Exercises in Secure Mobile Systems

FH Hagenberg, WS 2022
FH-Prof. DI Dr. Erik Sonnleitner

Exercise 5: Sniffing, App Analysis & CTF

Exercise 5.1: Sniffing

(1) **Sniffing with tcpdump**
Create syntactically and semantically valid command-lines for the `tcpdump` packet sniffer, in order to fulfill the following goals:

(1.1) Capture all packets at a network interface, which use the UDP protocol and any destination port from 1 to 1024.
(1.2) Capture all broadcast and multicast packets.
(1.3) Capture all packets, whose destination is host is given via a particular IP address (e.g. 192.168.1.1), except for packets to port 80 (http) or port 23 (telnet).
(1.4) Find a way to capture all TCP packets, where the RST flag is set (abnormal connection closure).
(1.5) Set up tcpdump to capture HTTP packets only, and save the captured packets to a file. Browse the web for a few minutes and stop capturing. Then use the `tshark` utility to open and print the captured packets, and use common shell utilities to extract a list of all distinct destination hosts/IPs.

Make sure to test your filters and create screenshots of captured packages.

Exercise 5.2: Application Analysis

(2) **Application analysis: Preparation**
A variety of tools are listed below, which can be used to analyze the behavior of programs. If not specifically given, the Arch/Manjaro package is named similarly.

- Quick tools overview + example usage:
  - **`$ ltrace <executable>`** → trace library calls of a program
  - **`$ strace <executable>`** → trace system calls (to OS) of a program
  - **`$ dstat 1`** → view continuous system stats (memory, block I/O, interrupts, context switches, ...)
  - **`# lsof`** → list open files and sockets for processes
  - **`# netstat -atnp`** → view open network connections for processes (part of package **`net-tools`**)
  - **`$ strings <file>`** → extract printable characters from binary and list all of them (part of package **`binutils`**)
  - **`$ xxd <file>`** → print hex-dump of given file (part of package **`vim`**)
  - **`$ objdump -d <binary>`** → print disassembly of binary

Moreover, there are several very powerful tools for application analysis, including debuggers (e.g. **`gdb`**, **`gdb-peda`**) and in-depth reverse engineering tools (like **`radare2`**) – all of which have a considerable learning curve but will improve analysis capability greatly.

Exercise 5.2: Application Analysis

(2) **Application analysis: The actual assignment**

- Download the mysterious application
  - for 32-bit Linux: **`https://delta-xi.net/download/mystery32`**
  - for 64-bit Linux: **`https://delta-xi.net/download/mystery64`**
  - Note: Of course, the application doesn't do any harm to your computer and can be safely run.
- There are **3** things to find: a magic password, and 2 operations the program executes.
- Use any CLI tool you like to get the answers. Only few are needed and there are multiple ways to get the answers.
- **Submission**: Explain what you have found, what tasks the application performs in your opinion, and how you came to your judgement. Include screenshots!

Bonus: CTF

(3) **Capture the flag challenge: SMS-CTF**
A *Capture the Flag* is traditionally a hacking event, where the goal is to hijack a host/network, and find some kind of virtual "flag". On the *SMSCTF* VM, two types of user accounts exist:

- `levelXX` is the user account for level XX, which you can use to log on to the system for solving that level. The password is the same as the username.
- `flagXX` is the user account for level XX, whose system permissions you will need to acquire in order to "capture the flag".

On most levels, you need to execute the (already installed) program `/bin/getflag` with the permissions of `flagXX` user which corresponds to your current level. You main goal is to find out, how to acquire those permissions. You can execute `getflag` as `levelXX` user too, but that doesn't get you the flag. Smart thinking is required.

On a few other levels, your goal is to read the contents of a particular file called the token. It is a plaintext file containing a character sequence – being able to read this file means you have mastered the level.

Your assignment is, to get the flags on levels 00 through 05. Hand-in a **detailed, step-by-step description** on how the flag can be retrieved, and how you managed to find the solution. The CTF ISO is available via **`https://delta-xi.net/download/sms_ctf_v5.iso`**, and the challenges are listed at **`https://delta-xi.net/smsctf/`**.

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA