

Class activity

A hands-on exercise from the class-activities appendix. The full text also appears in Appendix A of the book.

Accompanies *Computer Security: The Foundations*, Krerk Piromsopa, Ph.D., 2026.
<https://www.cp.eng.chula.ac.th/~krerk/books/ComputerSecurity/>

Authorized use only

These exercises involve real attack techniques. Carry them out only against systems you own, or that you have written permission from the owner to test. Unauthorized access, scanning or interception is a criminal offence under Thailand's Computer Crime Act and its equivalents elsewhere, whether or not any damage results. If you are unsure whether a target is in scope, ask your instructor before you start.

Activity 7: Secure Software / Simple Web Server

Overview

We will use Python in this exercise, although you may use any programming language. The goal is to appreciate the importance of *Secure by Design* by progressively hardening a minimalist web server.

The provided `simple_web_server.py` is a single-threaded HTTP server. While it serves its basic purpose, it is far from secure: one connection blocks all others (opening a denial-of-service vector), there is no access control (enabling elevation of privilege, information disclosure, and tampering), and there is no log (eliminating non-repudiation). Your task is to secure it.

Note: This is an educational assignment. Do not deploy it in any production environment.

Exercises

1. Here is the source code for `simple_web_server.py`. It is reproduced in full below, and can also be downloaded from the companion website.¹ Run it with:

```
$ python simple_web_server.py
```

Place an `index.html` file in the same directory. To access the server, open `http://localhost:8080/index.html` in a browser.

```
import socket

PORT = 8080

def serve_file(conn, pathname):
    if pathname.startswith('/'):
        pathname = pathname[1:]
    if pathname == '':
        pathname = 'index.html'

    try:
```

¹https://www.cp.eng.chula.ac.th/~krerk/books/ComputerSecurity/files/simple_web_server.py

```

        with open(pathname, 'r') as f:
            content = f.read()
    except Exception:
        conn.sendall(b"HTTP/1.0 404 Not Found\r\n\r\n")
        return

    conn.sendall(b"HTTP/1.0 200 OK\r\n\r\n")
    conn.sendall(content.encode())

def process_request(conn):
    data = conn.makefile('r').readline()
    parts = data.split()
    command = parts[0]
    pathname = parts[1]

    if command == 'GET':
        serve_file(conn, pathname)
    else:
        conn.sendall(b"HTTP/1.0 501 Not Implemented\r\n\r\n")

    conn.close()

def run():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:
        server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server.bind(('', PORT))
        server.listen(5)
        while True:
            conn, addr = server.accept()
            process_request(conn)

if __name__ == '__main__':
    run()

```

- (a) What happens if a client connects to SimpleWebServer but never sends any data and never disconnects? Test this with `telnet localhost 8080` (and leave it open). What type of attack is this?
- (b) Try the DoS attack described in class: attempt to download `/dev/random` (on Linux). Rewrite `serve_file()` to guard against this attack.
- (c) Implement logging: for each connecting client, record client information in a log file.
- (d) HTTP supports file uploads via the PUT command.
 - What threats must you consider if SimpleWebServer supports file uploads?
 - For each threat, what security mechanisms are needed to mitigate it?
 - Implement upload capability with a `store_file()` function. Apply the security measures you identified.
 - Once logging and `store_file()` are implemented, launch a defacement attack: replace `index.html` with one you have created. (This is a common attack against high-profile websites.)
2. What are the most important steps you would recommend for securing a new web server? A new web application?
3. What is Cross-Site Scripting (XSS)? What is the potential security impact on servers and clients?
4. What are *phishing* and *pharming*? What countermeasures can protect against each?

5. Explore the OWASP website (<https://owasp.org>).

- What are some vulnerabilities of web browsers?
- What modes of attack are used for Denial of Service? What preventive measures can be implemented?

From *Computer Security: The Foundations*, Kerk Piromsopa, Ph.D.
Reproduce and adapt freely for non-commercial classroom instruction, with attribution. Full terms: LICENSE.txt in the student package.