# Klassifikation von bösartiger Software und aktuelle Testergebnisse des Virus Test Centers von AntiMalware-Software unter Linux

Bodo Eggert

Michel Messerschmidt

Jan Seedorf



## Inhaltsverzeichnis



# Abstract

Der folgende Beitrag soll einen Überblick geben über verschiedene Arten von bösartiger Software ("Malware"), die Benutzer von vernetzten Computersystemen heutzutage im Allgemeinen und speziell unter Linux bedrohen, und aufzeigen, welche Antimalwareprogramme unter Linux gegen die vorher definierten Arten von Malware guten Schutz bieten.

Dabei werden zunächst unterschiedliche Arten von bösartiger Software definiert und voneinander abgegrenzt, zusätzlich werden Klassifikationen von Malware erläutert. Im zweiten Teil wird für die unterschiedlichen Arten von Malware angegeben, inwiefern Bedrohungen unter Linux existieren und einzelne Beispiele dazu aufgezeigt. Am Ende des Artikels werden aktuelle Testergebnisse des Virus Test Centers (VTC) der Universität Hamburg von AntiMalware-Software unter Linux im Detail vorgestellt.

# Einleitung

Der Benutzer eines Computersystems sieht die Daten auf seinem Rechner heutzutage einer Vielzahl von Bedrohungen ausgesetzt. Besonders, wenn das Computersystem an ein Netzwerk (z. B. das Internet) angeschlossen ist, besteht die Gefahr von Angriffen auf den Rechner aus dem Netz. Zusätzlich zu direkten Angriffen besteht auch

eine Bedrohung durch bösartige Software, die über Dienste wie e-mail und ftp-Download oder über Disketten auf den Rechner des Benutzers gelangen kann und dort lokal ausgeführt wird. Bösartige Software - im folgenden auch als Malware (engl. "malicious software") bezeichnet - verbreitet sich dabei teilweise selbstständig in Dateisystemen oder Netzwerken (zum Beispiel Würmer, die sich selber an sämtliche Adressen des e-mail Adressbuchs des befallenen Rechners weiter versenden). Teilweise gelangt sie nur mit Hilfe des Benutzers auf den Rechner, etwa beim Download von Dateien aus dem Internet oder beim Herunterladen von e-mail-Anhängen.

Prinzipiell bestehen dabei unabhängig von dem eingesetzten Betriebssystem - unter anderem - folgende Bedrohungen für Benutzer von (vernetzten) Computersystemen:

- Verlust von Daten
- Unbefugter Zugriff auf lokale Daten (aus dem Netzwerk)
- Unbefugter (eventuell unbemerkter) Versand von sensitiven Daten
- Unbefugte (eventuell unbemerkte) Veränderung von lokalen Daten (aus dem Netzwerk)
- Datenspionage bei Versand von Daten über ein Netzwerk
- Adreßfälschung: eine andere Person gibt sich im Netz als der Benutzer aus und verschickt unter dessen Identität Nachrichten oder führt andere Aktionen aus
- Verhinderung des Zugriffs auf Dienste im Netzwerk (denial of service)

Den Hintergrund dieses Beitrags liefert das Virus Test Center der Universität Hamburg ([VTC 2003]). Als Hauptstudiumsprojekt am Fachbereich Informatik werden im Virus Test Center unter Leitung von Prof. Dr. Klaus Brunnstein regelmäßig Tests von Anti-Malware-Software durchgeführt. Das VTC existiert seit 1992, regelmäßige Testberichte werden seit 1997 veröffentlicht. Die Verfasser arbeiten als Studenten in verschiedenen Bereichen an dem Projekt mit und haben so Erfahrungen im Umgang mit bösartiger Software und dem wissenschaftliche Testen von Anti-Malware-Software erlernt.

# Definitionen und Klassifikation von bösartiger Software

## Malware / bösartige Software

> *Definition*: "A software or module is called "malicious" ("malware") if it is intentionally dysfunctional, and there is sufficient evidence (e.g. by observation of behaviour at execution time) that dysfunctions may adversely influence the usage or the behaviour of the original software." ([Brunnstein 1999])

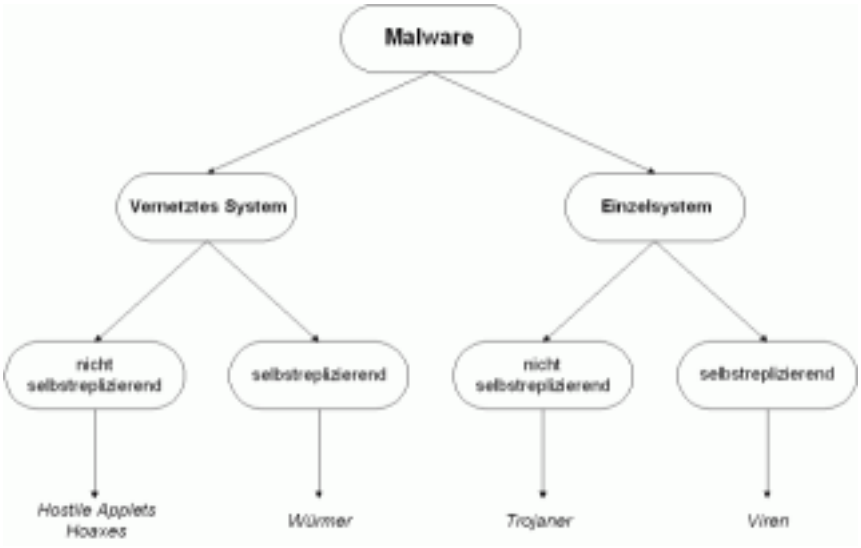
Malware fasst als Oberbegriff alle Arten von bösartiger Software zusammen, die für Benutzer eine Gefahr darstellt. Mit Malware wird jegliche Art von Software bezeichnet, die intentional dysfunktional ist. Dysfunktional bedeutet, daß die Software von der (formalen oder informalen) Spezifikation abweicht und somit für den Anwender "unerwünschte und unerwartete Funktionen besitzt". Intentional bedeutet, daß diese zusätzlichen, verborgenen Funktionalitäten vom Programmierer der Software beabsichtigt sind. Intentional dysfunktionale Software (Malware) beinhaltet als Begriff also nicht so genannte "Bugs" oder sonstiges, unbeabsichtigtes und im Rahmen von Softwareentwicklung oft versehentlich entstandenes Fehlverhalten.

