

RMS - ROSE_SWE's Malware Scanner

Ralph Roth

Table of Contents

1. Introduction	2
1.1. About Malware	2
1.2. What is Anti-virus Software?	2
2. What is RMS?	3
3. Compile Once - Run Everywhere	4
4. Installation	5
4.1. Windows 12 or Windows 11 26H2 Installation?	5
4.2. About/History/License	6
4.3. Different Operating System	7
4.4. Different Computing Platforms	7
4.5. Small Memory Version of RMS	8
4.6. Features	8
4.7. Build Schema	9
4.8. Known Bugs/To Do	9
4.9. Return/Error Codes	10
5. Command line Parameters	11
5.1. Notes on parameter usage	11
5.2. The environment variable RMS	11
5.3. Rollback of preset values	11
5.4. Options Move and Copy (Quarantine)	12
5.5. Exclude Option (-xdir=)	13
5.6. Option /CSV	13
5.7. Suggested parameters for virus scanner testing	13
6. Output when RMS found a virus	14
7. Stopping a running Scan?	15
8. Comparing RMS against F_Mirc	16
9. False Positives	17
10. License	18
11. Included Files	19
12. History	20
13. Copyright	23
Computer Viruses and Malware - A Short Overview	24
14. Malware	25
15. (Computer) Virus	26
15.1. Direct Action Viruses	26
15.2. (Computer) Boot Virus	26
15.3. Multipartite Virus	27
16. Trojan horses	28

17. Ransomware	29
17.1. Introduction to Ransomware	29
17.2. Origins of Ransomware	29
17.3. Evolution of Ransomware	29
17.4. Modern Ransomware Characteristics	30
17.5. Steps in a Ransomware Attack	30
17.6. Mitigation and Defense	30
17.7. Ransomware: Conclusion & Best Practices	30
18. Malicious Mining Software (Crypto-Miner)	32
19. Greyware	33
19.1. Scam	33
19.2. Adware	33
19.3. Spyware	34
19.4. Malvertising	34
20. Backdoors	35
21. Botnets	36
22. Macro viruses	37
23. Worms	38
24. Protestware	39
25. Stealth viruses	40
25.1. File stealth viruses	40
25.2. Full stealth viruses	40
25.3. Countermeasures against Stealth Viruses?	40
26. Encryption	41
26.1. What is a polymorphic virus?	41
27. What is an armored virus?	42
28. What is Phishing/Vishing?	43
29. Secure Boot/UEFI/Firmware Malware	44
29.1. UEFI Bootkits in General	45
29.2. Rootkits and Bootkits	45
30. Links / Pointers	46
31. Some very old (DOS) viruses that were very widespread in the past	47
31.1. CIH (Chernobyl)	47
31.2. Sasser	47
31.3. Melissa	47
31.4. Lehigh	47
31.5. Form	47
31.6. Elk Cloner	48
31.7. Ping Pong (Bouncing Ball)	48
31.8. The Brain Virus: The Birth of the Computer Virus Era	48
31.9. Cascade	49

31.10. Jerusalem.....	50
31.11. The Tequila Virus.....	51
31.12. Stoned.....	52
31.13. Michelangelo.....	53



Format: Plain text (ASCII-DOC), UTF8, Windows CR+LF, 80 chars max. per line

	—	\		\	\	/	—	
		—)			\	/	\
	—	<					—)
	—	\	\	\			—	/

Chapter 1. Introduction

RMS - ROSE SWE's Malware Scanner

RMS is a multi-platform command line antivirus program that detects known and unknown malware, Trojans and viruses.

Antivirus software, also known as antivirus or AV software, is a type of software designed to prevent, detect and remove malicious software, also known as malware, from a computer or network. Malware includes viruses, worms, Trojan horses, spyware, adware and other malicious programs that can damage or compromise a computer system. Antivirus software typically works by scanning a computer's hard drive and memory for known patterns of malicious code, and then either removing or quarantining the code so that it cannot cause any further damage. The software needs to be updated regularly to keep up with the new types of malware that are constantly being developed. The goal of antivirus software is to provide a first line of defense against malware threats and help keep a computer safe.

1.1. About Malware

The term "malware" refers to any type of malicious software, such as spyware, Trojans, ransomware, adware, or viruses, that can affect the performance or security of your computer. Most malware infections come from the Internet or from receiving emails. Infections can occur when you download and run files (click on attachments or downloads) - but also when you simply browse web pages.

A definition of malware you will find below.

1.2. What is Anti-virus Software?

Antivirus software protects your computer from known viruses, worms, Trojan horses and other unwanted intruders that can make your computer "sick". Computer viruses are very similar to biological viruses in that they attach themselves to programs or hosts and replicate continuously, although nowadays hosts take the form of USB drives, email attachments or files rather than living organisms. Viruses, worms, trojans and the like often perform malicious actions, corrupting or even deleting files or data, encrypting important files (→ ransomware), accessing personal data or using your computer to attack other computers, spreading to other computers or simply replicating and disrupting your system, making it unstable or more vulnerable to attack. A program that can detect such malware is called a virus or malware scanner.

Chapter 2. What is RMS?

RMS is both a very fast signature scanner and a heuristic-based scanner. RMS is a portable, cross-platform command line interface (CLI) scanner that can detect script viruses, worms, viruses and malware. This includes IRC worms (.INI), batch files (.BAT), Java scripts (.JS, .JSE), Visual Basic scripts (.VBS, .HTML, .SHS, .VBE), Trojans, backdoors, mail worms, spyware, key loggers, viruses (.COM, .EXE, .SHS, .SCR, etc.) and other script worms such as .CS and .WBT infectors. It can detect virus mutations; it looks for Trojans, fun and joke programs, DOS viruses, boot/MBR viruses, script viruses (VBS, HTML, etc.), IRC worms, malware and droppers. RMS is able to disassemble and decrypt files using a software emulator and entry point detection engine. This generic detection, known as heuristic analysis, is a technique that allows unknown viruses to be detected by looking for suspicious command sequences rather than relying on known signatures. RMS is therefore able to detect suspicious commands and anomalous sequences and detect unknown viruses! On the other hand, RMS can flag files that may contain malware as false positives. For example, F_Mirc (RMS) was able to detect the VBS.Love_Letter family of viruses using its heuristic scanning engines!

Chapter 3. Compile Once - Run Everywhere

Revolutionizing Portability with MPScan and RMS: Embracing the "Compile Once - Run Everywhere" Paradigm

The notion of "Compile Once - Run Everywhere" is both intriguing and essential in the realm of software development. This principle is centered around crafting portable software capable of seamless execution across diverse platforms, devoid of any need for modifications or runtime compilation.

In the dynamic landscape of software engineering, achieving cross-platform compatibility stands as a paramount objective. The ability to craft code once and witness it execute uniformly across various environments streamlines deployment, maintenance, and enhances overall user experience. Introducing MPScan and RMS embodying the spirit of CO-RE (Compile Once - Run Everywhere), these modern tools offer a contemporary approach to tackling this challenge head-on.

Chapter 4. Installation

RMS is delivered as a packed archive without an installation program. Simply unpack the package and copy the required files to a sub-directory of your choice. The package contains RMS for several different platforms, see "Different Operating System" below.



The virscan.* and rms.key are required files. Also one executable that fits our OS and architecture is required.

Example for Linux64 (Minimal Installation)

```
mkdir -p ~/tools/rms
cp virscan.* rms.key ~/tools/rms
cp LinuxAll/rms_86_x64 ~/tools/rms/rms
```

Example for Linux32 (Minimal Installation)

```
mkdir -p ~/tools/rms
cp virscan.* rms.key ~/tools/rms
cp LinuxAll/rms_i686 ~/tools/rms/rms
```

Example for Windows64 (Minimal Installation)

```
mkdir c:\tools
mkdir c:\tools\rms
copy virscan.* rms.key c:\tools\rms\
copy Windows\rms64.exe c:\tools\rms\rms.exe
```

Replace rms64.exe with rms32.exe if you are using Windows 32 bit (e.g. Win 8.1)



Run RMS without an option to get the basic command line options displayed. For the Windows platform, we provide a versatile batch file called windows.bat to scan your environment. This batch file automates the scanning process, making it easy for users to perform a comprehensive malware scan with minimal effort.

4.1. Windows 12 or Windows 11 26H2 Installation?

From a developer's perspective, **Smart App Control (SAC)** and the newly announced **Windows Baseline Security Mode** (part of Microsoft's 2026 "Secure Future Initiative") represent a fundamental shift in how Windows handles software execution. It essentially introduces a "walled garden" approach, similar to Apple's **Gatekeeper**, which aims to block all unverified or unsigned code by default.

4.1.1. The Core Model: Code Integrity by Default

This security model uses cloud-based AI and digital signatures to decide if an application is "safe." If a binary isn't signed by a Certificate Authority (CA) that Microsoft trusts, or if it doesn't have a high enough "reputation" in the cloud, it is blocked before it even starts. While this is a powerful defense against ransomware and "Living off the Land" attacks, it creates a massive friction point for anyone

shipping code.

4.1.2. The Impact on Developers and Organizations

- **Open-Source and Indie Devs:** This is a major blow. If you cannot justify the yearly cost of EV (Extended Validation) code-signing certificates, your software is effectively flagged as malware. This will stifle innovation and make it nearly impossible for small developers to distribute software on Windows without going through a costly and bureaucratic process.
- **Legacy Software:** For enterprise IT, the situation is particularly problematic. Unsigned legacy applications and internal tools will be blocked by default. Administrators face a massive workload managing manual exceptions, which risks creating a fragmented and hard-to-maintain IT environment.
- **The "Opt-Out" Loophole:** While the system allows for manual overrides and "opt-outs" to keep the OS usable, this remains a target. If an attacker gains sufficient privileges, they can theoretically manipulate these exceptions to bypass the protection entirely.

