Universität Hamburg
Fachbereich Informatik

Baccalaureatsarbeit

Analyse der aVTC-Tests und Verbesserung der Testauswertung

Michel Messerschmidt

12. April 2002

Betreuer: Prof. Dr. Klaus Brunnstein

Inhaltsverzeichnis

1 Einleitung **5**

2 Überblick über Ziele und Durchführung der aVTC-Tests **7**

2.1 Grundlegende Begriffe 7

2.2 Anforderungen und Kriterien zum Testen von Anti-Malware-Produkten . . 8

2.3 Ziele und Rahmenbedingungen der aVTC-Tests 9

2.4 Überblick über die verschiedenen aVTC-Tests 11

2.5 Aufbau der Malware-Datenbanken 12

2.6 Zeitliche Gliederung eines Tests 13

3 Analyse der bisherigen Testdurchführung **15**

3.1 Vorbereitungs-Phase 15

3.2 Scan-Phase 16

3.3 Auswertungs-Phase 17

3.3.1 Überblick 17

3.3.2 Struktur der Auswertungsumgebung 19

3.3.3 Einrichtung der Auswertungsumgebung 19

3.3.4 Durchführung der Auswertung 20

3.3.5 Einzelheiten der verwendeten Skripte 21

3.3.6 Qualitätssicherung 23

3.4 Berichts-Phase 25

4 Mögliche Verbesserungen **27**

4.1 Vorbereitungs-Phase 27

4.2 Scan-Phase 28

4.3 Auswertungs-Phase 28

4.4 Berichts-Phase 29

5 Beschreibung der durchgeführten Verbesserungen **31**

5.1 Anforderungen 31

5.2 Implementation 32

5.2.1 Statusinformationen 32

5.2.2 Neue Struktur 34

5.2.3 Ablauf der Auswertung 35

5.2.4 Qualitätssicherung 36

5.3 Bedienung 36

6 Ausblick **39**

Literaturverzeichnis **41**

1 Einleitung

Das Anti-Virus Test Center (aVTC) ist eine Einrichtung des Arbeitsbereichs ”Anwendungen der Informatik in Geistes- und Naturwissenschaften” (AGN) im Fachbereich Informatik der Universität Hamburg. Als eine von ca. vier Einrichtungen weltweit führt das aVTC regelmäßig Tests von Anti-Virus- bzw. Anti-Malware-Produkten durch. Es ist dabei die einzige Einrichtung, deren Tests vollständig unentgeltlich durchgeführt werden (sowohl für Hersteller wie für Anwender der Produkte).

Diese Arbeit beschreibt nach einer kurzen Einführung (Kapitel 2) die Vorgehensweise bei solchen Tests (Kapitel 3), zeigt mögliche Optimierungen auf (Kapitel 4) und beschreibt die im Rahmen dieser Arbeit durchgeführten Verbesserungen (Kapitel 5) an der aVTC Testauswertung.

Auch wenn die Analyse und die Verbesserungsvorschläge sich auf die gesamten Tests beziehen, liegt der Schwerpunkt dieser Arbeit aus Zeitgründen auf der Auswertung (und Qualitätssicherung) der Testergebnisse.

2 Überblick über Ziele und Durchführung der aVTC-Tests

2.1 Grundlegende Begriffe

In diesem Abschnitt werden einige notwendige Begriffe kurz erklärt. Es werden nur die für diese Arbeit wesentlichen Merkmale dargestellt, da eine umfassende Definition bzw. Diskussion der verschiedenen Ansätze viel zu umfangreich wäre.

Malware ist die Kurzform von "malicious software" und somit ein Oberbegriff für jegliche Art Software, die man als "bösartig" oder "schädlich" einschätzen würde. Die Entscheidung, ob Software Malware ist oder nicht, lässt sich leider nicht eindeutig treffen; manchmal ist dies sogar vom Anwendungskontext abhängig. Malware wird in etlichen Arten unterteilt (z.B. Viren, Trojaner, Hoaxes) und nach mehreren Kriterien klassifiziert (z.B. ob die Malware auf ein vernetztes System angewiesen ist). Eine ausführlichere Einführung in dieses Thema ist z.B. in [aVTC 2000] zu finden.

Virus Als (Computer-)Virus wird jede Software bezeichnet, die die Fähigkeit zur Selbst-Replikation besitzt. Es existieren verschiedene Definitionen für Viren – siehe z.B. [Bontchev 1998, p. 23ff] – unter anderem auch eine formale Definition [Cohen 1984]. Daher sind Viren die einzige Malware-Art, für die eindeutig entschieden werden kann, ob eine Software in diese Kategorie fällt oder nicht. Oft wird Malware deshalb in die Kategorien "viral" (Viren und Würmer) und "nicht-viral" (alles andere) unterteilt[1]. Üblicherweise wird dabei zwischen Viren und Würmern unterschieden; dieser Unterschied ist für die vorliegende Arbeit aber nicht von Bedeutung und wird hier deshalb nicht weiter beachtet.

Anti-Virus-Produkt Dies ist eine Software, die die Fähigkeit besitzt, virale Software zu erkennen. Außerdem sollte die Aktivierung und Verbreitung von Viren verhindert und bereits infizierte Software wieder desinfiziert werden können.

Anti-Malware-Produkt Diese Produkte haben im Gegensatz zu Anti-Virus-Produkten die Aufgabe nicht nur Viren, sondern jegliche Art von Malware erkennen zu können. Dies ist zwar unmöglich, da ein eindeutiges Unterscheidungsmerkmal fehlt, bekannte Malware sowie deren zugrundeliegenden Mechanismen können aber durch-

[1] Dies ist auch bei den aVTC-Tests der Fall

aus erkannt werden. Während es noch vor einigen Jahren ausschließlich Anti-Viren-Produkte gab, wurden im Laufe der Zeit zunehmend Funktionen zur Erkennung nicht-viraler Malware ergänzt, so daß sich viele Produkte zu Anti-Malware-Produkten weiterentwickelt haben[2]. Da alle Produkte in den aVTC-Tests sowohl als Anti-Virus- wie als Anti-Malware-Produkt getestet und bewertet werden, wird im folgenden nur noch der Begriff Anti-Malware-Produkt verwendet.

Malware-Datenbank Eine Viren- oder Malware-Datenbank ist ein Sammlung von Malware, die zum Testen der Anti-Malware-Produkte verwendet wird. Um aussagekräftige Ergebnisse zu erhalten, müssen die Datenbanken möglichst umfangreich sein – im Idealfall wäre die gesamte bekannte Malware enthalten[3]. Die Forderung nach möglichst umfangreichen Datenbanken ist durch die verwendeten Erkennungstechniken der Anti-Malware-Produkte begründet[4]. Da die Erkennung der Produkte manchmal nur auf bestimmte Dateien einer Malware ausgerichtet ist, sollte jeweils mehrere Exemplare (Samples) vorhanden sein, um solche Schwächen zu erkennen.

In-The-Wild (ITW) Da nicht alle bekannten Viren weit verbreitet sind und einige sogar nur bei Virenautoren und in Anti-Viren-Labors existieren, besteht die Notwendigkeit zu unterscheiden, welche Viren zur Zeit wirklich eine reale Bedrohung darstellen bzw. weit verbreitet sind und welche nicht. Deshalb werden von [Wildlist] jeden Monat Listen der ”in freier Wildbahn” vorkommenden Viren erstellt. Die Erkennung dieser Viren wird als besonders wichtig eingestuft. Allerdings muß beachtet werden, daß viele ITW-Viren vor ihrer weiten Verbreitung bereits bekannt und meist schon in den Datenbanken von Anti-Viren-Labors vorhanden waren. Daher ist es wichtig, Anti-Malware-Produkte nicht nur mit ITW-Datenbanken zu testen.

2.2 Anforderungen und Kriterien zum Testen von Anti-Malware-Produkten

Beim Testen von (Software-)Produkten gibt es viele Kriterien, nach denen bewertet werden kann: Funktionalität, Softwarequalität (innere Qualität), Ergonomie, Dokumentation, Wartbarkeit, Administrationsaufwand usw.. Allerdings sind einige dieser Kriterien nur sehr schwierig objektiv zu beurteilen (z.B. der Support des Herstellers). Andere Kriterien sind nicht für alle Anwendungsgebiete des Produkts sinnvoll (z.B. benötigen private Benutzer normalerweise keine Remote-Administration, während diese für größere Firmen unverzichtbar ist). Wieder andere Kriterien lassen sich nur mit großem Aufwand testen (z.B. die Wartbarkeit von Produkten, die für den Einsatz in großen Netzwerken mit mehreren

[2] Anhand der Testergebnisse ist leicht erkennbar, daß diese Entwicklung noch längst nicht abgeschlossen ist

[3] Dies ist für nicht-virale Malware natürlich nicht möglich

[4] Meistens werden heutzutage signatur-basierte Verfahren verwendet – also letztlich Pattern-Matching. Oft sind aber zusätzlich heuristische Verfahren im Einsatz. Die Grundidee dieser Verfahren wird z.B. in [Schmall 2002] beschrieben.

tausend Arbeitsplätzen gedacht sind). Die Ausstattung der Test-Einrichtung ist somit ein begrenzender Faktor für die zu testenden Kriterien.

Beim Testen von Anti-Malware-Produkten gibt es aber noch zusätzliche Anforderungen, die den Aufwand erhöhen. Zum Einen befinden sich Anti-Malware-Produkte in einem stetigen, raschen Entwicklungszyklus, um mit der schnellen Entwicklung der Malware Schritt halten zu können. Deshalb sind solche Produkte und damit auch die Testergebnisse wesentlich schneller veraltet als in anderen Bereichen. Dies hat zur Folge, dass Tests relativ häufig durchgeführt werden müssen und die Dauer der Tests möglichst kurz sein muß, um aussagekräftige Ergebnisse zu erhalten. Zum Anderen ist der Aufbau und die Pflege der zum Testen benötigten Virendatenbanken mit zusätzlichem Aufwand verbunden, der den Aufwand der eigentlichen Testdurchführung leicht übersteigen kann.

2.3 Ziele und Rahmenbedingungen der aVTC-Tests

Bei allen im aVTC durchgeführten Tests wird als einziges Testkriterium die maximal mögliche Erkennungsrate (also eine Aussage über die Funktionalität) der Produkte ermittelt. Das hat vor allem zwei Gründe. Malware richtet bereits heute beträchtliche Schäden an. Ein Anti-Malware-Produkt, das nicht in der Lage ist, diese Schäden in ausreichendem Maße zu verhindern, ist ungeachtet aller anderen Qualitäten nutzlos (wobei hier nicht näher darauf eingegangen wird, wie sich ”ausreichender Schutz” definieren lässt).

Dies bedeutet keineswegs, daß andere Kriterien unwichtig sind. Gerade eine einfache Bedienung und Wartbarkeit ist oft ausschlaggebend für den erfolgreichen Einsatz eines Produkts. Aber in einem Qualitätstest kann Funktionalität nicht durch andere Merkmale ersetzt, sondern nur ergänzt werden. Dies bedeutet allerdings auch, daß Funktionalität alleine noch keine allgemeine Aussage über die Qualität eines Produktes zulässt. Es ist aber eine notwendige Bedingung für gute Qualität. Insofern ist der Funktionalitätstest nur ein Bestandteil einer vollständigen Produktevaluation (wenn auch der wichtigste).

Eine vollständige Produktevaluation wird allerdings auch nicht als Ziel der aVTC-Tests angesehen. Als Einrichtung einer Universität bestehen die Ziele des aVTC vor allem in der Ausbildung von Studenten, der Entwicklung wissenschaftlicher Testmethoden und – in Hinblick auf die Testergebnisse – der allgemeinen Einschätzung aktueller Sicherheitsprobleme und der Wirksamkeit entsprechender Schutzmethoden. Deshalb wird in den Testberichten auch besonderer Wert auf die zeitliche Entwicklung der Produkte über mehrere Tests bzw. Jahre hinweg sowie die durchschnittliche Erkennungsrate aller Produkte gelegt, die für eine reine Produktevaluation nebensächlich wären. Trotzdem muß darauf geachtet werden, möglichst aktuelle Testergebnisse zu ermitteln, da diese sonst ausschließlich historischen Wert hätten und nichts über *aktuelle* Gefährdungen aussagen würden.

Da es jedoch kaum Einrichtungen gibt, die Evaluationen von Anti-Malware-Produkten durchführen oder deren Methoden wissenschaftlichen Ansprüchen nicht genügen (wie z.B. die meisten von Zeitschriften durchgeführten Tests) und deshalb nur geringe Aussagekraft besitzen, besteht ein gewisse Nachfrage in diesem Bereich. Daher werden die aVTC-Tests oft als Produktevaluationen interpretiert oder sogar dafür kritisiert, daß sie diese Aufgabe nicht richtig erfüllen.

Theoretisch wäre es zwar denkbar, die aVTC-Tests zu vollständigen Produktevaluationen weiterzuentwickeln, da im aVTC das notwendige Fachwissen vorhanden ist und herstellerunabhängig gearbeitet werden kann. Da entsprechende Erweiterungen die Qualität der Tests aber nicht beeinträchtigen dürfen, wäre dies ein erheblicher Aufwand[5], dessen Nutzen im universitären Rahmen nicht ersichtlich ist (natürlich wäre der Nutzen für Anwender und Hersteller enorm). Allerdings könnten solche Tests dann nicht mehr in dieser Form – nämlich komplett unentgeltlich und in das Studium eingebunden – durchgeführt werden.

Um jegliche Spekulationen gleich wieder zu beenden, muß nur noch der zweite Grund genannt werden, der in der stark begrenzten (materiellen wie personellen) Ausstattung des aVTC liegt. Somit sind umfassendere Tests nicht in einem sinnvollen Zeitrahmen durchführbar. Würden mehrere Kriterien evaluiert werden, würde der Test dermaßen in die Länge gezogen werden, daß die Ergebnisse veraltet und somit nicht zu gebrauchen wären. Aber auch mit der verfügbaren Hardware und der aktuellen Personalsituation sind umfangreichere Tests einfach nicht zu bewältigen.

Das aVTC Personal besteht aus mehreren Studenten unter der Leitung von Prof. Brunnstein. Weitere Mitarbeiter sind nicht verfügbar. Da die Arbeit im aVTC nur begrenzt als Studienleistung angerechnet werden kann, liegt die zur Verfügung stehende Zeit pro Student bei ca. 1-2 Stunden pro Woche (und in der vorlesungsfreien Zeit noch darunter). Da Studenten in der Regel nur einige Semester mitarbeiten können, besteht ein ständiger Wechsel der Mitarbeiter. Abgesehen von der Notwendigkeit der Einarbeitung, die einen gewissen Zeitaufwand aller anderen Mitarbeiter erfordert, aber als Teil des Studiums zu den Zielen des aVTC gehört, besteht immer die Gefahr, daß spezielle Kenntnisse nicht weitergegeben werden und somit verlorengehen.

Die zur Verfügung stehende Hardware ist größtenteils veraltet (der durchschnittliche Rechner ist ca. ein Pentium 200 MHz, 64 MB RAM), so daß nur wenige Rechner für aktuelle Betriebssysteme (wie Windows 2000) geeignet sind. Abgesehen von den Tests unter MSDos stehen ohnehin zu wenige Rechner zur Verfügung, um alle Produkte parallel testen zu können. Zusätzlich ist die Hardware nicht einheitlich, sondern es gibt höchst unterschiedliche Systeme (IDE und SCSI Laufwerke, verschiedene Netzwerkarten, ...). Somit entsteht ein zusätzlicher Aufwand durch häufiges Neuinstallieren von Betriebssystemen und Produkten, ohne einheitliche Methoden nutzen zu können. Zur Zeit sind die Clients auf 3 verschiedene Subnetze aufgeteilt, die jeweils mit einer Netzwerkkarte des Servers verbunden sind, da die Netzwerk-Bandbreite sonst nicht für alle Clients ausreichen würde.

Angesichts der allgemein nicht so positiven Finanzlage der deutschen Hochschulen sind von Seiten der Universität keine Mittel zu erwarten, die diese Lage verbessern könnten.

[5] Insbesondere, da es keine typisches Einsatzszenario dieser Produkte gibt, sondern vom Großunternehmen bis zum Privatanwender stark unterschiedliche Einsatzbedingungen für dasselbe Produkt existieren

2.4 Überblick über die verschiedenen aVTC-Tests

Im aVTC werden regelmäßig verschiedene Testarten durchgeführt, wobei versucht wird, einen jährlichen Zyklus einzuhalten. Im "großen" Test werden Produkte unter allen Betriebssystemen (zur Zeit sind das MSDos, Linux, Windows 98, Windows NT 4.0 und Windows 2000) mit dem aktuellen Stand aller Datenbanken getestet (Boot-, File-, Makro- und Skriptviren sowie gepackte, polymorphe und VirusKit-Datenbanken). Im "kleinen" Test wird nur mit Makro- und Skriptviren getestet, um den Aufwand zu verringern.

Zusätzlich gibt es den "Heureka"-Test. Dabei werden die Fähigkeiten der Produkte getestet, neue Viren und Virus-Varianten zu erkennen, die es zum Erscheinungszeitpunkt der Produkte noch nicht gab. Dazu werden 3 und 6 Monate nach dem jeweils letzten "großen" oder "kleinen" Test Datenbanken erzeugt, die nur die in diesem Zeitraum neu entdeckten Viren und Virus-Varianten enthalten. Der Heureka-Test wird nur unter einem Betriebssystem (zur Zeit Windows NT 4.0) durchgeführt und beinhaltet nur Datenbanken mit Makro- und Skriptviren.

Im Rahmen einer Diplomarbeit wurde außerdem der "Antivirus Repair Test" (ART) entwickelt, der die Fähigkeiten der Produkte zum Entfernen von (Makro-)Viren aus infizierten Office-Dokumenten bzw. dem Wiederherstellen der Dokumente testet.

Bisher werden ausschließlich "On-Demand" Produkte getestet, also Produkte, die manuell gestartet werden müssen. "On-Access" Produkte (Produkte, die beim Betriebssystemstart mitgestartet werden und im Hintergrund laufen) und spezielle Lösungen (wie z.B. Virenfilter auf Mailservern) werden bisher nicht im aVTC getestet. Ein Test für "On-Access" Produkte ist aber in der Entwicklung und hoffentlich bald einsatzbereit, da die "On-Access" Funktionen der Anti-Malware-Produkte heutzutage einen wesentlichen Anteil an den eingesetzten Schutzmethoden haben und daher nicht ignoriert werden dürfen. Es muß an dieser Stelle jedoch erwähnt werden, daß "On-Access" Tests, die mit dem gleichen qualitativen Niveau wie "On-Demand" Tests durchgeführt werden sollen, wesentlich aufwändiger in der Durchführung sind.

Bei allen Tests werden grundsätzlich nur englische Softwareversionen verwendet (sowohl der Betriebssysteme wie der zu testenden Produkte), da lokalisierte Versionen in der Regel nicht so aktuell sind. Außerdem ist die Aussagekraft der Testergebnisse größer, wenn international weit verbreitete Versionen verwendet werden. Liegt ein Produkt allerdings nur in einer nicht englischen Version vor, wird dieses ebenfalls getestet, sofern dies vom Aufwand her vertretbar erscheint. Dabei kommen deutsche Produktversionen am häufigsten vor, es wurden aber auch schon spanische und polnische Produktversionen getestet (dies erforderte einen erheblichen Mehraufwand bei der Bedienung dieser Produkte und erschwerte eine sorgfältige Konfiguration).

Sämtliche Tests sind so organisiert, daß die Virendatenbanken auf einem Server (zur Zeit Windows NT 4.0) als Netzwerkfreigaben bereitgestellt werden und auf den Clients als (Netz-)Laufwerke eingebunden werden. Somit lassen sich die Produkte einzeln auf den Clients installieren und auf die jeweiligen Laufwerke ansetzen. Dies ist notwendig, da nicht alle Produkte Netzwerkfreigaben direkt scannen können. Dadurch entsteht ein hohe Netzlast, die durch die Auswertung noch weiter erhöht wird, da die Auswertung zwar auf einem der Clients durchgeführt wird, aber alle Daten und Programme auf dem Server liegen.

2.5 Aufbau der Malware-Datenbanken

Jede Malware-Datenbank (im Testbericht ”testbed” genannt) ist keine Datenbank im eigentlichen Sinn, sondern vielmehr eine Verzeichnishierarchie[6].

Die obersten Verzeichnisebenen unterteilen die Malware nach Typ, also z.B. ”`VBS`” für VisualBasic-Skript-Viren, ”`X97M`” für Excel97-Makro-Viren oder auch ”`trojan`” für trojanische Pferde. Unterhalb dieser Verzeichnisse existiert für jede Virusfamilie ein eigenes Verzeichnis, z.B. ”`sircam`” oder ”`loveletter`”. Eine Verzeichnisebene tiefer wird dann noch nach Virus-Varianten unterschieden, so daß alle Samples von z.B. der ”Melissa”-Variante ”A” im Verzeichnis ”`W97M\Melissa\A`” zu finden sind, während die Variante ”B” im Verzeichnis ”`W97M\Melissa\B`” abgelegt ist.

Die kleinste Einheit, nach der in den aVTC-Tests differenziert wird, ist also die Virus-Variante. Im Folgenden wird dabei nicht mehr zwischen Viren und Virus-Varianten unterschieden, da die Auswertung diese Unterscheidung ebenfalls nicht trifft. Der Begriff ”Virus” wird also auch für die einzelnen Virus-Varianten verwendet.

Die verschiedenen Datenbanken im Test unterscheiden sich nur durch ihren Typ. Dabei wird nach mehreren Kriterien unterschieden. Zuerst werden alle Viren entsprechend ihrer Klassifikation in Boot-, File-, Makro- und Skript-Viren in Boot-, File-, Makro- und Skript-Datenbanken eingeteilt. Polymorphe Viren und Virenkits (also Viren, die mit einem Virus-Generator-Programm erzeugt wurden) werden aufgrund der großen Anzahl der Samples und etwas anderer Testbedingungen von dieser Unterteilung ausgenommen und in eigenen Poly- bzw. Viruskit-Datenbanken getestet.

Von jeder dieser Datenbankentypen gibt es dann noch Zoo-, ITW- und Malware-Datenbanken. Die Zoo-Datenbanken (also z.B. ”Makro-Zoo”) enthalten jeweils alle Viren dieser Art, während die ITW-Datenbanken nur die Teilmenge der Viren enthalten, die in der aktuellen ITW-Liste aufgeführt sind. Die Malware-Datenbanken enthalten alle Malware dieser Art (also z.B. Skript-Malware), die nicht selbst-replizierend ist (also z.B. Trojaner, Dropper, Backdoors, Intendeds). Die Bezeichnung ”Malware-Datenbank” ist leider irreführend und sollte besser durch ”Non-replicating”, ”Non-viral” o.ä. ersetzt werden, da ”Malware” als Obergriff auch die Viren in allen anderen Datenbanken umfasst. Diese Änderung hat sich aber noch nicht gegen die historischen Bezeichnungen durchsetzen können, obwohl der Begriff ”Malware” ansonsten mit der üblichen Bedeutung verwendet wird – auch im Testbericht.

Zuletzt werden die Datenbanken für die Auswertung noch unterschieden nach:

[6] Jede Datenbank im Test erhält dabei einen eigenen Laufwerksbuchstaben

- ”normal” – wie eben beschrieben
- ”packed” – alle Samples sind in Archiven komprimiert (z.B. als `ZIP`- oder `ARJ`-Archiv)
- ”false positives” – die Datenbank enthält zusätzlich nicht-maliziöse Dateien (sogenannte ”false positives”), um zu testen, wie hoch die Rate falscher Alarme der Produkte ist

2.6 Zeitliche Gliederung eines Tests

Jeder Test besteht aus mehreren Phasen: In der Vorbereitungs-Phase wird der Test angekündigt, zu testende Produkte werden empfangen und die Virendatenbanken werden erstellt. Die Testrechner werden vorbereitet, indem die benötigten Betriebssysteme installiert und Festplatten-Images erstellt werden (falls noch nicht vorhanden).

In der Scan-Phase werden alle Produkte installiert und auf die Virendatenbanken angesetzt.

Parallel dazu beginnt die Auswertungs-Phase, in der die von den Produkten erzeugten Logdateien ausgewertet werden und die so ermittelten Ergebnisse überprüft werden (Qualitätssicherung). Da sich hierbei die Notwendigkeit weiterer Scanvorgänge ergeben kann (etwa weil ein Produkt nur einen Teil einer Virendatenbank gescannt hat), sind diese beiden Phasen eng miteinander gekoppelt.

Liegen alle Ergebnisse vor, beginnt die Berichts-Phase, in der der Testbericht erstellt, Korrektur gelesen und schließlich veröffentlicht wird.

Die Aufgabenverteilung zwischen den aVTC-Mitarbeitern orientiert sich ebenfalls an diesen Phasen (und ist im jeweiligen Testbericht dokumentiert).

3 Analyse der bisherigen Testdurchführung

3.1 Vorbereitungs-Phase

Zuerst wird Kontakt mit den Herstellern aufgenommen, um den Test anzukündigen und die zu testenden Produkte zu empfangen. Dies wird hier nicht näher ausgeführt, ist aber in [Marx 2000] ausführlicher beschrieben. Danach werden die Datenbanken erstellt. Dazu werden zu einem festgelegten Zeitpunkt Kopien der bestehenden Datenbanken angelegt, sowie alle bis dahin neu erhaltenen Samples (soweit sie wirklich viral sind) einsortiert und dem Datenbank-Schema entsprechend umbenannt.

Dies wird (immer noch) per Hand durchgeführt, lediglich von einem kleinen Skript unterstützt, das einen Teil der Sortierung vornimmt. Da einige Datenbanken recht groß sind (über 150.000 Dateien), ist dies natürlich in höchstem Grad fehleranfällig. Insbesondere besteht bei so vielen manuellen Dateioperationen die Gefahr, einen Virus zu aktivieren. Ein derartiger Vorfall könnte beträchtliche Schäden bewirken (z.B. die Virendatenbanken zerstören und über das Netzwerk andere Rechner infizieren) und ist deshalb unbedingt zu vermeiden.

Deshalb müssen alle Rechner, die Zugriff auf die Virendatenbanken haben (sowohl zur Erstellung und Pflege der Datenbanken als auch zur Testdurchführung), isoliert von allen anderen Systemen betrieben werden. Der unvermeidbare Austausch mit anderen Systemen (neue Samples, Testergebnisse) sollte nur unter größtmöglicher Kontrolle durchgeführt werden.

In derselben Zeit können bereits die Testrechner vorbereitet werden. Im Idealfall muß pro Rechner nur ein Festplatten-Image wieder eingespielt werden, ansonsten muß das zu testende Betriebssystem mit allen notwendigen Patches, Updates und Service Packs sowie den notwendigen Treibern neu installiert werden. Leider kann bei der vorhandenen Hardware auch das Einspielen eines Festplatten-Images einige Zeit benötigen. Der einfachste Weg wäre, Images auf CDs zu speichern und per Bootdiskette zurückzuspielen. Leider scheitert dies an der Unfähigkeit der meisten im aVTC verfügbaren CD-Laufwerke, gebrannte CDs zu lesen. Ein anderer Ansatz, das Bereitstellen der Images auf dem Server, scheitert im Allgemeinen an mangelndem Festplattenplatz auf dem Server, ist aber für ältere Systeme mit geringerem Umfang (wie z.B. Windows NT 4.0) gerade noch machbar. Allerdings müssen dazu Bootdisketten mit Netzwerktreibern vorhanden sein, um die Serverfreigaben ansprechen zu können. Da viele verschiedene Netzwerkkarten existieren, werden natürlich ebenso viele verschiedene Bootdisketten benötigt. Das stellt zwar kein Problem dar, da die Hardwareausstattung jedoch bestenfalls mangelhaft dokumentiert ist, dauert es recht lange bis neue studentische Mitarbeiter jeden Rechner gut genug kennen, um ein Image problemlos aufzuspielen.

3.2 Scan-Phase

Das Testen eines Produkts in dieser Phase beginnt mit dem Lesen der mitgelieferten Dokumentation (sofern vorhanden) und dem vorangegangenen Emailverkehr mit dem Hersteller (da notwendige Seriennummern oder Empfehlungen zur Konfiguration meist auf diese Weise eintreffen). Zusätzlich wird in den Aufzeichnungen der letzten Tests nachgeschaut, da bei einigen Produkten zusätzliche Funktionen – wie z.B. Kommandozeilen-Scanner – bei der Installation extra aktiviert werden müssen.

Bisher hat jeder Tester solche wichtigen Hinweise (wenn überhaupt) nur für sich selber dokumentiert. Da aber nicht jedesmal dieselbe Person dasselbe Produkt testet und es (wie erwähnt) bei den studentischen Mitarbeitern einen ständigen Wechsel gibt, geht auf diese Weise viel Wissen wieder verloren, das diesen Teil des Tests deutlich beschleunigen könnte. Daher gibt es seit dem letzten Test ein zentrales Archiv, in dem Hinweise, Einstellungen und Optionen für alle Produkte abgelegt werden müssen. Somit ist die Einarbeitungszeit für ein Produkt wesentlich geringer, da auf diese Informationen zurückgegriffen werden kann (dies erfordert natürlich auch mehr Zeit, um die geleistete Arbeit zu dokumentieren, aber insgesamt wird doch einiges eingespart). Dies funktioniert allerdings auch nicht immer, da sich einige Produkte von einem Test zum nächsten so sehr verändern, das doch wieder von vorne begonnen werden muß.

Bei der Installation ist darauf zu achten, daß alle Signaturupdates eingespielt werden, damit alle Produkte auf dem gleichen Stand sind, wenn sie getestet werden. Außerdem darf nicht vergessen werden, evtl. erforderliche Lizenz-Dateien einzubinden, da solche Produkte sonst nur als Demoversion mit beschränktem Funktionsumfang installiert sind.

Schließlich muß der Scanner dem Test entsprechend konfiguriert werden. Hierzu gibt es verschiedene Meinungen. Zum einen gibt es die Meinung, daß Produkte nur mit den Standardeinstellungen getestet werden sollten, um das ”typische Einsatzszenario” zu simulieren. Solche Annahmen sind jedoch schwer zu verifizieren, insbesondere ist es fraglich, ob es überhaupt ein typisches Einsatzszenario gibt. Hiermit wird also unter Umständen nur die Fähigkeit eines Herstellers getestet, seine Produkte vernünftig vorzukonfigurieren, aber nicht die Erkennungsrate, zu der das Produkt fähig ist. Außerdem sind zum Testen einige oft vom Standard abweichende Einstellungen zwingend erforderlich, z.B. das Erstellen einer Logdatei. Dabei ist es schwierig, allgemeine Regeln aufzustellen, welche Einstellungen geändert werden dürfen und welche nicht. Die andere Meinung (nach der auch im aVTC verfahren wird) geht von dem Ziel aus, die maximale Erkennungsrate zu ermitteln. Dabei ist grundsätzlich jede Einstellung zulässig. Um einen Vergleich der Produkte zu ermöglichen, wird versucht, für jeden Scanner die optimale Einstellung zu finden. Dies wird natürlich wesentlich erleichtert, wenn ein Hersteller seinerseits die entsprechenden Einstellungen vorschlägt. Andererseits besteht natürlich die Gefahr, daß ein Hersteller speziell für solche Tests optimierte Einstellungen implementiert, so daß die Testergebnisse weit von jedem realen Einsatz entfernt sind.

Generell wird versucht, folgendes Verhalten zu konfigurieren:

- Scannen aller Dateien
- Scannen gepackter Dateien (sowohl laufzeitkomprimierte Dateien wie Dateien in Archiven)
- Heuristische Verfahren aktivieren
- Logdatei erzeugen
- alle gescannten Dateien in der Logdatei aufführen (nicht nur die als infiziert erkannten)
- Dateien mit vollständigem Pfad in der Logdatei aufführen (sonst ist eine Auswertung nicht möglich!)
- infizierte Dateien nur melden, aber nicht löschen, verschieben oder reparieren
- keine manuellen Eingaben anfordern
- keine akustischen Warnungen ausgeben

Leider bieten längst nicht alle Produkte diese Einstellungen an. Insbesondere das Protokollieren aller gescannten Dateien wird oft vermisst, ist jedoch die einzige Möglichkeit, eindeutig festzustellen, ob ein Produkt wirklich alle Dateien gescannt hat. Da die Optionen der Produkte höchst unterschiedlich benannt sind (und die Funktion etlicher Optionen unklar ist), hilft oft nur das Ausprobieren mit verschiedenen Einstellungen auf einer kleinen Virendatenbank.

Hat man dann endlich die bestmögliche Konfiguration gefunden, wird (nach dem Dokumentieren dieser Einstellungen) jede Virendatenbank einzeln gescannt und jeweils eine Logdatei erzeugt. Diese werden dann an die Auswerter weiter gegeben, die daraus die Erkennungsraten dieses Scanners ermitteln.

3.3 Auswertungs-Phase

3.3.1 Überblick

Dieses Kapitel beschreibt die bisherige Art der Testauswertung mit dem Ziel, Unzulänglichkeiten zu verdeutlichen und einen Vergleich mit der neuen Auswertung zu ermöglichen. Es stellt keine vollständige Beschreibung aller Skripte und Abläufe dar. Die Auswertung kann nur auf Windows-Systemen durchgeführt werden, da der Auswertungsablauf im Wesentlichen mittels MSDos-Batchdateien gesteuert wird. Weitere Voraussetzungen sind die Programme "`awk`" (genauer: `GNU awk`, Version 3.04), "`sort`", "`join`", "`wc`" (Bestandteile der `GNU textutils` Version 1.11), "`perl`" (Version 5.001 oder neuer) sowie das Komprimierungsprogramm "`arj`". Zuerst wird die Auswertungsumgebung[1] eingerichtet und die Schnittstellen zu der Scan- und der Berichts-Phase werden festgelegt. Dies erfolgt in Form von Verzeichnissen auf dem Server, in denen einerseits die Logdateien der Produkte von den Testern abgelegt werden müssen (im Folgenden 'Log-Verzeichnis' genannt) und andererseits die ermittelten Ergebnisse der Produkte bereitgestellt werden ('Result-Verzeichnis').

Neue Logdateien werden per Hand aus dem Log-Verzeichnis in die Auswertungsumgebung kopiert und danach archiviert. Dabei werden die Logdateien innerhalb der Auswertungsumgebung zuerst nach Betriebssystemen[2] und dann nach Datenbanken unterteilt. Daraufhin wird für jedes Betriebssystem und jede Datenbank das entsprechende Auswertungsskript gestartet, das die Ergebnisse erzeugt. Ein Auswertungsvorgang betrifft also immer alle Logdateien, die von allen bereits getesteten Produkten unter einem Betriebssystem von einer Datenbank erstellt wurden.

Schließlich müssen die Ergebnisse noch überprüft werden. Dazu werden die Dateien, die in den einzelnen Schritten der Auswertung erzeugt wurden, auf Übereinstimmung mit der jeweiligen Logdatei überprüft. Dies dient neben der Korrektheitsprüfung der Ergebnisse auch dazu zu überprüfen, ob wirklich alle Dateien in der Datenbank gescannt wurden. Ist letzteres nicht sicherzustellen werden bis zu zwei erneute Scans angesetzt (sogenannte Nach- oder Postscans) und dem entsprechenden Tester wird mitgeteilt, welche Dateien erneut gescannt werden müssen.

Auch die Logdatei selber wird auf Verfahrensfehler überprüft:

- Wurden beim Scannen Optionen vergessen
- Stimmt das Datum der Virensignaturen
- Wurde die richtige Produktversion und die richtige Lizenz verwendet

Erst wenn es keine Unstimmigkeiten mehr gibt, werden die Ergebnisse dieses Produkts akzeptiert.

Alle Produkte werden während der gesamten Auswertung durch eine eindeutige Abkürzung aus drei Zeichen identifiziert[3]. Diese Abkürzungen werden auch in den anderen Test-Phasen sowie im Testbericht verwendet, obwohl das nicht unbedingt notwendig wäre.

[1] Unter "Auswertungsumgebung" wird im Folgenden die Gesamtheit aller Dateien und Verzeichnisse verstanden, die für die Testauswertung verwendet werden

[2] Unter denen das Produkt getestet wurde

[3] Wenn notwendig, wird im Folgenden stellvertretend "FOO" verwendet

3.3.2 Struktur der Auswertungsumgebung

Pro Betriebssystem und Datenbank existiert ein Log-Verzeichnis (z.B. `P:\VTC20017\W98\MACROITW`) und ein entsprechendes Verzeichnis in der Auswertungsumgebung (im Folgenden "Auswertungs-Verzeichnis" genannt, z.B. `P:\auswert\W98\MACROITW`), in dem die Auswertung der entsprechenden Logdateien stattfindet. Da diese Namen willkürlich gewählt werden, sind sie nicht eindeutig[4] und können daher nicht von einem Skript automatisch verarbeitet werden.

Zum Archivieren werden Kopien der Logdateien in ein ARJ-Archiv verschoben, das sich im jeweiligen Log-Verzeichnis befindet.

Die Auswertungsumgebung besteht außer den Auswertungs-Verzeichnissen noch aus Verzeichnissen, in denen Auswertungsskripte zu finden sind (diese liegen in `P:\variant\batches`, `P:\variant\erzeugen`, `N:\scripts\zwi` sowie etlichen Unterverzeichnissen vor), Verzeichnissen in denen Informationen über die Virendatenbanken abgelegt sind (`P:\dirlist`, `P:\variant`) sowie einem Verzeichnis (`N:\scripts\zwi\muster`), in dem sich auf die einzelnen Produkte angepasste Skripte befinden. Letztere werden im Folgenden auch "Scanner-Pattern" genannt. Im Auswertungs-Verzeichnis befinden sich die auszuwertenden Logdateien sowie eventuell Informationen über den Status eines Produkts. Aus diesem Verzeichnis heraus wird auch die Auswertung gestartet. Alle Ergebnisse und Zwischenschritte befinden sich in einem Unterverzeichnis "NORMAL" des Auswertungs-Verzeichnisses.

3.3.3 Einrichtung der Auswertungsumgebung

Alle benötigten Verzeichnisse müssen per Hand erstellt werden, bevor die Auswertung gestartet wird. Außerdem muß für jedes Auswertungs-Verzeichnis eine Batchdatei erstellt werden, mit der die Auswertung in diesem Verzeichnis gestartet werden kann. Diese hat im Wesentlichen die Aufgabe eine andere, für diesen Virendatenbank-Typ geeignete Batchdatei mit den passenden Parametern (Dateiname der zugehörigen Variantliste, Betriebssystem, aktuelles Verzeichnis) zu starten. Es existieren Batchdateien für folgende Datenbanktypen: "normal", "packed" und "false positives". Andere Kriterien zum Unterscheiden der Datenbanktypen werden nicht verwendet.

Die Variantlisten müssen ebenfalls erstellt werden. Diese sind ein zentraler Bestandteil der Auswertung und enthalten jeweils eine Liste aller Viren (bzw. Virus-Varianten) und der Anzahl der zugehörigen Samples. Dabei wird jeder Virus durch das entsprechende Verzeichnis angegeben und die Anzahl der Samples entspricht (bis auf wenige Ausnahmen) der Anzahl der Dateien in diesem Verzeichnis. Für die Erstellung existieren mehrere Skripte und Batchdateien im Verzeichnis `P:\variant\erzeugen`, die aus den Dirlisten[5] die Vari-

[4] Es ist auch schon passiert, daß aus Versehen zwei alternative Log-Verzeichnisse angelegt wurden

[5] Eine Dirliste ist einfach eine Auflistung aller Dateien einer Datenbank und entspricht der Ausgabe des `dir` Befehls

antlisten erstellen. Allerdings arbeiten diese nur mit Dateien im selben Verzeichnis, so daß man zusätzlich jede Dir- und Variantliste per Hand kopieren und umbenennen muß. Für jede Datenbank existiert eine Variantliste. "false positive"-Datenbanken benötigen allerdings eine zweite Variantliste, die nur die "false positives" enthält. Allgemein ist es auch denkbar, mehrere Variantlisten pro Datenbank zu erstellen, um verschiedene Teile der Datenbank auszuwerten (z.B. nur die Teilmenge der ITW-Viren). Dies wird von den Auswertungsskripten allerdings nicht unterstützt.

3.3.4 Durchführung der Auswertung

Nachdem neue Logdateien in das Auswertungsverzeichnis kopiert wurden, muss die in diesem Verzeichnis erstellte Batchdatei (siehe oben) gestartet werden. Dabei ist darauf zu achten, dass man sich im richtigen Verzeichnis befindet, bevor die Batchdatei gestartet wird, da einige Skripte Dateien im aktuellen Verzeichnis erwarten und sonst fehlerhafte Ergebnisse produzieren würden. Dies ist natürlich immer wieder eine Fehlerquelle und eigentlich gar nicht notwendig, da den Skripten der Name des Auswertungsverzeichnisses zusätzlich als Parameter übergeben wird. Daher ist bei allen Änderungen an den Skripten darauf zu achten, daß solche fest einprogrammierten Abhängigkeiten vermieden werden.

Bevor die Logdateien an die Reihe kommen, werden von den Skripten noch die Variantlisten dieser Datenbank (es können mehrere sein, falls die Datenbank "false positives" enthält) ins Auswertungsverzeichnis kopiert sowie alle Ergebnisse früherer Auswertungsvorgänge aus dem Unterverzeichnis `NORMAL` zurückkopiert.

Jetzt werden die Logdateien verarbeitet. Dabei werden leider immer *alle* Logdateien im Auswertungsverzeichnis verarbeitet, auch wenn diese schon früher ausgewertet wurden. Dies lässt sich kaum vermeiden, da die Skripte nicht zwischen alten und neuen Logdateien unterscheiden können. Da Logdateien nicht gerade klein sind (eine Logdatei kann durchaus über 100 MB groß werden), ist der Zeitverlust hier enorm. Das Testen jeder noch so kleinen Änderung an irgendeinem Skript erfordert somit die komplette Neuauswertung aller Logdateien.

Ein Ansatz diesen Zeitaufwand zu verringern – schließlich erfordern gerade die Scanner-Pattern laufende Änderungen – ist die Einführung zusätzlicher Status-Dateien. Existiert im Auswertungsverzeichnis eine Datei namens `FOO.o_k`, wird die Logdatei dieses Produkts nicht neu ausgewertet, stattdessen werden nur alle Dateien dieses Produkts aus dem Unterverzeichnis `NORMAL` ins Auswertungsverzeichnis kopiert (nur um am Ende unverändert wieder ins Unterverzeichnis verschoben zu werden). Dadurch wird schon viel Zeit gespart, allerdings benötigt alleine das unvermeidbare Hin- und Herkopieren der Dateien einige Zeit, da es sich um Netzwerklaufwerke handelt.

Die anderen beiden Status-Dateien sind `FOO.n_s` ("no scanner"), um zu signalisieren, dass dieses Produkt nicht unter diesem Betriebssystem bzw. dieser Virendatenbank getestet wurde, sowie `FOO.n_r` ("no report"), um deutlich zu machen, dass keine Logdatei erzeugt werden konnte. Diese beiden Status-Dateien werden allerdings nicht weiter verwendet, sie sollen nur den Auswertern helfen, den Überblick zu behalten.

Nachdem die Logdateien in mehreren Schritten zerlegt wurden (siehe nächsten Abschnitt), steht das Ergebnis jedes Produkts in einer eigenen Datei (`FOO.dat`). Durch das anschließende Zusammenkopieren aller Ergebnisdateien entstehen so die aus dem Testbericht bekannten Tabellen.

Zum Schluß werden alle Dateien der ausgewerteten Produkte bis auf die Logdatei (also alle Zwischenschritte sowie das Ergebnis) ins Unterverzeichnis `NORMAL` verschoben. Dies soll die Übersicht erhöhen.

3.3.5 Einzelheiten der verwendeten Skripte

In diesem Abschnitt wird der typische Auswertungsablauf *einer* Logdatei beschrieben. Da die Logdatei jedes Produkts anders aussieht, wird diese zuerst in ein einheitliches Format überführt, bei dem jede Zeile nur genau einen Dateinamen mit vollständigem Pfad und (wenn zutreffend) dahinter mit einem Leerzeichen getrennt den gemeldeten Virusnamen enthält. Mittels des Scanner-Pattern Skripts werden dabei aus der Logdatei zwei neue Dateien in diesem einheitlichen Format generiert: Eine Datei (`FOO_1`) enthält alle als infiziert gemeldeten Dateien (und die entsprechenden Virusnamen), die andere (`FOO.sbo`) enthält alle als nicht infiziert gemeldeten Dateien (diese wird nicht weiter ausgewertet sondern dient nur zum Überprüfen, ob alle Dateien gescannt wurden). Zusätzlich werden alle anderen Zeilen der Logdatei in einer weiteren Datei gespeichert (`FOO.res` – gewissermaßen der nicht verwertbare Rest). Dies ist nützlich, um Fehler in den Scanner-Pattern zu finden. Leider ist die Aufteilung zwischen diesen Dateien nicht disjunkt, da manchmal auch die Zeilen der `FOO.sbo` zusätzlich in der `FOO.res` gespeichert werden[6]. Außerdem werden Angaben über die Summe aller gescannten bzw. aller infizierten Dateien extrahiert (soweit dies vom Produkt ausgegeben wird). Da diese Angaben nicht sehr zuverlässig sind, werden sie nicht weiter ausgewertet, sondern nur zur Kontrolle angegeben.

Die Scanner-Pattern erfordern dabei laufende Änderungen: Zum Einen erzeugt ein Produkt unter verschiedenen Betriebssystemen meist auch unterschiedliche Logdateien, es existiert aber für jedes Produkt nur ein Scanner-Pattern für alle Betriebssysteme. Aber auch die verwendeten Optionen und die Art des Aufrufs (ob von der Kommandozeile oder per grafischer Oberfläche) haben erheblichen Einfluss auf das Aussehen der Logdatei. Zum Anderen ändert sich das Format der Logdatei bei einigen Produkten fast mit jeder Produktaktualisierung.

Da für die Ergebnisse bei jeder Datei nur zwischen ”infiziert” und ”nicht infiziert” unterschieden wird, muß bereits an dieser Stelle begonnen werden, die Ausgaben der Produkte zu interpretieren. Denn in den Logdateien finden sich häufig nur Meldungen wie

- ”`Possibly infected with an unknown virus`”

[6] Manchmal ist das Ganze eben auch weniger als die Summe seiner Teile

- ”`Could be a destructive program`”
- ”`suspicious code found`”
- ”`suspect: Macro.VBA`”
- ”`Is a joke program`”

Sollen solche Meldungen nun als ”infiziert” oder als ”nicht infiziert” gewertet werden ?

Die vermeintlich einfachste Lösung, jede Scannermeldung außer ”`ok`” als ”infiziert” zu zählen funktioniert schon deshalb nicht, weil viele Produkte eine Menge zusätzlicher Informationen zu den einzelnen Dateien ausgeben, die für eine reine Malware-Erkennung unwichtig sind. Beispiele für solche Meldungen sind

- ”`archive HTML`”
- ”`packed: ASPack`”
- ”`contains macros`” (dies ist *kein* eindeutiger Hinweis auf maliziöse Inhalte!)

Aber auch unter wissenschaftlichen Gesichtspunkten lassen sich kaum verbindliche Kriterien angeben, um die Frage ”infiziert oder nicht” eindeutig zu entscheiden, da der Begriff ”Malware” nicht so exakt definiert ist wie der Begriff ”Virus”[7].

Für die Auswertung im aVTC werden alle Meldungen, die eindeutig *keinen* Hinweis auf irgendeine Art von Malware darstellen (wie z.B. in der zweiten Liste) als ”nicht infiziert” gewertet und alle anderen als ”infiziert”.

Manchmal finden sich auch widersprüchliche Meldungen:

```
K:\KIT\VBS\SVBSVC\A\VBS_001_.VBS  ok
K:\KIT\VBS\SVBSVC\A\VBS_001_.VBS/(SCRENC)  infected: VBS.Creator.Gen.B
```

Diese werden in der Regel zugunsten der Produkte als ”infiziert” gezählt, aber zusätzlich im Testbericht in der Problemliste erwähnt.

Als nächstes wird die Datei `FOO_1` sortiert und es entsteht die Datei `FOO_2`. Doppelte Zeilen werden dabei gleich mit entfernt, was die bei Nachscans unvermeidbar vorkommenden mehrfachen Meldungen der gleichen Datei zwar eliminiert, etwaige Mehrfachmeldungen des Produkts selber so aber nicht mehr ermittelt werden können.

Danach wird aus der Liste der infizierten Dateien eine mit der Variantliste vergleichbare Datei erzeugt. Dazu wird jede Zeile durchgegangen, der Dateiname vom Pfad abgeschnitten und gleichzeitig die Anzahl der Dateien in dem so bestimmten Verzeichnis gezählt. Der vom Produkt gemeldete Virusname wird nur verwendet, um zu überprüfen, ob alle Dateien in einem Verzeichnis (bzw. alle Samples eines Virus) mit dem gleichen Namen erkannt wurden. Ist dies nicht der Fall, wird dies als unzulässige Identifizierung (”unreliable identification” im Testbericht) dieses Virus gezählt. Allerdings nur, wenn auch alle Dateien dieses

[7] Und schon der Begriff ”Virus” wird mit sehr unterschiedlichen Bedeutungen verwendet

Virus erkannt wurden, andernfalls wird dies als unzuverlässige Erkennung ("unreliable detection") gezählt. Da es keine einheitlichen Namenskonventionen für Malware gibt, ist es nicht möglich bzw. nicht sinnvoll die Korrektheit der Namen auszuwerten.

Diese vier Angaben (Pfadname, Anzahl der gefundenen Dateien in diesem Verzeichnis, unzuverlässige Identifizierung, unzuverlässige Erkennung) werden in eine weitere Zwischendatei geschrieben. Mit einem `join` gegen die Variantliste entsteht daraus jetzt eine Ergebnisliste, die alle Viren mit allen notwendigen Angaben enthält.

Jetzt braucht nur noch zusammengezählt werden und das Ergebnis steht fest. Dabei werden folgende Werte ermittelt:

- Die Anzahl der erkannten Viren (genauer: der Virus-Varianten) – dies ist gleich der Anzahl der Zeilen in der Ergebnisliste, in denen die Anzahl der gefunden Dateien größer Null ist.
- Die Anzahl der erkannten Samples – dies ist gleich der Summe der gefundenen Dateien in jeder Zeile der Ergebnisliste
- Die Anzahl unzuverlässig erkannter Viren – dies ist gleich der Summe des entsprechenden Feldes in jeder Zeile der Ergebnisliste
- Die Anzahl unzuverlässig identifizierter Viren – dies ist gleich der Summe des entsprechenden Feldes in jeder Zeile der Ergebnisliste

Außerdem werden jeweils die Prozentwerte bezogen auf die gesamte Datenbank berechnet. Dafür muß natürlich die Anzahl der Viren und der Samples in der Datenbank bekannt sein. Da diese nirgendwo gespeichert werden, werden sie jedes mal neu berechnet. Die Anzahl der Viren in der Datenbank ist gleich der Anzahl der Zeilen in der Ergebnisliste und die Anzahl der Samples ist gleich der Summe des entsprechenden Feldes in jeder Zeile der Ergebnisliste. Es wäre sicherlich besser, diese Daten bei der Erstellung der Variantlisten einmal zu berechnen und dann nur noch auf die gespeicherten Werte zurückzugreifen.

3.3.6 Qualitätssicherung

Die Qualitätssicherung muß im Wesentlichen per Hand erledigt werden, da die Auswertungsskripte kaum Hilfen geben. Lediglich die Sortiervorgänge lassen sich leicht überprüfen, da die Zeilenanzahl der unsortierten und der sortierten Dateien ausgegeben werden[8].

Die Angaben der Produkte selber, wieviele Dateien gescannt wurden und wieviele infiziert sind, sind meistens nicht brauchbar. Oft werden nicht nur Dateien sondern auch Verzeichnisse gezählt, manchmal wird die Anzahl der "gescannten Objekte" gezählt (wobei ein Word-Dokument mit vielen Makros dann auch entsprechend viele "gescannte Objekte" enthält). Es gibt aber auch Produkte, bei denen die Zählweise gar nicht nachvollziehbar ist.

[8] Auf diese Weise wurde ein Fehler in einer früher verwendeten `sort`-Version entdeckt

In einigen wenigen Fällen können diese Angaben allerdings auf Fehler in der Auswertung hinweisen.

Nur die Anzahl der Zeilen in allen einzelnen Dateien zu vergleichen ist manchmal nicht ausreichend, da es durchaus vorkommt, das dieselbe Datei als infiziert und als nicht infiziert einsortiert wird (z.B. eine gepackte Datei). Manchmal wird auch eine Datei mehrfach aufgeführt. Fehlen dann dafür andere Dateien, kann die Anzahl trotzdem stimmen.

Die meisten Fehler werden in der Regel von falsch oder unzureichend funktionierenden Scanner-Pattern verursacht. Typische Fehler in der `FOO_1` sind dabei:

- Die letzte Zeile der Logdatei fehlt oder es existiert eine zusätzliche Leerzeile am Anfang – dies passiert häufig bei Produkten, bei denen sich Angaben zu einer Datei über mehrere Zeilen erstrecken.
- Es wird in jeder Zeile statt des Datei- bzw. Virusnamens nur Unsinn ausgegeben – hier wird das falsche Feld aus der Logdatei extrahiert.
- Hinter dem Dateinamen steht noch ”Anhänge” (z.B. `W97_000_.DOC/Module3`), oft wird diese Datei dann mehrfach ausgegeben – hier werden die Dateinamen nicht richtig gefiltert.
- Die `FOO_1` und die `FOO.sbo` sind leer – die Meldungen des Produkts werden falsch oder gar nicht interpretiert.

Notwendig ist auf jeden Fall, die gefundenen infizierten Dateien (`FOO_2`) und die gefundenen nicht infizierten Dateien (`FOO.sbo`) mit der Variantliste zu vergleichen, um sicherzustellen, daß wirklich jede Datei in der Virendatenbank gescannt wurde. Dies lässt sich z.B. mit der folgenden Befehlsfolge erledigen:

```
copy FOO_2 + FOO.sbo FOO.tmp
sort -u FOO.tmp > FOO_srt.tmp
join -v 1 variant.lst FOO_srt.tmp > FOO_chk.txt
```

Allerdings ist es aufwändig, dies jedesmal von Hand durchführen zu müssen.

Werden hierbei Dateien gefunden, bedeutet dies noch nicht, dass diese nicht gescannt wurden. Zuerst muß überprüft werden, ob diese Dateien nur bei der Auswertung nicht berücksichtigt wurden. Dann sollte sich die entsprechende Zeile allerdings in der `FOO.res` finden lassen. Dieser Fall ist typisch für neue oder veränderte Meldungen eines Produkts (wenn z.B. statt ”infection:” jetzt ”infected by” ausgegeben wird oder statt ”could be infected” jetzt ”possibly infected” gemeldet wird). In so einem Fall, muß das Scanner-Pattern Skript angepasst werden und die Auswertung neu gestartet werden. Hat die Suche keinen Erfolg, muß noch die Logdatei selber nach dem Dateinamen durchsucht werden (z.B. mittels ”grep”), da auch Fehler in den Scanner-Pattern auftreten können, bei denen ganze Zeilen verschluckt werden.

Wird auch in der Logdatei nichts gefunden, ist somit sicher, dass die gesuchte Datei nicht korrekt gescannt wurde. Es ist natürlich auch denkbar, dass eine Datei zwar gescannt, aber nicht gemeldet wurde – mangels Nachweismöglichkeiten[9] macht dies in der Praxis keinen Unterschied (es ist beides ein Fehler des Produkts, nicht des Auswertungsvorgangs). In diesem Fall wird ein Nachscan angesetzt (insgesamt aber maximal zwei, mehr wären vom Aufwand her nicht vertretbar). Dazu wird dem Tester eine Liste der fehlenden Dateien übergeben. Da diese bei einigen Produkten recht umfangreich ausfallen, ist es manchmal nicht machbar, nur diese Dateien erneut zu scannen. In diesem Fall werden die ganzen Verzeichnisse oder sogar das ganze Laufwerk erneut gescannt (das liegt im Ermessen des Testers).

Die Logdateien der Nachscans werden (wenn sie im Log-Verzeichnis eingetroffen sind) per Hand an die erste Logdatei angehängt und daraufhin die Auswertung wiederholt. Dadurch ergeben sich eventuell viele Samples, die mehrmals als infiziert gemeldet werden (bei jedem Scan einmal), da mehrfache Meldungen bei der Auswertung jedoch ignoriert werden, macht dies für die Ergebnisse keinen Unterschied.

3.4 Berichts-Phase

Die Testberichte werden ausschließlich im Textformat (ASCII) erstellt und veröffentlicht. Dies hat zum einen historische Gründe, bietet die beste Kompatibilität und ist vor allem das einzige Format, in dem keine Viren verbreitet werden können. Die Nachteile dieses Formats sind die beschränkten Formatierungsmöglichkeiten (insbesondere da viele Tabellen enthalten sind) sowie schlechte Gliederung (es gibt nur die Möglichkeit, den Text auf mehrere Dateien zu verteilen).

Der Testbericht wird weitgehend per Hand erstellt. Um Zeit zu sparen, werden dazu oft alte Testberichte als Grundlage genommen und an den aktuellen Test angepasst. Die Ergebnistabellen werden per "Copy & Paste" in den Bericht eingefügt. Angesichts des Umfangs, den ein Testbericht annehmen kann (200 Seiten), ist dies wiederum sehr fehleranfällig. Typische Fehler sind z.B. alte Überschriften oder fehlende Zeilen in den Tabellen. Da zudem die gleichen Tabellen an jeweils zwei verschiedenen Stellen in den Testbericht eingefügt werden (einmal in den Ergebnisabschnitt und einmal in den Bewertungsabschnitt), müssen alle Texte zusätzlich auf Konsistenz miteinander überprüft werden. Zusätzliche Daten wie Durchschnittswerte, etc. werden ebenfalls per Hand berechnet.

