

HS USER MANUAL

For

HACKSTOP

Revised documentation for HackStop 1.21 (HS.TXT Revision 19.31)

1. QuickReference

Synopsis: HackStop (HS) protects **DOS COM** and **DOS EXE** files against hacking, modification, analysis, reverse engineering and unpacking. HackStop is distributed as shareware.

A few questions:

- Have you ever written a program, and then found out the next day that your program had been cracked and uploaded to all the warez ftp sites across the world?
- Have you ever invested a lot of time and money in developing a program, which you released as shareware with a registration key option? Just to find out that someone created a generic key generator to register the current and future versions of your work?

HackStop can help you!

HackStop encrypts your DOS programs, attaches a decryptor and makes the program start at this decryptor. Whenever someone executes a HackStopped program, it will automatically be decrypted. But not before HackStop has checked its environment. If you try to debug the HackStop decryptor, you might have a hard night...

HackStop was not written to make money. Originally, the author used it for protecting his own software. HackStop was released first as shareware in 1994. It has become the best-known DOS protector, stable and pretty secure. HackStop inspired many coders to write unpackers or programs with the same functionality. The registration fee is very low; looking at the time it took us to develop HackStop.

1.1. Table of Contents

1.	Quick Reference	1
1.1.	Table of Contents	2
1.2.	How to Use HackStop?	3
1.3.	Command Line Parameters	3
1.4.	What does HackStop do?	4
1.5.	Files That Can't Be HackStopped	5
1.6.	Why Should I Use HackStop?	6
1.7.	Requirements	6
2.	Technical Notes	7
2.1.	Version numbering	7
2.2.	COM versus EXE	8
2.3.	How Does an Unpacker Work?	8
2.3.1.	Ungeneric unpackers	9
2.3.2.	Library-based unpackers	9
2.3.3.	Tracers	10
2.3.4.	Memory Dumpers	10
2.3.5.	Offline protection	11
2.3.6.	Recovery programs	11
2.4.	About HackStop	11
2.5.	Protection against DOS Viruses?	12
3.	Legal Terms and Disclaimer	12
3.1.	Disclaimer	12
3.2.	Documentation	13
3.3.	License - Shareware	13
3.4.	Distribution Restrictions	14
4.	Closing	14
4.1.	Registration	15
4.2.	Personalised Versions of HS	15
4.3.	How to get the newest version of HackStop	15
4.4.	Contacting the Author	16
4.5.	Enhancements In Future Versions	16
4.6.	Credits	16

1.2. How to Use HackStop?

This is HackStop's command-line syntax:

hs [filename] [-options]

Options can start with a slash or even a comma, too.

For example, to protect megaprg.exe in the current directory:

hs megaprg.exe

HackStop will only protect files ending with ".COM" and ".EXE". After a file has been protected by HackStop, HackStop can not „un-HackStop“ it. You can imagine why! Be sure to test your protected program. We have tested HackStop a lot ourselves, but some incompatibilities possibly arise with, for example, overlaid EXE programs.

HackStop automatically creates back-up files with the extension .BAK for EXE-files. In fact, your original file is renamed to the extension ".BAK" and HackStop creates a new EXE based on the back-up file, which is opened read-only. To use the extension .EHS rather than .BAK, invoke HS with the additional option "-bh". Example:

hs myfile.exe -bh

The time and date stamp of myfile.exe remains the same. There are a lot of other options...

1.3. Command Line Parameters

HackStop can be invoked with the following additional command line parameters:

Parameter	Meaning
-? -h	Displays a short help, how to use HackStop.
-bh	Makes a back-up file with the extension ".EHS" for EXE files and ".CHS" for COM files. Warning, old back-up files are overwritten!
-I	Shows a little intro with greetings and other nice stuff - with Adlib sound! For the intro you need a 386 CPU!
-ii	Shows HS internal compiler information.
-k	Kills the "HsxxMsDos" signature at the end

Parameter	Meaning
	of HackStopped files. You can use this option to fool unpacker tools. This option is only available in the registered version.
-nb	Does not encrypt the body of EXE files
-nr	Does not encrypt the relocations of EXE files
-p	Shows the version number, compile time and the personalised text of HS.
-pb	Shows the build number of HackStop. A build is a program recompilation after changes that just aren't major enough to raise the version number. The two other numbers are COM decryptor size and the EXE decryptor size.

