CS2107 Tutorial 7
(Network Security + Privilege Escalation)

School of Computing, NUS

18–22 October 2021

Part A: Network Security

Relevant Networking Background

Some of you have not completed Networking module yet, and thus may not familiar with the role of port numbers in a client-server connection and communication. A service offered by a server can be accessed via some predefined port number. For instance, consider a SMTP email server, which listens to port number 25. If a client wants to communicate with the server for a SMTP communication, the client sends packets to the server's IP address at port number 25. The server responses by sending packets to the client's IP address at another port number selected by the client, which is greater than 1,023.

Hence, in summary, if the client sends packets to the SMTP server, the destination port of these packets must be 25, but the source port can be any number greater than 1,023. Conversely, packets sent by the SMTP server have source port 25, and their destination port is the one specified by the client.

Also notice that a source IP address can be easily forged. A reply packet sent to this spoofed IP address is delivered to the address, and is therefore generally inaccessible to the packet spoofer. In this tutorial, we assume that a sent packet will reach its destination as specified in the destination IP address. Man-in-the-middle between a firewall and a host is out-of-scope in the firewall design. It needs to be separately dealt with by other measures.

Questions

A-1. (*Firewall design*:) Suppose you are the system administrator of a new secondary school, and your first task is to design the school's network and its firewalls. You have decided to have 2 firewalls to protect your network.

The machines in the network include:

(a) `Lab`: There is a total of ∼100 machines in a few labs for students to prepare their reports, search for materials in the Web, etc. There are also network printers in the labs.

(b) `Teachers`: Every teacher has a PC in the teacher room. The teachers use the PCs to enter students' grades, send/receive emails, prepare teaching materials, print exam questions, perform Web searches, etc. There are also network printers in the teacher rooms.

(c) `Web-server`: the school's Web server.

(d) `Email-server`: the school's SMTP email server.

(e) `SQL-server`: This is a SQL server that stores the student database. Some information can be accessed via a Web-based application hosted in the Web server. For example, a Web-based application can allow the students to update their mobile phones. Some other information can be accessed only by the teachers.

You need to configure firewall rules in the firewall table of each deployed firewall. Each rule occupies a row in the table. The fields of each rule, together with their possible values, are as follows:

No	*Source IP*	*Dest IP*	*Source port*	*Dest port*	*Protocol*	*Direction*	*Action*

(a) *No*: the serial no of each rule.

(b) *Source IP*: a set of source IP addresses. It can be specified as a boolean expression over the above-listed predefined names, e.g. `Email-server`, and other newly-defined names, e.g. `Internal` to refer to all machines in the school.

(c) *Dest IP*: a set of destination IP addresses. Likewise, its boolean expression can be defined over the predefined names listed above, or any newly-added names.

(d) *Source port, Destination port*: the source and destination port number, respectively. Notice that some services use fixed predefined port numbers. The firewall can recognize some well-known port numbers, like `HTTP` (port no 80), `HTTPS` (port no 443), `SMTP` (port no 25), `LPR` (a network printing protocol, which listens to port no 515), and `SQL`.

(e) *Protocol*: the protocol used, which can be: TCP, UDP, ICMP, IP. Note that various applications, such as HTTP, HTTPS, SMTP, LPR, are connection-oriented applications and run on top of TCP. Yet, there are some UDP-based applications, such as DNS (for query and response).

(f) *Direction*: this can be `IN` or `OUT`. A firewall divides the network into two sectors, say S_1, and S_2. This field indicates which direction a referred packet is moving, either from S_1 to S_2, or the other way. Your design has to indicate the meaning/context of `IN` and `OUT`.

(g) *Action*: either `Block` or `Allow`.

Similar to the explanation given in the lecture notes, when a packet arrives, the action of the first rule that matches the packet applies. For example, the following three rules

No	*Source IP*	*Dest IP*	*Source port*	*Dest port*	*Protocol*	*Direction*	*Action*
1	`Web-server`	*	`HTTP`	*	`TCP`	`OUT`	`Allow`
2	*	`Web-server`	*	`HTTP`	`TCP`	`IN`	`Allow`
3	*	*	*	*	*	*	`Block`

allow packets to be sent from (into) the `Web-server` to (from) any other IP addresses, and block everything else. Note that the symbol "*" means "any", and it matches anything.

Remarks on the Requirements:

(a) It is important to prevent cases where student exam questions get mistakenly printed in the `Lab`.

(b) It is also important to protect the SQL server.

(c) We know that source IP addresses can be spoofed. The school is worried that some students are running some hacking tools that generate spoofed source IP addresses. Hence, the school wants to block outbound packets from the school that do not have legitimate source IP addresses.

(d) In this question, we ignore the detailed issue of routing. So, we do not consider the Internet gateway and Network Address Translation (NAT). For simplicity, we just assume that all machines use "public" IP addresses.

(e) Will you increase the security of the school network if you use 3 firewalls instead? You can think of a good strategy of placing the third firewall, and consider what machines to be placed on the new network segment.

Do you think the partition of the networks below is reasonable, and the given firewall rules below are appropriate for the school's need?

