C:\` der Prompt. 37

proprietär

Eine Software wird proprietär (lateinisch *Proprietarius* = der Eigentümer) genannt, wenn sie unter einer nicht-freien Lizenz steht. Beispiele sind Microsoft Windows und Adobe Flash. 33, 43

Prüfsumme

Die Bildung einer *Prüfsumme* ist ein Verfahren zur Fehlererkennung. Sie errechnet sich über einen mathematischen Algorithmus aus dem Inhalt einer Datei oder eines Datenpakets. 23

Repositorium

Ein Repositorium (englisch *repository* = Ablage oder Aufbewahrungsort) ist eine Informationsquelle, in der die systembeschreibenden Objekte und Verfahren in einer Datenbank oder Verzeichnisstruktur abgelegt sind. Es kann Informationen über verschiedene Versionen und Konfigurationen enthalten und verschiedenen Anwendungen den Zugriff auf eine einheitliche Informationsbasis ermöglichen. 39

Security Content Automation Protocol

Das *Security Content Automation Protocol* ist eine Zusammenfassung von mehreren offenen Standards zur Erfassung, Nomenklatur, Bewertung und Überprüfung von Schwachstellen in Software. 39

Server

Unter einem Server (englisch *to serve* = bedienen) wird ein Rechner oder Programm verstanden, das anderen Rechnern oder Programmen Dienstleistungen anbietet und erfüllt. In den meisten Fällen ist ein Server sehr leistungsfähig und wird nur auf Anfrage aktiv. 11, 48

Shell

Die *Shell* (englisch für „Hülle“ oder „Außenhaut“) ist Bestandteil einer Anwendung oder eines Betriebssystems und vereinfacht als Schnittstelle zwischen Benutzer und Rechner die Bedienung. Der Name stammt von Muschelschalen, die als Oberfläche zwischen den inneren Komponenten und dem Anwender von außen steht. 36

Software

Als *Software* bezeichnet man die Gesamtheit aller nicht physikalischen Bestandteile eines elektronischen Datenverarbeitungssystems, also Programme und Daten. 11

Spamfilter

Ein Spamfilter (englisch *spam* = Nachricht mit unerwünschtem Inhalt) ist eine Software oder ein Software-Modul, das eingehende E-Mails überprüft und diese bei einem Verdacht auf Spam entfernt oder in einen bestimmten Ordner verschiebt. Der Spamfilter kann seine Funktion entweder bereits im Mail-Server oder beim Anwender im E-Mail-Programm ausführen. 19

SQL

Die strukturierte Abfragesprache *Structured Query Language* wurde Mitte der 1970er Jahre von der Firma IBM für den Einsatz mit relationalen Datenbanken entwickelt. Die Syntax ist einfach aufgebaut und an die englische Umgangssprache angelehnt. 33, 43

Stylesheet

Ein Stylesheet (englisch *style* = Stil, *sheet* = Papierbogen) ist eine Art Formatvorlage und enthält Regeln zur Formatierung und Darstellung von Elementen. 51

Tag

In Auszeichnungssprachen wie XML oder HTML dient ein *Tag* (englisch für „Schlagwort“) der Auszeichnung von Textelementen. Der Titel einer HTML-Seite wird beispielsweise zwischen dem öffnenden `<title>`- und dem schließenden `</title>`-Tag beschrieben. 50

URL

Der *Uniform Resource Locator* (englisch für „einheitlicher Quellenzeiger“) ist ein standardisiertes Darstellungsverfahren zur Lokalisierung und Identifizierung von Ressourcen in Computernetzwerken in der Form `Protokoll://Host/Pfad`. 42

Virensignatur

Die *Virensignatur* stellt ein möglichst eindeutiges Erkennungsmerkmal eines Computervirus dar und kann als eine Art Fingerabdruck zu seiner Identifizierung verstanden werden. 22

Webbrowser

Ein Webbrowser (englisch *to browse* = stöbern, durchsuchen) ist eine Anwendung, mit dem der Benutzer auf Webseiten und andere Internet-Dienste zugreifen kann. 27, 51

Website

Eine Website (englisch *web* = Netz, *site* = Ort oder Platz) bezeichnet die Gesamtheit aller zusammengehörigen Webseiten eines Internetangebots. 34, 39

XML

Die *Extensible Markup Language* (englisch für „erweiterbare Auszeichnungssprache“) ist eine Standardsprache zum Beschreiben hierarchisch strukturierter Daten in maschinell interpretierbaren Dokumenten, wobei XML selbst die Regeln für die Struktur definiert. 43, 48

XML-Schema

Ein *XML-Schema* definiert die XML-Struktur nicht anhand einer DTD, sondern in Form eines separaten XML-Dokuments. Die Endung für die Dateien lautet üblicherweise „.xsd“, was für „XML Schema Definition“ steht. 49

XSL

Die *Extensible Stylesheet Language* ist eine erweiterte Beschreibungssprache für Stylesheets, um Layouts für XML-Dokumente zu definieren. 52

XSLT

Die XSL-Subsprache *Extensible Stylesheet Language for Transformations* kann zwischen verschiedenen XML-Sprachen konvertieren und die Ausgabe von XML-Dokumenten in beliebige Formate festlegen. 51

Anhang A

Anlagen zur Diplomarbeit

A.1 Angaben zu Quellen

Die Abbildung „Schematische Darstellung zum Aufbau eines Datenbanksystems“ ist mit der Anwendung „OpenOffice.org Draw“ entworfen und besitzt daher keine Quellenangabe. Die Diagramme im Kapitel „Auswertung der Schwachstellen“ hat die Tabellenkalkulations-Software „Microsoft Office Excel 2007“ auf Grundlage der abgefragten Werte aus der Datenbank erstellt.