Options are not case sensitive. You can use "-", "/" and "," to introduce an option.

Examples:

```

hs -?
hs /ii
hs -p
hs /pb
hs comfile.com ,bh
hs realexe.exe -k
hs -i

```

1.4. What does HackStop do?

HackStop encrypts and secures your executable program files by placing a special security envelope around them. Except the automatic decrypting process, the security envelope takes care of the actual protection.

To protect a specific file, simply run HackStop on it, and you and your users will not know that it's there unless somebody tries to crack or analyse it. This is usually done using a debugger or dumper. Do not expect to be able to easily trace through HackStop's security envelope using many debuggers - advanced debug traps help destroy this option. Furthermore HackStop has different levels of encryption to stop tracing or analysing attempts on protected programs. HackStop is one of the most advanced executable protection programs of its type to keep your programs from being altered or reverse engineered.

HackStop does not only use simple 'XOR' encryption techniques. HS includes a simple "mini mutation engine" to ensure that the encryption keys are unique. In the current version HackStop has several layers of encryption. HackStop includes code against popular hacking tools like:

- TR, IceUnp, TRON -p, UNP, X-Tract, debug, eDump

- DeGlucker, CUP386, ERP, Turbo Debugger, SoftIce
- TEU, UPC, Entpack, XPack -UX, AutoHack, SnapShot, Intruder, CUNP, XO

Please refer to the file HISTORY.TXT to see all "supported" unpackers.

1.5. Files That Can't Be HackStopped

First, HackStop only protects executable files, recognisable on its .COM and .EXE extension.

Second, Windows, OS/2 or BeOS .EXE files can not be protected by HackStop. Technically, these files have the PE „Portable Executable“ format, defined by Microsoft. A PE-program is basically a small DOS program (the „stub“) plus a huge „overlay“ (additional bytes in the program on disk that will not be loaded into memory when the program is executed). The stub says something like "This program requires Windows" and contains a pointer to the actual program. DOS will display the error message; Windows will start the real program. HackStop will automatically detect whether a file has a Windows, OS/2 or a linear executable header and does not waste your time trying to protect it.

Third, HackStop has its own limitations:

- HackStop tries to load an EXE header of 64 bytes to determine the file type. Because of this, files smaller than 64 bytes can not be HackStopped.
- COM files larger than approximately 61000 bytes can not be HackStopped because the decryptor would not fit into the maximal COM size of 64 KB - 256 bytes (PSP) - stack size. You can find HackStop's decryptor size using the -pb switch.
- Files with overlays cannot be HackStopped because the "Load Window" of DOS could now be too small for this file. However, you could try the alternative solution using Ben Castricum's classic UNP 4.12b tool:

```
unp l program.exe out.exe -r+
```

```
hs out.exe
```

```
unp o program.exe out.exe
```

- HackStop will not protect files with a weird EXE header for your own safety. This case includes files with more relocations than the header could include or files with maxmem=0 allocation.
- HackStop will not protect files with more than ~16000 relocation items - files that never have been seen in real life. This is because the HS decryptor does not scan for a new segment if there are so many relocations, a speed omission because most files do not need it. In the case you have such a file you must compress the file first (we recommend UPX).

Hint: Use an on-line compressor like PkLite, WWPack or UPX before protecting the file with HackStop. The program will be much smaller. HackStopped files normally are no longer compressible after protecting due to the strong encryption!

You can technically protect files like COMMAND.COM, but in our opinion it makes no sense to protect DOS system files. We did not test it either.

