Security in Mobile Systems - UE/EX

FH Hagenberg, WS 2017
DI Dr. Erik Sonnleitner

Exercise 5: Sniffing, App Analysis & CTF

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Exercise 5: Sniffing, App Analysis & CTF

(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 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.

Exercise 5: Sniffing, App Analysis & CTF

(2) Application analysis: Preparation

- Install the following packages on your machine (package names valid for Arch/Antergos): `ltrace`, `strace`, `vmstat` (package `procps-ng`) `lsof`, `dstat`, `netstat` (package `net-tools`), `strings` (package `binutils`) and `xxd` (part of package `vim`)

- Package installation notes:
  - You may need to update your system before, using `pacman -Syu --force`
  - You can install all packages at once: `pacman -S ltrace strace ...`
  - On non-Arch systems, packages may be named differently, and may or may not be already installed.

- Quick tools overview + example usage:
  - `ltrace <executable>` → trace library calls of a program
  - `strace <executable>` → trace system calls (to OS) of a program
  - `vmstat 1` → view continuous system stats (memory, block I/O, interrupts, context switches, ...)
  - `dstat 1` → same as vmstat, but nicer interface
  - `sudo lsof` → list open files and sockets for processes
  - `netstat -atnp` → view currently open network connections for processes
  - `strings <file>` → extract printable characters from binary and list all of them
  - `xxd <file>` → print hex-dump of given file

Exercise 5: Sniffing, App Analysis & CTF

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

- Download the mysterious application
  - for 32-bit Linux: **`http://dl.delta-xi.net/mystery32`**
  - for 64-bit Linux: **`http://dl.delta-xi.net/mystery64`**
- Use the tools above (or ones we have already used) to get a grasp on what this application is doing, and try to find the magic password.
- The previous slide only shows *one* example of how to use each tool – however, several tools have many more functions which can be used; go through the manual pages if necessary.
- The application does not actually deal with data – it just simulates certain actions.
- Note: You really don't need *every* tool above - it's just a collection of basic tools for application testing; 1-2 may be enough, if used correctly. Also, there is more than one way 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!

Exercise 5: Sniffing, App Analysis & 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. Grab the ISO from me and go to `http://smsctf.delta-xi.net`.

Exercise 5: Sniffing, App Analysis & CTF

(5) **Bonus**: Change the MAC address of your primary network interface to any valid random value. Force the Linux kernel to enable packet routing, and use `iptables` to enable masquerading/NATting for one particular client IP address (you will need to use the `-j REDIRECT` iptables target to forward traffic to another local port, where you continue data investigation).

Once your masquerading is set up, the client which you route and masquerade packets for, has to set its default gateway to your IP address.

Learn how to use the Python tool `sslstrip`, which can peel off the SSL layer from encrypted packets (sslstrip doesn't necessarily work on *every* HTTPS connection – but on those with weak configuration).

Then use the `ettercap` tool to capture credentials for a particular service used by the client (website login, e-mail, etc).

Hand-in a step-by-step guide how to make the setup work, and explain in detail how every tool works (including used options, etc), as well as how they work together to perform the task.