## Klassifikation von Malware

Die Gesamtmenge aller bösartigen Software kann man grundlegend danach unterscheiden, ob die Verbreitung auf einem Einzelsystem oder über vernetzte Systeme stattfindet. Zusätzlich lässt sich Malware danach differenzieren, ob die bösartige Software selbstreplizierend ist oder nicht. Selbstreplizierende Software verbreitet sich bei Ausführung selber, nicht-selbstreplizierende Software verbreitet sich nicht bei der Ausführung ([KittelTicak 2002]).

Nach diesen zwei Unterscheidungsmerkmalen lassen sich vier Gruppierungen von Malware vornehmen (in Klammern jeweils der Oberbegriff für diese Art von Malware):

- Selbstreplizierende Software auf einem Einzelsystem (Viren)
- Selbstreplizierende Software auf vernetzten Systemen (Würmer)
- Nicht-selbstreplizierende Software auf einem Einzelsystem (Trojanische Pferde)
- Nicht-selbstreplizierende Software auf vernetzten Systemen (Hostile Applets)




Viele maliziöse Softwareobjekte sind nicht eindeutig einer der oben genannten Kategorien zuzuordnen, da sie Elemente mehrere Kategorien in sich vereinen. So gibt es zum Beispiel viele Würmer, die auf befallenen Systemen trojanische Pferde oder Viren installieren.

> *Definition*: "Any software that reproduces (or "self-replicates"), possibly depending on specific conditions, at least in 2 consecutive steps upon at least one module each (called "host") on a single system on a given plattform, is called a "virus" (for that plattform). A virus may be compiled (e.g. boot and file virus) or interpreted (e.g. script virus)." ([Brunnstein 1999])

Ein Virus ist eine bestimmte Art von Malware, nämlich bösartige Software, die durch Infektion andere Dateien befällt und sich so reproduziert. Die befallene Datei heißt Wirt. Ein Virus muß sich mindestens zweimal (durch Befall anderer Dateien) reproduzieren können. Das heißt, auch von dem Virus befallene Dateien können wieder andere Dateien infizieren. Ein Virus verbreitet sich auf lokalen Systemen; verbreitet sich maliziöse Software selbstständig über Netzwerke, so wird sie als Wurm bezeichnet.

## Klassifikationen von Viren

Da es sich bei einem Großteil der verbreiteten bösartigen Software um Viren handelt, haben sich verschiedene, allgemein akzeptierte Klassifikationen für diese spezielle Art von Malware durchgesetzt. Zum einen kann man Viren nach der Plattform, für den der Virus programmiert wurde und auf dem er lauffähig ist (das heißt, auf dem er sich durch Replikation verbreiten kann), einteilen. Zum anderen werden Viren häufig danach eingeteilt, ob sie verbreitet sind oder ob es sich um seltene, nicht häufig gefundene Viren handelt.

Eine Einteilung in verschiedene Klassen ist sinnvoll, um einen Überblick über die große Masse an verschiedenen Viren zu erlangen und um neu auftauchende Exemplare einordnen zu können. Allerdings lassen sich einige Viren nicht eindeutig zu den beschriebenen Klassen (s.u.) zuordnen. Da anhand dieser Klassifikationen auch die Malware-Datenbanken des Virus-Test-Centers strukturiert sind, werden in den folgenden beiden Unterabschnitten die Klassifikation nach Plattform und nach Verbreitung kurz erläutert.

## Klassifikation nach Plattform

Jede Art von programmiertem Computercode ist jeweils nur auf einer bestimmten Plattform lauffähig. Da ein Computervirus jeweils nur auf dieser Plattform ausgeführt werden kann, kann man Viren nach der Plattform, für die der Code geschrieben wurde, unterscheiden:

- Boot-Viren (engl. boot viruses)
- Datei-Viren (engl. file viruses)
- Makro-Viren (engl. macro viruses, z. B. .doc, .xls)
- Skript-Viren (engl. script viruses, z. B. .js, .vbs)

### Klassifikation nach Verbreitung

Eine Aussage, wie hoch die Gefahr ist, sich mit einem bestimmten Virus zu infizieren, kann über die Analyse der Verbreitung von Viren getroffenen werden. Viele Hersteller von Anti-Viren-Software stellen auf ihren Webseiten Übersichten bereit, auf denen sie angeben, welche Viren am meisten verbreitet sind ("Top 10" und ähnliches). Diese Übersichten basieren auf den Daten des jeweiligen Herstellers. Obwohl einige Hersteller von Anti-Malware-Software über eine große Anzahl an Kunden verfügen, sind solche Auflistungen nicht notwendigerweise repräsentativ für die tatsächliche Verbreitung von bestimmten Viren.

Eine allgemein akzeptierte Einteilung von Viren nach ihrer Verbreitung ist die Einteilung von der WildList Organisation ([Wildlist 2003]). Die WildList Organisation teilt Viren in zwei Klassen ein:

- Viren, die als in-the-wild betrachtet werden
- Viren, die nicht als in-the-wild betrachtet werden

Beim Testen von Anti-Malware-Software werden die Viren, die auf der aktuellen WildList stehen, als in-the-wild Viren bezeichnet (abgekürzt ITW). Die restlichen Viren einer Testmenge bezeichnet man als Zoo-Viren, da sie kaum verbreitet sind und sich deshalb - die Analogie zu Tieren aufrechterhaltend - nicht in "freier Wildbahn" sondern in einem "Zoo" befinden.

# Einfallstore für Malware

Malware nutzt prinzipiell die normalen Schwachstellen des Systems, um sich zu verbreiten. Die Ausnutzung einer konkreten Schwachstelle ("vulnerability") durch bösartige Software bezeichnet man als "Exploit". Im folgenden sollen dafür zunächst einige allgemeine Verbreitungsbeispiele (unabhängig vom Betriebssystem) betrachtet werden, bevor im darauf folgenden Abschnitt konkrete Beispiele für Malware unter Linux kurz aufgezeigt werden.