1.6. Why Should I Use HackStop?

For the few DOS freaks out there, we don't need to say that compressing files with UPX or something similar does not offer enough protection. Anyone can decompress PKLITE-compressed files quite easily. Not only UNP does the job, but also with a little interaction TR will do everything for you due to the complete lack of anti-TR-debug tricks. Even if a program is compressed with the supposedly "invincible" -E option of the professional version of PKLITE or the "pu" option of WWPack.

There are a lot of tools that can do this. We have at least about 40 different unpackers that can unpack PKLITE or LZEXE. I even have at least three batch files that are able to remove protection from COM files. If you are interested in such unpackers, then try to request the UNTINY archive at your local ROSE SWE Distribution Site (see ROSEBBS.TXT).

After decompressing a compressed program, anyone can change your program (remove copyright screens, disassemble code, etc) and spread it around. NO software protection program will stop all crackers, for sure in 2003. But HackStop does a good job protecting your programs against most of them.

Compared to other protections nowadays, the main advantage of HackStop is clearly its compatibility. Been famous for this in the past, we recently have had some unstable releases. However, this version is as reliable as usual.

All programs from ROSE SWE are protected with HackStop - this means that about 75.000 users are running HackStopped programs!

1.7. Requirements

The requirements to run files **protected** by HackStop are basically zero: DOS 3.30, an 8086 processor with 256 KB, etc... From version 1.19 on, the HS program **itself** requires at least an 80386/SX CPU to protect files or view the intro.

HS86: Protected files can run on my 8 MHz XT, and we have done much testing of HackStop on even this lowest common denominator type of machine, to make sure that HackStop will run on every type of computer from Intel 8088 to AMD K7 and beyond. Start HackStop with the option "hs -te" to see our test bed.

HS386: This version uses advanced anti debugger features from the 80386 instruction set. For this reason HS and protected files requires at least a 80386 CPU or better.

2. TechnicalNotes

2.1. Version numbering

As you might know, HackStop has had a turbulent history. Having had its best time around 1995, it is now back again, being updated by two enthusiast authors, who have since a long time just rewritten the documentation you are reading now.

In the past, HackStop was criticised for its version numbering. Indeed there were about ten versions 1.17 floating around. In fact, this was the ideal weapon against all ungeneric HackStop unpackers at that time. The next information can help you to identify the HackStop version you have.

Starting from version 1.11, HackStop adds a signature to programs protected with HackStop. You will find at the last 9 bytes at the end of the file the following code:

"HS", verhi, verlo, "MsDos"

"verhi" and "verlo" are the version numbers of the used HackStop program. If you have used version 1.11 then "verhi" is 1 and "verlo" is 11.

printf ("Version used: %i.%02i", verhi, verlo);

From HS 1.18 on, a program called ChkHS is included which uses this detection of protected files. However, this will not work on files that are protected using the "-k" switch in the registered version of HackStop.

From version 1.18 on as well, HackStop has the option "-pb". This option shows the currently build version and the actual protector length. This option was written for ChkExe, ScanExe and other tools to determine the different HackStop versions. Please note that personalised registered versions of HackStop have a different build than registered versions or the shareware version.

This is a typical output of `hs -pb`

HS-ID = HS.386, Build=61.2867.3078 - pre-release for...

HS-ID = HS, Build=68.3058.3130 - Special X-Mas release!

HS-ID = HS, Build=233.3364.3854 - Release for the ROSE SWE mailing list!

The HS-ID tells you if it is the 8086 (HS) or the 80386 (HS.386) version of HackStop. Build is split in build counter, COM protector length and EXE protector length. Sometimes an internal remark will follow just like "beta version" etc.

Some popular online compressors like WVPack, UPX, PkLite, LZEXE and Diet put a signature into the exe-header. If HackStop finds one of these known signatures it will replace (overwrite) them with "HS".

2.2. COM versus EXE

COM files belong to the CP/M age, when a code, data and stack of a program together had to fit within one 64-kb segment. COM files do not have relocation items, because of this one segment.