We can place two firewalls, F_1 (front-end/outer firewall) and F_2 (back-end/inner firewall), to separate the public Internet and the school's internal network (`Internal`). In between these two firewalls, we designate the school's DMZ, where the `Web-server`, `Email-server`, `Lab`, and `Lab-printers` are placed. We put `Lab` in the DMZ instead of `Internal` since:

- The `Lab` PCs are considered not to contain any important data; and
- We want to segregate `Lab` and `Teachers` as required.

In `Internal`, we place `Teachers`, `Teacher-printers`, and `SQL-server`.

The network partitioning set-up thus looks like in the following diagram:

$$\texttt{Internal} \leftarrow (\text{IN})\ F_2\ (\text{OUT}) \rightarrow \quad \text{DMZ} \quad \leftarrow (\text{IN})\ F_1\ (\text{OUT}) \rightarrow \text{Internet}$$

The diagram above also indicates the context of `IN` and `OUT` directions on the two employed firewalls F_1 and F_2. We can configure the two firewalls with the rule sets shown in Table 1 and Table 2, respectively.

Remarks on the Firewall Configuration and Rules:

- Notice that the requirement (c) above, which aims to block outgoing packets with illegitimate source IP addresses (*egress filtering*), is automatically met by the given rule sets.
- Any other Firewall rules can be added as necessary, for examples those needed to allow DNS and HTTPS traffic.

No	*Source IP*	*Dest IP*	*Source port*	*Dest port*	*Protocol*	*Direction*	*Action*
1	Web-server	*	HTTP	*	TCP	OUT	Allow
2	*	Web-server	*	HTTP	TCP	IN	Allow
3	Email-server	*	SMTP	*	TCP	OUT	Allow
4	*	Email-server	*	SMTP	TCP	IN	Allow
5	Lab	*	*	HTTP	TCP	OUT	Allow
6	*	Lab	HTTP	*	TCP	IN	Allow
7	Teachers	*	*	HTTP	TCP	OUT	Allow
8	*	Teachers	HTTP	*	TCP	IN	Allow
9	*	*	*	*	*	*	Block

Table 1: Firewall rules for the front-end/outer firewall F_1.

No	*Source IP*	*Dest IP*	*Source port*	*Dest port*	*Protocol*	*Direction*	*Action*
1	SQL-server	Web-server	SQL	*	TCP	OUT	Allow
2	Web-server	SQL-server	*	SQL	TCP	IN	Allow
3	Teachers	*	*	HTTP	TCP	OUT	Allow
4	*	Teachers	HTTP	*	TCP	IN	Allow
5	Teachers	Email-server	*	SMTP	TCP	OUT	Allow
6	Email-server	Teachers	SMTP	*	TCP	IN	Allow
7	*	*	*	*	*	*	Block

Table 2: Firewall rules for the back-end/inner firewall F_2.

If so, answer the following questions:

(a) Which rule(s) allow Teachers to access external web servers?

(b) Which rule(s) block Lab from accessing an external SFTP server?

(c) What rules should you additionally set if you want to allow Lab to access external SFTP servers?

Solution

(a) Rules 3 and 4 of the inner firewall (F_2), as well as Rules 7 and 8 of the outer firewall (F_1) allow `Teachers` to send outgoing and receive incoming HTTP packets to and from external web servers.

(b) Rules 9 of the outer firewall (F_1) blocks `Lab` from accessing an external SFTP server.

(c) The following rules below can be added before Rule 9, e.g. as the new Rules 9 and 10:

No	*Source IP*	*Dest IP*	*Source port*	*Dest port*	*Protocol*	*Direction*	*Action*
9'	`Lab`	`*`	`*`	`SFTP`	`TCP`	`OUT`	`Allow`
10'	`*`	`Lab`	`SFTP`	`*`	`TCP`	`IN`	`Allow`

A-2. (*Firewall design*:) The lecture notes list a few types of firewall. The firewalls above inspect only a few important fields of network packets. Which type of firewall do they belong to?

Solution

The firewalls are *packet filtering firewalls*, which operate at the IP layer.

Part B: Privilege Escalation

Relevant Backgrounds

Shell program. Consider a UNIX-based OS and an executable program named `mysh`, which essentially is an existing command `sh` known as "shell". Suppose the file `file1` just contains the following two lines:

```
echo Happy Cat
ls -al
```

Running the program `mysh` with parameter `file1` will read and execute all commands contained in the file `file1`. Hence, executing the program:

```
$ mysh file1
```

will display the text `Happy Cat` (due to the command `echo Happy Cat`), and perform a directory listing (because of the command `ls -al`).

Backdoor program. Suppose an attacker, who is assumed to already have a local access, is given just a few seconds exercising your account. Can he plant a backdoor so that later he can come back to your account? The answer is yes. He can run the following commands to set up a set-UID shell program:

```
$ cp /bin/sh /tmp/mysh
$ chmod u+s /tmp/mysh
```

Notice that you are the owner of the planted shell program. Therefore, when later the attacker run the planted set-UID program, the program's effective UID will be your UID. (Notice, however, that a modern shell like bash/dash has an extra security protection as described below. Nonetheless, there is a way to bypass the protection as explained in the link below.)