# Einfallstore für Malware

## Lokale Verbreitung, Social Engeneering:

Egal, ob eine Datei durch einen Virus infiziert wurde oder ob sie "Klick mich, ich bin ein Bild von AnnaKournikova!.jpg.vbs" heißt, wird dem Benutzer ein vertrautes, harmloses Objekt vorgespiegelt, das dieser gerne nutzen will. Für Virusscanner ist dies der einfachste Fall, denn die Malware liegt als Datei vor, und das Betriebssystem bietet die Möglichkeit, vor der Ausführung die Datei zu scannen.

Anders sieht es aus, wenn in einem Hoax zum Beispiel die Anleitung zum Löschen der "gefährlichen" Datei mit dem Teddybärsymbol (vom Java-Debugger) immer wieder in einem neuen Format weitergeleitet wird. Dies ist kein maliziöser Programmcode, den ein Virusscanner erkennen sollte, sondern die Dummheit der Nutzer, vor der nur Aufklärung und Rechtevergabe schützt.

## Freigaben:

Viele Nutzer geben Platten mit Systemverzeichnissen für das lokale Netzwerk frei und vertrauen darauf, daß niemand böse Absichten hat. Hierbei wird gerne übersehen, daß erstens das Internet ein Netzwerk ist, und zweitens die böse Absicht nicht notwendig ist, um Schaden anzurichten. Wenn Malware die Rechte des Benutzers erlangt hat, dann kann sie alle erreichbaren Dateien löschen oder unbemerkt ändern. Noch schlimmer ist es, wenn ein Wurm Zugriff auf die Konfigurationsdateien nehmen kann, denn so kann dieser nicht nur den Inhalt der Dateien auslesen, sondern auch den Startmechanismus verändern und sich somit zur Ausführung bringen oder Virusscanner abschalten.

Diese Angriffe können theoretisch sehr leicht vermieden werden, indem man Verzeichnisfreigaben auf notwendige Verzeichnisse beschränkt und dort keine ausführbaren Dateien lagert oder anklickt. Dies funktioniert genau so lange, wie alle Nutzer sich daran halten, was der Administrator nur selten sicherstellen kann.

Ein Virusscanner kann gegen derartige Angriffe schützen, aber es ergeben sich einige Einschränkungen. Wichtig ist, daß die Dateien bei jedem Öffnen gescannt werden, denn eine Datei, die nicht für Netzwerkzugriffe gesperrt ist, kann gleichzeitig von einem anderen Nutzer ausgeführt werden. Hier ergibt sich eine Race-Condition; maliziöser Code könnte nach den Öffnen, aber vor dem Ausführen geschrieben werden. Ein Sicherheitsrisiko bleibt also auch bei Nutzung eines Virusscanners bestehen. Besonders On-Demand-Scanner (Überprüfung nach Aufruf) können nicht zuverlässig auf zum Schreiben freigegebenen Verzeichnissen für Sicherheit sorgen, denn zwischen dem Scannen und dem Ausführen einer Datei kann diese wesentlich leichter durch einen Wurm ersetzt werden als bei Verwendung eines On-Access-Scanners (Überprüfung beim Zugriff). Auf keinen Fall dürfen System- oder Konfigurationsdateien mit Schreibzugriff freigegeben sein.

## Wechseldatenträger:

Ein verseuchtes System kann eingelegte Datenträger so modifizieren, daß diese eine Kopie der Malware enthalten, welche beim Einlegen oder beim Booten ausgeführt wird. Da Disketten immer weniger benutzt werden, ist diese Art der Verbreitung derzeit wenig bedeutend. Mit der Verbreitung von wiederbeschreibbaren Medien in Form von USB-Sticks und ähnlichen Geräten wird diese Verbreitungsart jedoch neues Potential erhalten. Steckt man ein Wechselmedium an ein laufendes System, so kann ein dort laufender On-Access-Scanner die Schadsoftware vor der Ausführung erkennen, denn es besteht kein prinzipieller Unterschied zu von Benutzern aufgerufenen Dateien.

Ein neu startendes System, das versucht, von einem Medium zu booten, startet meistens ohne aktives AntiMalware-Programm, so daß ein Virus sich verbreiten kann. Hier besteht jedoch für einen Virus die Schwierigkeit, den Start des Betriebssystems zu überleben, und diese Aufgabe wird zunehmend kompliziert, da die Startroutinen heutiger Betriebssysteme recht komplex sind.

## Buffer Overflow (zu lange Eingabedaten):

Ist eine Programmeingabe zu lang, so kann die Rücksprungadresse der aufgerufenen Funktion durch einen Wert in der Eingabe ersetzt werden. Dies erlaubt bei Kenntnis über das angegriffene System, mit der Anfrage ein Code-Fragment mitzuschicken, welches beim Verlassen der Funktion angesprungen wird. Dieses Code-Fragment kann zum Beispiel einen Wurm nachladen (Morris- und Ramen-Wurm), bereits den gesamten Wurm enthalten (SQL-Slammer) oder eine Shell für den Angreifer aufrufen. Solch ein Angriff läuft meistens direkt über ein Netzwerk und im Hauptspeicher ab, so daß kein scanbares Objekt im Dateisystem vorliegt (der Hauptspeicher wird höchstens beim Start des Scanners, unter Linux bisher jedoch nicht geprüft). Erst eine nachgeladene Datei könnte erkannt werden, dann ist die Malware jedoch schon aktiv. Man braucht andere Verfahren, um diesen Angriff abzuwenden, zum Beispiel einbruchsvermeidende Systeme (intrusion avoidance systems). Besser ist es jedoch, möglichst sauber programmierte Software zu verwenden.

## Auswertung im falschen Kontext:

Eine Anfrage enthält Sonderzeichen oder zusätzliche Angaben, die fälschlicherweise zu einer geänderten Bearbeitung der Anfrage führen.