COM files are easy to hack, because DOS is an unsecured OS and does not clear the used memory, so the unpacked programs can be found in memory. COM files fit into one segment (so CS=DS=ES=SS) and have no relocations. They don't even have a header: the entry point of the program is the first byte of the file. Therefore, dumps of the unprotected program are directly executable. Unpackers are far easier to write for COM files than for EXE files.

EXE files have a header of varying size. EXE files can have any length and have an overlay if the file is longer than it should be according to the EXE header. EXE files have several segments.

As well, the EXE header contains the relocation items. These are offsets in the exe body where DS+10 have to be added. DOS does this automatically after having loaded the exe body image into memory, but because HackStop has encrypted the exe body and the relocation items, the HackStop decryptor has to do this itself.

If you have the choice between .COM and .EXE file type, choose .EXE files, they are safer! For this reason an additional program (COM2EXE) is included in the HackStop package to convert .COM files to .EXE. As well as you can protect files twice with HackStop (remember that back-up files are overwritten!) you could protect .COM files, convert them with COM2EXE and protect them again. This adds a two-level security envelope around your file that makes hacking a little bit harder, but the loading time a bit slower.

HackStop protects COM and EXE files in a total different way.

2.3. How Does an Unpacker Work?

Ah, the technical part... Haven't you been waiting for that? :)

There are several ways to destroy HackStop's protection, as shown in the past. When an unprotection utility unprotects a file, it creates a virtual DOS environment for the file to run in - until the security envelope has finished decrypting or unpacking itself. When this is done, the unprotect utility writes back to disk what is in memory. Now you have your unprotected file.

Unpackers are usually divided in types. Compare this documentation to that of other protectors and you'll see that nobody really agrees about what the type „generic unpacker“ means (in my opinion that's a program that removes any security envelope from any file). By origin, generic unpackers were meant at least to remove future versions of a specific packer.

2.3.1. Ungeneric unpackers

Ungeneric unpackers are meant to remove one specific protection. These programs usually use neat tricks; they hook an interrupt, set a breakpoint on a place where HackStop by mistake allowed that or misuse other protection holes. Many unpackers for HackStop have been written this way (Ka0t's unHS, Mega Devil's unPHS, Stefan Esser's HSR, Rand0m's KillHS, the rIDDLER's xHS, Gabler's HackStop remover ...) Previous versions of HackStop did not include a relocation decryptor, so ungeneric unpackers were relatively easy to code; starting with HS v1.19 future versions of these tools will require a relocation rebuilder.

2.3.2. Library-based unpackers

Intruder was the first unpacker based on this system. UPC was an Intruder-recode by my personal coding god Synopsis, the great TEU program by JVP followed the idea, the undocumented XPack-option UX contained TEU-code and the latest addition to this breed is the fine Blast Wave unpacker.

If your program is written in a DOS high level language like Pascal, C or Basic, the compiled program uses some interrupt calls which are typical for the used compiler. Tools like UPC, Intruder and TEU simply hook these interrupts and wait for these calls. Then, they search the original entry point of the program (because meanwhile the protector has done its work).

So how does HackStop stop such unpackers? Well, it simulates the start-up code. Tools like TEU will detect a false start-up code and dump up to 2 MB of scratch to disk, claiming that this is the unpacked program. HackStop has simulated more than 10 different start-up codes in the past. We have cleaned up a bit in the meantime. We still work on a main generic shield, which already renders all these nice tools useless, but there are compatibility problems. A prototype of this shield is present in Mess 1.30 /T; once it is finished it will be included in HackStop.

As an alternative for failing protections in protectors like HackStop, things like Hold's ANTIUPC and MaX' Anti-TEU have been written and some people have even come up with changed system units for Turbo Pascal.

2.3.3. Tracers

Tracers break after every instruction. There are several ways to code a tracer. First, there's the Intel method: using the breakpoints the processor provides. In 1994, everyone still used the standard int1-method using the 8086 trap flag („software breakpoints“), IUP and UNP t were completely based on it. Disabling this single step tracing, or playing around with the stack, will kick this kind of tracers.

Turbo Debugger 286 does not really use single step tracing. It hooks interrupt 3. When tracing, it overwrites every next instruction with an „int 3“ instruction. When control is returned to TD, TD restores the instruction and starts all over again. Playing around with the interrupt table is enough to kick TD286.

The UCF coder Alex Petroukine (aka Sage) set standards in 1996, by using the 386+ specific „hardware breakpoints“ we knew from SoftIce and TD386 for his generic unpacker CUP386 /3. The internal debugger was even greater. It defeated HackStop immediately at that time and forced the scene to invent even more tracing-based antidebugging tricks. Currently, the best tracer of this kind is GTR (again by an excellent UCF coder, Hendrix), while Lado's LTR and CrazyMax' DeGlucker still do a very nice job. The HackStop version for the 8086 does not disable hardware breakpoints. Therefore, a HS386 alias HackStop/32 version is available with several 386+ antidebugging tricks. We advise you to use HS386 instead of HS86 because of this better security.

Inventing tracing-based antidebugging tricks is even more essential for the second way of tracing. CUP386 /7, and more known, debuggers like Liu TaoTao's TR and tracers like Jauming Tseng's Ice-Unp emulate every instruction instead of executing it. HackStop tries to kick these by checking emulation bugs. For example, TR 2.53 treats 386+ instructions like BT as a NOP.

2.3.4. Memory Dumpers

These tools do often not produce working executables, but memory snapshots while the program is unpacking. Usually these snapshots are made using the timer interrupt or hardware breakpoints. HackStop tries to detect these programs on their hooked interrupts. Martin Malik's hAWeD! protection even disables interrupt 13h, but this approach would slow down concurrent processes.

Seldom described in protector documentation is the relatively new technology by EliCZ to use Windows-specific breakpoints to dump protected programs. Currently, HackStop does not contain tricks against EliCZ' EDump program which makes use of this technology. The problem is that EDump operates on ring0-level while Hack-Stopped programs are ring3. EDump can not be removed by the Hack-Stop decryptor without harming Windows. At the moment that this documentation is written, all protectors that claim to kick EDump do not.

2.3.5. Offline protection

Hardly a protection, HackStop after five years still contains the infamous, often simulated „opcode encryption“, alias Nebelbombs. Even TEU, Ciphator and PELockNT are full of this anti-disassembling code. The so-called Nebelbombs take up about 25% of the amount of bytes of a HackStop decryptor. In fact Nebelbombs are just harmless instructions which jump to a location within another opcode. This causes the HackStop decryptor to be difficult to disassemble using tools like IDA or Sourcer 7 by V Communications. However, tools like MOW and AHCR (by ROSE SWE) exist to replace some Nebelbombs with NOP instructions. Needless to say, that if a HackStopped programs is “edited” by tools like MOW or AHCR it won’t run anymore!

2.3.6. Recovery programs

The name of this program is borrowed from the documentation of the only program of this type: ERP by Richie. As explained in the COM versus EXE part, EXE files can have relocations. These things involve knowledge of how DOS loads EXE files, which is not entirely trivial. It is easy to find out whether the author of your favourite protector has this knowledge: copy any EXE file with more than 0 relocations (say ARJ.EXE) to a temporary directory and protect it. Next, view the resulting file. Search for a nice string („copyright“ is a safe bet). Does it occur? Yes? In that case, the string has **not** been encrypted, even while you protected the program. Even more funny; try to change the string with a hex-editor. Does the changed program still run? If yes, consider how difficult it must be for a cracker to change your code...

Run this procedure with HackStop 1.19 and you’ll see that you’re safe. But some protectors, among them earlier HackStop versions, do not encrypt the EXE body if there are relocations in it. Except for the risk that people are changing the code, it is very easy to remove the protection itself then – the only things missing compared to the unprotected file are the starting point of your program (not necessarily the first byte) plus the starting point of the stack.