4.1.3. The Implementation Gap

Microsoft is planning a phased rollout, but concrete details remain scarce. Current industry analysis and Insider builds suggest a significant push starting with **Version 26H2** (expected late 2026) or potentially a future **Windows 12**.

What is most frustrating for developers is the lack of a clear roadmap. Microsoft hasn't provided specific **APIs or tools** to help us adapt. Instead, we are told to "watch the blogs," which is an incredibly thin basis for such a breaking change to the Windows ecosystem. For a platform historically defined by its "run anything" freedom, this shift toward a closed model is a significant departure.



So expect to add an exclamation to run RMS on Windows Platform enforcing SAC or WBSM!

4.2. About/History/License

Version 1.00 of RMS is based on technology from the following virus scanners from ROSE_SWE

- RHBVS
- VSP (VirScan Plus)
- MPScan
- f_mirc
- AntiLink package
- MemScan

RMS replaces F_Mirc 7.84 and therefore the usage of F_Mirc is **deprecated**. Also RMS has a better detection rate than F_Mirc. See also below "History" for more details.

RMS is Freeware for non-commercial usage by ROSE SWE. All Rights Reserved! See paragraph

4.3. Different Operating System

RMS is available for different operating system. When you start RMS a banner with the program version, build number and target platform is printed.

E.g.:

```
-----[ RMS/Linux-x86_64 2.7.3-9.136 - ROSE_SWE's Malware Scanner ]-----
Platform -----/
Program version -----/
Build -----/
```

Program version naming had changed to Semantic Versioning 2.0.0 (<https://semver.org/>)

4.4. Different Computing Platforms

The following platforms are currently provided/supported

- Win32+Win64 (32/64) - Windows console program, runs under Win95/98/ME, NT, 2000, XP, Win10 & Win11 etc., 32bit and 64bit Windows are supported. A Pentium CPU or higher is required. Long file names (LFN) supported on all Windows platforms.
- DOS32 - runs under DOS, Pentium required, a DPMS extender (v0.9/1.0) is required. Long file names supported under Windows 98, 2000 & XP and better. A DPMS (DOS Protected-Mode Interface) server must be present to run RMS. When running in raw DOS (ie: not a Windows 3x/9x/NT/XP/etc command-prompt), the CWSDPMS server must be on your PATH environment variable (note: FreeDOS comes with it already installed). See https://en.wikipedia.org/wiki/DOS_Protected_Mode_Interface As Windows only supports 64MB of DPMS the DOS32 version of RMS will not run on Windows. We plan to drop DOS32 support with the next releases. Requires at least 128MB of memory to run.
- Linux (32/64) - runs under 2.6.x and higher kernels. LFN under native Linux and mounted Win32/FAT/NTFS supported. Requires a Pentium. Fasted platform! The 64 bit Linux version needs a COREAVX2 or better CPU and a AVX2 or better co-processor! The 32 bit version requires a SSE2 co-processor and a Pentium-4 CPU or better. A modern GLIBC is required, details see below. WSL2 also works with the Linux64 binary!
- ARM 32/64 - Linux binaries for Aarch64 provided for feedback
- Support for additional platforms or OS versions (e.g. older GLIBC versions) on request, contingent upon a commercial requirement and the intent to purchase a commercial key file.



Long file names (LFN) supported on all platforms. All platforms requires at least 256 MB of free memory and a Pentium CPU if not otherwise noted.



RMS is optimized for terminal usage. RMS does dynamically adjust the output to the terminal width, but for best performance and display we recommend to use a terminal with at least 120 characters width. When running under Windows, we recommend to use the Windows Terminal (<https://aka.ms/terminal>) instead of the default CMD terminal (e.g. 80 characters) for better performance and display. The Windows Terminal also supports long file names (LFN) and Unicode characters, which can enhance the user experience when using RMS.

4.5. Small Memory Version of RMS

RMS now offers a small memory version optimized for DOS and all Raspberry Pi models, designed to operate efficiently on platforms with limited resources. The standard RMS version's memory footprint of approximately 200 MB is simply too large for these systems. This optimized version maintains the same high-quality malware detection capabilities for most threat types while significantly reducing the memory footprint.

4.6. Features

- **Reduced Memory Usage:** Optimized to run on systems with constrained resources, such as older DOS machines and various Raspberry Pi models. The standard version's 150 MB footprint is too large for these platforms, making this version essential.
- **Broad Platform Compatibility:** Functions across a wide range of hardware, from legacy DOS systems to the latest Raspberry Pi boards.
- **Maintained Detection Capabilities:** Preserves full detection efficiency for most malware types.

The small memory version offers a slight trade-off in detection capabilities:

- **Windows Malware Detection:** Slightly reduced efficiency compared to the full version.
- **Other Malware Types:** Maintains the same high level of detection as the full memory-intensive version.

This optimized version allows users with older or resource-limited systems to benefit from robust malware protection without overwhelming their hardware. It's particularly useful for maintaining security on legacy systems or when deploying RMS on Raspberry Pi devices for various projects.

RMS now offers a small memory version optimized for DOS and all Raspberry Pi models, designed to operate efficiently on platforms with limited resources. The standard RMS version's memory footprint of approximately 150 MB is simply too large for these systems. This optimized version maintains the same high-quality malware detection capabilities for most threat types while significantly reducing the memory footprint.

4.6.1. Linux:GLIBC



The Linux version of RMS 4.0 and higher is compiled as a static executable, which means it does not rely on shared libraries at runtime anymore. This design choice enhances portability and simplifies deployment across different Linux distributions, as it eliminates dependencies on specific library versions. However, it is important to note that the static executable still requires a compatible GLIBC version to run successfully. The required GLIBC version may vary depending on the specific build and platform, but generally falls within the range of GLIBC 2.31 to GLIBC 2.38. Users should ensure that their Linux environment meets this requirement to avoid compatibility issues when running RMS.



If you run into an error message like this, you need a more modern setup:

```
sle15sp5-jeos:/tmp # ./rms
./rms: /lib64/libc.so.6: version `GLIBC_2.34' not found (required by ./rms)

sle15sp5-jeos:/tmp # ldd -v rms
./rms: /lib64/libc.so.6: version `GLIBC_2.34' not found (required by ./rms)
linux-vdso.so.1 (0x00007fffe2eeb000)
libc.so.6 => /lib64/libc.so.6 (0x00007fea939be000)
/lib64/ld-linux-x86-64.so.2 (0x00007fea93bc2000)

Version information:
./rms:
  libc.so.6 (GLIBC_2.34) => not found          <== !!
  libc.so.6 (GLIBC_2.2.5) => /lib64/libc.so.6
/lib64/libc.so.6:
  ld-linux-x86-64.so.2 (GLIBC_2.3) => /lib64/ld-linux-x86-64.so.2
  ld-linux-x86-64.so.2 (GLIBC_PRIVATE) => /lib64/ld-linux-x86-64.so.2
```

How to determine your installed GLIBC version?

```
$ ldd --version
ldd (Ubuntu GLIBC 2.35-0ubuntu3.8) 2.35
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
```

4.7. Build Schema

The build number is a unique increasing number that is incremented with each build of RMS. A higher build number means a newer program version.

4.8. Known Bugs/To Do

- 1.) Done: The command line engine should now handle spaces, e.g. `-log="C:\Documents and Settings..."`
- 2.) Corrupted DOS EXE Entry Point versus a working Windows Entry Point may report

```
"Corrupted MS-DOS Header! Size=10.992, EP=157.686"
```

This means the EXE file is in general corrupted. This is very typical for malware.

3.) Directory depth (path length) greater than 255 characters is not yet fully implemented

4.) Scanning of Office documents is very limited. But this type of malware is now almost distinct.

4.9. Return/Error Codes

Return codes can be used within shell scripts (Linux) or batch files (Windows):

```
0      all OK, nothing found
4      One of the signatures files is damaged or the access is denied!
5      One or more viruses found
6      can not change to directory
7      online help (maybe wrong parameters)
8      file not found, e.g. virscan.*

11..18 Operating System Errors [Linux/DOS/Windows], please report it to
      ROSE SWE!
xx     Internal error, please report it to ROSE SWE!
```

Chapter 5. Command line Parameters

Run RMS with the option `-?` and `-??` to see all current supported command line arguments!

5.1. Notes on parameter usage

Customers familiar with the American or UNIX parameter syntax (minus sign) instead of the slash (/) can also use the minus sign (-) to start an option. Under Linux the use of the minus sign for command line arguments is mandatory!

Example: `-all` is equivalent to `/ALL` (applies for DOS and Windows)



There must be at least one blank between the individual arguments! The arguments are not case sensitive.

5.2. The environment variable RMS

Instead of always calling RMS with arguments, RMS can be controlled with a so-called environment variable. For example, enter the following at the DOS or CMD (Windows) prompt:

```
SET RMS=/cde -log
```

If you start RMS now, RMS reads all required arguments from the variable. This assumes that the RMS binary is named `RMS.exe`

On Linux do a:

```
export RMS=-log
```

5.3. Rollback of preset values

Sometimes it might be desired to reset already set options (i.e. set by `SET RMS=...`) This can simply be done by a minus sign following the option on the command line. With this action the option is being switched off.

For example, you have entered the following:

```
SET RMS=/all
```

Then start RMS with the following argument:

```
RMS c: /all-
```

In this case the command line option overrides the option set by the environment variable!

Command line always override environment options.

5.4. Options Move and Copy (Quarantine)

You can use the `-copy=Fullpath` or `-move=Fullpath` options to copy or move suspicious files to a quarantine folder "Fullpath" (Fullpath means e.g. under Linux `-copy=/home/ralph/found`). If that folder does not exist it will be created. It is important that you use the full path here. Files to be copied or moved must be smaller than the virtual available memory.

If `Fullpath` is not specified then the folder 'found_rms' in the user home directory is used and created.