*Beispiel:*

```
GET /cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
```

Diese Web-Server Anfrage sorgt dafür, daß in einer Telefonbuchanwendung der Variable "alias" der String

```
"x<Zeilenende>/bin/cat /etc/passwd"
```

zugewiesen wird. Wenn die Variable nun ungeschützt verwendet wird, so kann das Zeilenende als Ende des Befehls interpretiert werden, wodurch der Rest zu dem nächsten auszuführenden Befehl wird. Dieser Angriff unterscheidet sich für den Virusscanner nicht von einem Buffer Overflow, auch hier kann er nur dann eingreifen, wenn eine Datei nachgeladen wird.

## Unzureichende Prüfung der Zugriffsrechte, Ausnutzung einer Vertrauensstellung:

Eine bekannt unsichere Funktion ist nicht oder nicht an allen Stellen, an denen sie maliziöse Handlungen erlaubt, vor Aufrufen geschützt, oder ein Zugang wird über unsichere Methoden kontrolliert.

*Beispiel* IFRAME-Exploit: Zitat aus http://www.microsoft.com/technet/security/bulletin/MS99-042.asp [http://www.microsoft.com/technet/security/bulletin/MS99-042.asp]:

> "The Internet Explorer security model normally restricts the Document.ExecCommand() method to prevent it from taking inappropriate action on a user's computer. However, at least one of these restrictions is not present if the method is invoked on an IFRAME."

*Beispiel* vim: Der Texteditor Vim erlaubte es, für die verschiedensten Aufgaben eigene Funktionen zu schreiben und dabei auf Systemfunktionen zurückzugreifen. Diese Funktionen können auch im Text zugewiesen werden, so daß der Autor eines Textes bei den Lesern die Ausführung beliebigen Codes auslösen kann. Ein Beispiel für einen damit möglichen Wurm kann man unter http://www.securiteam.com/unixfocus/5RP0L0A8US.html [http://www.securiteam.com/unixfocus/5RP0L0A8US.html] nachlesen.

Beispiel Morris-Wurm: Wenn der Morris-Wurm (auch als Internet-Wurm bekannt, 1988) auf einem System aktiv war und bereits zu einem Benutzeraccount Zugang hatte, dann versuchte er, weitere Systeme über die Programme "rexec" oder "rsh" zu erreichen. Vertrauten diese Systeme oder dessen Benutzer dem befallenen System, so konnte der Wurm sich dort zur Ausführung bringen.

Diese Exploits können über die enthaltenen Strings leicht erkannt werden, so lange sie als Datei vorliegen und das Format richtig erkannt wird. Dies kann aber zumindest beim Morris-Wurm erst passieren, wenn die Schadroutine bereits läuft. Zu diesem Zeitpunkt könnten jedoch schon Nutzerdateien gelöscht werden sein. Zudem gibt es auch die Möglichkeit der nichtmaliziösen Nutzung der Schnittstellen, so daß man mit "false positives" rechnen muß, die Erkennung ist also sehr unzuverlässig. Wesentlich besser ist die Vermeidung von Software, die unnötige Funktionen und Schnittstellen anbietet oder auf den nötigen Schnittstellen inhärent unsicher ist.

# Schützt Open Source vor Trojanern?

Open Source gibt normalerweise die Sicherheit, daß eingebauter maliziöser Code entdeckt werden kann. Open-Source-Programme sollten daher weniger bewusst eingebaute Hintertüren als kommerzielle Software enthalten. Dies hilft jedoch nicht direkt gegen versehentlich eingebaute Fehler, kompromitierte Download-Server und trojanisierte Binaries. Besonders letztere werden unter Umständen nie entdeckt.

Man kann auch nicht darauf vertrauen, daß ein Trojaner im Sourcecode auftaucht und dort irgendwann gefunden wird, denn wie schon im Paper "Reflections on Trusting Trust" von Ken Thompson (erschienen in "Communication of the ACM" Vol. 27, No. 8, August 1984) beschrieben, kann man einen Compiler so gestalten, daß er beim Compilieren in beliebige Programme, zum Beispiel dem Login-Programm oder dem Compiler selbst, Hintertüren und Schadfunktionen einbauen kann. Zitat:

> "You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.) No amount of source-level verification or scrutiny will protect you from using untrusted code."

# Beispiele für Malware unter Linux

## Slapper

"Slapper" ist ein Internet-Wurm, der sich über den OpenSSL-Exploit in Apache/mod_ssl verbreitet. Die Verwundbarkeit wurde im August 2002 gefunden, der Wurm trat am Freitag, den 13. September auf. Verwundbar sind Web-Server, die SSL mit der OpenSSL-Bibilothek 0.96d, 0.97beta2 oder älter anbieten.

Der Exploit beginnt damit, daß verschiedene Rechner auf Port 80 gescannt werden. Meldet sich dort ein Apache-Server, so verbindet sich der Wurm zu Port 443 dieses Rechners und schickt den Exploit-String. Dieser bewirkt, daß eine UU-codierte Kopie des Wurms in /tmp/.uubugtraq gespeichert wird, woraus dann bugtraq.c extrahiert und mit gcc compiliert wird. Danach liegt der Wurm als ausführbare Datei vor und wird gestartet.

Neben der Verbreitungsroutine enthält der Wurm einen Client für ein Peer-to-Peer-Netzwerk, das dazu dient, DDoS-Attacken ("distributed denial of service") durchzuführen.

## Winux (alias Lindows, PEELF)

Der Winux-Virus ist ein Dateivirus, der sowohl ELF-Dateien (Linux-Programme) als auch EXE-Dateien im PE-Format (Windows-Programme) infizieren kann. Es handelt sich hierbei um ein Proof-of-Concept, der sich nicht weit verbreitet hat.

Dieser Virus verbleibt nicht im Speicher und führt auch keine Programme aus. Wenn eine mit diesem Virus infizierte Datei ausgeführt wird, sucht der Virus sowohl nach Dateien im ELF- und PE-Format. In PE-Dateien wird der Relokationsbereich überschrieben, es sei denn, er ist nicht groß genug, um den Virus aufzunehmen. In ELF-Dateien wird der Virus am Anfang des Programms eingefügt und dabei die Datei um 2748 Bytes vergrößert. Hierbei kann es vorkommen, daß das infizierte Programm nicht mehr korrekt funktioniert, die Reparatur ist jedoch trivial.

# Testergebnisse des VTC von Antimalware-Produkten unter Linux

## Die Antimalware-Tests des VTC

Programme, die den Benutzer vor selbstreplizierender und anderer Malware schützen, bezeichnet man als AntiMalware-Programme. Das Virus Test Center (VTC) führt regelmäßig Tests von Antimalware-Produkten unter verschiedenen Betriebssystemen durch (zur Zeit: Windows98, Windows2000, WindowsXP, Linux). An dieser Stelle werden nur die Testergebnisse unter Linux behandelt, alle anderen Testergebnisse finden sich auf der Webseite des VTC ( http://agn-www.informatik.uni-hamburg.de/vtc/ [http://agn-www.informatik.uni-hamburg.de/vtc/]). Getestet wurde unter Suse Linux 7.1.

## Der VTC-Test 2002-12

Der letzte Linux-Test wurde mit Malware-Datenbanken durchgeführt, die insgesamt 207.724 infizierte Dateien bzw. 38.681 verschiedene Viren/Malware enthielten. Diese waren auf folgende Datenbanken aufgeteilt:

**Tabelle 1. Verwendete Malware-Datenbanken**

| Malware-Art | Datenbank-Typ | Viren/Malware | Infizierte Dateien |
|---|---|---|---|
| File | Zoo-Viren | 21790 | 158747 |
| File | In-The-Wild-Viren | 50 | 442 |
| File | nicht-replizierende Malware | 8001 | 18278 |
| Makro | Zoo-Viren | 7306 | 25231 |
| Makro | In-The-Wild-Viren | 124 | 1337 |
| Makro | nicht-replizierende Malware | 450 | 747 |
| Skript | Zoo-Viren | 823 | 1574 |
| Skript | In-The-Wild-Viren | 20 | 122 |
| Skript | nicht-replizierende Malware | 117 | 202 |

Zusätzlich wurden den File-Zoo- und Makro-Zoo-Datenbanken noch einige hundert ausgesuchte nicht infizierte Dateien hinzugefügt, um stichprobenartig die Anfälligkeit der Produkte für "false positives" zu testen. Unter "false positive" versteht man Dateien, die irrtümlich als infiziert erkannt werden, obwohl sie gar nicht infiziert sind. Es wurde auch die Fähigkeit der Produkte getestet, Malware in komprimierter Form zu erkennen. Dazu wurden die Inhalte der In-The-Wild Datenbanken zusätzlich mit mehreren verbreiteten Kompressionsprogrammen komprimiert.

Folgende 8 Produkte wurden getestet:

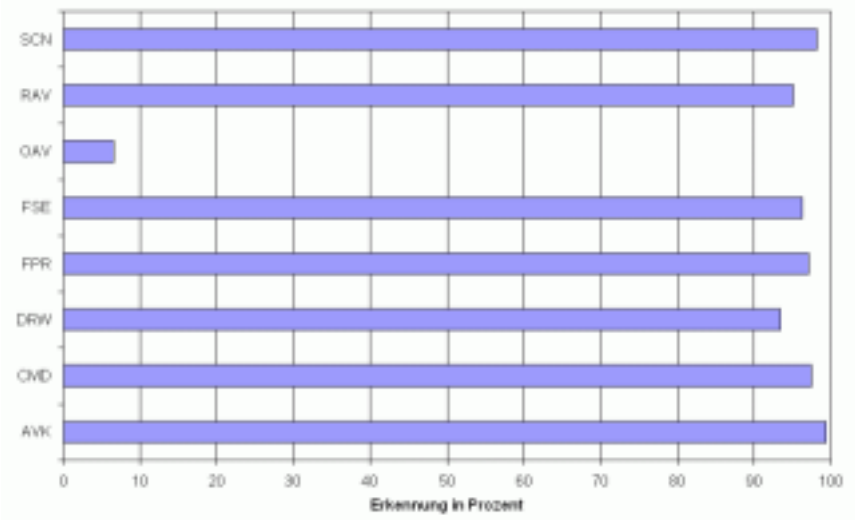
- *AVK*: GData AntiVirenKit 10 (entspricht Kaspersky Antivirus v3.0 beta 1.1)
- *CMD*: Command Antivirus v4.64.1
- *DRW*: Dr.Web for Linux v4.26
- *FPR*: F-Prot for Linux v3.11b
- *FSE*: F-Secure Antivirus v4.13 build 3360
- *OAV*: OpenAntivirus ScannerDaemon v0.2.0
- *RAV*: Reliable Antivirus v8.3.1
- *SCN*: McAfee Viruscan v4.16.0