This is where ERP comes in. ERP recognizes entry points in EXE bodies. After successfully ERPing a protected file, the protection is simply not executed. It is like digging a tunnel below a wall. But HackStop does handle relocations, thus turning the sand into stone...

2.4. About HackStop

HackStop is written entirely in assembly language (MASM 6.xx, macros and ASM libraries). We have written and tested HackStop on

different development platforms. Furthermore we have tested HackStop on more than 50 different machines with different DOS and Windows versions. HackStop's anti-debugging techniques (which use the same anti-debugger macros like ROSETINY, ChkPc or HMS) have been around for over ten years, with many people pitching in ideas and unpackers to make it more secure. We credit them below and in the HackStop intro (hs -i).

If you have any suggestions, questions, comments etc. about HackStop, you can contact us. We are happy to get feedback!

By origin, HackStop is an idea of Ralph Roth. All versions from 0.90 up to 1.18 are mainly his work. The main development on version 1.19 has been done by Stonehead. Major development on version 1.20 was done again by Ralph.

2.5. Protection against DOS Viruses?

HackStopped programs are immunised against the standard Jerusalem (1808) virus family. For this reason all HackStopped programs end with the sign "MsDos" - this makes the Jerusalem virus think the HS-protected file is already infected. Besides, tracing viruses like Happy_Shiny or DAME:Trigger will hang the system if they are trying to infect programs protected with HackStop. If you need additional protection, protect your file with a shield like VSS¹, RecAV², F-Xlock or FileShield too.

It may be possible for a protected program to set off some anti-viral programs that have heuristic abilities. This has not happened to my knowledge (it seems they cannot trace through the decryption algorithm) but some (AVP, F-Prot, RHBVS, Suspicious etc.) are set to alert the user if they detect a decryption algorithm at the beginning of a file's execution. So far, none of the heuristic programs I have tested seem to be able to identify the decryption algorithm as being such. TBSCAN can indicate some heuristic flags, however, but TBSCAN is meanwhile obsolete.

3. Legal Terms and Disclaimer

3.1. Disclaimer

HackStop basically has no legal guarantee and warranty because I do not want to get sued over it, and should be used "as is". Here is the official disclaimer:

HackStop ("program") DOES ALTER EXECUTABLE FILES and may have or cause compatibility problems with them (that is why YOU should keep a back-up file, in case of incompatibility with a particular file) in certain circumstances.

¹ By ROSE SWE, not public available, ask for a copy.

² By ROSE SWE, available on our home page!

Under no circumstances ROSE SWE, Ralph Roth ("author") may be held liable or accountable for any damage to system files, executable files, data files, or any other system or data damage due to use or misuse of his program. The author also may not be held accountable for loss of profits or for any other damages incurred by the use or misuse of his program. The author has forewarned any users that damage to files may occur with use or misuse of his program, and in executing the program, the user fully understands these risks and this disclaimer.

3.2. Documentation

Information in the documentation is subject to change without notice and does not represent a commitment on the part of ROSE SWE.

The name „HackStop“ is historically defined. In the current age, the terms hacking and cracking are often redefined and confused. We assure you that this program will not stop anyone hacking the root account on your Linux box. :)

3.3. License - Shareware

The supplied software contains NO public domain program(s). The program and all accompanying documentation are Copyright (c) 1994-2003 by ROSE SWE. All rights reserved.

The Copyright laws of Germany protect this software and accompanying documentation. Any use of this software in violation of Copyright law or the terms of this limited licence will be prosecuted to the best of our ability. The conditions under which you may copy this software and documentation are clearly outlined below under 'Distribution Restrictions'.

HackStop is distributed as SHAREWARE. You may use HackStop for the purposes of evaluating it (after understanding the disclaimer and the documentation) for 60 days. No files protected by HackStop during this trial period may be distributed to OTHER computers at all, commercially or non-commercially. If you find HackStop to be of use to you, you must register HackStop with the author.