Gleichzeitig werden auch die Produktbewertungen erstellt. Da die Kriterien bereits vorher feststehen, beschränkt sich dies im Wesentlichen auf das Anwenden der Kriterien auf die Ergebnisse und ließe sich somit gut automatisieren. Da die Kriterien z.T. sehr differenziert sind, bedeutet dies zur Zeit noch viel Handarbeit.

Am Korrekturlesen sind alle aVTC-Mitarbeiter beteiligt, allerdings haben die Auswerter den Vorteil die Ergebnisse besser zu kennen und sind daher stärker beteiligt als die Tester.

Nachdem der Testbericht den Qualitätsansprüchen genügt, wird er den Herstellern der Produkte zugesandt, damit diese noch vor der Veröffentlichung ebenfalls Korrekturen anmelden

[9] Dies ist einer der Fälle, in denen die gemeldete Summe gescannter Dateien einen Hinweis geben kann

können. Diese Korrekturen werden allerdings nicht einfach übernommen, sondern einzeln geprüft. Abgesehen von bisher nicht gefundenen Fehlern betrifft dies vor allem Reklamationen von nicht-viralen Dateien in den Virendatenbanken. In so einem Fall wird die jeweilige Datei nochmals analysiert. Lässt sich die virale Eigenschaft dabei nicht eindeutig nachweisen, wird die Datei aus der Datenbank entfernt und die Ergebnisse werden entsprechend angepasst. Sind alle Korrekturwünsche entsprechend behandelt worden, wird der Testbericht im Internet veröffentlicht.

4 Mögliche Verbesserungen

In diesem Kapitel werden Schwachpunkte der bisherigen Testdurchführung genannt und Verbesserungen vorgeschlagen. Die Umsetzung dieser Vorschläge beschränkt sich im Rahmen dieser Arbeit ausschließlich auf die Auswertungs-Phase.

4.1 Vorbereitungs-Phase

Die Pflege der Datenbanken sowie die Vorbereitung der Datenbanken für einen neuen Test wird immer komplexer und nimmt zunehmend mehr Zeit in Anspruch, da die Größe der Datenbanken ständig zunimmt[1]. Es kommt neuerdings vor, dass noch nach Beginn der Scan-Phase Fehler in den Datenbanken entdeckt werden (z.B. nicht-virale Dateien oder falsch einsortierte Samples). Deshalb muß das Einsortieren neuer Samples stärker automatisiert werden, insbesondere dürfen keine Kopier- und Verschiebe-Operationen von Hand durchgeführt werden, um die Gefahr versehentlich aktivierter Viren zu verringern. Es wäre von Vorteil, wenn die Datenbanken täglich aktualisiert werden, so daß für einen neuen Test nur eine neue Kopie erstellt werden muß und keine separaten (und somit aufwändigen) Sortiervorgänge notwendig sind. Zusätzlich müssen Konsistenzchecks entwickelt werden (z.B. um ungültige Dateinamen herauszufiltern).

Alternativ ist zu überlegen, ob die Virendatenbanken nicht wie bisher als Verzeichnishierarchie gespeichert werden, sondern als ”echte” Datenbanken organisiert werden. Das hätte den Vorteil, dass alle Verwaltungsoperationen (inklusive dem Erstellen der Testdatenbanken) mit einem dafür geeigneten Programm erledigt werden, statt dies per Hand erledigen zu müssen oder eigene Sortier- und Verwaltungsprogramme schreiben zu müssen.

Da die Fähigkeit eines Anti-Malware-Produkts auf aktuelle Bedrohungen zu reagieren ein entscheidendes Qualitätsmerkmal ist, sollten die Virendatenbanken keinen älteren Stand haben als die zu testenden Produkte. Wie bereits jetzt dürfen die Virendatenbanken auch nicht neuer als die Produkte sein, da Signatur-basierte Scan-Methoden sonst nicht vernünftig getestet werden können (davon ausgenommen sind natürlich die Heureka-Tests).

Um die Aktualität des Tests zu erhöhen, sollte die Scan-Phase direkt nach dem Eintreffen der Produkte beginnen - alle anderen Vorbereitungen sollten dann bereits erledigt sein.

[1] Einige der dabei auftretenden Probleme sind in [Bontchev 1998, Kapitel 8] beschrieben.

Um die Transparenz zu erhöhen, sollten Bewertungskriterien bereits vor Beginn der Auswertungs-Phase veröffentlicht werden, insbesondere wenn diese seit dem letzten Test verändert wurden.

Das Neuinstallieren bzw. Wiedereinspielen der Images sollte bereits in der Vorbereitungs--Phase auf *jedem* Client getestet werden, um zeitraubende Überraschungen in der Scan-Phase zu vermeiden (z.B. falls die CDR doch nicht gelesen werden kann).

4.2 Scan-Phase

Es sollte überlegt werden, ob es sinnvoll ist, Anforderungen für die Testdokumentation aufzustellen. Bisher ist jeder Tester selbst dafür verantwortlich, was, wie und wieviel er dokumentiert. Dadurch entstehen leicht Lücken, wenn jemand aufhört oder auch nur eine Weile nicht mitarbeiten kann (z.B. wegen Prüfungen).

4.3 Auswertungs-Phase

Es sollten Statusinformationen zu den einzelnen Produkten angelegt werden, so dass unnötige, wiederholte Auswertungen nicht mehr vorkommen müssen. Dadurch würde es auch einfacher werden, einen Überblick über den Fortschritt des Test zu gewinnen. Zusätzliche ”Zettelwirtschaft” wäre dann nicht mehr notwendig.

Die Auswertungsskripte sollten flexibler werden. Es sollte möglich sein, sowohl alle Logdateien auf einmal wie auch einzelne Logdateien auszuwerten.

Generell ist die Benutzerschnittstelle der Auswertungsskripte verständlicher zu gestalten. Dies beinhaltet sowohl eine intuitive Bedienung wie auch zusätzliche Dokumentation. Es sollte möglich sein, die Auswertung durchzuführen ohne mit den Auswertungsskripten im Detail vertraut zu sein [2].

Zur Zeit ist die Fehlersuche in den Skripten sehr mühsam, da diese willkürlich über verschiedene Verzeichnisse verteilt sind. Da etliche Verzeichnisse fest einprogrammiert sind, lassen sich Änderungen nur mit großem Aufwand durchführen. Um diese Probleme zu beheben muß die Verzeichnisstruktur wesentlich vereinfacht werden. Ebenfalls müssen alle Abhängigkeiten von fest einprogrammierten Verzeichnissen beseitigt werden.

Zum Vereinheitlichen der Logdateien sollte nicht mehr ein Scanner-Pattern für alle Betriebssysteme genutzt werden. Dadurch werden zwar ca. fünfmal so viele Scanner-Pattern benötigt, aber die Komplexität der Scanner-Pattern lässt sich erheblich reduzieren. Auch der

[2] Natürlich ist es für die Qualitätssicherung unerläßlich, die Skripte im Detail zu kennen – schließlich können Fehler auch durch die Skripte verursacht werden. Allerdings sollte der zeitaufwändige Teil der Auswertung von jedem erledigt werden können, der gerade zur Verfügung steht. Und auch die Einarbeitung neuer Auswerter würde deutlich erleichtert werden

erhöhte Aufwand bei der Anpassung an neue Logformate ist geringer einzuschätzen als die Fehlersuche in komplexen Scanner-Pattern.

Die Qualitätssicherung sollte besser von den Auswertungsskripten unterstützt werden. Es gibt einige Abläufe, die für beinahe jede Logdatei per Hand durchgeführt werden – dies lässt sich leicht per Skript erledigen. Zur Zeit hat jeder Auswerter seine eigenen Testmethoden für die Qualitätssicherung. Hier sollten einheitliche Methoden entwickelt werden und soweit möglich automatisiert werden. Zusätzlich sollte der "Datenfluss" durch die verschiedenen Auswertungsschritte so organisiert werden, daß sich in jedem Auswertungsschritt Gültigkeitsaussagen über die entstandenen Daten treffen lassen (etwa in Form von Programm-Invarianten).

Die Ermittlung der Daten für "unzuverlässige Identifikation" ("unreliable identification") muß sorgfältiger durchgeführt werden. Z.B. sollten auch alle Viren, die nur allgemein erkannt werden (etwa als "`is a destructive program`") ebenfalls als unzuverlässige Erkennung gewertet werden. Eventuell ist es sogar sinnvoll, Meldungen wie "`possibly infected by an unknown virus`", die keine deutliche Entscheidung treffen, getrennt zu erfassen und auszuwerten. Dieser Vorschlag konnte allerdings im Rahmen dieser Arbeit aus Zeitgründen nicht umgesetzt werden.

4.4 Berichts-Phase

Die Lesbarkeit des Testberichts muß deutlich verbessert werden. Insbesondere die vielen aufeinander folgenden Tabellen erschweren im jetzigen Format die Lesbarkeit. Auch die Zusammenfassung ("executive summary") ist mit bis zu 30 Seiten viel zu umfangreich.

Daher sollte der Testbericht anders gegliedert werden. Es sollte möglich sein, die wichtigsten Ergebnisse innerhalb der ersten paar Seiten zu finden. Die Zusammenfassung sollte sich ausschließlich auf die wichtigsten Ergebnisse beschränken und nicht mehr als ein paar Seiten umfassen. Die folgenden Kapitel des Testberichts sollten zunehmend mehr Details enthalten, wobei Wiederholungen vermieden werden sollten.

Alle Inhalte, die sich auf mehrere Tests beziehen (Zeitreihen, Entwicklung der Produkte), sollten in einem separaten Kapitel zusammengefasst werden, da diese Inhalte nicht notwendigerweise mit den test-spezifischen Ergebnissen zusammen veröffentlicht werden müssen.

Allgemeine Kapitel, deren Inhalt sich von Test zu Test kaum ändert, sollten eher am Ende des Testberichts zu finden sein – eventuell auch als Anhang.

Da oft ein alter Testbericht als Vorlage genommen wird, ändern sich viele Formulierungen nicht. Daher würde es viel Zeit sparen und "copy & paste"-Fehler verhindern, wenn diese Teile des Testberichts von einem Skript generiert würden. Dazu müssten Vorlagen erstellt werden, in die dann nur noch die spezifischen Daten (Testname, -zeitraum, Ergebnisse, Produktinformationen) eingesetzt werden bräuchten. Einige Kapitel (z.B. die Beschreibung der aufgetretenen Probleme) müssen natürlich weiterhin per Hand erstellt werden.

Die Produkt-Bewertungen könnten bis zu einem gewissen Grad ebenfalls auf diese Weise generiert werden.

Die Lesbarkeit könnte deutlich erhöht werden, wenn der Testbericht im HTML-Format erstellt würde. Vor allem die Darstellung der Tabellen kann davon deutlich profitieren. Heutzutage kann davon ausgegangen werden, daß HTML-Formate genauso weit verbreitet sind wie Text-Formate, daher sollte die Verbreitungsmöglichkeit davon nicht eingeschränkt sein. Das hauptsächliche Argument für ASCII-Text war, daß nur in diesem Format keine Viren übertragen werden können. Dies trifft für HTML nicht zu. Allerdings gibt es keine direkt in HTML programmierten Viren, es können nur maliziöse Inhalte in eine HTML-Datei eingebettet werden ("active content"). Da der Testbericht kein "active content" benötigt sondern rein statisch geschrieben werden kann, lassen sich solche Schwachstellen vermeiden. Somit besteht zwischen Test- und HTML-Format in dieser Hinsicht kein großer Unterschied.

Der einzige Nachteil des HTML-Formats ist in der schlechteren Druckunterstützung zu sehen. Gerade bei größeren Tabellen haben viele Programme Probleme, diese korrekt auszudrucken. Daher ist es sinnvoll, eine HTML-Version des Testberichts nur zusätzlich zu erstellen.