This is a flat copy operation, no sub-directories are retained. Depending on the platform file permissions and timestamps of the copied file may be restored. It also does not check if the file already exists in the quarantine directory, in which case it is simply overwritten. Also you must ensure that you do not scan or cross the quarantine path itself. Therefore RMS adds the quarantine path to the automatically excluded list of directories.

Files will get a new name, currently:

```
Virusname__FileName__HexCounter.Extension
```

Example for the copy option under Linux:

```
rms ~/Zoo/hashed/ -copy=/home/ralph/shaldir
```

or

```
rms "/tmp/dir with spaces" "-copy=/home/ralph/f o u n d" -log
```

Examples of renamed files (SHA1 as file name) afterwards:

```
InfectionAsh.B__04C53F52__0070.com
InfectionAsh.B__f579c710952bcba6b2fae223fb9da8f294cfc67e__0048__006A.Bin
InfectionAvispa__04C53E59__000D.exe
InfectionBW__04C53EA3__0017.exe
InfectionBW__c2a00b5582c88d32211722be7113b877ee8fccf0__003C__0065.Bin
InfectionBW__e208cff69552d0b475edef11c2e9a6d49e53ba3b__0053__006F.Bin
InfectionEightTunes__04C53E78__000E.exe
InfectionEightTunes__cfff5e7d00c4f399a048c6d91291777176f37c05__003B__0064.Bin
InfectionFiledate__04C53E82__000F.exe
InfectionFiledate__04C53E83__0010.exe
InfectionLiberty.2857.O__b5f67d7637fabd94ead47efeb46bab9b43592329__0022__0056.Bin
InfectionLiberty.2857.O__c0594812f046bc87575bdd5d55c3dc5e7e4b6734__003E__0066.Bin
InfectionNovember__04C53F53__0071.exe
InfectionTrivial__04C53F3B__003D.exe
Type_File03.85C0+C0C0D5__11229a7c54ecd bfb319fb72aff6683dcd66e935e__0007.Bin
Type_File03.98FD+C0BFD5__c80640e8089e7be77b8d20fc65feef5b9c0b83c8__003A.Bin
Type_File03.A207+C0C0D5__3325b97949a43fc8f9bfe63cbe53dbc4b2a82502__0028.Bin
Type_File03.C6DA+C0C0D5__bf5133c058d1bba83bb89bd63590c87aba9d450a__001F.Bin
Type_FileOpen.DEB7-B82D__2e59520b2fcd67b53f2014d279d79831af1978b4__0011.Bin
```


5.5. Exclude Option (-xdir=)

With the command line option `-xdir="fullpath1,fullpath2"` it is possible to exclude (multiple) paths from scanning. On Linux, the pseudo/kernel file systems `/sys`, `/dev` and `/proc` are added to the exclude list by default. On Windows, "`C:\System Volume Information`" is excluded by default.

Multiple paths can be added, separated by a comma, but protected with `"x,y,z"`.

Linux example

```
$ rms ~/temp -log -copy -xdir=/home/ralph/temp/2
```

```
-----[ RMS/Linux-x86_64 2.61-5.125 - ROSE_SWE's Malware Scanner ]-----
(c) 1989, 1994-2026 by ROSE SWE, Ralph Roth, http://rose.rult.at

| Registered to   = Freeware User, for non-commercial and private use only!
| Keyfile        = /home/ralph/tp/exe/rms.key
| Logging to file = ./rms.log
| Quarantine path = /home/ralph/found_rms   Copy [OK],   Move [--]
| Excluded paths  = /dev,/home/ralph/found_rms,/home/ralph/temp/2,/proc,/sys
| Commandline     = |/home/ralph/temp|-log|-copy|-xdir=/home/ralph/temp/2|

-----[ /home/ralph/temp (Malware and viruses) ]-----
| Settings: Scan all files: [--] Delete files: [--]

| Scanning "/home/ralph/temp" for malware and viruses.
/home/ralph/temp/Eddie.1459._VBA_.exe           Infection: Rape.1639
/home/ralph/temp/Email-Worm.VBS.generic._VBA_.vbs Infection: Email.Worm.VBS.Generic
/home/ralph/temp/1/Spartak.1453._VBA_.exe       Infection: Spartak_II.1453
/home/ralph/temp/1/Tequila._VBA_.exe           Infection: Tequila.2468.A Dropper
[!] Note: Directory /home/ralph/temp/2 not scanned, found in exclude list=/home/ralph/temp/2

-----[ Statistics ]-----
```



On Linux we recommend also to exclude the mountpoints `/run` and `/media`

5.6. Option /CSV

This option logs found malware to a comma separated values file (CSV) in the current directory named `RMS-0000.csv`

If the file already exists, the next free filename is used by increasing the filename hexadecimal counter, e.g. `RMS-0001.csv`, `RMS-0002.csv` etc. until `RMS-ffff.csv`

5.7. Suggested parameters for virus scanner testing

For testing RMS against other virus scanners we suggest the following options:

```
RMS directory_to_scan -all -log=vtc_13062022.log -logall -logdel
```



Long file names in the command line are supported, spaces must be quoted!

Chapter 6. Output when RMS found a virus

The output of found malware is to the following schema:

```
/home/.../Koniec.432.A.exact.com   Unknown: Type_FileOpen.D5CA-5769
/home/.../Konkoor.1844.A.exe       Infection: Konkoor.1844.A
/home/.../HTML.Phish.BBR.html      Warning: Generic.JScript.Encrypted!
/home/.../Cracky.596.com           Note: Corrupted MS-DOS Header! Size=596, EP=25.937
```

- Unknown = Detected by a heuristic detection module. Chances are high that this could be a new virus!
- Infection = Detected virus/malware with the name of the malware
- Warning = Unusual file format or obscure file structure
- Note = File is damaged

Chapter 7. Stopping a running Scan?

Simply press the [ESC] key (Escape) to stop the program!

Chapter 8. Comparing RMS against F_Mirc

Scanning of a larger virus collection with unique virus strains

```
F_Mirc 7.84-5.633, 04.01.2022 - 34.682/66.550  
RMS 1.00-92, 04.01.2022      - 41.403/66.550
```

Chapter 9. False Positives

In Feb. 2024 the program MS Defender falsely finds the malware "Win32/Wacapew.C!ml" (or other variants) in the file rms64.exe or sometimes in the file rms32.exe the following malware.

- Program:Win32/Wacapew.C!ml
- Program:Win32/Cayunamer.A!ml
- Trojan:Win32/Wacatac.H!ml
- Trojan:Win32/Wacatac.C!ml

Seems MS Defender detect the executable compression as a false positive.

Chapter 10. License



RMS is distributed as AnyWare. This means that the author (ROSE SWE) has full copyright on the entire program package, utilities and documentation. Use of the program for personal use is free (just like freeware). **Commercial use of the program requires a commercial licence** (technically implemented by a licence key file). If you would like to use RMS in a commercial environment, please email us with the number of users you would like a site licence for to get a quote.

If you find this program useful and want to see it improved, just send me **anything** (hence the name AnyWare) that you think might be helpful, i.e. email, malware/viruses, bugs, reports or even money :-)

See ROSEBBS.TXT for contact details.

Chapter 11. Included Files

RMS_*.EXE RMS - virus scanner. Win32 (and Win64 bit) console version and 32 bit version for DOS (requires a DPMS host).

rms_elf32/64 Version for "modern" Linux (32bit or 64bit) distributions.

RMS.key License file for RMS. RMS is free for non commercial users. If you want to use RMS in a commercial environment, please send an email for a quote about how many users you need a site license.

virscan.wsm Signature database to detect VBS/JS viruses (Windows Scripting Malware)

virscan.irc Signature database to detect Batch/ISS/IRC-Worm viruses

virscan.trj Signature database to detect Trojans, viruses, Backdoor and malware

virscan.mpv Signature database to detect 'Multi Partite Viruses'

mbrpartdumper.sh For Linux only: Helper script to dump all MBR and Partitions to be scanned by RMS. Requires root rights.

DiskRead.com For DOS only: Small tool to dump complete floppy disks (all major formats) to be scanned by RMS. Additional similar tools are available from the ROSE SWE Freeware Tools (e.g. RFWT2202)

SCOBP Only for DOS: Displays the contents of MBR, boot and partition table. Advanced program with Int13 tracer and option to save or restore MBR and partition table.