ROSE SWE hereby guarantees you a limited licence to use this software for evaluation purposes for a period not exceeding sixty (60) days. If you intend to continue using this software (and/or its documentation) after the sixty (60) day evaluation period, you must make a registration payment to ROSE SWE. Using this software after the sixty (60) day evaluation period without registering the software is a violation of the terms of this limited licence!

You shall not use, copy, emulate, clone, rent, lease, sell, modify, decompile, disassemble, otherwise reverse engineer, or transfer the program, or any subset of the program, except as provided for in this agreement. Any such unauthorised use shall result in immediate and automatic termination of this licence. ROSE SWE reserves all rights not expressly granted here.

3.4. Distribution Restrictions

As the copyright holder, ROSE SWE authorises distribution by individuals only in accordance with the following restrictions.

The package is defined as the entire file either as 'self extracting executable' or an 'archive' as distributed by ROSE SWE. The authenticity of the package can be verified by contacting ROSE SWE or using the program CrCheck. The original archive is packed using the ZIP format. If the package is changed in any way, the distribution is forbidden. Please contact ROSE SWE to obtain a complete package suitable for distribution. You are hereby granted permission by ROSE SWE to copy the package for your own use or for others to evaluate, ONLY when the following conditions are met:

- The package - including all related program files and documentation files - ARE NOT modified in any way and must be distributed as a complete unchanged package, without exception. Small supplements to the package, such as the introductory or installation batch files are acceptable. This should always be done by supplying EXTRA files, never by altering the package (file) as distributed by ROSE SWE.
- No price or other compensation may be charged for the package. A distribution cost may be charged for the cost of the diskette, shipping and handling, as long as the total (per package) does not exceed Euro 10. The package CANNOT be sold as part of some other inclusive package, nor can it be included in any commercial or non-commercial software-packaging offer, without a written agreement from ROSE SWE.
- ROSE SWE prohibits the distribution of outdated versions of the package, without written permission from ROSE SWE. If the version you have obtained is over twelve (12) months old, please contact ROSE SWE to ensure that you have the most current version.
- The package, program(s) or documentation cannot be 'rented' or 'leased' to others. If you wish to add any of our packages to a CD-ROM or other collection, please check the release date of the version you have. If the version is over twelve (12) months old then please contact ROSE SWE to ensure that you have the most current version.
- If you would like to distribute the package as a 'Disk-of-the-Month', or as part of a subscription or monthly service, then you must contact ROSE SWE in advance to ensure that you have the most current version of the software. Only current versions may be shipped as 'Disk-of-the-Month' disks.
- You may not list this product in advertisements, catalogues, or other literature that describes this product as 'FREE SOFTWARE'. This is 'Try-Before-You-Buy' software, it is not free!

4. Closing

There is no doubt that HackStop can save you time, effort, energy and money. There are no "run-time fees", "royalties" or anything of the type attached to the cost of HackStop. You can protect and

distribute as many files as you want with HackStop ONCE YOU REGISTER. The cost is Euro 15, -- per copy of HackStop. Please use the file REGISTER.TXT to order a registered version of HackStop. Please send the register form to my address, even if you have transferred the money to my bank account, because our address will often be unreadable on checks!

4.1. Registration

There is almost no difference between the registered and unregistered version of HackStop except for the "beg remark" and the ASCII remark in HackStopped programs, saying that it is an UNREGISTERED SHAREWARE version. The registered version of HackStop has a different 'data offset', other antidebugging macros and a different protector length, so programs protected with the shareware version will always differ from the registered versions. The registered version of HackStop supports the "-k" switch to remove the HackStop signature. Along with registering HackStop you will receive the newest currently available version of HackStop. To register your copies of HackStop please print out the file REGISTER.TXT.

Additionally with the registered version of HackStop you will receive the latest versions of **ROSE COM Crypt/286**, **ROSE EXE Cryptor (REC)** and **ROSETINY** (Freeware) as well as beta versions of HackStop or other file protection tools, if available! German users will additionally receive some bonus antivirus programs in German written by ROSE SWE. We always try to pack as many programs as possible on the disc containing HackStop.