Alle im Rahmen dieser Diplomarbeit verwendet elektronischen Quellen befinden sich auf der beiliegenden CD-ROM[1]. Im Verzeichnis `Quellen` befinden sich die gespeicherten Websites zum Zeitpunkt des im Literaturverzeichnis angegebenen Abrufdatums. In der selben Form sind im Verzeichnis `Antiviren-Software` alle Produktübersichten und Hersteller-Websites der im Anhang A.6 dargestellten Antiviren-Software gesichert.

Weiterhin sind alle im Abschnitt A.8 aufgeführten Quelltexte im Verzeichnis „MySQL Import“ der CD-ROM abgelegt. Auch die XML-Feeds und die knapp 120 Megabyte große Protokoll-Datei `MySQL.log` sind dort abgelegt.

Im Verzeichnis `MySQL Datenbank` befindet sich ein vollsändiger *Dump*[2] der MySQL-Datenbank. Diese Datei ermöglicht das Herstellen der gesamten Datenbank durch den einfachen Aufruf von `mysql -uroot -p<Passwort> nvd < nvd-dump.sql` über die Kommandozeile.

[1] Compact Disc Read-Only Memory
[2] englisch für „Speicherausdruck“

A.2 Bezeichnungen für einen erkannten Virus durch verschiedene Sicherheitssoftware

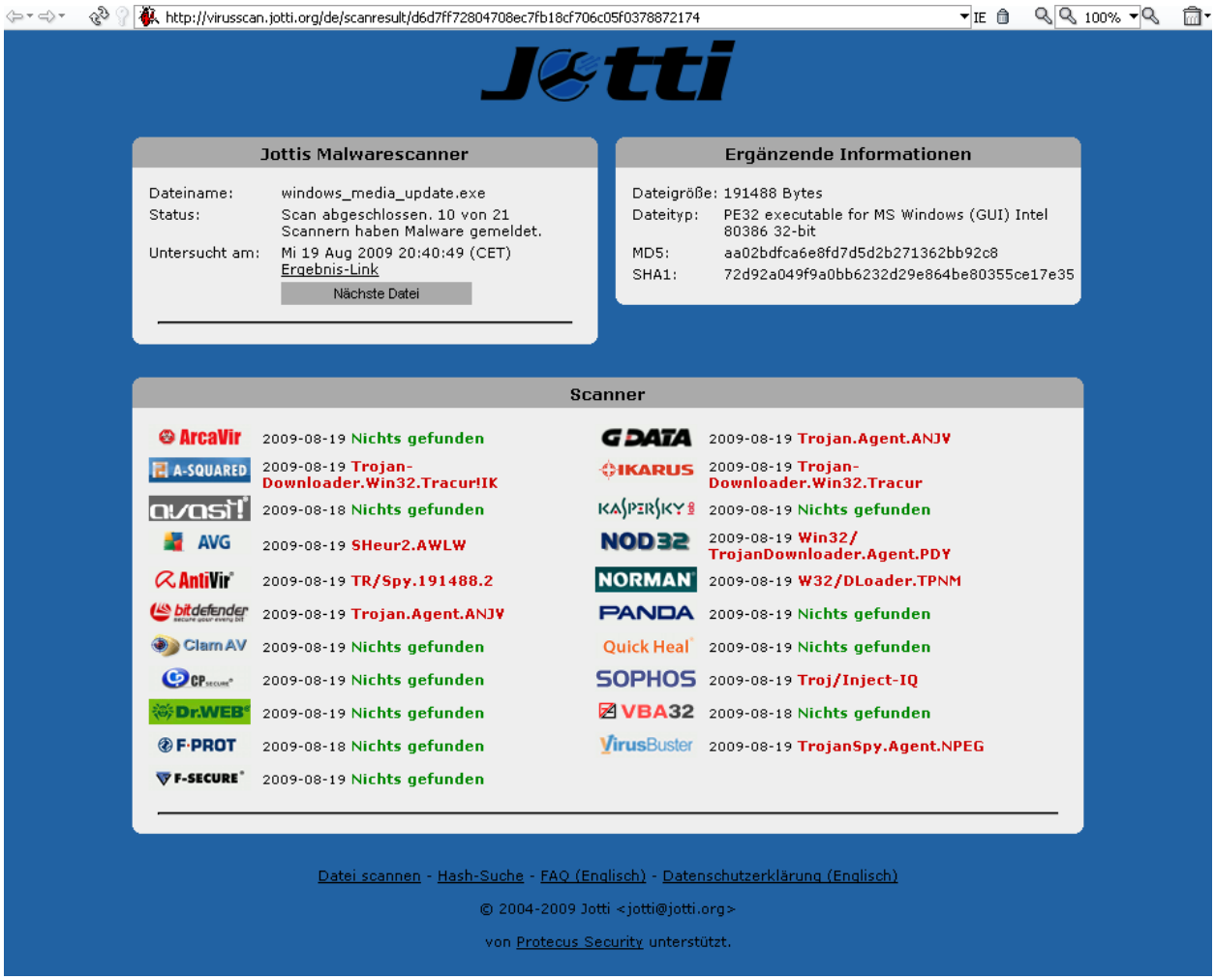


Abbildung A.1: Scan-Ergebnis verschiedener Antiviren-Programme nach dem Überprüfen der infizierten Datei „windows_media_update.exe“ – Stand: 26. August 2009 [Jot]

