How to Encrypt Your Bash Shell Script on Linux Using SHC

by RAMESH NATARAJAN *on* MAY 15, 2012

Q: How do I encrypt my bash shell script on Linux environment? The shell script contains password, and I don't want others who have execute access to view the shell script and get the password. Is there a way to encrypt my shell script?

A: First, as a best practice you should not be encrypting your shell script. You should really document your shell script properly so that anybody who views it understands exactly what it does. If it contains sensitive information like password, you should figure out a different approach to write the shell script without having to encrypt it.

That being said, if you still insist on encrypting a shell script, you can use SHC utility as explained below. Please note that encrypted shell script created by shc is not readable by normal users. However someone who understands how this works can extract the original shell script from the encrypted binary created by shc.

SHC stands for shell script compiler.

1. Download shc and install it

Download shc and install it as shown below.

Verify that shc is installed properly.

```
$ ./shc -v
shc parse(-f): No source file specified

shc Usage: shc [-e date] [-m addr] [-i iopt] [-x cmnd] [-l lopt] [-rvDTCA
```

2. Create a Sample Shell Script

Create a sample bash shell script that you like to encrypt using shc for testing purpose.

For testing purpose, let us create the following random.sh shell script which generates random numbers. You have to specify how many random numbers you like to generate.


```
$ vi random.sh
#!/bin/bash

echo -n "How many random numbers do you want to generate? "
read max

for (( start = 1; start <= $max; start++ ))
do
  echo -e $RANDOM
```

```
1678
491
```

3. Encrypt the Shell Script Using shc

Encrypt the random.sh shell scripting using shc as shown below.

```
$ ./shc -f random.sh
```

This will create the following two files:

```
$ ls -l random.sh*
-rwxrw-r--. 1 ramesh ramesh   149 Mar 27 01:09 random.sh
-rwx-wx--x. 1 ramesh ramesh 11752 Mar 27 01:12 random.sh.x
-rw-rw-r--. 1 ramesh ramesh 10174 Mar 27 01:12 random.sh.x.c
```

- random.sh is the original unencrypted shell script
- random.sh.x is the encrypted shell script in binary format
- random.sh.x.c is the C source code of the random.sh file. This C source code is compiled to create the above encrypted random.sh.x file. The whole logic behind the shc is to convert the random.sh shell script to random.sh.x.c C program (and of course compile that to generate the random.sh.x executable)

```
$ file random.sh
random.sh: Bourne-Again shell script text executable

$ file random.sh.x
random.sh.x: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamic

$ file random.sh.x.c
random.sh.x.c: ASCII C program text
```

4. Execute the Encrypted Shell Script

Now, let us execute the encrypted shell script to make sure it works as expected.

```
10494
29627
```

Please note that the binary itself is still dependent on the shell (the first line provided in the random.sh. i.e /bin/bash) to be available to execute the script.

5. Specifying Expiration Date for Your Shell Script

Using shc you can also specify an expiration date. i.e After this expiration date when somebody tries to execute the shell script, they'll get an error message.

Let us say that you don't want anybody to execute the random.sh.x after 31-Dec-2011 (I used last year date for testing purpose).

Create a new encrypted shell script using "shc -e" option to specify expiration date. The expiration date is specified in the dd/mm/yyyy format.

```
$ ./shc -e 31/12/2011 -f random.sh
```

In this example, if someone tries to execute the random.sh.x, after 31-Dec-2011, they'll get a default expiration message as shown below.

```
$ ./random.sh.x
./random.sh.x: has expired!
Please contact your provider
```

If you like to specify your own custom expiration message, use -m option (along with -e option as shown below).

```
$ ./shc -e 31/12/2011 -m "Contact admin@thegeekstuff.com for new version

$ ./random.sh.x
./random.sh.x: has expired!
Contact admin@thegeekstuff.com for new version of this script
```

- -r will relax security to create a redistributable binary that executes on other systems that runs the same operating system as the one on which it was compiled.
- -T will allow the created binary files to be traceable using programs like strace, ltrace, etc.
- -v is for verbose

Typically you might want to use both -r and -T option to craete a redistributable and tracable shell encrypted shell script as shown below.

```
$ ./shc -v -r -T -f random.sh
shc shll=bash
shc [-i]=-c
shc [-x]=exec '%s' "$@"
shc [-l]=
shc opts=
shc: cc  random.sh.x.c -o random.sh.x
shc: strip random.sh.x
shc: chmod go-r random.sh.x

$ ./random.sh.x
How many random numbers do you want to generate? 3
28954
1410
15234
```

Finally, it is worth repeating again: You should not be encrypting your shell script in the first place. But, if you decided to encrypt your shell script using shc, please remember that a smart person can still generate the original shell script from the encrypted binary that was created by shc.

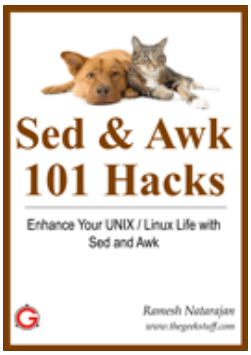
If you enjoyed this article, you might also like..

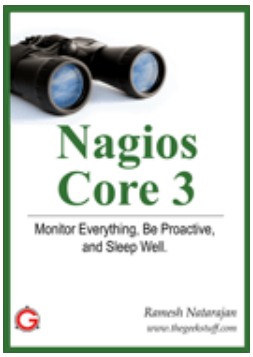
1. 50 Linux Sysadmin Tutorials
2. 50 Most Frequently Used Linux

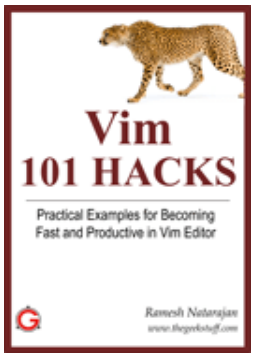
- Awk Introduction – 7 Awk Print Examples

4. Mommy, I found it! – 15 Practical Linux Find Command Examples
5. Linux 101 Hacks 2nd Edition eBook **Free**

- 25 Most Frequently Used Linux IPTables Rules Examples
- Turbocharge PuTTY with 12 Powerful Add-Ons







Comments on this entry are closed.

Anurag Sharma May 15, 2012, 5:44 am

very nice...its helped me a lot.thanks a ton

LINK

john May 15, 2012, 11:24 am

Ramesh,

Enjoy your articles and short summaries.

This tool encodes, it does not encrypt. It converts to binary and provides some nice features that make it
more secure. Encryption would require a key. I suppose you could say that if you don't have the shc tool you couldn't decode the file, making it "encrypted", but that is a very weak cipher that could be broken quite possibly with nothing more than the strings command, at least for the plain text component.

passed as it would be encoded as plain ASCII. I use strings on microsoft executable files and attachments to look for strings of text that point to dll's or registry entries… spotted a lot of viruses that way.

Have you tried using strings on the encoded file? It would be interesting to see what is shown. I'm not
going to download the tool to try it.

So, this is a merely a compiler, converting the plain ASCII of the shell script to binary.
NAME shc – Generic shell script compiler

A truly encrypted file would not be executable without another step or a key. It might be possible using PGP or some combination of ssh and file permissions, sudo, sticky bits or any other convoluted method to provide true security. The use of expect with DBA files and FTP presents the same kind of challenges. Have come up with a number of ways of masking this info.

But even though the author of the tool uses the term encryption, it is not.

thanx,
john

LINK

Sylvain Senechal May 15, 2012, 11:53 am

I am always suspicious when a 3rd party tool needs root privileges to run properly.

On early Ubuntu distribution (for example), to use shc without root privileges you need to use the -T argument. (Try the following to see if you need the -T switch: $ strace strace -p $$)

Since Ubuntu 10.10, PTRACE_ATTACH is not allowed against arbitrary matching-uid processes. The traced "child" must be a descendant
of the tracer or must have called prctl(2) using PR_SET_PTRACER,
with the pid of the tracer (or one of its ancestors). For more

– https://wiki.ubuntu.com/SecurityTeam/Roadmap/KernelHardening#ptrace
– http://ubuntuforums.org/archive/index.php/t-1872947.html

Also, not sure if it's a realy safe way to encrypt scripts. (But its better than nothing)

cf.:
http://www.linuxjournal.com/article/8256
http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=327263
http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=508109

Thx for your post!
Regards,

– Sylvain

LINK

Jalal Hajigholamali May 16, 2012, 11:22 am

Hi Ramesh

Thanks a lot…

i used SHC for many years and very useful utility

LINK

Krishnan May 17, 2012, 6:42 am

After encrypting can i get rid of the .sh file? What if i want to make chnages to the script again, can i decrypt from .sh.x or .sh.x.c? If i pass on these encrypted files to some one else for execution can they decrypt these files and look into the contents?