Registered users can order the newest version of HS for half price.

4.2. Personalised Versions of HS

You can obtain a so-called "**personalised**" version of HS. The difference between the normal version and this version is that your copy of HS carries your name and address or an advertising slogan. If you want, you can include up to 6-10 lines of text! You can send us your ASCII text logo to be included in HS.EXE. There is also a version available with no text.

For this reason personalised HS versions will produce HackStopped programs with a different length and a different offset that means that they are harder to attack than the registered (standard) or Shareware version. COMMERCIAL USE OF HACKSTOP REQUIRES A REGISTRATION!

4.3. How to get the newest version of HackStop

First take a look at the file ROSEBBS.TXT - it contains addresses providing new HackStop versions for downloading.

Hanno Boeck has established the "EXE distribution list". Over this mailing list we send the newest HackStop (and related programs from ROSE SWE).

Join your mailing list at subscribe-rose_swe@yahooGroups.com get all programs from ROSE SWE via email - see ROSEBBS.TXT for further details!

4.4. Contacting the Author

ROSE SWE
Dipl.-Ing. Ralph Roth

rose_swe@gmx.net
http://come.to/rose_swe

Check the file REGISTER.TXT and ROSEBBS.TXT for the complete address, PGP key and Email address!

4.5. Enhancements In Future Versions

Personally I think the MS-DOS age is gone, so therefore HS development had slowed down. HackStop 1.20 is a quite stable version currently unpackable by all the old HackStop unpackers. Therefore main development will be enhancing HS for new operating systems, new unpackers or bug fixing.

If there is enough interest, the following features could be implemented:

- Enhanced encryption of the HackStop decryptor using a mutation engine, to implement the earlier claimed „memory encryption“ 100%. This could be done using the SHAME mutation engine from Stonehead's Mess. However, a major rewrite will be necessary.
- Optional password protection.
- Optional 80386 checking of HackStopped programs.
- HackStop for coff2 files (produced by GNU DJGPP compiler)
- HackStop for Linux (elf file format)
- Handling of Windows 9x/NT (PE) EXE programs.

Thank you for evaluating HackStop and reading the documentation. Happy HackStop'ing! All improvements and suggestions are welcome!

4.6. Credits

We would like to thank and send "greetings" to the following peoples for pitching in ideas, finding bugs and doing beta testing of HS the last years:

- Andreas Marx (author of CGL AV and TScan)

© by ROSE SWE, Ralph Roth

http://come.to/rose_swe

- Ben Castricum (author of UNP)
- Christian Ghisler (author of Win-Commander)
- Grischa Brockhaus (author of SkullCheck)
- Hanno Boeck (author of ChkExe)
- J.H. Dinges
- Peter Hubinsky (Sysop SAC BBS/SAC ftp) for being the first distributor of HackStop world-wide.
- Michael Hering (author of FI etc.)
- Rafal Wierzbicki & Piotr Warezak (authors of WWPack & Sac-View)
- Ralph Biedermann (formerly sysop of LionBox/Germany), all the sysops distributing HackStop!
- Rene Rudolf
- Stefan Kurtzhals (author of SSC)
- **Stonehead (author of Mess and co-author of HackStop)**
- VrtikSoft - J. Valky & L. Vrtik (authors of TraceLook). THX guys for pitching in so many ideas in HackStop.
- Walter Gabor and Thorsten Weber (Mr.D) for correcting the docs.
- Willi Marquardt for antidebugging tricks and for being the first person hacking HS.
- Everyone on the HackStop betatester-mailinglist
- The 31337 side (X-Adi, UE, UCF, TPiNC, Vandals etc.) - see the intro. You know documentation sucks! ;)
- All those we have forgotten to mention: Run HS.EXE with the option "-i"
- Jeremy Lilley and Andrew Kacy. The earliest version of this documentation was written around 1995 using the documentation of their programs Protect! and LockMaster as an example.

/* the end */