A.3 Secunia Personal Software Inspector

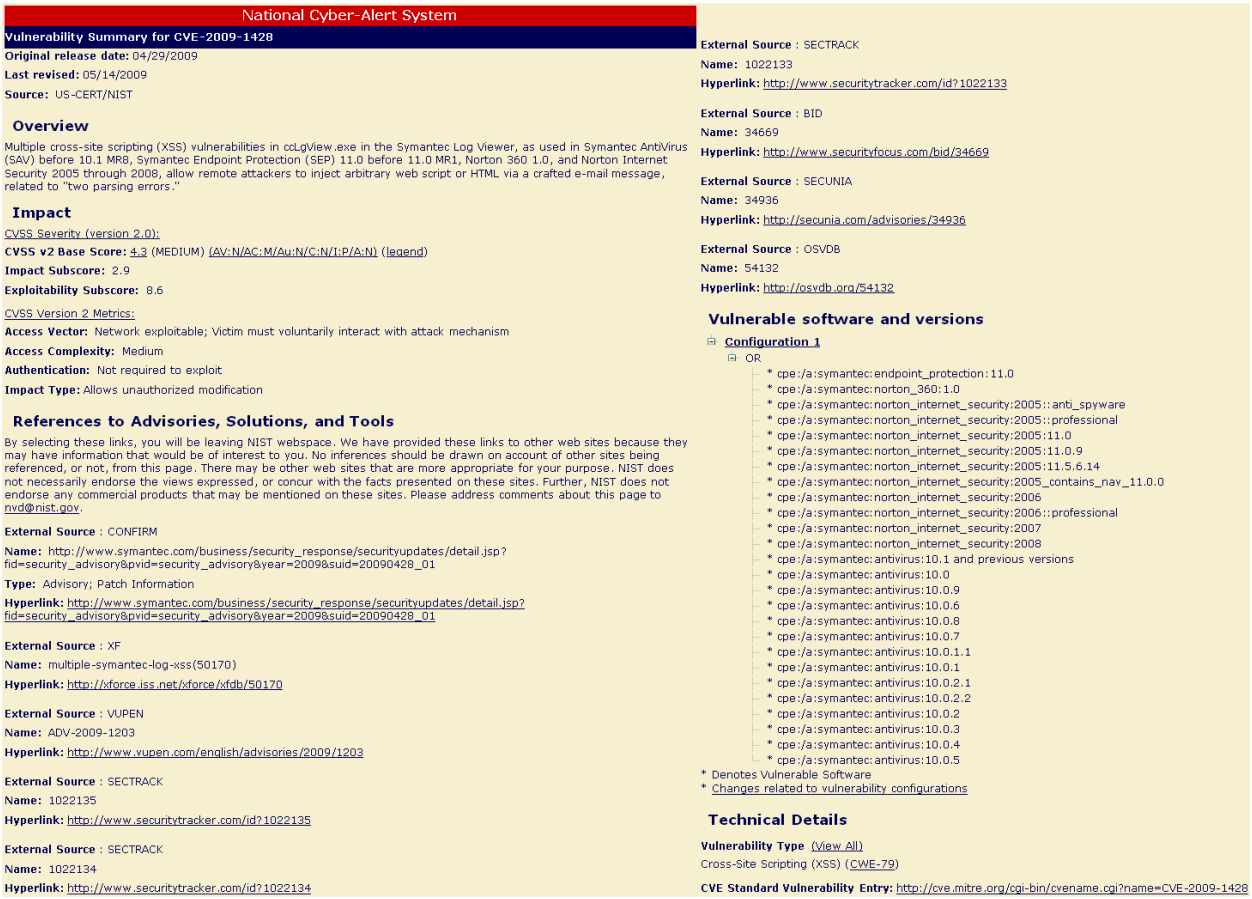
Abbildung A.2: Ergebnis der Suche nach unsicheren Programmen durch die Software „Secunia Personal Software Inspector“ [Secb]

A.4 Übersicht zu wichtigen MySQL-Befehlen

MySQL-Befehl	Beschreibung
`SHOW DATABASES;`	Anzeige aller auf dem System verfügbaren Datenbanken
`CREATE DATABASE Testdatenbank;`	Anlegen einer neue Datenbank ohne Tabellen
`USE Testdatenbank;`	Auswahl der aktiven Datenbank
`SHOW TABLES;`	Anzeige aller Tabellen in der aktiven Datenbank
`CREATE TABLE Testtabelle (Vorname VARCHAR(30), Nachname VARCHAR(30));`	Erstellen einer „Testtabelle“ mit den Spalten „Vorname“ und „Nachname“ des Spaltentyps Zeichenkette variabler Länge mit maximal 30 Zeichen
`INSERT INTO Testtabelle (Vorname, Nachname) VALUES ("Max", "Mustermann");`	Einfügen der Werte „Max“ und „Mustermann“ in die entsprechenden Spalten „Vorname“ und „Nachname“ der „Testtabelle“
`SELECT Vorname FROM Testtabelle ORDER BY Nachname;`	Abfrage und Ausgabe aller Werte der Spalte „Vorname“ aus der „Testtabelle“ nach „Nachname“ sortiert ab
`DROP TABLE Testtabelle;`	Löschen der „Testtabelle“
`DROP DATABASE Testdatenbank;`	Löschen der „Testdatenbank“ einschließlich der Tabellen