LINK

Thanks we have been using SHC for years and found it , very useful.

Karthik.P.R
MySQL DBA

LINK

KB May 24, 2012, 10:51 am

Summary:

SHC is a cool geeky tool, but it doesn't address what I feel are the root causes – a failure to separate code from configuration and a myopic view of security.

Details:

Ramesh – nice article and you're definitely pointing folks in the right direction but I see a deeper issue here (you alluded to it but didn't address it head-on).

As I see it, when developers (whether system administrators or not) want to hard-code things (like passwords for example), they're writing code that's hard to maintain and typically much less secure / flexible over the long term.

Passwords are just one of many things that programs need to access as part of the programs' configuration. Rather than storing configuration details directly in a program, it's often much more maintainable to store configuration in a configuration-specific area like a config.ini or similar file. This makes it a lot easier to a) find, and b) change without risking breaking the executable code (whether shell script or binary executable), c) should help remind the developer that the values need to be checked before being used, and d) avoids re-building a system each time the configuration changes. After all – we do this for executable programs – why can't we do the same for scripts?

The other item to consider – when storing sensitive data in a file system (or other persistence layer / store), administrators need to consider how safe the store is from prying eyes. If a store is broken into, it helps to know where all the passwords are stored so those passwords can be changed easily. It is also a good idea to change passwords from time to time even if a store hasn't been broken into

If the password being stored is not being used to get into something but to allow access to something, most programs will store a secure hash of a password or better yet, store a hash of a password per-authorized-user. Often, access requirements are not just user and password combinations but may also include a source IP or range of addresses as well as a potential secure key that is changed out from time to time as well. Bonus points if access is specific to the user's needs (like user a can SELECT, user b can SELECT, INSERT, UPDATE, DELETE, and user c has administrative access).

KB

LINK

eshwar August 1, 2012, 5:35 am

hi just i have encrypted with key my shell script by SHC utility nice tool .
bt i want to know how to decrypt the encrypted shell script.please let me know anyone knows this ...thanks in advance

regards,
eshwar

LINK

kunwar January 7, 2013, 3:02 am

I am getting some junk charector as a output after running the converted binary of the source shell script.

LINK

Gaby April 25, 2013, 1:13 am

Hi,

Thanks a lot, this applications is very useful (on my Linux machine), but I have problems to use shc in Mac OS. Not works. I compiled this one and ejecute the

My OS is MAC OS X Lion.

Can you help me, please?

Thanks!!

LINK

Syafeuq May 11, 2013, 3:31 am

After i encrypted it my script did not working. Is it shc still available to use?

[root@my shc-3.8.7]# ./op.sh.x

[8]+ Stopped ./op.sh.x

LINK

Manish September 22, 2013, 1:41 am

bash-2.04$ gcc aaa.sh.x.c
bash-2.04$ ./a.out
HI,

While i am encryted the script. Same will be given below error. In first statement (db2 connect to manis) is successfull but in 2nd statement (db2 “select count(1) from amnish.test”) given below error after establishing the connection successfully.

Database Connection Information

Database server = DB2/AIX64 9.5.9
SQL authorization ID = MANISH
Local database alias = MANISH

SQL1024N A database connection does not exist. SQLSTATE=08003

whyitellu November 13, 2013, 1:51 am

@Syafeuq

try to compile programe with shc -v -r -T -f /op.sh

it will probably solve the problem

LINK

Sepahrad Salour February 20, 2014, 8:37 am

Hi,

Thanks for your great article 🙂

LINK

naveen February 21, 2014, 10:23 pm

thanks...

LINK

Conti April 2, 2014, 11:56 pm

warning! shc will not protect your sourcecode!

After executing the “protected script” run:

ps aux | grep “name of shc compiled script”

You will notice it uses a single process to execute the entire contents of your script and it is now displayed in it’s entirety in your shell window. You can then simply find and replace ^J with \n and ^I with \t and you have the full formatted source code.

technique is irreversible but the point is to at least make it complex enough that it becomes more effort than it is worth to a would be code thief.

shc however, can be reversed in seconds as I have just described above.

