

Forum:[Hack Forums](#)

Thread: [shc - Generic shell script compiler \[Protection\]\[*nix\]](#)

Loki123 - March 13, 2014, 6:21 pm

What is shc and what will you be showing us?

For those whom don't know what shc is you might want to take a look at this

<http://www.datsi.fi.upm.es/~frosal/sources/shc.html>

Basically it is to protect your shell script. Your Bash, csh or other script. I will then be showing you two different things. How to protect the file and then how to get it back in plain text. Thus it isn't secure but does somewhat a fine job protecting something that isn't supposed to be easily read.

Downloading and installing shc

wget <http://www.datsi.fi.upm.es/~frosal/sources/shc-3.8.7.tgz>

```
tar -xvzf shc-3.8.7.tgz
```

```
cd shc-3.8.7
```

```
make
```

```
./shc
```

If the last one ran successfully you should see this on your terminal screen:

PHP Code

```
<?php
```

```
$ shc
```

```
shc parse(-f): No source file specified
```

```
shc Usage: shc [-e date] [-m addr] [-i iopt] [-x cmdnd] [-l lopt] [-rvDTCAh] -f script
```

```
$
```

Protecting the shell script

Now at this point we should have a script we want to protect. After shc has done its job you should receive a binary file with the .x extension.

To protect a file simply do

PHP Code

```
<?php
```

```
$ ./shc -f /script_path
```

There are a lot of options. For example expiration date and more that can be found in the manual.

Retrieve plain text from binary

So.. what happens when we run the binary file we created it decrypts and loads the script into memory when started. So right after we start the binary file we can segfault it and retrieve the script from the core dump. Sounds and is simple..

You might want to take a look at core dumps, what it is and the usage of them. They are useful for debugging in several situations.

http://en.wikipedia.org/wiki/Core_dump

So, by default most linux distributions have turned off core file creation. To check if it is turned on or off simply do

PHP Code

```
<?php
```

```
$ ulimit -c
```

and if the output is 0, it is tuned off. To set it to say, 50000 bytes add 50000 after the -c

PHP Code

```
<?php
```

```
$ ulimit -c 50000
```

If the script is too large you can adjust the core file size.

To start the binary and segfault it right away you would do it like this :

PHP Code

```
<?php
```

```
./script.sh.x& ( sleep 0.02 && kill -SIGSEGV $! )
```

sleep 0.02 should be enough time to start it up and decrypt the original script. The \$! variable contains the pid of the last background process started so we can kill it with the segmentation fault signal SIGSEGV.

Next thing we do is this :

PHP Code

```
<?php
```

```
cat core | strings >plain_text
```

Now if you would open up plain_text file that was just created and you should find your script somewhere inside there. You could try to search for BASHPID.. mine was inside it. Thus :

Code BASHPID

SCRIPT

...

BASHPID

Hope it helped some of you. Enjoy. ?

P.S

if anyone knows about something newer that achieves this in a better and more secure way. Post it! I'd like to take a look at it.

?CARUs L?VE?™ - March 13, 2014, 6:29 pm

Nice tutorial you got here. I have no use for it, but still nice. Good work.

Loki123 - March 13, 2014, 6:35 pm

(Mar 13, 2014 06:29 PM)?M?LE ZE?O Wrote: [\[CITE\]](#) Nice tutorial you got here. I have no use for it, but still nice. Good work.

haha.. I don't expect many users to have use for this. I just thought it was interesting and decided to share it with you and those whom might find it also interesting or find a use for it. ?