Chapter 12. History



In chronological order

17.04.2026	4.1.8	Many internal enhancements for better virus detection. Added new viruses.
10.04.2026	4.00.1	Added more than 125.000 viruses. Linux executable now are static and don't require GLIBC anymore
29.03.2026	3.31.1	Added more than 3000 viruses. Non user visible enhancements.
25.03.2026	3.30.1	Sharpened the detection of polymorph viruses like the Natas family. Added new viruses. Non user visible enhancements.
14.02.2026	3.26.2	Added new viruses. Non user visible enhancements.
30.11.2025	3.20.5	Added new viruses. Non user visible enhancements.
17.10.2025	3.19.1	Added new viruses. Non user visible enhancements.
19.08.2025	3.17	Added new viruses. Enhanced heuristics.
18.06.2025	3.15.1	Added new viruses. Enhanced heuristics. Non user visible enhancements. Better Trivial and SillyComp detection.
02.05.2025	3.14.1	Added 30000 new viruses. Enhanced heuristics. Entry point detection enhanced
11.04.2025	3.13.1	Better Symlink handling added. New viruses added.
17.03.2025	3.11.2	Massive internal rewrite because after adding too many new viruses the DOS compiler bailed out :-)
14.02.2025	3.8.1	Added 7000 viruses. Added an new AVR heuristic search engine
28.01.2025	3.6.4	Small enhancements. New viruses added.
05.12.2024	3.6.1	New viruses, enhanced AVR modules....
28.11.2024	3.5.4	New viruses....
24.09.2024	3.5.1	Added a new AVR module. New viruses added.
04.09.2024	3.4.1	Consolidated and rebuild all the virus detection databases. New viruses added. Small enhancements
16.08.2024	3.3.2	New viruses added. Small internal enhancements
07.08.2024	3.2.1	Small enhancements. New viruses added. Updated documentation.
03.07.2024	3.1.3	Better entrypoint handling, better handling of false positives. New viruses added. Updated documentation.
17.06.2024	3.0.0	RMS now supports scanning multiple drives (Windows) and directories (Windows/Linux)


```

=====
10.06.2024    2.8.2    Massive updates on the RMS documentation
07.06.2024    2.8.1    Public release (added viruses). Consolidated signatures

12.03.2024    2.7.2    Public release (added viruses)

11.02.2024    2.7.2    Better detection. File handling rewritten.
                    Enhanced entry-point engine. New viruses added.
                    Internal enhancements.

Feb. 2024     2.6.4    Internal version.

02.02.2024 - 2.5.3-7817 Better Trojan detection. Enhanced entry-point
                    engine. New viruses added. Internal enhancements.

18.01.2024 - 2.4.1-7611 Re-consolidated virus signatures. New viruses.

07.11.2023 - 2.3.4-7166 Small enhancements and 21000+ viruses added.

31.10.2023 - 2.2.1-7061 Small enhancements and 10000+ viruses added.

19.10.2023 - 2.1.1-6912 Small enhancements and new viruses added.

13.09.2023 - 2.0.1-6678 Added the scan engine for DOS and Boot/MBR viruses
                    from MPScan to RMS

29.08.2023 - 1.77-6621 Small enhancements and new viruses added.

13.07.2023 - 1.76-6374 Small enhancements around the AVR-Trivial engine.
                    New viruses added.

28.06.2023 - 1.75-6284 Changes to the Trojan Signature database
                    Small enhancements and new viruses added.

09.06.2023 - 1.73-6009 Small enhancements and new viruses added.

12.05.2023 - 1.72-5771 Enhanced entry-point engine. New viruses added.
                    Internal enhancements.

08.04.2023 - 1.71-5735 Small enhancements and new viruses added.

03.03.2023 - 1.70-5173 Added the option (-xdir=) from MPScan to RMS. Internal
                    changes to the file/path parser. New viruses added

22.02.2023 - 1.61-5003 Statistics now include start and end time and total
                    time required. New viruses added

10.02.2023 - 1.60-4877 Added scan engine for *NIX malware

30.01.2023 - 1.50-4746 Rewritten IRC and Batch search engine. Rewritten
                    command-line engine. New viruses added

30.12.2022 - 1.24-4177 Added new viruses. -copy & -move without argument will
                    assume a default path

18.12.2022 - 1.23-3988 Massive enhancements for the modules AVR:Tiny, AVR:Mini,
                    AVR:Trivial and AVR:Silly to provide better detection.
                    Added 50.000 viruses. Enhancements (AVR-Modules)

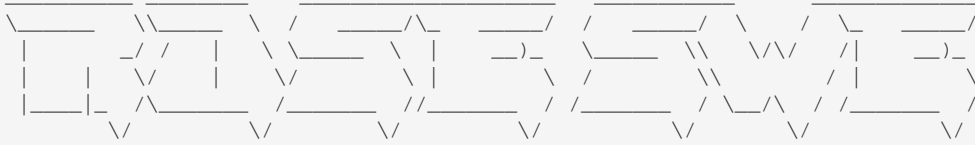
26.11.2022 - 1.21-3739 Added 25.000 viruses. No user visible enhancements
06.11.2022 - 1.20-3345 Added 35.000 viruses. Entry-point Engine improved
13.10.2022 - 1.19-3169 Added viruses. Small enhancements. +40k viruses
26.09.2022 - 1.18-2959 Added 20.000 viruses. Entry-point Engine improved
17.09.2022 - 1.17-2671 Added 50.000 viruses. No user visible enhancements
09.09.2022 - 1.16-2542 Added 6.000 viruses
10.07.2022 - 1.15-2311 Added 10.000 viruses
07.07.2022 - 1.14-2187 Added 12.000 viruses
01.07.2022 - 1.13-2059 Added 6.000 viruses.
12.06.2022 - 1.12-1924 Added 9.000 viruses. Enhanced entry point detection
                    engine
08.06.2022 - 1.10-1847 Added 5.000 viruses. Fixed 3 False Positives.
15.05.2022 - 1.09-1632 Added 14.000 viruses.
06.05.2022 - 1.08-1523 Added 13.000 viruses. Optimization (speed)
07.04.2022 - 1.07-1414 Added 10.000 viruses.
25.03.2022 - 1.06-1308 Added 40.000 viruses. Changed look and feel.
18.03.2022 - 1.05-1170 Enhancements around the scanning engine.

```

03.03.2022 - 1.04-1088	Enhanced entry point engine. 25.000 viruses added
14.02.2022 - 1.03-0834	Added 3.000 viruses
10.02.2022 - 1.03-0692	Added 30.000 viruses. Small internal optimizations
07.02.2022 - 1.02-0559	Added 80.000 viruses
22.01.2022 - 1.01.0401	Added detection engine for DOS debug scripts Added 70.000 viruses
19.01.2022 - 1.00.0329	Public release, added 210.000 viruses (on top)
03.01.2022 - 1.00.0084	Internal initial release.

Chapter 13. Copyright

©copyright by ROSE SWE (ALL RIGHTS RESERVED!)



ROSE SWE
Dipl.-Ing. Ralph Roth
<http://rose.rult.at>
rose_swe@hotmail.com
rose_swe@gmx.net

See ROSEBBS.TXT for
full address, FAX and PGP keys.

All Rights Reserved!

Computer Viruses and Malware - A Short Overview

A computer virus is a piece of code (software) that is installed on a computer either by a hacker, by another compromised computer (replication), malicious attachments/emails or a website (drive-by infection). It performs functions that the computer owner does not authorize and does not want.

Viruses are sometimes also referred to as malware. This is usually where they have adverse effects on the computer user, such as logging each keystroke (through a keylogger), audio recording or snapshots of each screen.

Such infection can lead to identity theft, endangerment of bank or purchase card data or loss of confidential data. It is more likely to occur on home computers that are normally not as security managed as corporate computers.

Chapter 14. Malware

Malware, or malicious software, is a generic term for a variety of malicious or intrusive software, including computer viruses, worms, Trojans, ransomware (ransoms), spyware, adware, scareware and other malicious programs. It can take the form of executable code, scripts, active content and other software. Malware is defined by its malicious intent, which violates the requirements of the computer user - and therefore does not include software that causes unintentional damage due to a defect.

Programs officially delivered by companies can be considered malware if they secretly violate the interests of the computer user.

Chapter 15. (Computer) Virus

A computer virus is a type of malicious software program ("malware") that, when executed, replicates itself by modifying other computer programs and appending or inserting its own code. When this replication succeeds, the affected programs are then said to be "infected" with a computer virus.

The term "virus" is also commonly, but erroneously, used to refer to other types of malware. "Malware" encompasses computer viruses along with many other forms of malicious software, such as computer "worms", ransomware, spyware, adware, Trojan horses, keyloggers, rootkits, bootkits, malicious Browser Helper Object (BHOs) and other malicious software. The majority of active malware threats are actually Trojan horse programs or computer worms rather than classic computer viruses.

Roughly you can distinguished between - Memory resident (fast) infecting viruses and - Direct action viruses

15.1. Direct Action Viruses

Direct action viruses are a type of malware that infect individual files on a computer, rather than the boot sector or Master Boot Record (MBR). They are called "direct action" viruses because they are executed each time a specific file is opened or executed, which allows the virus to infect other files on the computer.

Some of the simpler computer viruses do not actively manifest themselves in computer memory. The very first file infector viruses on the IBM PC, such as VirDEM and Vienna, belong to this category. As a rule, direct viruses do not spread quickly and are not easily spread in the wild.

Direct action viruses load themselves into computer memory with the host program. Once they have taken control, they search for new objects to infect by searching for new files. For this very reason, one of the most common types of computer viruses is the direct action infector. This type of virus can be created relatively easily by the attacker in binary or scripting languages on a variety of platforms.

Direct action viruses typically use a FindFirst, FindNext sequence to search for a number of victim applications to attack. Typically, such viruses only infect a few files when executed, but some viruses infect everything at once, enumerating all the directories for victims.

Direct action viruses typically spread by attaching themselves to executable files, such as .exe, .com, or .bat files. When an infected file is executed, the virus infects other files on the computer and may also cause other malicious activity.

15.2. (Computer) Boot Virus

Boot viruses are the oldest known computer viruses. They were the most common type of virus until 1995, but are now extinct. Today, there are almost no boot sector viruses anymore because BIOS and operating systems usually have well-functioning software or hardware protection.

A boot virus is a computer virus that becomes active when the computer starts (boots) before the operating system (DOS, Linux or Windows) is fully loaded. Boot sector viruses take advantage of the fact that the boot sector is always loaded first. On floppy disks, the virus is at least partially in the boot sector, so even floppy disks with no files on them can be infected. On hard disks, the virus infects the master boot record (MBR) or logical boot sector.

A boot sector virus infects the boot sector of floppy disks and the master boot record (MBR) of a hard drive. The boot sector is the first physical part of a floppy disk and is a sector (512 bytes). The boot sector is used by boot floppies to boot from the floppy. When a user tries to boot from an infected boot floppy, or leaves an infected floppy in the floppy drive when the computer starts up, the BIOS accesses this sector and executes it with the appropriate BIOS boot setting. The virus then attempts to infect the hard disk's MBR every time the computer is started. When an infected computer is started, the MBR, which is normally responsible for recognising the different partitions on the hard drive, is loaded. Once loaded, the virus remains in memory and monitors access to floppy disks. When a floppy disc is inserted into a computer infected with a boot sector virus, the virus infects the boot sector of the floppy disc.