LINK

Ziegfred May 16, 2014, 3:25 am

Hi, Thanks for this post. Btw, Is this also applicable in UNIX format?
Thanks

LINK

rinkupal June 5, 2014, 10:19 am

when encryption done it works but after some hours it not work
sometimes says expired or sometimes weird out

just used sic -v -f script.sh in terminal

any help

LINK

Justin Buser June 26, 2014, 6:29 am

Obviously not a big deal but in step 1 the following:

tar xvfz shc-3.8.7.tgz

should actually be:

tar -xvzf shc-3.8.7.tgz

or any derivation thereof provided that the f comes last as tar looks for the name of the file to extract after the f switch. Nit-picky to be sure but I just thought I'd

Ap.Muthu October 1, 2014, 7:18 am

wget http://www.datsi.fi.upm.es/~frosal/sources/shc-3.8.9.tgz
tar xzvf shc-3.8.9.tgz
cd shc-3.8.9
tar xzvf shc-3.8.9.tgz

v3.8.7 is nolonger available.
wget does not seem to work but browser download does.
Only if the -T switch is used to compile, does the executable produced run in debian squeeze (i386) in an OpenVZ container.

LINK

Elgrande October 13, 2014, 4:10 am

When using with expect script i can see password of “encrypted” script
via ps -ef

LINK

Lorenzo a November 26, 2014, 6:22 am

Hi.
I don’t know if this is the right place where I can ask about shc and his functionalities but I have a question about this product. If this is not the right place where can I ask for my problem?
The bug related to the script size limitation and _SC_ARG_MAX. I get this error when I do a ‘make test’. However after a ‘make’ the program is compile successfully (and the program shc is created). There is a method to solve this bug?
Thanks in advance and sorry if this is not the right place to ask of the shc program.

LINK

reliable December 8, 2014, 6:45 am

While running ./test.sh -x i am getting below error

Walker December 8, 2014, 7:04 am

Two questions

1. first is when I run make, it gives me
———————
*** ?Do you want to probe shc with a test script?
*** Please try… make test
———————

and when I run make test, some error occurs. When there should be .sh file in my PATH?
———————
*** Running a compiled test script!
*** It must show files with substring “sh” in your PATH…
./match.x sh
make: *** [make_the_test] Killed: 9
———————

2. another question is that
does this supports mac system? I though unix based mac os x should be working fine with shc, but after I run shc, the generated file (.x) just give me kill -9 as output.
not sure what is wrong

LINK

Vikas February 2, 2015, 5:27 am

Hi,
I am unable to execute converted script on CentOS 6
shc -rv -f ../script-api-add-device.sh
shc shll=sh
shc [-i]=-c
shc [-x]=exec ‘%s’ “$@”
shc [-l]=
shc opts=

[root@localhost shc-3.8.9]# cd ..
[root@localhost etc]# ./script-api-add-device.sh.x

[20]+ Stopped ./script-api-add-device.sh.x
[root@localhost etc]# uname -a
Linux localhost.localdomain 2.6.32-431.el6.x86_64 #1 SMP Fri Nov 22 03:15:09 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux

LINK

showain February 25, 2015, 1:23 pm

i encrypted my file but i have problem
“test.sh” its my file name //original code
test.sh.x // the encrypted shell script in binary format
test.sh.x.c

how can i set encrypted file to original code (test.sh.x to test.sh)
please help me fast

LINK

dedicaid March 4, 2015, 5:07 pm

./mach.x
works fine
/bin/sh ./match.x
fails cannot execute binary file
usually when called via screen or php
ive used this years ago dont recall this being a problem is it new?

LINK

test March 11, 2015, 2:58 pm

[root@localhost mnt]# ./abc.sh
hi

[1]+ Stopped ./abc.sh.x
[root@localhost mnt]# killall -9 abc.sh.x
[1]+ Killed ./abc.sh.x
[root@localhost mnt]#

hi
i am getting the above error while running the encrypted scriptkindly help me

LINK

Piyush Sharma May 30, 2015, 7:46 pm

Get an error:
[2] + stopped ./dnsfull.sh.x

LINK

Reza June 6, 2015, 11:18 am

Hi this is very nice !
but i have a problem. thats is:
root@static [~]# ./tool.sh

[1]+ Stopped ./tool.sh
root@static [~]# ./tool.sh

[2]+ Stopped ./tool.sh
root@static [~]# ./tool.sh

[3]+ Stopped ./tool.sh

i don’t know what i do, please help me
thanks

LINK

I see many of you getting [x] Stopped message while executing the *.x script. This indicates, your script was passed for a while in background, similar to pressing “Ctrl+z”. This would result the same when you execute script.

to make it run in “foreground” again, just run the below command you would see the script executes as defined and you will get the desired o/p,

fg <>

where <> is the number you see in [x] stopped message. 🙂
eg:

fg 1

Cheers!
NSP

LINK

JFK September 2, 2015, 1:19 am

This is for those who has encountered the + Stopped issue.