5 Beschreibung der durchgeführten Verbesserungen

Die Auswertungs-Phase wurde komplett neu konzipiert und implementiert, da eine Änderung der bestehenden Skripte mindestens ebenso aufwändig gewesen wäre. Um alle Anforderungen zu erfüllen (siehe unten) hätte kein größerer Skriptteil unverändert übernommen werden können.

5.1 Anforderungen

An die Auswertungsskripte werden folgende Anforderungen gestellt:

- Der zum Auswerten notwendige Zeitaufwand soll minimiert werden.
- Alle nicht notwendig per Hand zu erledigenden Aufgaben sollen automatisiert werden.
- Es soll jederzeit möglich sein, schnell einen Überblick über den aktuellen Stand zu bekommen.
- Die Auswertung soll auch nach einiger Zeit noch nachvollziehbar bzw. wiederholbar sein. Insbesondere sind alle dafür notwendigen Informationen zu archivieren.
- Es muß möglich sein, die Auswertung auf einzelne Betriebssysteme, Virendatenbanken oder Produkte zu beschränken. Eine Auswertung aller Logdateien muß trotzdem einfach durchzuführen sein.
- Die Auswertungsskripte müssen einfach bedienbar und gut dokumentiert sein, um eine schnelle Einarbeitung (sowohl in die Bedienung wie in die zugrunde liegenden Mechanismen) zu ermöglichen. Insbesondere soll die Bedienung auch ohne Kenntnis der zugrunde liegenden Mechanismen möglich sein.
- In den Auswertungsskripten sollen keine Annahmen über Verzeichnisse fest implementiert sein. Alle notwendigen Verzeichnis-Definitionen müssen an einer zentralen, leicht änderbaren Stelle (Konfigurationsdatei) gesammelt sein. Das gesamte Auswertungssystem soll sich mit minimalen Änderungen an andere Verzeichnisstrukturen anpassen lassen.

- Es soll möglich sein, mehrere verschiedene Tests gleichzeitig auszuwerten.
- Um unnötige Serverbelastungen zu vermeiden, sollen möglichst viele Berechnungen lokal auf dem Client durchgeführt werden. Alle anfallenden Daten müssen natürlich auf dem Server gespeichert werden.
- Da der Server nur begrenzt Festplattenplatz besitzt, ist die Möglichkeit vorzusehen, die Daten nur komprimiert auf dem Server abzulegen.
- Die Auswertung sollte möglichst wenig zusätzliche Software erfordern, um die Administration der Rechner und die Fehlersuche zu vereinfachen.
- Die Auswertung sollte nicht von einem Betriebssystem abhängig sein, sondern plattformunabhängig auf allen Clients durchgeführt werden können, um eine optimale Ausnutzung der vorhandenen Hardware zu ermöglichen.

5.2 Implementation

Aufgrund der Anforderungen wurde für die Implementation die Programmiersprache `Perl` ausgewählt, da diese Skriptsprache gut zum Verarbeiten von Texten geeignet ist. Somit kann auf die Programme `awk`, `sort`, `join` und `wc` verzichtet werden[1]. Da bereits einige `Perl`-Skripte in der alten Auswertung verwendet wurden, ist der Einsatz von `Perl` ohne größere Umstellungsrisiken realisierbar. Ein weiterer Vorteil ist die Möglichkeit, in `Perl` relativ einfach plattformunabhängig programmieren zu können. Da `Perl` im Gegensatz zu `awk` auch modulare und objektorientierte Programmierung unterstützt, läßt sich die Auswertungsumgebung wesentlich besser strukturieren als bisher. Dies erhöht sowohl die Bedienbarkeit als auch die Übersicht über die gesamte Auswertungsumgebung. Zudem lassen sich mit `Perl` relativ einfach formatierte Text- und HTML-Dokumente erstellen. Insgesamt besteht die neue Auswertung aus nur noch einem Skript, bei dem die verschiedenen Auswertungs-Funktionen als Module realisiert sind. Auf diese Weise wird die Bedienung stark vereinfacht, während sich zukünftige Erweiterungen leicht als zusätzliche Module einbinden lassen.

5.2.1 Statusinformationen

Eine der wichtigsten Änderungen ist die Verwendung von Statusinformationen. Zwar gab es auch im alten System erste Ansätze (Dateien, bei deren Existenz bestimmte Auswertungsschritte übersprungen wurden), aber erst jetzt wird der Status aller Produkte im Test festgehalten. Erst dadurch ist es möglich, unnötige Auswertungsvorgänge ohne Änderungen an den Skripten zu erkennen und zu überspringen. Als weiterer Vorteil lassen sich jetzt

[1] sort wird allerdings weiterhin verwendet

jederzeit Berichte über das Voranschreiten des Tests erstellen und die Notwendigkeit anderer aufwändiger ”Buchführungen” (in der Regel durch Zettelsammlungen an der Wand) entfällt.

Während einige Statusänderungen vom Auswertungsskript automatisch durchgeführt werden, darf dies bei anderen nicht der Fall sein – z.B. darf der Status erst dann auf ”fertig” geändert werden, wenn die Qualitätssicherung erfolgt ist. Deshalb sollte dieser Status nicht automatisch vergeben werden dürfen. Um in Ausnahmefällen – wenn etwa mit falschen Einstellungen getestet wurde und neue Logdateien erstellt werden müssen – nicht den Status vieler Logdateien per Hand ändern zu müssen, gibt es einen neuen Auswertungsbefehl, der dies vereinfacht (siehe auch Abschnitt 5.3).

Jeder Logdatei bzw. jedem Produkt pro Betriebssystem und Datenbank wird genau ein Status aus folgender Liste zugeordnet:

NO TEST: Dieses Produkt ist nicht für dieses Betriebssystem bzw. diese Virendatenbank geeignet.

NO REPORT: Es konnte keine Logdatei erzeugt werden. Daher konnten auch keine Ergebnisse ermittelt werden.

-S1: Es wird auf die Logdatei gewartet

S1: Die Logdatei wurde gefunden und archiviert aber noch nicht ausgewertet

S1E: Die Logdatei wurde ausgewertet, die Ergebnisse sind aber noch nicht überprüft worden (Qualitätssicherung)

OK S1E: Die Auswertung ist abgeschlossen (nach einem Scan-Vorgang)

-S2: Es wird auf die Logdatei des ersten Nachscans gewartet

S2: Die Logdatei des ersten Nachscans wurde archiviert aber noch nicht ausgewertet

S2E: Die Logdateien des ersten Scans und des Nachscans wurden zusammenkopiert und erneut ausgewertet, die Ergebnisse sind aber noch nicht überprüft worden (Qualitätssicherung)

OK S2E: Die Auswertung ist abgeschlossen (nach 2 Scan-Vorgängen)

-S3: Es wird auf die Logdatei des zweiten (und letzten) Nachscans gewartet

S3: Die Logdatei des zweiten Nachscans wurde archiviert aber noch nicht ausgewertet

S3E: Die Logdateien wurden zusammenkopiert und erneut ausgewertet, die Ergebnisse sind aber noch nicht überprüft worden (Qualitätssicherung)

OK S3E: Die Auswertung ist abgeschlossen (nach 3 Scan-Vorgängen), auch wenn jetzt noch nicht alle Dateien nachweisbar gescannt wurden

Jeder Status, der mit **OK** oder **NO** beginnt, wird dabei als ”fertig” interpretiert und alle weiteren Auswertungsschritte für diese Logdatei werden übersprungen.

Dieses Schema ist leicht erweiterbar – etwa um weitere Ausnahmefälle oder Scanvorgänge.

5.2.2 Neue Struktur

Im Gegensatz zu vorher ist jedem Test jetzt ein eindeutiger Testname zugeordnet, der den Testzeitpunkt kennzeichnet (z.B. "2002_02"). Ebenso gibt es jetzt eindeutige Abkürzungen für Betriebssysteme und Datenbanken. Somit lässt sich jetzt eine einheitliche Verzeichnishierarchie basierend auf diesen Namen definieren. Damit besteht endlich die Möglichkeit, die Dateien in diesen Verzeichnissen ohne Änderungen am Skript automatisch zu verarbeiten. Alle Daten eines Tests liegen daher innerhalb der Auswertungsumgebung in einem Verzeichnis, das dem Testnamen entspricht (z.B. `P:\auswert\2002_02`). In diesem Verzeichnis findet man dann die Unterverzeichnisse

- `LISTS` für Variantlisten, Dirlisten, etc,
- `KEYS` für die Scanner-Pattern
- `RESULT` für die Ergebnisse
- `EVAL` für Bewertungen der Ergebnisse ("Grading")

sowie wie früher die Auswertungs-Verzeichnisse (nach Betriebssystemen und Datenbanken gegliedert) – aber mit dem Unterschied, dass die Verzeichnisnamen jetzt eindeutig und nicht willkürlich festgelegt sind. Dementsprechend wird auch nicht wieder für jeden Test eine neues Verzeichnis für die neuen Logdateien (Schnittstelle zur Scan-Phase) festgelegt, sondern es wird nur noch ein allgemeines Verzeichnis festgelegt und dann der Testname verwendet (z.B. `P:\scanlogs\2002_02` für den Test "2002_02"). Die Logdateien werden jetzt in einem separaten Verzeichnis (`P:\:bak_logs`) archiviert, um diese Sicherheitskopien besser vor versehentlichem Überschreiben oder Löschen durch Tester oder Auswerter zu schützen[2].

Die Scanner-Pattern sind jetzt nach Betriebssystemen unterteilt – für jedes Produkt existieren nun also bis zu fünf Scanner-Pattern. Dafür lassen sich diese schneller und einfacher anpassen.

Alle Skripte befinden sich unterhalb des Verzeichnisses `P:\scripts`. In diesem Verzeichnis befinden sich das Frontend `avtctest.pl` sowie die Konfigurationsdatei `test.cfg`, während sich die meisten Funktionen in Perl-Modulen im Unterverzeichnis `Avtc` befinden[3].

Alle Verzeichnisse werden nur in der zentralen Konfigurationsdatei `test.cfg` (im Textformat) festgelegt und lassen sich somit leicht ändern. Da sich dem Auswertungs-Frontend auch eine andere Konfigurationsdatei als Parameter übergeben lässt, kann die Auswertung

[2] Es ist durchaus schon vorgekommen, dass aus Versehen Logdateien oder Ergebnisse gelöscht wurden und die Scans wiederholt werden mussten

[3] Dies ist sinnvoll, um Kollisionen innerhalb der Modulhierarchie von Perl zu vermeiden – außerdem erhöht es die Übersichtlichkeit

auch simultan auf verschiedenen Systemen durchgeführt werden. Z.B. ist zum Auswerten unter Linux nur eine Kopie der Konfigurationsdatei an die entsprechenden Verzeichniskonventionen anzupassen und beim Start als Parameter zu übergeben[4]. Aber auch zum Test von Änderungen an den Skripten ist dies hilfreich. Ebenso ist damit die gleichzeitige Auswertung mehrerer Tests möglich oder die Verwendung mehrerer Variantlisten für eine Datenbank.

Im Gegensatz zu früher werden viel mehr Informationen über den Test gespeichert, um die Ausführung zu beschleunigen und um Test-spezifische Informationen von den Skripten zu trennen. Deshalb enthält das Verzeichnis `LISTS` nicht nur Dir- und Variantlisten, sondern zusätzlich

- Die Gesamtzahl der Viren und Samples in jeder Virendatenbank
- Eine Liste aller Produkte im Test
- Eine Liste aller verwendeten Betriebssysteme
- Eine Liste aller Virendatenbanken mit Angabe des Typs, des Laufwerksbuchstabens und des Verzeichnisnamens in der Auswertung
- Eine Liste aller verwendeten Komprimierungsprogramme, falls gepackte Virendatenbanken getestet werden

Somit werden erstmals alle test-spezifischen Daten getrennt von den Auswertungsskripten und für jeden Test separat gespeichert. Erst dadurch lässt sich die spätere Wiederholbarkeit des Tests wirklich sicherstellen. Die Menge der dafür mindestens zu sichernden Daten reduziert sich pro Test – abgesehen von den Logdateien – auf die Verzeichnisse `LISTS` und `KEYS` sowie die Konfigurationsdatei[5].

