



Mission Name

ReconCar

History Context

Claire and Ethan are inside a cave trying to search the Recon CAR for the bug left by Dr. Pinche.

Technical High-Level Overview

A Smart Fridge forensic image is provided to the player. The goal of this challenge is to identify which bug was left by Dr.Pinche inside the Reconcar. In this scenario player will be investigating filesystem to locate any clue to a backdoor. Finally, considering filesystem dates and forensic artifacts related to persistence, the challenge could be solved.

Short Description

You're going to analyse ReconCar system. This time ReconCar has a Raspberry Pi connected to infotainment system. Your goal will be to locate any identifier which involves Dr.Pinche's malicious activity.

Mission Description

The goal of this challenge is to identify which bug was left by Dr.Pinche inside the ReconPr. In this scenario you will be investigating a filesystem to locate any clue to a backdoor or other method which involves Dr.Pinche's malicious activity. Your goal will be to locate the identifier, which leads that Dr.Pinche tampered ReconCar.

Location

RECON CAR - AIR / ALTAI MOUNTAINS



Tools

- Autopsy
- Openssl

Questions

Which program was used to modify the profile?

- nano

What was the last command launched by the user pi related to store passwords?

- Impas gui

Items

1. Use Autopsy to mount evidence.
2. Identify all persistence Linux artifacts.
3. Use comments as passwords.

Write Up

First of all, player will have to mount the evidence provided using Autopsy:

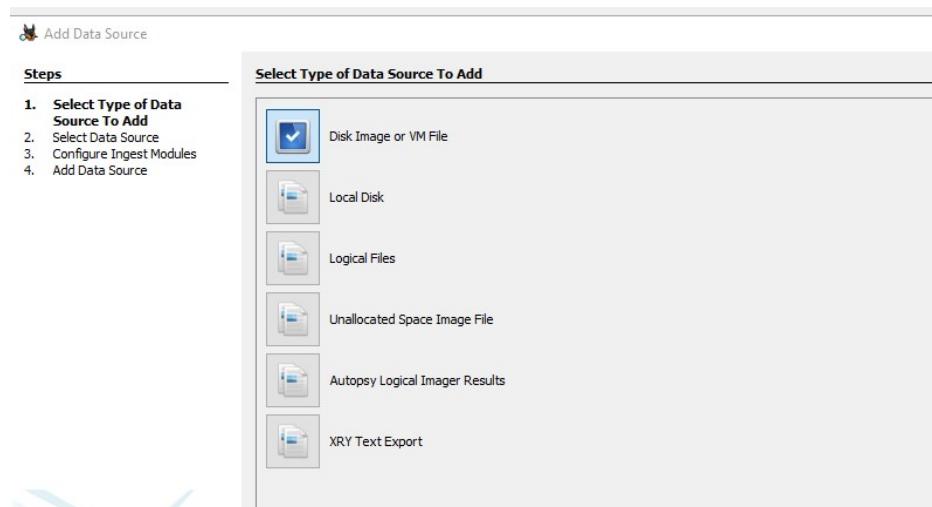


Figure 1

In this scenario, there is no need to index the evidence, just explore it. So, unselect all ingest modules,

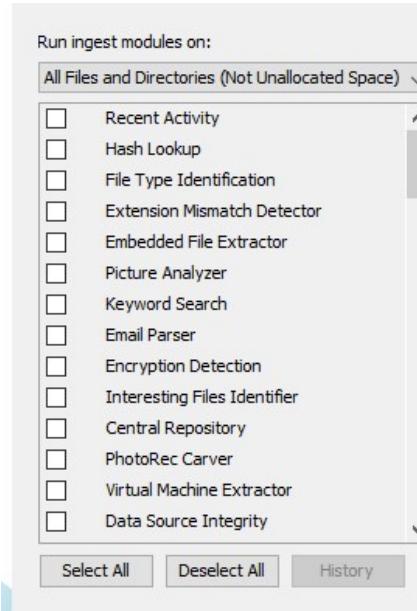


Figure 2

Then, player must select filesystem, in order to check files and folders:

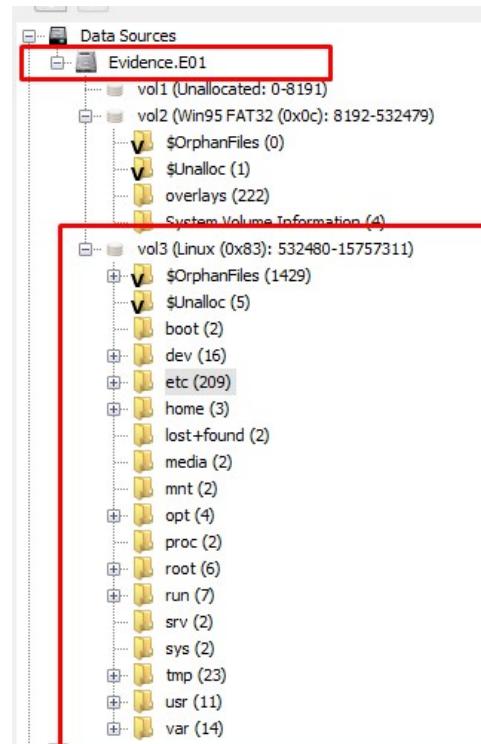


Figure 3



One possible wa, would be to analyse the bash_history file, to check commands launched, but as you can see below, this way could raise the complex of the case: there are two tools related to save secure passwords password an impass.

The screenshot shows a hex editor interface with several tabs at the top: Hex, Text, Application, File Metadata, Context, and Results. The Text tab is selected, displaying a list of commands. Below the tabs is a search bar with "Page: 1 of 1" and "Page" buttons. The command list includes:

```
uname /a
uname -a
history
openssl
ifconfig
service ssh status
netstat -putan
sudo netstat -putan
ping 192.168.1.1
password
password -h
impass
impass gui
ifconfig
ls -l
shutdown -h now
sudo shutdown -h now
```

Figure 4

Considering the challenge description, Dr.Piche could hide something into ReconCard, so player could think about all persistence artifact in Linux. One place to check would be /etc/shadow to see, if there is any user related to Dr.Pinche.

The screenshot shows a terminal window with the following text output:

```
sync:*:18754:0:99999:7:::
games:*:18754:0:99999:7:::
man:*:18754:0:99999:7:::
lp:*:18754:0:99999:7:::
mail:*:18754:0:99999:7:::
news:*:18754:0:99999:7:::
uucp:*:18754:0:99999:7:::
proxy:*:18754:0:99999:7:::
www-data:*:18754:0:99999:7:::
backup:*:18754:0:99999:7:::
list:*:18754:0:99999:7:::
irc:*:18754:0:99999:7:::
gnats:*:18754:0:99999:7:::
nobody:*:18754:0:99999:7:::
systemd-timesync:*:18754:0:99999:7:::
systemd-network:*:18754:0:99999:7:::
systemd-resolve:*:18754:0:99999:7:::
_apt:*:18754:0:99999:7:::
pi:$6$CbZcwof0Y90cCJZL$onwoiCoXybme.6GQrfqTjbFDqqHyyR8iMrNN/yCdZ206cXGQx209Ou9mblJzzhogf069cIBhzmCCPkGH11xk/1:18754:0:99999:7:::
messagebus:*:18754:0:99999:7:::
_rpc:*:18754:0:99999:7:::
statd:*:18754:0:99999:7:::
sshd:*:18754:0:99999:7:::
avahi:*:18754:0:99999:7:::
lightdm:*:18754:0:99999:7:::
rtkit:*:18754:0:99999:7:::
pulse:*:18754:0:99999:7:::
saned:*:18754:0:99999:7:::
hplip:*:18754:0:99999:7:::
colord:*:18754:0:99999:7:::
systemd-run@unit=11+18754.....
```

Figure 5



Other place to check would be /etc/crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# Example of job definition:
# .----- minute (0 - 59)
# .----- hour (0 - 23)
# | .---- day of month (1 - 31)
# | | .-- month (1 - 12) OR jan,feb,mar,apr ...
# | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | |
# | |
L>>t>>t>>
user-name command to be executed
17 *      * * *      root    cd / && run-parts --report /etc/cron.hourly
25 6      * * *      root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6      * * 7      root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6      1 * *      root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

Figure 6

Another place would be /home/pi/.bash_rc:

```

```
esac
enable color support of ls and also add handy aliases
if [-x /usr/bin/dircolors]; then
 test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
 alias ls='ls --color=auto'
 #alias dir='dir --color=auto'
 #alias vdir='vdir --color=auto'
 alias grep='grep --color=auto'
 alias fgrep='fgrep --color=auto'
 alias egrep='egrep --color=auto'
colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'
some more ls aliases
#alias ll='ls -l'
#alias la='ls -A'
#alias l='ls -CF'
Alias definitions.
You may want to put all your additions into a separate file like
~/.bash_aliases, instead of adding them here directly.
See /usr/share/doc/bash-doc/examples in the bash-doc package.
if [-f ~/.bash_aliases]; then
 . ~/.bash_aliases
enable programmable completion features (you don't need to enable
this, if it's already enabled in /etc/bash.bashrc and /etc/profile
sources /etc/bash.bashrc).
if ! shopt -o posix; then
 if [-f /usr/share/bash-completion/bash_completion]; then
 . /usr/share/bash-completion/bash_completion
 elif [-f /etc/bash_completion]; then
 . /etc/bash_completion
 fi
fi

```

**Figure 7**



Finally we've found a clue in /home/pi/.profile

```
~/.profile: executed by the command interpreter for login shells.
This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
exists.
see /usr/share/doc/bash/examples/startup-files for examples.
the files are located in the bash-doc package.
the default umask is set in /etc/profile; for setting the umask
for ssh logins, install and configure the libpam-umask package.
#umask 022
if running bash
if [-n "$BASH_VERSION"]; then
 # include .bashrc if it exists
 if [-f "$HOME/.bashrc"]; then
 . "$HOME/.bashrc"
 fi
set PATH so it includes user's private bin if it exists
if [-d "$HOME/bin"] ; then
 PATH="$HOME/bin:$PATH"
set PATH so it includes user's private bin if it exists
if [-d "$HOME/.local/bin"] ; then
 PATH="$HOME/.local/bin:$PATH"
cat treh.txt.enc | openssl aes-256-cbc -d -a); exit
#Always use the same:cjklmno
```

Figure 8

This commands inside the **.profile** file, could have been modified by Dr.Pinche:

The screenshot shows a hex editor interface with the file 'treh.txt.enc' open. The file contains several lines of binary data followed by the highlighted command. The command is enclosed in a red box.

| Page:       | 1 of 1                                                           | Page | Go to Page: | Jump to Offset | 0 | Launch in HxD |
|-------------|------------------------------------------------------------------|------|-------------|----------------|---|---------------|
| 0x00000000: | 55 32 46 73 64 47 56 6B 58 31 2F 64 49 53 74 45 U2FsdGVkX1/dISxE |      |             |                |   |               |
| 0x00000010: | 4E 79 4B 45 32 6B 46 64 43 5A 77 6C 6F 58 63 30 NyKE2kFdCZwloXc0 |      |             |                |   |               |
| 0x00000020: | 73 4E 4C 38 44 62 49 39 35 4C 6E 4D 55 75 32 46 sNL8DbI95LnMUu2F |      |             |                |   |               |
| 0x00000030: | 52 53 63 33 71 41 4B 52 35 6E 55 45 77 52 73 64 RSc3qAKR5nUEwRsd |      |             |                |   |               |
| 0x00000040: | 0A 79 68 5A 56 34 44 39 75 56 6B 4E 41 70 4C 6E .yhZV4D9uVkJApLn |      |             |                |   |               |
| 0x00000050: | 68 47 49 67 58 73 58 54 2B 55 4D 43 62 32 62 52 hGIgXsXT+UMCb2bR |      |             |                |   |               |
| 0x00000060: | 62 67 47 6B 51 44 76 45 78 59 41 4D 3D 0A bgGkQDvExYAM=.         |      |             |                |   |               |

Figure 9

If we check file treh.txt.enc

```
jmma@demowindows:/mnt/c/THREATIA/C3-M2/scripts$ file treh.txt.enc
treh.txt.enc: openssl enc'd data with salted password, base64 encoded
```

Figure 10



So, the file contains encrypted data. If we launch the command:

- cat treh.txt.enc | openssl aes-256-cbc -d -a

We will need the password to open the encrypted piece of information. So, the key is the comment on the profile file:

```
cat treh.txt.enc | openssl aes-256-cbc -d -a); exit
#Always use the same:cjklmno
```

Finally we can open the container and get flag: 93203023208

```
jmma@demowindows:/mnt/c/THREATIA/C3-M2/scripts$ cat treh.txt.enc | openssl aes-256-cbc -d -a
enter aes-256-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
#!/bin/bash
echo "Dr.Pinche was here: 93203023208"
jmma@demowindows:/mnt/c/THREATIA/C3-M2/scripts$
```

Figure 11

## Flag Information

flag{93203023208}