Tabelle A.1: Übersicht zu wichtigen MySQL-Befehlen [KK07]

A.5 Darstellung einer Schwachstelle in der National Vulnerability Database



National Cyber-Alert System

Vulnerability Summary for CVE-2009-1428

Original release date: 04/29/2009

Last revised: 05/14/2009

Source: US-CERT/NIST

Overview

Multiple cross-site scripting (XSS) vulnerabilities in ccLgView.exe in the Symantec Log Viewer, as used in Symantec AntiVirus (SAV) before 10.1 MR8, Symantec Endpoint Protection (SEP) 11.0 before 11.0 MR1, Norton 360 1.0, and Norton Internet Security 2005 through 2008, allow remote attackers to inject arbitrary web script or HTML via a crafted e-mail message, related to "two parsing errors."

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 4.3 (MEDIUM) (AV:N/AC:M/Au:N/C:N/I:P/A:N) (legend)

Impact Subscore: 2.9

Exploitability Subscore: 8.6

CVSS Version 2 Metrics:

Access Vector: Network exploitable; Victim must voluntarily interact with attack mechanism

Access Complexity: Medium

Authentication: Not required to exploit

Impact Type: Allows unauthorized modification

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

External Source : CONFIRM

Name: http://www.symantec.com/business/security_response/securityupdates/detail.jsp?fid=security_advisory&pvid=security_advisory&year=2009&suid=20090428_01

Type: Advisory; Patch Information

Hyperlink: http://www.symantec.com/business/security_response/securityupdates/detail.jsp?fid=security_advisory&pvid=security_advisory&year=2009&suid=20090428_01

External Source : XF

Name: multiple-symantec-log-xss(50170)

Hyperlink: http://xforce.iss.net/xforce/xfdb/50170

External Source : VUPEN

Name: ADV-2009-1203

Hyperlink: http://www.vupen.com/english/advisories/2009/1203

External Source : SECTRACK

Name: 1022135

Hyperlink: http://www.securitytracker.com/id?1022135

External Source : SECTRACK

Name: 1022134

Hyperlink: http://www.securitytracker.com/id?1022134

External Source : SECTRACK

Name: 1022133

Hyperlink: http://www.securitytracker.com/id?1022133

External Source : BID

Name: 34669

Hyperlink: http://www.securityfocus.com/bid/34669

External Source : SECUNIA

Name: 34936

Hyperlink: http://secunia.com/advisories/34936

External Source : OSVDB

Name: 54132

Hyperlink: http://osvdb.org/54132

Vulnerable software and versions

- Configuration 1
  - OR
    - * cpe:/a:symantec:endpoint_protection:11.0
    - * cpe:/a:symantec:norton_360:1.0
    - * cpe:/a:symantec:norton_internet_security:2005::anti_spyware
    - * cpe:/a:symantec:norton_internet_security:2005::professional
    - * cpe:/a:symantec:norton_internet_security:2005:11.0
    - * cpe:/a:symantec:norton_internet_security:2005:11.0.9
    - * cpe:/a:symantec:norton_internet_security:2005:11.5.6.14
    - * cpe:/a:symantec:norton_internet_security:2005_contains_nav_11.0.0
    - * cpe:/a:symantec:norton_internet_security:2006
    - * cpe:/a:symantec:norton_internet_security:2006::professional
    - * cpe:/a:symantec:norton_internet_security:2007
    - * cpe:/a:symantec:norton_internet_security:2008
    - * cpe:/a:symantec:antivirus:10.1 and previous versions
    - * cpe:/a:symantec:antivirus:10.0
    - * cpe:/a:symantec:antivirus:10.0.9
    - * cpe:/a:symantec:antivirus:10.0.6
    - * cpe:/a:symantec:antivirus:10.0.8
    - * cpe:/a:symantec:antivirus:10.0.7
    - * cpe:/a:symantec:antivirus:10.0.1.1
    - * cpe:/a:symantec:antivirus:10.0.1
    - * cpe:/a:symantec:antivirus:10.0.2.1
    - * cpe:/a:symantec:antivirus:10.0.2.2
    - * cpe:/a:symantec:antivirus:10.0.2
    - * cpe:/a:symantec:antivirus:10.0.3
    - * cpe:/a:symantec:antivirus:10.0.4
    - * cpe:/a:symantec:antivirus:10.0.5

* Denotes Vulnerable Software

* Changes related to vulnerability configurations

Technical Details

Vulnerability Type (View All)

Cross-Site Scripting (XSS) (CWE-79)

CVE Standard Vulnerability Entry: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1428

Abbildung A.3: Detaillierte Darstellung der Schwachstelle „CVE-2009-1428“ auf der Website der National Vulnerability Database – Stand: 18. August 2009 [Natg]

A.6 Übersicht zu Sicherheitssoftware für Microsoft Windows

Produkt	Hersteller	Website zum Produkt
avast! Antivirus Home Edition	ALWIL Software a.s.	http://www.avast.de/index.php/avast-Home-Edition.html
AVG Anti-Virus Free Edition	AVG Technologies	http://www.avg.com/de.product-avg-anti-virus-free-edition
Avira AntiVir Personal	Avira GmbH	http://www.free-av.de/de/download/index.html
BitDefender Free Edition	Softwin	http://www.bitdefender.com/site/Main/view/Download-Free-Products.html

Tabelle A.2: Verfügbare kostenlose Sicherheitssoftware für Microsoft Windows

Produkt	Hersteller	Webseite zum Produkt
avast! Antivirus 4.8 Professional Edition	ALWIL Software a.s.	http://www.avast.de/index.php/avast-Professional-Edition.html
Avira AntiVir Premium 2009	Avira GmbH	http://www.avira.com/de/produkte/avira_antivir_premium.html
AVG Anti-Virus	AVG Technologies	http://www.avg.com/de.2213
BitDefender Antivirus 2009	Softwin	http://www.bitdefender.de/PRODUCT-2216-de--bitdefender-antivirus-2009.html
CA Anti-Virus 2009	CA Inc.	http://ca-store.de/virus/antivirus.aspx?sc_lang=de-DE
eScan AntiVirus Edition	MicroWorld Technologies Inc.	http://www.mwti.com/products/escan/escan_antivirushomeuser/escanav_homeuser.asp
ESET NOD32 Antivirus Home Edition	ESET	http://www.eset.de/produkte/eset-nod32-antivirus-home-edition
F-PROT Antivirus	FRISK	http://www.f-prot.com/products/corporate_users/win
F-Secure Anti-Virus 2009	F-Secure Corporation	http://www.f-secure.com/de_DE/products/home-office/antivirus/index.html
G Data AntiVirus 2010	G DATA Software AG	http://www.gdata.de/onlineshop/anti-virus-produkte/shop/2-privatanwender/883-g-data-antivirus-2010.html
Kaspersky Anti-Virus 2010	Kaspersky Lab	http://www.kaspersky.com/de/anti-virus
McAfee VirusScan Plus	McAfee Inc.	http://home.mcafee.com/store/package.aspx?pkgid=276
Norman Antivirus & Antispyware	Norman	http://www.norman.com/home/all_products/antivirus/norman_antivirus_antispyware/de
Norton AntiVirus 2009	Symantec Corporation	http://www.symantec.com/de/de/norton/antivirus
Panda Antivirus Pro 2010	Panda Security	http://www.pandasecurity.com/germany/homeusers/solutions/antivirus
Steganos AntiVirus 2009	Steganos GmbH	http://www.steganos.com/de/produkte/virenfrei-surfen/antivirus/uebersicht
TrustPort Antivirus 2009	TrustPort Inc.	http://www.trustport.com/en/products/home-and-small-office/trustport-antivirus

Tabelle A.3: Verfügbare kostenpflichtige Sicherheitssoftware für Microsoft Windows

A.7 Anzahl der Datensätze in den MySQL-Tabellen

Jahr	MySQL-Tabelle		
	data	*refs*	*products*
2002	6 650	21 830	27 335
2003	1 472	7 188	10 840
2004	2 630	15 795	27 155
2005	4 605	27 704	35 309
2006	6 968	56 162	40 407
2007	6 407	51 649	36 657
2008	6 494	44 361	65 680
2009	1 546	10 502	36 593
Summe	36 772	235 191	279 976