5.2.3 Ablauf der Auswertung

Die wesentliche Neuerung besteht darin, dass möglichst viele Schritte auf dem lokalen Client ausgeführt werden, so dass der Server und das Netzwerk möglichst wenig belastet werden.

Es wird in jedem Fall zuerst der Status der ausgewählten Logdateien geprüft, um überflüssige Aktionen zu vermeiden. Danach werden die zu bearbeitenden Logdateien aus dem Archiv (auf dem Server) extrahiert und auf den jeweiligen Client kopiert. Alle weiteren Auswertungsschritte laufen nur auf dem Client ab. Erst danach werden die Ergebnisse sowie alle Zwischenschritte (diese aber nur komprimiert) auf dem Server abgelegt und danach der Status aktualisiert.

Die Dateien auf dem Client werden nicht sofort gelöscht, sondern erst beim nächsten Auswertungsvorgang. Somit besteht auch die Möglichkeit, dieselben Logdateien wiederholt

[4]Natürlich muß auch Perl installiert sein, aber das ist heutzutage meistens Standard
[5]Statt der Konfigurationsdatei genügt auch der Testname

auszuwerten, ohne diese erneut über das Netzwerk kopieren zu müssen. Während es bisher leicht vorkommen konnte, dass man beim Erstellen neuer Scanner-Pattern aufgrund der vielen Kopiervorgänge über das Netzwerk bei jedem Auswertungsvorgang über 30 Minuten warten musste und somit schnell 1-2 Tage zur Erstellung benötigte, lässt sich dies jetzt ohne wesentliche Verzögerung erledigen.

Einige Auswertungsschritte wurden geändert, um die Qualitätssicherung zu erleichtern, ansonsten ist der Ablauf der bisherigen Auswertung weitgehend übernommen worden. Allerdings wurden wie bei den Verzeichnissen auch bei den Dateinamen Veränderungen vorgenommen, um eine einheitliche und besser nachvollziehbare Struktur zu erhalten.

Die Scanner-Pattern sind jetzt als `Perl`-Skripte realisiert, die zur Laufzeit dynamisch in das Auswertungsskript eingebunden werden. Der Aufwand alle Scanner-Pattern von `awk` nach `Perl` zu portieren ist angesichts deren Kurzlebigkeit allerdings zu hoch. Deshalb wurde zusätzlich die Möglichkeit vorgesehen, die alten Scanner-Pattern zu verwenden[6]. Somit lässt sich der Umstellungsaufwand zeitlich besser verteilen, allerdings wird solange noch `awk` benötigt.

5.2.4 Qualitätssicherung

Die einzelnen Auswertungsschritte wurden dahingehend geändert, daß zu den Daten weder Zeilen hinzugefügt noch gelöscht werden. Jede Zeile der jeweiligen Eingabedateien wird also auch wieder ausgegeben, wobei die Ausgabedateien disjunkte Teilmengen bilden. Somit lassen sich jetzt in jedem Auswertungsschritt Programm-Invarianten über die Zeilenanzahl der Eingabe- und Ausgabedateien definieren, deren Gültigkeit sich leicht berechnen lässt. Diese Berechnungen werden zusammen mit einem deutlichen Hinweis, ob die Invariante erfüllt ist oder nicht, in eine Kontroll-Datei (`FOO.con`) geschrieben und ermöglichen es, mit einem Blick zu erkennen, ob die einzelnen Verarbeitungschritte fehlerfrei abgelaufen sind. Die Notwendigkeit jedes Zwischenergebnis per Hand zu kontrollieren entfällt also.

Werden Fehler festgestellt, lässt sich zudem schnell erkennen, in welchem Auswertungsschritt diese aufgetreten sind.

5.3 Bedienung

Statt wie bisher für jede Aufgabe und jede Datenbank ein eigenes Skript starten (oder sogar erstellen) zu müssen, gibt es jetzt nur noch das Skript `avtctest.pl`, mit dem sich alle Aufgaben erledigen lassen. Dazu werden dem Skript verschiedene Befehle übergeben. Optional lässt sich angeben, auf welche Logdateien sich der Befehl bezieht. Folgende Befehle sind möglich:

[6] Dies ist als "Fallback"-Mechanismus implementiert – nur wenn das `Perl`-Skript nicht gefunden wurde, wird auf das `awk`-Skript zurückgegriffen

new Die Auswertungsumgebung für einen neuen Test einrichten (dies umfasst das Erstellen der Verzeichnisse, Variantlisten, usw.). Nach Ausführung dieses Befehls kann die Auswertung der Logdateien beginnen.

update Es werden neue Listen erstellt – dies ist notwendig, falls sich nach Testbeginn noch Änderungen an den Datenbanken ergeben

eval Dieser Befehl startet die Auswertung neuer Logdateien

reeval Bereits ausgewertete Logdateien werden mit diesem Befehl erneut ausgewertet (der Status wird also implizit zurückgesetzt). Um die Scanner-Pattern anzupassen sind in der Regel mehrere Auswertungsvorgänge hintereinander notwendig, wofür dieser Befehl sinnvoll ist, da der Status nicht jedesmal per Hand zurückgesetzt werden muß.

archive Hiermit besteht die Möglichkeit, Logdateien nur zu archivieren aber noch nicht auszuwerten (z.B. weil gerade kein Rechner zum Auswerten zur Verfügung steht, aber der Speicherplatz im Log-Verzeichnis knapp wird).

reset Hiermit lässt sich der Status verändern. Dies kann natürlich auch durch Editieren der Statusdateien geschehen, hiermit kann der Status allerdings auch für alle Logdateien (oder alle Logdateien eines Produkts) auf einmal zurückgesetzt werden. Der gewünschte Status muß als Parameter angegeben werden, andernfalls wird auf den initialen Status (”-S1”) zurückgesetzt.

textresult Dient zum Erzeugen der Ergebnistabellen für den Testbericht. Früher wurden diese bei jedem Auswertungsvorgang automatisch miterzeugt, dies ist aber unnötig, da die Tabellen erst viel später gebraucht werden.

reportstatus Erzeugt die Datei `status.txt`, in der aufgelistet ist, welche Produkte fertig, im Test oder noch nicht getestet sind. Zusätzlich wird für jedes Produkt angegeben, welche Logdateien noch fehlen. Dieser Befehl wird bei jedem Auswertungsvorgang mittels ”eval” oder ”reeval” implizit mit ausgeführt.

rank Dies ist ein erster Ansatz, einen Teil der Produkt-Bewertung skript-gesteuert zu erzeugen.

help Beinahe trivial, darf aber nicht fehlen

Ohne weitere Parameter wird der angegebene Befehl für alle Logdateien ausgeführt. Er kann aber durch Angabe des Betriebssystems, der Datenbank oder eines Produkts auch auf die entsprechenden Logdateien beschränkt werden. Z.B. werden mit dem Befehl

```
avtctest eval w2k all foo
```

die Logdateien, die von dem Produkt ”FOO” unter dem Betriebssystem Windows 2000 auf allen Datenbanken erzeugt wurden, ausgewertet.

Eine kleine Hürde gibt es dabei allerdings noch: Da `Perl`-Skripte im allgemeinen nicht direkt ausführbar sind, müsste das Skript jedesmal über den `Perl`-Interpreter aufgerufen werden, z.B.:

```
C:\perl\bin\perl.exe avtctest.pl eval w2k all foo
```

Außerdem muß vorher noch eine Umgebungsvariable gesetzt werden, damit unsere eigenen Perl-Module auch gefunden werden:

```
set PERL5LIB=p:\auswert\scripte
```

Um die Bedienung zu vereinfachen wurde deshalb eine Batch-Datei `avtctest.bat`[7] erstellt, in der diese Kommandos zusammengefasst sind. Somit ist die direkte Ausführung des oben angegebenen Kommandos möglich. Falls das Skript-Verzeichnis `P:\scripts\` in den Suchpfad aufgenommen wurde, ist es sogar egal, aus welchem Verzeichnis das Auswertungsskript aufgerufen wird.

Außerdem existieren für spezielle Anwendungsfälle optionale Parameter:

-c Hiermit kann eine andere Konfigurationsdatei angegeben werden

-s Hiermit werden die Logdateien beim Auswerten nicht vom Server kopiert, sondern es wird darauf vertraut, daß diese bereits auf dem lokalen Rechner im richtigen Verzeichnis liegen. Diese Option ist praktisch, wenn man Scanner-Pattern anpassen muß, da man dabei typischerweise oft hintereinander dieselbe Logdatei auswertet und diese somit nicht ständig über das Netzwerk kopiert werden muß.

Die Anzahl der vom Auswerter durchzuführenden Arbeitsschritte wird also minimiert. Dies sollte die Einarbeitung neuer Auswerter stark vereinfachen, da es leichter fällt den Gesamtablauf nachzuvollziehen.

[7] Natürlich gibt es auch eine entsprechende `avtctest.sh` für den Linux-Rechner

6 Ausblick

Es ließen sich zwar trotz der Beschränkung auf die Auswertung und Qualitätssicherung nicht alle Verbesserungs-Vorschläge umsetzen, das neue Auswertungssystem ist aber bereits einsetzbar und hat sich im laufenden Test 2002-03 (Heureka-2) bereits bewährt. Da in diesem Test nur "normale" Datenbanken verwendet wurden (also keine "packed" oder "false positive" Datenbanken) konnten Funktionen zur Auswertung der anderen Datenbank-Typen noch nicht vollständig implementiert bzw. getestet werden. Dies wird jedoch während des nächsten Tests nachgeholt. Obwohl die relativ kleinen Heureka-Datenbanken ohnehin vergleichsweise einfach auszuwerten sind, hat sich die Auswertung und Qualitätssicherung spürbar vereinfacht und gibt Anlaß zu der Hoffnung, diese Testphase auch in zukünftigen Tests deutlich schneller und mit wesentlich geringerem Aufwand durchführen zu können.

Insgesamt gibt es jedoch noch viele verbesserungswürdige Abläufe bei den aVTC-Tests. Dies betrifft vor allem die Ergebnisdarstellung sowie die Pflege der Datenbanken – wie der aktuelle Testbericht [Brunnstein 2002] mit einem Umfang von über 100 Seiten und einer Gesamtdauer von ca. 3 Monaten zeigt.

Literaturverzeichnis

[aVTC 2000] aVTC: "Viren und Malware – Eine Einführung", 2000, Universität Hamburg

[Bontchev 1998] Bontchev, Vesselin: "Methodology of Computer Anti-Virus Research", Dissertation, 1998, Universität Hamburg

[Brunnstein 2002] Brunnstein, Klaus & aVTC-Team: "aVTC Testreport 2002-03", `ftp://agn-www.informatik.uni-hamburg.de/pub/text/pc-av/2002-03`

[Cohen 1984] Cohen, Fred: "Computer Viruses - Theory and Experiments", 1984, `http://www.all.net/books/virus`

[Marx 2000] Marx, Andreas: "A Guideline to Anti-Malware-Software Testing", EICAR 2000 Best Paper Proceedings, pp. 218-253

[Schmall 2002] Schmall, Markus: "Heuristic Techniques in AV Solutions: An Overview", 2002, `http://www.securityfocus.com/infocus/1542`

[Wildlist] The Wildlist Organization International, `http://www.wildlist.org`