# Testergebnisse

## Das beste Produkt ?

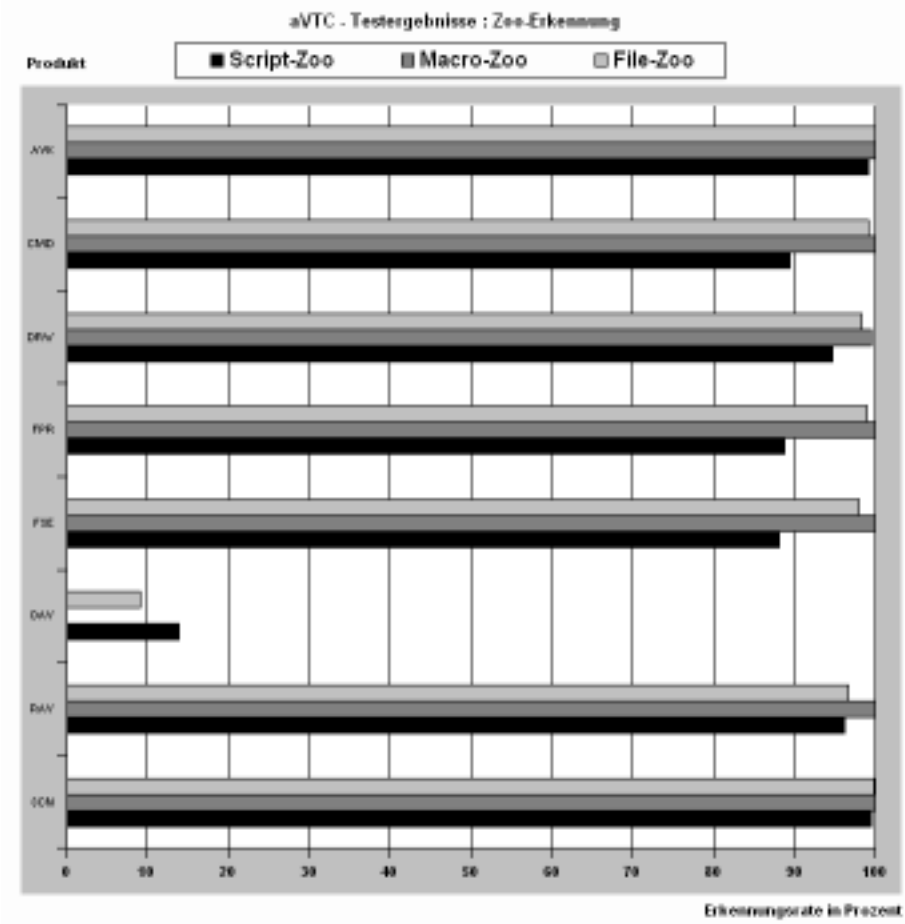
Die erste Frage, die meistens gestellt wird, ist, welches Produkt denn nun das beste sei. Diese Frage wird hier nicht beantwortet werden, denn die Antwort hängt einerseits sehr vom jeweiligen Einsatzbereich ab und setzt andererseits voraus, daß ein Produkt in allen relevanten Aspekten getestet wird. Da der Einsatzbereich von Antimalware-Produkten von Privatanwendern bis zu Großunternehmen (mit Tausenden von Clients) reicht, ist es nicht sinnvoll zu versuchen, allgemeine Aussagen dieser Art zu treffen. Die VTC-Tests beschränken sich auf die Erkennungsrate der Produkte; andere für den Einsatz durchaus relevante Aspekte (wie z. B. Geschwindigkeit, Bedienung, Remote-Administration) werden nicht berücksichtigt. Deshalb ist es auch nicht möglich, im Rahmen der VTC-Tests umfassende Bewertungen der Produkte vorzunehmen. Trotzdem werden auch in VTC-Tests Produkte bewertet, aber ausschließlich nach funktionalen Aspekten.

Die folgende Grafik zeigt, welcher Prozentsatz aller infizierten Dateien von den verschiedenen Produkten als infiziert gemeldet wurde.




Das sieht zwar aussagekräftig aus, man sollte sich jedoch nicht davon täuschen lassen, denn die Aussagekraft dieser Grafik ist eher gering. Z. B. wäre es schlichtweg falsch, hieraus eine Rangliste der besten Produkte zu erstellen (leider findet man solch oberflächliche Bewertungen oft). Es lassen sich höchstens eingeschränkte Aussagen treffen, z. B. das die Erkennungsrate von OAV deutlich schlechter ist als die aller anderen Produkte.

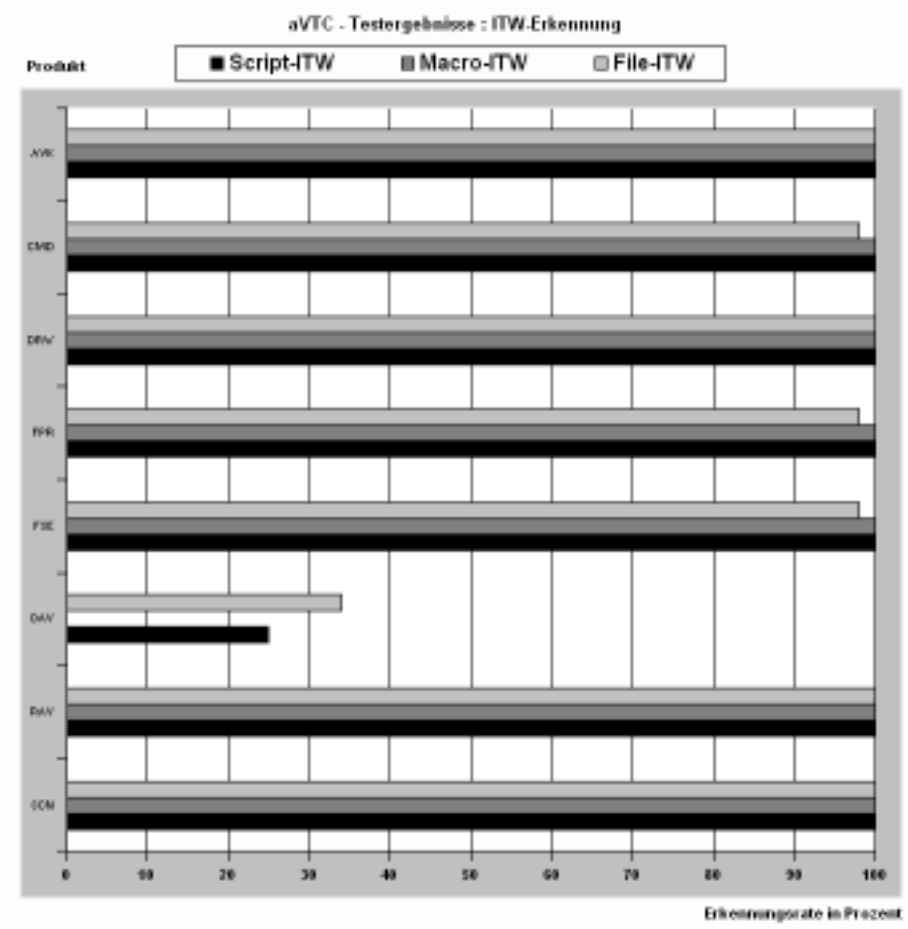
Einer der Gründe, warum diese Zahlen nicht ausreichend sind, liegt in der unterschiedlich guten Erkennung der verschiedenen Virenarten. Während einige Produkte z. B. File-Viren sehr gut erkennen aber mit Skript-Viren Probleme haben, sind andere überall mittelmäßig. Es gibt auch Produkte, die nur bestimmte Virenarten erkennen können, dies jedoch sehr präzise. Somit lassen sich die Gesamtergebnisse nicht unbedingt vergleichen. Und für ein konkreten Einsatz ist es wichtig zu wissen, daß ein Produkt in den für diesen Zweck wichtigen Bereichen gut funktioniert.




Wie man sieht, zeigen FPR, FSE und CMD (die intern alle die gleiche Scan-Engine verwenden) deutliche Schwächen bei der Erkennung von Skript-Viren, während RAV eher Probleme mit File-Viren hat (und dabei über 700 Viren nicht erkennt).

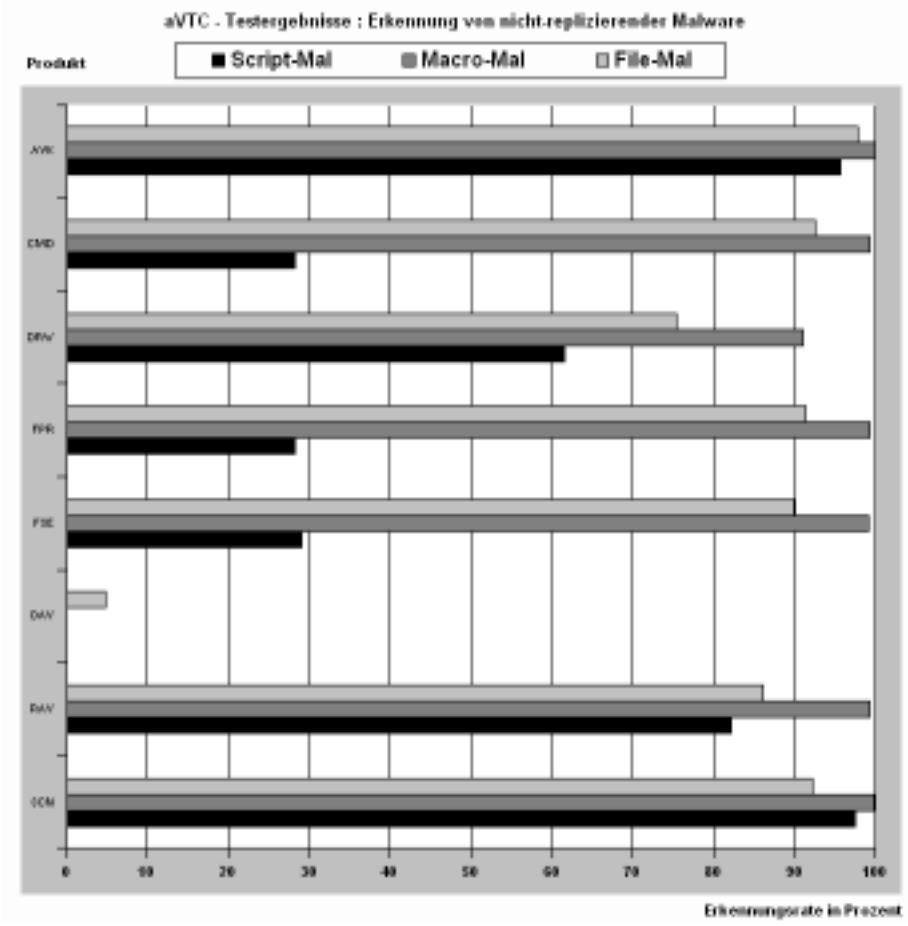
## Erkennung von ITW-Viren und nicht-replizierender Malware

Viel bedeutender ist jedoch, wie weit verbreitet die nicht erkannten Viren sind. Auch wenn ein Produkt insgesamt 99,9% aller Dateien erkennt, kann es trotzdem unbrauchbar sein, falls sich unter den nicht erkannten Dateien viele In-The-Wild-Viren befinden. Deshalb werden In-The-Wild-Viren in separaten Datenbanken getestet. Während z. B. 100 nicht gefundene Dateien in einer (riesigen) Zoo-Datenbank kaum einen Unterschied machen, ist dies in einer In-The-Wild-Datenbank nicht akzeptabel.




Insgesamt ergibt sich hier ein ganz anderes Bild als bei der vorhergehenden Betrachtung. Im Gegensatz zu den Zoo-Datenbanken ist hier bei FPR, FSE und CMD die Erkennung der Skript-Viren sogar besser als die der File-Viren (wobei allerdings die unterschiedliche Größe der Datenbanken berücksichtigt werden muß). Bereits die relativ geringe Anzahl nicht erkannter Dateien in den ITW-Datenbanken ist ähnlich schwerwiegend einzustufen wie die Schwächen von RAV auf den Zoo-Datenbanken.