Tabelle A.4: Anzahl der Datensätze in den MySQL-Tabellen

A.8 Quelltexte

A.8.1 XML-Beschreibung für die Schwachstelle „CVE-2009-1428“

Listing A.1: XML-Beschreibung für die Schwachstelle „CVE-2009-1428“

```
<entry CVSS_vector="(AV:N/AC:M/Au:N/C:N/I:P/A:N)" CVSS_base_score="4.3"
    CVSS_exploit_subscore="8.6" CVSS_impact_subscore="2.9" name="CVE-2009-1428" seq
    ="2009-1428" severity="Medium" type="CVE" published="2009-04-29" CVSS_version="2.0"
    CVSS_score="4.3" modified="2009-05-14">
    <desc>
        <descript source="cve">Multiple cross-site scripting (XSS) vulnerabilities in
            ccLgView.exe in the Symantec Log Viewer, as used in Symantec AntiVirus (SAV)
             before 10.1 MR8, Symantec Endpoint Protection (SEP) 11.0 before 11.0 MR1,
            Norton 360 1.0, and Norton Internet Security 2005 through 2008, allow remote
             attackers to inject arbitrary web script or HTML via a crafted e-mail
            message, related to "two parsing errors."</descript>
    </desc>
    <loss_types>
        <int />
    </loss_types>
    <range>
        <network />
        <user_init />
    </range>
    <refs>
        <ref source="CONFIRM" patch="1" url="http://www.symantec.com/business/
            security_response/securityupdates/detail.jsp?fid=security_advisory&amp;pvid=
            security_advisory&amp;year=2009&amp;suid=20090428_01" adv="1">http://www.
            symantec.com/business/security_response/securityupdates/detail.jsp?fid=
            security_advisory&amp;pvid=security_advisory&amp;year=2009&amp;suid=20090428
            _01</ref>
        <ref source="XF" url="http://xforce.iss.net/xforce/xfdb/50170">multiple-symantec
            -log-xss(50170)</ref>
        <ref source="VUPEN" url="http://www.vupen.com/english/advisories/2009/1203">ADV
            -2009-1203</ref>
        <ref source="SECTRACK" url="http://www.securitytracker.com/id?1022135">1022135</
            ref>
        <ref source="SECTRACK" url="http://www.securitytracker.com/id?1022134">1022134</
            ref>
        <ref source="SECTRACK" url="http://www.securitytracker.com/id?1022133">1022133</
            ref>
        <ref source="BID" url="http://www.securityfocus.com/bid/34669">34669</ref>
        <ref source="SECUNIA" url="http://secunia.com/advisories/34936">34936</ref>
        <ref source="OSVDB" url="http://osvdb.org/54132">54132</ref>
    </refs>
    <vuln_soft>
        <prod vendor="symantec" name="antivirus">
            <vers num="10.0" />
            <vers num="10.0.1" />
            <vers num="10.0.1.1" />
            <vers num="10.0.2" />
            <vers num="10.0.2.1" />
```

```
30                <vers num="10.0.2.2" />
31                <vers num="10.0.3" />
32                <vers num="10.0.4" />
33                <vers num="10.0.5" />
34                <vers num="10.0.6" />
35                <vers num="10.0.7" />
36                <vers num="10.0.8" />
37                <vers num="10.0.9" />
38                <vers num="10.1" prev="1" />
39            </prod>
40            <prod vendor="symantec" name="endpoint_protection">
41                <vers num="11.0" />
42            </prod>
43            <prod vendor="symantec" name="norton_360">
44                <vers num="1.0" />
45            </prod>
46            <prod vendor="symantec" name="norton_internet_security">
47                <vers edition="" num="2005" />
48                <vers edition=":anti_spyware" num="2005" />
49                <vers edition=":professional" num="2005" />
50                <vers edition="11.0" num="2005" />
51                <vers edition="11.0.9" num="2005" />
52                <vers edition="11.5.6.14" num="2005" />
53                <vers num="2005_contains_nav_11.0.0" />
54                <vers edition="" num="2006" />
55                <vers edition=":professional" num="2006" />
56                <vers num="2007" />
57                <vers num="2008" />
58            </prod>
59        </vuln_soft>
60    </entry>
```

A.8.2 XSL-Stylesheet

Listing A.2: XSL-Stylesheet (`xsl-style.xsl`)

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">            <!-- Cursor auf die Wurzel des Dokuments setzen -->
    <html>                          <!-- ab jetzt folgt die HTML-Ausgabe -->
        <head>
            <script language="Javascript">
                <![CDATA[   var countID = 1;
                            function zaehlen() {
                                countID = countID+1;
                            }
                ]]>
            </script>
            <!-- das JavaScript erstellt einen Zähler, der später als ID verwendet wird -->
            <!-- der Parser interpretiert den Abschnitt "CDATA" nicht als XML-Quellcode -->
            <title>XML-Transformation</title>   <!-- Titel des HTML-Dokuments -->
        </head>
        <body>
            <xsl:apply-templates/>      <!-- HTML-"body" mit generierten MySQL-Befehlen -->
        </body>
    </html>
</xsl:template>

<xsl:template match="nvd">
    <xsl:for-each select="entry">       <!-- Auswahl für jedes Element <entry> -->
        INSERT INTO data (id, name, type, modified, published, description, seq, severity,
            cvss_vector, cvss_base_score, cvss_exploit_subscore, cvss_impact_subscore,
            cvss_version, cvss_score) VALUES (
        <!-- MySQL-Befehl zum Einfügen der folgenden Werte in die angegebenen Spalten -->
        <script language="Javascript">
            <![CDATA[document.write(countID);]]>
        </script>,
        <!-- JavaScript setzt die generierte "countID" als Argument für Spalte "ID" ein -->
        "<xsl:value-of select="@name"/>",
        <!-- den Attributswert des Attributs "name" in die Spalte "name" einsetzen -->
        "<xsl:value-of select="@type"/>",
        "<xsl:value-of select="@modified"/>",
        "<xsl:value-of select="@published"/>",
        "<xsl:apply-templates select="desc"/>",
        <!-- Elementinhalt vom Element <entry><desc> in Spalte "description" einsetzen -->
        "<xsl:value-of select="@seq"/>",
        "<xsl:value-of select="@severity"/>",
        "<xsl:value-of select="@CVSS_vector"/>",
        "<xsl:value-of select="@CVSS_base_score"/>",
        "<xsl:value-of select="@CVSS_exploit_subscore"/>",
        "<xsl:value-of select="@CVSS_impact_subscore"/>",
        "<xsl:value-of select="@CVSS_version"/>",
        "<xsl:value-of select="@CVSS_score"/>");
        <br/>   <!-- Zeilenumbruch nach Abschluss des ersten MySQL-Befehls einfügen -->

        <!-- Beginn des Abschnitts für die Tabelle "refs" -->
```

```
49          <xsl:for-each select="refs/ref">    <!-- Auswahl für das Element <refs><ref> -->
50              INSERT INTO refs (pkey_id, ref_source, patch, adv, url) VALUES (
51              <script language="Javascript">
52                  <![CDATA[document.write(countID);]]>
53              </script>,
54              "<xsl:value-of select="@source"/>",
55              "<xsl:value-of select="@patch"/>",
56              "<xsl:value-of select="@adv"/>",
57              "<xsl:value-of select="@url"/>");
58          </xsl:for-each>
59          <br/>
60
61          <!-- Beginn des Abschnitts für die Tabelle "products" -->
62          <xsl:for-each select="vuln_soft/prod/vers">
63          <!-- Cursor wird auf das Element <vuln_soft><prod><vers> gesetzt -->
64              INSERT INTO products (pkey_id, vendor, name, version) VALUES (
65              <script language="Javascript">
66                  <![CDATA[document.write(countID);]]>
67              </script>,
68              "<xsl:value-of select="../@vendor"/>",
69              "<xsl:value-of select="../@name"/>",
70              <!-- Elemente "vendor" und "name" aus der nächsthöheren Ebene auswählen -->
71              "<xsl:value-of select="@num"/>");
72              <!-- Version aus <vuln_soft><prod><vers> in die Spalte "version" einfügen -->
73              </xsl:for-each>
74          <br/>
75          <br/>
76
77          <script language="Javascript">
78              <![CDATA[zaehlen();]]>
79          </script>
80          <!-- nach jedem Durchlauf die in Zeile 8 beschriebene Funktion "zaehlen" ausführen,
                um "countID" jeweils um 1 zu erhöhen -->
81      </xsl:for-each>
82  </xsl:template>
83
84  <xsl:template match="desc">
85      <xsl:variable name="varQuote" select="'&quot;'"/>
86      <!-- Suche nach Sonderzeichen, die den Datenimport stören können -->
87      <xsl:variable name="varSpecialChar" select="concat($varQuote,&quot;'&quot;)"/>
88      <xsl:value-of select="translate(., $varSpecialChar, '')"/>
89      <!-- Sonderzeichen werden ersetzt -->
90  </xsl:template>
91  </xsl:stylesheet>
```

A.8.3 Erstellen der drei MySQL-Tabellen

Listing A.3: Erstellen der drei MySQL-Tabellen (`tabellen-anlegen.txt`)

```
CREATE TABLE data(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(30),
    type VARCHAR(30),
    modified VARCHAR(30),
    published VARCHAR(30),
    description VARCHAR(4000),
    seq VARCHAR(30),
    severity VARCHAR(30),
    cvss_vector VARCHAR(30),
    cvss_base_score FLOAT,
    cvss_exploit_subscore FLOAT,
    cvss_impact_subscore FLOAT,
    cvss_version VARCHAR(30),
    cvss_score FLOAT
);


CREATE TABLE refs(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    pkey_id INTEGER,
    ref_source VARCHAR(100),
    patch VARCHAR(1),
    adv VARCHAR(1),
    url VARCHAR(1000)
);


CREATE TABLE products(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    pkey_id INTEGER,
    vendor VARCHAR(100),
    name VARCHAR(100),
    version VARCHAR(100)
);
```

A.8.4 HTML-Dokument zum Aufruf der Transformation

Listing A.4: HTML-Dokument zum Aufruf der Transformation (`xmltransformation09.html`)

```
<html>
    <body>
        <script type="text/javascript">
            var xml_path = "nvdcve-2009.xml";
            var xslt_path = "xsl-style.xsl";

            if (window.ActiveXObject) {
                <!-- XML-Code für den Webbrowser Mircosoft Internet Explorer -->
                xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
                xmlDoc.async=false;
                xmlDoc.load(xml_path);                                  <!-- XML laden -->

                xsl = new ActiveXObject("Microsoft.XMLDOM");
                xsl.async = false;
                xsl.load(xslt_path);                                    <!-- XSL laden -->

                document.write(xmlDoc.transformNode(xsl))               <!-- Transformieren -->
            }

                <!-- XML-Code für andere Webbrowser wie Firefox, Opera, etc. -->
            else if (document.implementation && document.implementation.createDocument) {

                var src_doc = document.implementation.createDocument("","", null);
                src_doc.async = false;
                src_doc.load(xml_path);                                 <!-- XML laden -->

                var processor = new XSLTProcessor();
                var xslt = document.implementation.createDocument("", "", null);
                xslt.async = false;
                xslt.load(xslt_path);                                   <!-- XSL laden -->
                processor.importStylesheet(xslt);

                var result = processor.transformToDocument(src_doc); <!-- Transformieren -->
                var xmls = new XMLSerializer();
                var output = xmls.serializeToString(result);
                document.write(output);
            }

            else {
                alert('Der verwendete Brwoser kann dieses JavaScript nicht ausführen');
            }
        </script>
    </body>
</html>
```

A.8.5 Vollständiger Datenimport in MySQL

Listing A.5: Vollständiger Datenimport in MySQL (`nvd-import.txt`)