When you run the actual script, no issue at all.
When you run the encrypted script, there is a [1]+ Stopped which means it is passing the script to be run as a background (similar to when you press [CTRL+Z].
You need to enter ‘fg’ twice in order to continue with the script.

The solution that I have is that you should compile the script with the ‘-T’ parameters.

Instead of compiling with # ./shc -r -f test.bash
Use this instead # ./shc -r -T -f test.bash

Cheers,
JFK

LINK

i often encountered “[x]+ Stopped”.
but i found a significant difference among shc version.
3.8.9b distribued on Francisco’s web site: often
3.9.2 distributed on github: never

according to “CHANGE”, 3.8.9b works as daemon.

LINK

Josh January 12, 2016, 7:44 pm

Walker, I have the same issues as you.

The encrypted file results in “Killed: 9”

LINK

Gowtham May 15, 2016, 2:44 am

my bash script ‘s binary not working!
$ ./sudo-gdb.x
./sudo-gdb.x: Operation not permitted
Killed

$ sudo-gdb
GNU gdb (Ubuntu 7.11-0ubuntu1) 7.11
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type “show copying”
and “show warranty” for details.
This GDB was configured as “x86_64-linux-gnu”.
Type “show configuration” for configuration details.
For bug reporting instructions, please see:
.
Find the GDB manual and other documentation resources online at:
.

Inadget February 22, 2017, 7:31 pm

Hello,

Thx for article, it's fine.

I've try this :
fct.sh
#!/bin/bash
function hello() {
echo "hello "$1
}

main.sh
#!/bin/bash

. ./fct.sh

hello John

This not work because the file fct.sh is an executable ...
I've try with export -f hello too, but not working.

Someone can help me ?

Thx a lot 🙂
Regards

LINK

Alejandro Antonio García Garay May 16, 2017, 12:40 pm

I need help. When I tray to install with make a error occurs :

make
cc -Wall shc.c -o shc

LINK

Next post: How to Calculate IP Header Checksum (With an Example)
Previous post: Intro to DOCSIS Architecture, CM CMTS Protocol for Cable Modems

RSS | Email | Twitter | Facebook | Google+

Search

EBOOKS

Free
Linux 101 Hacks 2nd Edition eBook - Practical Examples to Build a Strong Foundation in Linux

Bash 101 Hacks eBook - Take Control of Your Bash Command Line and Shell Scripting

Sed and Awk 101 Hacks eBook - Enhance Your UNIX / Linux Life with Sed and Awk

Vim 101 Hacks eBook - Practical Examples for Becoming Fast and Productive in Vim Editor

Nagios Core 3 eBook - Monitor Everything, Be Proactive, and Sleep Well

The Geek Stuff
16,589 followers

Follow Page Share

POPULAR POSTS

The Ultimate Bash Array Tutorial with 15 Examples

3 Steps to Perform SSH Login Without Password Using ssh-keygen & ssh-copy-id

Unix Sed Tutorial: Advanced Sed Substitution Examples

UNIX / Linux: 10 Netstat Command Examples

The Ultimate Guide for Creating Strong Passwords

6 Steps to Secure Your Home Wireless Network

Turbocharge PuTTY with 12 Powerful Add-Ons

CATEGORIES

Linux Tutorials

Vim Editor

Sed Scripting

Awk Scripting

Bash Shell Scripting

Nagios Monitoring

OpenSSH

IPTables Firewall

Apache Web Server

MySQL Database

Perl Programming

Google Tutorials

Ubuntu Tutorials

PostgreSQL DB

Hello World Examples

C Programming

C++ Programming

DELL Server Tutorials

Oracle Database

VMware Tutorials

troublesh ooting tips and tricks on Linux, database, hardware, security and web. My focus is to write articles that will either teach you or help you resolve a problem. Read more about Ramesh Natarajan and the blog.

drop me a line to say hello!.

Follow us on Google+

Follow us on Twitter

Become a fan on Facebook

Sed and Awk 101 Hacks eBook

Vim 101 Hacks eBook

Nagios Core 3 eBook