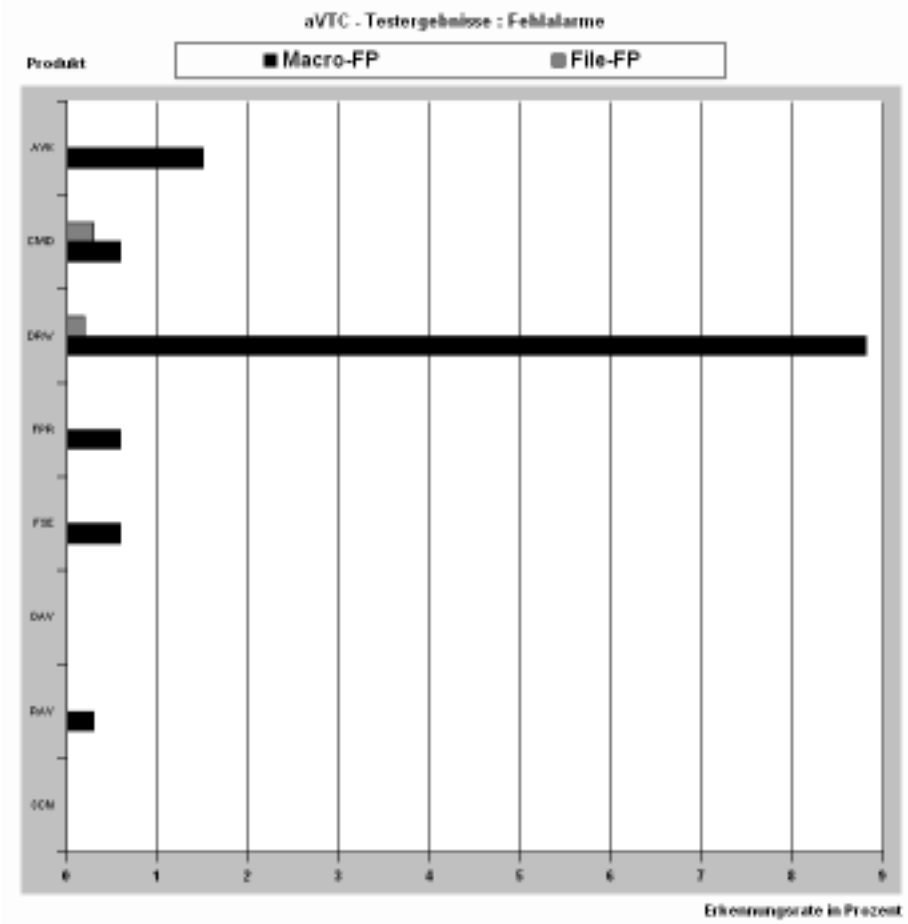
Wieder anders sieht es aus, wenn man die nicht-replizierende Malware betrachtet.




Hier schaffen es nur AVK und SCN, akzeptable Erkennungsraten zu erzielen.

## Falschmeldungen und Erkennung von Malware in komprimierten Dateien

Für den Praxiseinsatz ist es nicht nur wichtig, das ein Produkt möglichst jede Malware erkennt, sondern auch, das keine "sauberen" Dateien als Malware klassifiziert werden. Auch wenn es nicht möglich ist, dies repräsentativ zu testen, kann bereits ein Stichprobentest ausreichen, um Produktschwächen zu offenbaren. Dazu wurden vor allem Word und Excel-Dokumente verwendet, die harmlose Makros enthalten.




Wie man sieht, gibt es zumindest im Makroviren-Bereich bei vielen Produkten deutliche Probleme, zwischen schädlichen und harmlosen Makros zu unterscheiden. Lediglich SCN bietet hierbei eine zufriedenstellende Leistung. OAV hat zwar ebenfalls ein gutes Ergebnis, das allerdings angesichts der unzureichenden Erkennung von Makro-Viren nicht zu hoch bewertet werden darf.

Letztlich ist nicht unwichtig, wie gut die Produkte Malware in komprimierten Dateien, Mailbox-Archiven und anderen Kodierungen erkennen können. Dies wird wegen des großen Aufwands nur für wenige Kompressionsformate getestet (Zip, Arj, Cab, Lha, Rar1.5, Rar2.0, Rar3.0). Da die vollständigen Ergebnisse für diesen Beitrag viel zu umfangreich sind, wird hier nur gezeigt, welche Formate von den getesteten Produkten unterstützt werden (auch wenn diese Unterstützung nicht ganz zuverlässig ist):

- AVK: Zip, Arj, Cab, Lha, Rar1.5, Rar2.0
- CMD: Zip, Arj, Cab, Lha, Rar1.5, Rar2.0
- DRW: Zip, Arj, Cab, Rar1.5, Rar2.0
- FPR: Zip, Arj, Cab, Lha, Rar1.5, Rar2.0
- FSE: Zip, Arj, Lha
- OAV: -
- RAV: Zip, Arj, Cab, Rar1.5, Rar2.0, Rar3.0
- SCN: Zip, Arj, Cab, Lha, Rar1.5, Rar2.0

## Bewertung der Linux-Ergebnisse

Um zu einer differenzierteren Bewertung (unter den genannten Einschränkungen) zu gelangen, werden alle erwähnten (und weitere) Kategorien getrennt bewertet und mit unterschiedlicher Gewichtung zu einem Gesamtergebnis summiert. Dadurch ergibt sich (bei max. 24 erreichbaren Punkten) folgendes Ergebnis:

1. SCN (21 Pkt.)
2. AVK (19 Pkt.)
3. DRW, RAV (11 Pkt.)
4. CMD, FPR (10 Pkt.)
5. FSE (7 Pkt.)
6. OAV (0 Pkt.)

Dabei liegt OAV in der Erkennungsrate und der Funktionalität (keine Erkennung von Makroviren, keine Erkennung in komprimierten Dateien) mit deutlichem Abstand zurück und kann für einen tatsächlichen Einsatz nicht empfohlen werden. Dies war aber aufgrund des experimentellen Charakters des Projekts sowie des frühen Entwicklungsstands nicht anders zu erwarten.

Die vollständigen Bewertungskriterien sind im Testbericht auf der Webseite des VTC dokumentiert ([VTC 2002-12]).

# Referenzen


- [Brunnstein 1999]

  Klaus Brunnstein: "From AntiVirus to AntiMalware software and beyond: Another approach to the protection of customers from dysfunctional system behaviour", paper submitted to 22nd National Information Systems Security Conference, 1999

- [KittelTicak 2002]

  Martin Kittel, Mario Ticak: "Viren und Malware - Eine Einführung", Informationsheft des Virus Test Centers, Stand Januar 2003

- [VTC 2003]

  Homepage des Virus Test Centers, http://agn-www.informatik.uni-hamburg.de/vtc/

- [VTC 2002-12]

  Testreport zum VTC-Test 2002-12, ftp://agn-www.informatik.uni-hamburg.de/pub/texts/tests/pc-av/2002-12/

- [Wildlist 2003]

  Homepage von "The Wild Organization International", http://www.wildlist.org/