Known boot viruses include the Form virus, Parity Boot and Boot-437.

15.3. Multipartite Virus

A multipartite virus is a computer virus that infects and spreads in multiple ways. The term was introduced to describe the first viruses that included DOS executable files and PC BIOS boot sector virus code, where both parts are viral themselves. Prior to the discovery of the first of these, viruses were categorized as either file infectors or boot infectors. Because of the multiple vectors for the spread of infection, these viruses could spread faster than a boot or file infector alone.

The first virus that infected COM files and boot sectors, Ghostball (more a dropper than a real multipartite virus), was discovered by Fridrik Skulason in October 1989. Another early example of a multi-part virus was Flip, Frodo, Delwin and Tequila. Tequila for example could infect both DOS EXE files and the MBR (master boot sector) of hard disks.

Chapter 16. Trojan horses

A Trojan horse is a program that does something undocumented which the programmer intended, but that users would not accept if they knew about it. By some definitions, a virus is a particular case of a Trojan horse, namely, one which is able to spread to other programs (i.e., it turns them into Trojans too). According to others, a virus that does not do any deliberate damage (other than merely replicating) is not a Trojan. Finally, despite the definitions, many people use the term "Trojan" to refer only to a non-replicating malicious program.

Chapter 17. Ransomware

Ransomware is a particularly invasive form of malware that hijacks a victim's data or device and holds it hostage (or makes false claims of illegal activity, pornography use, or suggests a system is already infected with viruses) until a sum of money is paid to secure its release.

17.1. Introduction to Ransomware

Ransomware is a type of malware that hijacks a victim's data or device, demanding a ransom—usually in cryptocurrency—for its release. Sometimes, it also makes false claims about illegal activity or pre-existing infections to pressure victims. Paying the ransom doesn't guarantee recovery, as attackers may refuse to decrypt the data or demand more payments. According to Statista, only 54 per cent of organisations regained access to their data or systems after the first payment in 2021. Paying the ransom also encourages attackers to continue their malicious activities. In addition, the vulnerability still exists and can be exploited by another criminal group.

Ransomware has become a global threat, with attacks growing more sophisticated. Understanding its history, evolution, and tactics is critical to combating it.

17.2. Origins of Ransomware

Ransomware first appeared in 1989 with the DOS-AIDS Trojan (PC Cyborg), created by Joseph L. Popp. Distributed via floppy disks labeled "AIDS Information," it used simple encryption and demanded \$189 for unlocking files.

Popp exploited public fear of the AIDS epidemic to spread the malware. Despite its basic design, many fell victim, incurring financial losses and data breaches. Popp's erratic behavior led to his arrest, but he was deemed mentally unfit to stand trial. This event set the stage for ransomware to evolve into a significant cybercrime.

17.3. Evolution of Ransomware

Ransomware has transformed into a sophisticated, profitable industry:

- **Targeted Attacks:** Criminals now target high-value victims like corporations, hospitals, and government agencies.
- **Stronger Encryption:** Modern ransomware uses advanced encryption, making decryption without a key nearly impossible.
- **Cryptocurrencies:** Bitcoin and Monero enable anonymous payments, fueling ransomware's growth.
- **New Tactics:** Techniques like double extortion (encrypting data and threatening leaks) and Ransomware-as-a-Service (RaaS) allow even non-experts to launch attacks.

Notable attacks like WannaCry and NotPetya caused massive global disruptions, targeting businesses and government agencies.

17.4. Modern Ransomware Characteristics

Ransomware attacks today are more targeted and damaging:

- **Anonymous Payments:** Cryptocurrencies make tracking transactions difficult.
- **Time Pressure:** Deadlines threaten data deletion or leaks, forcing quick victim responses.
- **Unreliable Decryption:** Many victims don't recover access, even after paying. In 2021, only 54% regained access after the first payment.
- **Encouraging Crime:** Paying ransoms perpetuates these attacks and leaves vulnerabilities open for future exploitation.

17.5. Steps in a Ransomware Attack

Ransomware attacks follow a common progression:

1. **Gaining Access:** Attackers infiltrate systems using phishing emails, malware downloads, or exploiting vulnerabilities.
2. **Spread:** Malware spreads within the network, either automatically or manually in targeted attacks.
3. **Hostage Taking:** Data is encrypted, and a ransom note demands payment for decryption or preventing data leaks.

17.6. Mitigation and Defense

Defending against ransomware requires proactive measures:

- Stay vigilant against phishing attempts and suspicious files.
- Regularly update systems and software to patch vulnerabilities.
- Use multi-factor authentication to secure accounts.
- Maintain offline backups of critical data.
- Invest in robust cybersecurity tools and training.

17.7. Ransomware: Conclusion & Best Practices

Ransomware is an ever-evolving threat. Staying informed and implementing strong defenses can help mitigate the risk and minimize impact.

Ransomware attacks can be extremely damaging and complex, and the timeframe for action is very limited. The best way to deal with them is to avoid them in the first place, and use mechanisms to prevent and mitigate their impact. The best way to prevent a malware attack is to follow good operational and security practices, such as

- Keep all software and operating systems up to date.
- Use anti-virus and anti-malware software on desktop systems.

- Regularly scan for vulnerabilities and comply with security policies, the key is to do this regularly.
- The best way to do this is to automate it so that it does not become a problem and can be integrated as part of the deployment process.
- Ensure that the software supply chain is properly secured. From an attacker's perspective, attacking the supply chain may be the easiest way to reach most, if not all, of an organisation's systems.
- Implement proactive measures and adopt a zero-trust policy. This applies to containers as well as traditional environments.
- Implement password validation best practices, such as avoiding common words and using long phrases that are easier for humans to remember but harder for machines to crack.
- Educate staff on basic security principles, such as being wary of suspicious emails, recognising suspicious links and managing data to avoid storing critical data in unsecured locations.
- Perform regular backups and always keep a cold backup in a separate physical location with no network access. Ensure that recovery procedures are tested regularly.
- Automate the provisioning of your infrastructure so that you can restore your systems quickly - time is money.
- Have a disaster recovery plan in place and ensure it is tested regularly.

Chapter 18. Malicious Mining Software (Crypto-Miner)

Starting in 2018 Malware authors are increasingly relying on malicious mining software. This year for the first time there have been more infections of this type than with ransomware. More and more online criminals seem to turn their backs on ransomware and rely on crypto-miner. They secretly dig crypto money on infected computers - Monero is particularly popular. This is obviously extremely lucrative, as the latest figures show.

Reasons for the turnaround? If a ransomware/Trojan strikes and encrypts data from victims, they usually have to pay a ransom in the form of bitcoins. This is an obstacle that not every victim can or will take. Crypto-miner, on the other hand, only needs to infect computers. Afterwards, they dig in secret without any sacrifices and make silently sure that they bring the authors big profits - and not too short when you look at the exploding prices of different crypto currencies.

Nowadays even commercial antivirus software tries to use the user computer when idle for mining. So this kind of software is both a malware scanner and malware itself :-(

Chapter 19. Greyware

Grayware (or greyware) is a general term sometimes used as a classification for applications that behave in a way that is annoying or unwanted, but less serious or problematic than malware. Grayware includes spyware, adware, dialers, joke programs, remote access tools and any other unwanted files and programs other than viruses that are designed to affect the performance of computers. The term has been in use since at least September 2004.

Grayware refers to applications or files that are not classified as viruses or Trojans, but can still affect the performance of computers on the user's network and pose significant security risks to the user's business. Grayware often performs a number of unwanted actions, such as annoying users with pop-up windows, tracking user habits and unnecessarily exposing the computer to attacks.

19.1. Scam

"Scam is a term used to describe a fraudulent scheme or deception in which someone is tricked into giving away money or personal information. Scams can take many different forms, such as phishing scams, investment scams, lottery scams and technical support scams, to name a few.

Phishing scams are attempts to trick people into revealing sensitive information, such as passwords or credit card numbers, by posing as a trustworthy entity. Investment scams persuade people to invest money in a bogus business or financial scheme with the promise of high returns. Lottery scams are messages informing people that they have won a large sum of money in a lottery, but asking them to pay a small fee or provide personal information to claim the prize. Tech support scams are attempts to trick people into paying for unnecessary computer support services by pretending to be from a reputable tech company.

Scammers often use persuasion and urgency to get people to hand over money or personal information. It is important to be wary of unsolicited messages or offers, and to independently verify the legitimacy of any request for personal information or money. You can protect yourself against fraud by being aware of common scams, being wary of unsolicited messages or offers, and never giving out personal information or money without verifying the identity of the recipient.

19.2. Adware

Adware is software that displays advertising banners in web browsers such as Chrome, Internet Explorer and Mozilla Firefox. Although it is not classified as malware, many users find adware invasive. Adware programs often have undesirable effects on a system, such as annoying pop-up ads and general degradation of network connection or system performance. Adware programs are usually installed as separate programs bundled with certain free software from websites. Many users inadvertently agree to install adware by accepting the End User License Agreement (EULA) of the free software. Adware is also often installed together with spyware programs. Both programs benefit from each other's features - spyware programs profile users' Internet behavior, while adware programs display targeted advertisements that correspond to the collected user profile.

19.3. Spyware

Spyware is a type of computer virus that hides on your computer or mobile device, records your private data and sends that information back to whoever created it or monitors it. The tricky thing about spyware, and what separates it from the growing threat of ransomware is the fact that, spyware is designed to both install discretely and operate silently in the background.