```
1  CREATE DATABASE nvd02;
2  USE nvd02;
3  SOURCE tabellen-anlegen.txt;
4  SOURCE insert02.txt;
5
6  CREATE DATABASE nvd03;
7  USE nvd03;
8  SOURCE tabellen-anlegen.txt;
9  SOURCE insert03.txt;
10
11 CREATE DATABASE nvd04;
12 USE nvd04;
13 SOURCE tabellen-anlegen.txt;
14 SOURCE insert04.txt;
15
16 CREATE DATABASE nvd05;
17 USE nvd05;
18 SOURCE tabellen-anlegen.txt;
19 SOURCE insert05.txt;
20
21 CREATE DATABASE nvd06;
22 USE nvd06;
23 SOURCE tabellen-anlegen.txt;
24 SOURCE insert06.txt;
25
26 CREATE DATABASE nvd07;
27 USE nvd07;
28 SOURCE tabellen-anlegen.txt;
29 SOURCE insert07.txt;
30
31 CREATE DATABASE nvd08;
32 USE nvd08;
33 SOURCE tabellen-anlegen.txt;
34 SOURCE insert08.txt;
35
36 CREATE DATABASE nvd09;
37 USE nvd09;
38 SOURCE tabellen-anlegen.txt;
39 SOURCE insert09.txt;
```

A.8.6 Verändern der Identifikationsnummern in den Tabellen

Listing A.6: Verändern der Identifikationsnummern in den Tabellen (`nvd-neue-id.txt`)

```
1  USE nvd02;
2  UPDATE data SET id = (200200000 + id);
3  UPDATE refs SET id = (200200000 + id);
4  UPDATE refs SET pkey_id = (200200000 + pkey_id);
5  UPDATE products SET id = (200200000 + id);
6  UPDATE products SET pkey_id = (200200000 + pkey_id);
7  USE nvd03;
8  UPDATE data SET id = (200300000 + id);
9  UPDATE refs SET id = (200300000 + id);
10 UPDATE refs SET pkey_id = (200300000 + pkey_id);
11 UPDATE products SET id = (200300000 + id);
12 UPDATE products SET pkey_id = (200300000 + pkey_id);
13 USE nvd04;
14 UPDATE data SET id = (200400000 + id);
15 UPDATE refs SET id = (200400000 + id);
16 UPDATE refs SET pkey_id = (200400000 + pkey_id);
17 UPDATE products SET id = (200400000 + id);
18 UPDATE products SET pkey_id = (200400000 + pkey_id);
19 USE nvd05;
20 UPDATE data SET id = (200500000 + id);
21 UPDATE refs SET id = (200500000 + id);
22 UPDATE refs SET pkey_id = (200500000 + pkey_id);
23 UPDATE products SET id = (200500000 + id);
24 UPDATE products SET pkey_id = (200500000 + pkey_id);
25 USE nvd06;
26 UPDATE data SET id = (200600000 + id);
27 UPDATE refs SET id = (200600000 + id);
28 UPDATE refs SET pkey_id = (200600000 + pkey_id);
29 UPDATE products SET id = (200600000 + id);
30 UPDATE products SET pkey_id = (200600000 + pkey_id);
31 USE nvd07;
32 UPDATE data SET id = (200700000 + id);
33 UPDATE refs SET id = (200700000 + id);
34 UPDATE refs SET pkey_id = (200700000 + pkey_id);
35 UPDATE products SET id = (200700000 + id);
36 UPDATE products SET pkey_id = (200700000 + pkey_id);
37 USE nvd08;
38 UPDATE data SET id = (200800000 + id);
39 UPDATE refs SET id = (200800000 + id);
40 UPDATE refs SET pkey_id = (200800000 + pkey_id);
41 UPDATE products SET id = (200800000 + id);
42 UPDATE products SET pkey_id = (200800000 + pkey_id);
43 USE nvd09;
44 UPDATE data SET id = (200900000 + id);
45 UPDATE refs SET id = (200900000 + id);
46 UPDATE refs SET pkey_id = (200900000 + pkey_id);
47 UPDATE products SET id = (200900000 + id);
48 UPDATE products SET pkey_id = (200900000 + pkey_id);
```

A.8.7 Vereinen aller Datensätze in einer gemeinsamen Datenbank

Listing A.7: Vereinen aller Datensätze in einer gemeinsamen Datenbank (`nvd-vereinen.txt`)

```
1  CREATE DATABASE nvd;
2  USE nvd;
3  SOURCE tabellen-anlegen.txt;
4  INSERT data SELECT * from nvd02.data;
5  INSERT data SELECT * from nvd03.data;
6  INSERT data SELECT * from nvd04.data;
7  INSERT data SELECT * from nvd05.data;
8  INSERT data SELECT * from nvd06.data;
9  INSERT data SELECT * from nvd07.data;
10 INSERT data SELECT * from nvd08.data;
11 INSERT data SELECT * from nvd09.data;
12 INSERT refs SELECT * from nvd02.refs;
13 INSERT refs SELECT * from nvd03.refs;
14 INSERT refs SELECT * from nvd04.refs;
15 INSERT refs SELECT * from nvd05.refs;
16 INSERT refs SELECT * from nvd06.refs;
17 INSERT refs SELECT * from nvd07.refs;
18 INSERT refs SELECT * from nvd08.refs;
19 INSERT refs SELECT * from nvd09.refs;
20 INSERT products SELECT * from nvd02.products;
21 INSERT products SELECT * from nvd03.products;
22 INSERT products SELECT * from nvd04.products;
23 INSERT products SELECT * from nvd05.products;
24 INSERT products SELECT * from nvd06.products;
25 INSERT products SELECT * from nvd07.products;
26 INSERT products SELECT * from nvd08.products;
27 INSERT products SELECT * from nvd09.products;
```