Questions

B-1. Suppose Bob was an unhappy system administrator, and had a root access on a system (i.e. he knew the root password). He was going to quit his job soon. Bob was very unhappy, and wanted to plant a backdoor into the target system. Let us assume that Bob would either still be given a local user, or knew the password of an existing local user, or alternatively had created a local user that would go unnoticed. Bob wanted to make a very small change to the system so that later, when he logged in as a normal user, he could still issue commands with root privilege.

Bob knew that his root password would be changed immediately after he left, and that his home directory would be erased. In addition, the new system administrator would also compare the system-level directories, such as `/usr/bin`, to make sure that no malicious changes had been made.

Describe in principle how Bob could create the backdoor.

(*Hint*: by creating a program like `mysh` with an elevated privilege, and "hiding" that program somewhere in the system).

(*Note:* The above-mentioned method of using set-UID probably can't work in your favourite modern UNIX-based OS. This is because, due to the above security concerns, many OSes ignore the elevated effective UID when running shell scripts. This can be viewed as an "ad-hoc" patch to the security concern. Nevertheless, there are methods for Bob to work around it. See, for example, `http://koltsoff.com/pub/getroot/`. To understand the code better, you may also want to read `https://www.gnu.org/software/libc/manual/html_node/Users-and-Groups.html#Users-and-Groups`. Fundamentally, there is no way of stopping Bob. The "ad-hoc" patch only makes it slightly more difficult for Bob to create a backdoor.)

Solution

To successfully plant a backdoor shell, first of all, you need to understand the concept of a set-UID program, particularly a *set-UID-root* program. As discussed in the lectures, a set-UID-root program is an executable program owned by the root, and has its set-UID bit enabled. When it is invoked by a local non-root user, the process will run with its effective user ID (euid) set to the root, thus running with an elevated/escalated privilege.

To help you understand this better, let's take a loot at this `print-uids.c` code.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
   printf ("Real user id = %d, Effective user id = %d\n", getuid(), geteuid());
   return 0;
}
```

With a root privilege, you can compile the code, and then enable the executable's set-UID bit as follows: (Notice that in Ubuntu, you will need to first run the command `sudo su -` to switch to the root user.)

```
# gcc -o print-uids print-uids.c
# chmod u+s print-uids
```

The `print-uids` executable needs to have the following file attributes: `-rwsr-xr-x 1 root root 7468 Mmm dd hh:mm print-uids*`.

As a local non-root user, you can then run the `print-uids` executable, and inspect its output. Additionally, again as the root, you can disable the executable's set-UID bit; and subsequently observe the executable's output when it is invoked by a local non-root user. Do inspect the difference.

Solution (Continued)

Now, back to our given scenario, when Bob still has a root access, as the root, he can attempt to plant a copied shell as follows:

```
# mkdir -p /games/pokemon/scores
# cp /bin/sh /games/pokemon/scores/removescores
# chmod u+s /games/pokemon/scores/removescores
```

Unfortunately, when later Bob logs in as a non-root local user and executes `removescores`, the invoked shell won't run with a root privilege. You can check this by using `id` command. *Why is that so?*

As you can observe, in Ubuntu, `/bin/sh` actually points to `/bin/dash` (Debian Almquist shell), which is a lightweight `bash` variant. As mentioned above, even if we can invoke a set-UID-root shell, we will not get a root privilege since, as a security protection measure, `bash` automatically downgrades its privilege. This is possible, since if `bash` detects that it is executed in a set-UID process, then it will immediately change its effective user ID to the process' real user ID, thus essentially dropping the escalated privilege. (Note that `bash` in older versions of Ubuntu, e.g. version 12.04, does not have this protection measure.)

However, there are ways to bypass this protection measure. The *real root* can still invoke `bash` without its privilege gets dropped. Hence, if we can make `bash` run with its *real user ID* set to 0, then the root privilege will be retained.

Thus, instead of copying `/bin/sh`, as the root, Bob can just compile `getroot.c` to crate an executable named `/games/pokemon/scores/removescores`, and enable its set-UID bit. As you can see, `getroot.c` first turns the current *set-UID* process into a *real root* process by invoking, among others, `setuid(0)`. As explained in `https://www.gnu.org/software/libc/manual/html_node/Setting-User-ID.html#Setting-User-ID`, if the calling process is privileged, `setuid(0)` sets both the real and effective user IDs of the process to 0. Once both real and effective user IDs are set to 0, a shell `/bin/sh` is subsequently invoked. As a result, when later Bob executes the planted executable as a non-root local user, he will obtain a shell running with a root privilege as planned!

Additional Notes:
Notice that the technique above is just one way of creating a backdoor. There can be other ways, such as: setting a non-root local user to have UID 0, including a non-root local user into the `sudo` group, etc.

B-2. Terminology: *Insider threat.*

Solution

Please google the term. In the scenario above, Bob represents an insider threat.

— End of Tutorial —