Spyware is software that installs components on a computer to record browsing habits (primarily for marketing purposes). Spyware sends this information to its creator or to other interested parties when the computer is online. Spyware is often downloaded along with other components that are referred to as "free downloads" or "freeware" without informing the user about their existence or asking for permission to install them. The information that spyware components collect can include user's keystrokes (keylogging), which means that private information such as login names, passwords and credit card numbers can be stolen. Spyware collects data, such as account names, passwords, credit card numbers and other confidential information, and transmits it to third parties.

19.4. Malvertising

Malvertising, a combination of "malicious" and "advertising", refers to the distribution of malware through online advertising. Cybercriminals use legitimate ad networks to place malicious ads on trusted websites. Users can become infected by clicking on or even just viewing these ads.

How it works

- Inclusion in ad networks: Malicious ads are injected into legitimate networks.
- Spread on websites: These ads appear on popular websites.
- Distribution of malware: Clicking on or viewing the ad can lead to malware infection.

Types of malware

- Ransomware: Encrypts files and holds them for ransom.
- Spyware: Steals confidential information.
- Adware: Displays unwanted advertisements.
- Trojans: Allow unauthorised access to systems.

Protective measures

- Ad blockers: Prevent malicious ads from loading.
- Updated software: Reduces risks from known vulnerabilities.
- Security software: Antivirus programs and firewalls provide protection.
- Be careful what you click: Avoid clicking on suspicious ads.

Chapter 20. Backdoors

A point of access to a hidden program/system. Backdoors are usually intentionally created by a programmer for debugging or maintenance purposes, but if compromised, they can pose a security risk to unauthorized users or software, allowing access and causing damage. Malware often installs Backdoors on compromised systems!

Chapter 21. Botnets

A bot is a programs that run automated tasks over the Internet. Botnets are collection of bots that run autonomously and automatically. Typically they perform repetitive tasks at a much higher rate than a human is capable of. They can be used for malicious purposes, such as denial of service attacks or infecting other computers. An infected computer is called a bot or zombie.

Chapter 22. Macro viruses

A macro is a piece of code that can be embedded in a data file. A macro virus is thus a virus that exists as a macro attached to a data file. In most respects, macro viruses are like all other viruses. The main difference is that they are attached to data files (i.e., documents) rather than executable programs. Document-based viruses are, and will likely continue to be, more prevalent than any other type of virus.

Chapter 23. Worms

Worms are very similar to viruses in that they are computer programs that replicate functional copies of themselves (usually to other computer systems via network connections) and often, but not always, contain some functionality that will interfere with the normal use of a computer or a program. Unlike viruses, however, worms exist as separate entities; they do not attach themselves to other files or programs. Because of their similarity to viruses, worms also are often referred to as viruses.

Chapter 24. Protestware

In March 2022, a developer of **node-ipc** was caught adding malicious code to the popular open source package that deleted files on computers in Russia and Belarus. This was part of a protest that angered many users and raised concerns about the security of free and open source software. The node-ipc update is just one example of what some researchers call **protestware**. Most protest programs related to the Russian invasion of Ukraine simply display anti-war and pro-Ukrainian messages. However, in at least one project, virus-like code was added that aimed to cripple computers in Russia and Belarus. This led to criticism and accusations of causing collateral damage. But there are also examples of protest in the open source scene. Observers of the scene so far found about two dozen software projects that inserted "code against war."

Open-source programs can be modified and viewed by anyone, making them more transparent - and, at least in this case, more vulnerable to sabotage. The protestware event highlights some of the risks that arise when legions of volunteer developers create the code that is critical to running hundreds or thousands of other applications. Some open source software automatically downloads and integrates new versions, and even for those that don't, the vast amount of code often makes manual review infeasible. This means that an update by a single person can mess up an untold number of downstream applications. In that sense, this can be considered a "game changer."

Russia's largest bank has asked its customers to stop updating its software because it is under threat from "protestware". In response to the threat, Russian state-owned bank Sberbank even advised its Russian customers to manually check the source code of the software they need - a security measure that is not feasible for most users. "We urge users to stop updating software and developers to tighten monitoring when using external code," Sberbank said, according to Russian media and cybersecurity firms.

Chapter 25. Stealth viruses

What is a stealth virus? A stealth virus is one that, while active, hides the modifications it has made to files or boot records. It usually achieves this by monitoring the system functions used to read files or sectors from storage media and forging the results of calls to such functions. This means that programs that try to read infected files or sectors see the original, uninfected form instead of the actual, infected form. Thus the virus's modifications may go undetected by antivirus programs. However, in order to do this, the virus must be resident in memory when the antivirus program is executed, and the antivirus program may be able to detect its presence.

The very first DOS virus, Brain, a boot-sector infector for example monitored physical disk input/output and redirected any attempt to read a Brain-infected boot sector to the disk area where the original boot sector was stored.

25.1. File stealth viruses

In addition to hiding the boot information, DOS file stealth viruses attack .com and .exe files when opened or copied, and hide the file size changes from the DIR command. The major problem arises when you try to use the CHKDSK/F command and there appears to be a difference in the reported files size and the apparent size. CHKDSK assumes this is the result of some cross-linked files and attempts to repair the damage. The result is the destruction of the files involved.

25.2. Full stealth viruses

With a full stealth virus, all normal calls to file locations are cached, while the virus subtracts its own length so that the system appears clean.

25.3. Countermeasures against Stealth Viruses?

You need a clean system so that no virus is present to distort the results of system status checks. Thus you should start the system from a trusted, clean, bootable diskette before you attempt any virus checking.

Chapter 26. Encryption

One method of evading malware detection is to use simple encryption to encipher (encode) the body of the malware, leaving only the encryption module and a static cryptographic key in clear text which does not change from one infection to the next.

26.1. What is a polymorphic virus?

A polymorphic virus is one that produces varied but operational copies of itself. This strategy assumes that virus scanners will not be able to detect all instances of the virus. One method of evading scan-string driven virus detectors is self-encryption with a variable key. Polymorphic code was the first technique that posed a serious threat to virus scanners.

More sophisticated polymorphic viruses (e.g., V2P6) vary the sequences of instructions in their variants by interspersing the decryption instructions with "noise" instructions (e.g., a No Operation instruction (NOP), or an instruction to load a currently unused register with an arbitrary value), by interchanging mutually independent instructions, or even by using various instruction sequences with identical net effects (e.g., Subtract A from A, and Move 0 to A). A simple-minded, scan-string based virus scanner would not be able to reliably identify all variants of this sort of virus; in this case, a sophisticated scanning engine has to be constructed after thorough research into the particular virus.

One of the most sophisticated forms of polymorphism used so far is the Mutation Engine (MtE) or the Trident Polymorph Engine (TPE), which comes in the form of an object module. With such mutation engines, any virus can be made polymorphic by adding certain (API) calls to its assembler source code and linking to the mutation-engine and provided random-number generator modules.

The advent of polymorphic viruses has rendered virus scanning an increasingly difficult and expensive endeavor; adding more and more search strings to simple scanners will not adequately deal with these viruses.

Chapter 27. What is an armored virus?

Armored viruses use special tricks to make the tracing, disassembling, and understanding of their code more difficult. A good example is the Whale virus. An armored virus uses various techniques to evade detection, such as encrypting its code, obfuscating its code, and using anti-debugging and anti-tampering methods.

Armored viruses pose a serious threat because they can be used to perform malicious activities such as stealing sensitive information, altering or corrupting data, and slowing performance without being detected. They can also be used as part of more complex attacks, such as advanced persistent threats (APTs), to maintain a foothold on a target network over an extended period of time.

Chapter 28. What is Phishing/Vishing?

Phishing and vishing are types of scams used to steal sensitive information such as passwords, credit card numbers and other personal data.

Phishing is a type of scam that tricks people into providing sensitive information through fake emails or websites that appear to be from a reputable source, such as a bank or a well-known company. The goal of phishing scams is to trick people into revealing personal information, such as passwords or credit card numbers, by posing as a trustworthy entity.

Vishing, short for voice phishing, is a type of phishing scam where people are tricked into revealing sensitive information over the phone. In vishing scams, scammers often pretend to be from a bank, government agency or technology company and use persuasive techniques to get people to reveal sensitive information.

Both phishing and vishing scams are becoming increasingly sophisticated and it is important to be wary of unsolicited emails or phone calls. To protect yourself from these types of scams, never provide sensitive information in response to an unsolicited request and independently verify the identity of the recipient before providing any personal information.

Chapter 29. Secure Boot/UEFI/Firmware Malware

In 2012, an industry-wide coalition of hardware and software makers adopted Secure Boot as a crucial defense mechanism against a growing and sophisticated security threat: malware targeting the system's firmware, specifically the BIOS. The BIOS, or Basic Input/Output System, is the firmware responsible for initializing hardware components and loading the operating system every time a computer is powered on. Malware that infects the BIOS poses an especially insidious threat because it can establish a foothold deep within the system, evading traditional detection methods and persisting through operating system reinstalls. Once entrenched, such malware can execute before the operating system and any security software, making it extremely difficult to detect and remove.

The threat of BIOS-dwelling malware had long been considered theoretical, heightened by the creation of the ICLord BIOS rootkit by a Chinese researcher in 2007. ICLord was a proof-of-concept rootkit, a type of malware designed to gain and maintain privileged access to a system while remaining hidden from standard security measures. This proof of concept not only demonstrated the feasibility of BIOS rootkits but also underscored their potential power. While ICLord remained a theoretical threat, it set the stage for the realization of more dangerous malware.

In 2011, the landscape of firmware security changed dramatically with the discovery of Mebromi, the first-known BIOS rootkit to be observed in the wild. Mebromi marked the transition from theoretical to actual threat, underscoring the vulnerability of BIOS firmware to sophisticated attacks. Mebromi was capable of infecting the BIOS, overwriting it with malicious code, and maintaining persistence even after the operating system was reinstalled—a stark reminder of the critical need for stronger security measures.

Recognizing the severity of the threat posed by Mebromi and other potential firmware attacks, the architects of Secure Boot developed a sophisticated security framework aimed at fortifying the pre-boot environment. Integrated into the Unified Extensible Firmware Interface (UEFI)—which was designed to replace the aging BIOS system—Secure Boot leverages public-key cryptography to verify the integrity and authenticity of firmware and software components before they are loaded. Specifically, Secure Boot only allows the execution of code that is signed with a recognized and trusted digital signature, effectively preventing unauthorized or malicious code from compromising the system at such an early and vulnerable stage.

To this day, Secure Boot is regarded by security experts and organizations—including Microsoft and the US National Security Agency—as a foundational element in protecting devices, particularly in critical environments such as industrial control systems and enterprise networks. Its role in establishing a chain of trust from the hardware through to the operating system is considered essential in defending against the sophisticated and persistent malware threats that continue to evolve. The adoption of Secure Boot represents a significant milestone in the ongoing effort to enhance cybersecurity and protect against the increasingly complex landscape of malware attacks.

In 2024 Secure Boot is considered to be broken because cryptographic keys to protect secure boot were leaked (PKfail)

29.1. UEFI Bootkits in General

UEFI bootkits are a type of malware that hides in the UEFI firmware of a computer. UEFI is a software program that controls the boot process of boot process and execute malicious code before the operating system even starts.

- UEFI bootkits are difficult to detect and remove because they hide in the UEFI firmware.
- UEFI bootkits can be used to install persistent malware that remains active even after the operating system is reinstalled.
- UEFI bootkits can be used to steal sensitive data, such as passwords and bank details.

Countermeasures:

- Enable UEFI Secure Boot to prevent unauthorized code from running during the boot process.
- Keep your system firmware and operating system up to date to patch vulnerabilities.
- Use reliable antivirus software to detect and remove UEFI bootkits.

29.2. Rootkits and Bootkits

Rootkits and bootkits are sophisticated forms of malware that compromise system integrity by gaining unauthorized control over devices. Operating at low levels within an operating system or firmware, they are particularly challenging to detect and remove. This document outlines their functionality, types, techniques, and strategies for mitigation.

- Rootkits are tools that allow attackers to gain and maintain privileged access to a system while concealing their presence.
- Bootkits are a specific type of rootkit that infects the boot process, compromising systems before the operating system loads.

Both rootkits and bootkits pose significant security risks due to their stealthy operations and privileged access capabilities. Both rootkits and bootkits often conceal their presence by altering system internals. Traditional antivirus solutions may struggle to detect them due to their low-level operations.

Rootkits and bootkits represent significant threats to modern computing environments due to their stealthy nature and privileged access capabilities. Effective detection and mitigation require robust security practices, including hardware-based protections, monitoring tools, and proactive patching. Understanding these threats is essential for safeguarding critical systems and sensitive data.

Chapter 30. Links / Pointers

https://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms

Chapter 31. Some very old (DOS) viruses that were very widespread in the past

MS-DOS viruses were particularly prevalent during the early days of personal computing, exploiting the relatively limited security measures of that era. Here are some notable MS-DOS viruses:

31.1. CIH (Chernobyl)

- **Origin:** Taiwan, 1998
- **Type:** File virus
- **Description:** Known for its destructive payload that triggered on April 26, it could overwrite critical parts of the BIOS, rendering computers unbootable. Although not an MS-DOS virus, it affected Windows 9x systems that were based on DOS.

31.2. Sasser

- **Origin:** Germany, 2004
- **Type:** Worm
- **Description:** While not strictly an MS-DOS virus, Sasser exploited vulnerabilities in Windows systems to spread. It caused widespread disruptions in the early 2000s.

31.3. Melissa

- **Origin:** USA, 1999
- **Type:** Macro virus
- **Description:** This virus spread through Microsoft Word documents and email, causing substantial email server disruptions. While primarily a macro virus, its impact was significant across various Windows environments.

31.4. Lehigh

- **Origin:** USA, 1987
- **Type:** Boot sector virus
- **Description:** One of the early boot sector viruses, it specifically targeted the master boot record and could corrupt the entire hard disk.

31.5. Form

- **Origin:** Probably Swiss, early 1990s
- **Type:** Boot sector virus

- **Description:** This virus became widely known for its payload that activated on the 18th of each month, causing the keyboard to behave erratically.

31.6. Elk Cloner

- **Origin:** USA, 1982
- **Type:** Boot sector virus (on Apple II, not MS-DOS, but historically significant)
- **Description:** One of the earliest known viruses, it displayed a poem on the 50th boot of an infected system. Although not an MS-DOS virus, it is significant in the history of computer viruses.

31.7. Ping Pong (Bouncing Ball)

- **Origin:** Italy, 1988
- **Type:** Boot sector virus
- **Description:** This virus caused a bouncing ball effect on the screen and infected the boot sector of floppy disks.

These viruses highlight the diverse strategies employed by malware developers in the MS-DOS era, from boot sector infections to file-based and polymorphic techniques, illustrating the early challenges of computer security.

31.8. The Brain Virus: The Birth of the Computer Virus Era

The Brain virus, often cited as the first IBM PC-compatible virus, marked a significant milestone in the history of computer security. Created in 1986 by two brothers in Pakistan, it initiated an era of growing cybersecurity threats and responses.

The Brain virus was developed by Basit and Amjad Farooq Alvi, who operated a computer store in Lahore, Pakistan. Frustrated with the piracy of their medical software, they created the virus as a form of copy protection, embedding their contact information within the code to raise awareness about piracy.

The Brain virus serves as a historical landmark in cybersecurity, highlighting the early challenges of digital security and the unintended consequences of technological interventions. It underscored the need for ongoing vigilance and education in the evolving landscape of cybersecurity.

31.8.1. Technical Details

The Brain virus is a boot sector virus, infecting the boot sector of storage media like floppy disks. It becomes resident in memory when the computer boots from an infected disk and then infects any clean disks accessed by the system. It displays a message with the authors' names and contact details, an unusual feature among viruses.

Infection Mechanism:

- **Boot Process:** Loads into memory during boot from an infected disk.
- **Replication:** Spreads to other disks accessed by the infected computer.
- **Infected Message:** Displays a message with the creators' contact information.

Attributes:

- **Memory Resident:** Remains active until the computer is turned off.
- **Stealth Techniques:** Hides its presence by intercepting system calls.
- **Non-Destructive:** Does not delete or corrupt files, focusing instead on spreading and delivering a message.

31.8.2. Impact and Spread

The Brain virus's impact was significant, as it was the first widely recognized PC boot virus. It spread rapidly through the common practice of sharing floppy disks, reaching users worldwide by the late 1980s.

Geographical Spread:

- **Local to Global:** Initially spread within Pakistan, then globally through shared software.

Economic and Social Impact:

- **Awareness and Fear:** Raised awareness about computer security and vulnerabilities.
- **Economic Consequences:** Prompted investments in antivirus software and improved security practices.

31.9. Cascade

Cascade virus (also known as Herbstlaub in Germany) is a well-known DOS computer virus that is a memory-resident virus written in assembly language. Cascade was widely spread in the 1980s and early 1990s. It infected DOS .COM files and caused the text on the screen to cascade down and form a pile at the bottom of the screen. It was notable for the fact that it used an encryption algorithm to avoid detection. However, it could be seen that the size of the infected files increased by 1701 or 1704 bytes. In response, IBM developed its own anti-virus software.

When a file infected with Cascade is introduced into a system and executed, the virus checks the BIOS for the string "COPR. IBM", an IBM copyright notice in the BIOS. If it finds the string, it tries to stop there, but fails, and the virus becomes memory resident. Every time a .COM file is executed, the virus starts infecting it. It replaces the first three bytes of the new host file with code that references the virus code. The virus places the original first three bytes of the host into its own code.

Cascade's payload is executed when an infected file is executed between October 1 and December 31, 1988. It causes characters on a DOS screen to randomly drop down in a pile of numbers and letters. Variants can also cause noise.

The virus has a number of variants. Cascade-17Y4, which is believed to have originated in

Yugoslavia, is almost identical to the most common 1704-byte variant. One byte has been changed, probably by a random "mutation". However, this has resulted in a "bug" in the virus. Another mutated variant is also known - it infects the same file over and over again.

31.10. Jerusalem

Jerusalem is a DOS virus which was first detected in Jerusalem in October 1987. Its origin is uncertain, as it was thought to have originated in Israel, but evidence from 1991 suggests that it may have originated in Italy. As of 1993, Jerusalem was still in the wild and many variants had been created. The last reported case of Jerusalem was in 1995, almost 8 years after its discovery. The virus has gone by many names, some referring to its possible origin and its Friday the 13th payload date. Jerusalem was initially very common (for a virus at the time) and spawned a large number of variants. However, since the advent of Windows, these DOS interrupts are no longer used, so Jerusalem and its variants have become obsolete.

Once infected, the Jerusalem virus becomes memory resident (using 2kb of memory) and then infects every executable file that is run, except for COMMAND.COM. COM files grow by 1,813 bytes when infected by Jerusalem and are not re-infected. EXE files grow between 1,808 and 1,823 bytes each time they are infected. The virus re-infects .EXE files each time they are loaded until they are too large to load into memory. Some .EXE files are infected but do not grow because multiple overlays follow the real .EXE file in the same file. Sometimes .EXE files are infected by mistake, so that the programme fails to run when it is run.

The virus code itself hooks into interrupt processing and other low level DOS services. For example, code in the virus suppresses the printing of console messages if, for example, the virus is not able to infect a file on a read-only device such as a floppy disk. One of the clues that a computer is infected is the mis-capitalization of the well-known message "Bad command or file name" as "Bad Command or file name".

The program contains one destructive payload that is set to go off on Friday the 13th, all years but not in 1987. On that date, the virus deletes every program file that was executed. Jerusalem is also known as BlackBox because of a black box it displays during the payload sequence. If the system is in text mode, Jerusalem creates a small black rectangle from row 5, column 5 to row 16, column 16. The rectangle is scrolled up by two lines.

As a result of the virus hooking into the low-level timer interrupt, PC-XT systems slow down to one fifth of their normal speeds 30 minutes after the virus has installed itself. The slowdown is less noticeable on faster machines. The virus contains code that enters a processing loop each time the processor's timer tick is activated.

Symptoms also include spontaneous disconnection of workstations from networks and creation of large printer spooling files. Disconnections occur since Jerusalem uses the 'interrupt 21h' low-level DOS functions that Novell Netware and other networking implementations required to hook into the file system.

Variants

Over the years that Jerusalem spread, many virus coders created variants of the virus, making Jerusalem one of the largest families of viruses ever created. It even includes many sub-variants and a few sub-sub-variants. Most variants are unimaginative, simply changing the payload date,

text displayed or even nothing at all. Some variants contain fixes for the bugs of the original.

```
Jerusalem.1013 Jerusalem.1024 Jerusalem.1234 Jerusalem.1237 Jerusalem.1238
Jerusalem.1241 Jerusalem.1244 Jerusalem.1264 Jerusalem.1291 Jerusalem.1329
Jerusalem.1347 Jerusalem.1348 Jerusalem.1349 Jerusalem.1353 Jerusalem.1356
Jerusalem.1361 Jerusalem.1363 Jerusalem.1364 Jerusalem.1390 Jerusalem.1399
Jerusalem.1408 Jerusalem.1427 Jerusalem.1446 Jerusalem.1448 Jerusalem.1455
Jerusalem.1459 Jerusalem.1477 Jerusalem.1478 Jerusalem.1487 Jerusalem.1488
Jerusalem.1489 Jerusalem.1500 Jerusalem.1503 Jerusalem.1504 Jerusalem.1511
Jerusalem.1518 Jerusalem.1521 Jerusalem.1522 Jerusalem.1523 Jerusalem.1524
Jerusalem.1525 Jerusalem.1526 Jerusalem.1530 Jerusalem.1533 Jerusalem.1536
Jerusalem.1548 Jerusalem.1552 Jerusalem.1558 Jerusalem.1562 Jerusalem.1568
Jerusalem.1570 Jerusalem.1587 Jerusalem.1589 Jerusalem.1591 Jerusalem.1596
Jerusalem.1598 Jerusalem.1605 Jerusalem.1607 Jerusalem.1624 Jerusalem.1631
Jerusalem.1639 Jerusalem.1640 Jerusalem.1653 Jerusalem.1664 Jerusalem.1682
Jerusalem.1692 Jerusalem.1715 Jerusalem.1716 Jerusalem.1720 Jerusalem.1721
Jerusalem.1728 Jerusalem.1733 Jerusalem.1735 Jerusalem.1747 Jerusalem.1756
Jerusalem.1765 Jerusalem.1767 Jerusalem.1768 Jerusalem.1783 Jerusalem.1792
Jerusalem.1807 Jerusalem.1808 Jerusalem.1813 Jerusalem.1824 Jerusalem.1845
Jerusalem.1884 Jerusalem.1888 Jerusalem.1899 Jerusalem.1960 Jerusalem.1968
Jerusalem.1970 Jerusalem.1975 Jerusalem.1984 Jerusalem.1991 Jerusalem.2000
Jerusalem.2012 Jerusalem.2027 Jerusalem.2053 Jerusalem.2064 Jerusalem.2080
Jerusalem.2082 Jerusalem.2083 Jerusalem.2116 Jerusalem.2126 Jerusalem.2128
Jerusalem.2132 Jerusalem.2187 Jerusalem.2208 Jerusalem.2223 Jerusalem.2224
Jerusalem.2272 Jerusalem.2291 Jerusalem.2350 Jerusalem.2358 Jerusalem.2368
Jerusalem.2389 Jerusalem.2437 Jerusalem.2465 Jerusalem.2472 Jerusalem.2490
Jerusalem.2576 Jerusalem.2758 Jerusalem.2880 Jerusalem.2886 Jerusalem.3887
Jerusalem.4112 Jerusalem.5120 Jerusalem.641 Jerusalem.662 Jerusalem.679
Jerusalem.878 Jerusalem.880 Jerusalem.986 Jerusalem.A Jerusalem.CVEX
Jerusalem.Curse Jerusalem.June11_T3Scan Jerusalem.Plastique
Jerusalem.Roger Jerusalem.a Jerusalem.com Jerusalem.sURIV_3
```

31.11. The Tequila Virus

It was one of the early examples of a polymorphic virus, making it harder to detect by antivirus software because it changed its code each time it infected a new file. The Tequila virus, originating in Switzerland in 1991, is one of the earliest examples of polymorphic malware. This advanced virus significantly impacted the evolution of malware and cybersecurity strategies by evading detection through code mutation.

31.11.1. Technical Details

The Tequila virus infected executable (.EXE) files on DOS-based systems. It was a memory-resident virus that used polymorphism to alter its code with each new infection, making it difficult for signature-based antivirus programs to detect it.

Infection Mechanism:

1. **File Infection:** Attaches to executable files, embedding its encrypted code.
2. **Memory Residency:** Becomes resident in memory upon execution of an infected file, allowing it to infect other executables.
3. **Polymorphism:** Uses a mutation engine to change its code with each infection, evading detection.

Technical Attributes:

- **Encryption:** Hides its presence within infected files using encryption.

- **Mutation Engine:** Alters its code with each infection, complicating detection.
- **Payload:** Displays a fractal, alerting users to its presence.

Tequila pioneered polymorphic techniques, influencing future malware development and antivirus strategies. The Tequila virus was a landmark in the history of computer viruses, demonstrating sophisticated polymorphic techniques that challenged and shaped the development of antivirus technologies and cybersecurity strategies.

31.12. Stoned

The Stoned Virus Family: A Legacy of Boot Sector Infections

Stoned is a very large family of boot sector viruses on the DOS platform, originating in early 1988. This family of viruses became infamous for its persistent and insidious nature, with notable members such as the Michelangelo virus, which incited widespread panic in the early 1990s, and the Angelina virus from 1994, which notably resurfaced on infected laptops as late as 2007. The Stoned virus was allegedly created by a student at the University of Wellington in New Zealand.

31.12.1. Mechanism of Infection

When a computer boots from an infected hard drive, Stoned becomes resident in memory, establishing a foothold in the system. If the computer boots from a disk other than the hard disk, Stoned checks the master boot record (MBR) of the hard disk and infects it if it is clean.

- **Floppy Disk Infection:** When infecting a floppy disk, Stoned relocates the master boot record to sector 11 and places its own code in sector 0.
- **Hard Disk Infection:** When infecting the hard disk, it moves the MBR to page 0, cylinder 0, sector 7, and places itself in page 0, cylinder 0, sector 1.

The original variant of the Stoned virus targets only 360-kilobyte 5.25-inch floppy disks and hard disks.

Once resident in memory, Stoned infects the MBRs of all accessed floppy disks but does not reinfect the hard disk. Even if the virus is removed from the MBR while it is still in memory, it does not attempt to reinfect the hard disk.

31.12.2. Payload and Effects

The Stoned virus has a 1 in 8 chance of releasing its payload during the boot process. When this happens, the infected computer emits a beep and displays the following message:

Your PC is now stoned! LEGALIZE MARIJUANA!

31.12.3. Prominent Members

1. **Michelangelo Virus:** This variant caused significant panic in the early 1990s. Despite the media hype, it infected only a few thousand computers, demonstrating how fear and misinformation can amplify perceived threats.

2. **Angelina Virus:** Identified in 1994, this variant became notable again in 2007 when it was discovered on laptops that had been sold with pre-existing infections. This resurgence highlighted the enduring nature of some viruses and the importance of rigorous cybersecurity measures.

31.12.4. Legacy and Impact

The Stoned virus family exemplifies the early challenges of cybersecurity and the evolving nature of computer threats. Despite its relatively simple mechanics by today's standards, the impact of the Stoned virus family was profound, illustrating the vulnerabilities in early computing systems and the significant role of media in shaping public perception of cybersecurity threats.

31.13. Michelangelo

Michelangelo Virus: A Case of Media-Driven Panic

The Michelangelo virus, which stems from the Stoned boot virus family, is renowned for being one of the first computer viruses to garner widespread media attention. While it incited substantial panic, the actual damage it inflicted was minimal. The virus infected only a few thousand computers, making it a classic example of media-induced hysteria.

The media frenzy began in January 1992 when two coincidental events sparked interest. One computer manufacturer inadvertently shipped 500 computers infected with the Michelangelo virus. On the same day, another manufacturer announced it would begin shipping computers with pre-installed antivirus software. This concurrence captured the media's attention, leading to a flurry of sensationalist reporting.

United Press International played a pivotal role in escalating the panic by interviewing key figures in the cybersecurity field. Among them was the International Partnership Against Computer Terrorism and John McAfee, the president of a prominent antivirus company. These interviews fueled fears, with projections suggesting that hundreds of thousands of computers could be destroyed by the virus. Data recovery consultant Martin Tibor further amplified concerns with dramatic statements like "I find virus disasters everywhere" and "I see victims of viruses all the time."

In the weeks leading up to the virus's activation date, the media focused heavily on the potential local impact. Some outlets chose to report more on the growing hysteria than on factual information about the virus itself. Few took the initiative to consult with real experts to mitigate the panic. As a result, a significant number of computer users rushed to purchase antivirus software, driven by the fear of imminent digital disaster.

In hindsight, the Michelangelo virus serves as a poignant reminder of the power of media in shaping public perception and the importance of critical evaluation of such reports. While the virus itself was relatively harmless, the surrounding hype created a disproportionate sense of urgency and fear among computer users.

/* End of Document */