4.5 스니핑 후 스푸핑하기

- 많은 상황에서 먼저 패킷을 캡처한 다음, 캡처된 패킷을 기반으로 응답을 스푸핑해야 한다.

- 절차(예로서 UDP 이용)
  - PCAP API를 이용하여 관심 있는 패킷 캡처
  - 캡처된 패킷에서 복사본 만들기
  - UDP 데이터 필드를 새 메시지로 교체하고 원본 및 타겟 필드를 교체한다.
  - 스푸핑된 응답 보내기

UDP 패킷 스니핑/스푸핑

sniff_spoof_udp.c

```
134 UDP 패킷을 변조하여 출발지 포트와 목적지 포트를 바꾸고, 새로운 데이터를 포함하는 패킷을 생성.
135 IP 헤더를 수정하고, UDP 헤더를 수정하여 새 패킷을 구성.
136 변조된 패킷을 전송하기 위해 send_raw_ip_packet 함수를 호출.
137 */
138 void spoof_reply_udp(const struct ipheader* ip)
139 {
140   printf("\n----------------------------------------------------\n");
141   const char buffer[PACKET_LEN];
142
143   int ip_header_len = ip->iph_ihl * 4;
144   struct udpheader *udp = (struct udpheader *)((u_char *)ip + ip_header_len);
145
146   if(ntohs(udp->udp_dport) != 9999) {
147     // only spoof UDP packet with destination port 9090
148     printf("111 ===> udp->udp_dport = %d\n", ntohs(udp->udp_dport));
149     printf("777 ===> udp->udp_sport = %d\n", ntohs(udp->udp_sport));
150     //printf("bbb ===> udp->udp_sport = %d\n", udp->udp_sport);
151     return;
152   }
153   else {
154     printf("222 ===> udp->udp_dport = %d\n", ntohs(udp->udp_dport));
155     printf("888 ===> udp->udp_sport = %d\n", ntohs(udp->udp_sport));
156   }
157
158   // Step 1: Make a copy from the original packet
159   memset((char *)buffer, 0, PACKET_LEN);
160   memcpy((char *)buffer, ip, ntohs(ip->iph_len));
161
162   struct ipheader *newip = (struct ipheader *)buffer;
```

ICMP 패킷 스니핑/스푸핑

sniff_spoof_icmp.c

```
119 void spoof_reply_icmp(struct ipheader* ip)
120 {
121         printf("\n-----------------------------------------------------\n");
122         int ip_header_len = ip->iph_ihl*4;
123         const char buffer[PACKET_LEN];
124         // 원본 IP 헤더와 ICMP 헤더를 복사하여 새로운 패킷을 생성
125         memset((char*)buffer,0,PACKET_LEN);
126         memcpy((char*)buffer,ip,ntohs(ip->iph_len));
127
128         // Construct icmp header
129         // ICMP 응답을 생성하고 체크섬을 계산
130         struct ipheader *newip=(struct ipheader*)buffer;
131         struct icmpheader *newicmp=(struct icmpheader*) (buffer +ip_header_len);
132
133         newicmp->icmp_type=0;  //0 for reply
134         newicmp->icmp_chksum=0;
135         newicmp->icmp_chksum=in_cksum((unsigned short *)newicmp, sizeof(struct icmpheader));
136         newicmp->icmp_seq=count++;
137
138         // Construct ip header
139         // IP 헤더의 출발지와 목적지 주소를 반대로 설정
140         newip->iph_sourceip=ip->iph_destip;
141         newip->iph_destip=ip->iph_sourceip;
142         newip->iph_ttl=50;
143         newip->iph_len=ip->iph_len;
144
145         //Send Spoofed reply
146         send_raw_ip_packet(newip);
147
148 }
```

하이브리드 방식 스니핑 스푸핑 예

Scapy → 패킷 구성이 쉬움, 속도 느림
C 프로그램 → 패킷 구성이 쉽지 않음, 속도 빠름

짧은 시간에 많은 패킷을 생성해서 보낼 수 있는 쉬운 방법이 필요
Scapy와 C프로그램을 하이브리드로 사용

generate_udp.py

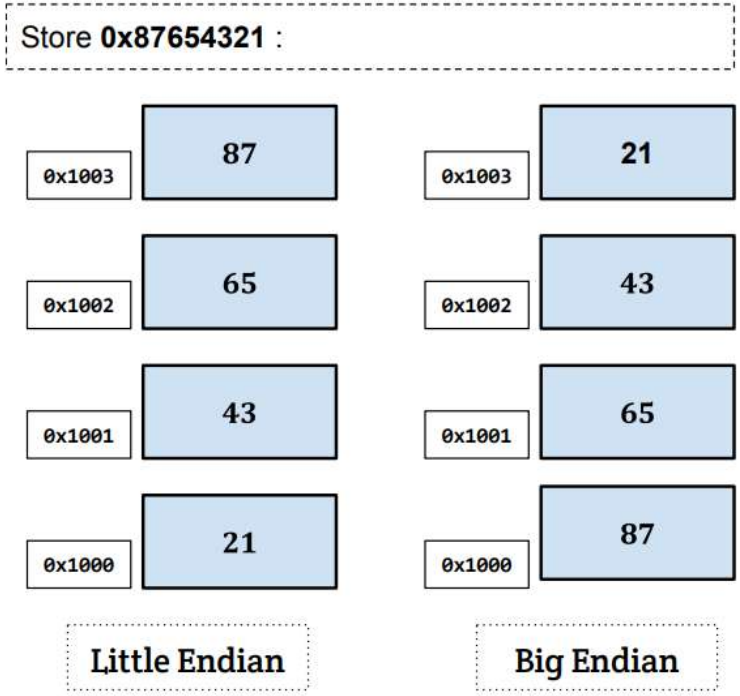
```python
#!/usr/bin/python3
from scapy.all import *

IPpkt = IP(dst='10.9.0.5', chksum=0)
UDPpkt = UDP(dport=53, chksum=0)
pkt = IPpkt/UDPpkt

# Save the packet data to a file
with open('ip.bin', 'wb') as f:
  f.write(bytes(pkt))
```

send_premade_udp.c

```c
int main()
{
  // create raw socket
  int enable = 1;
  int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
  setsockopt(sock, IPPROTO_IP, IP_HDRINCL, &enable, sizeof(enable));

  // read the UDP packet from file
  FILE *f = fopen("ip.bin", "rb");
  if(!f) {
    perror("Can't open 'ip.bin'");
    exit(0);
  }
  unsigned char ip[MAX_FILE_SIZE];
  int n = fread(ip, 1, MAX_FILE_SIZE, f);

  // modify and send out UDP packets
  srand(time(0)); // initialize the seed for random # generation.

  for(int i = 1; i < 10; i++) {
    unsigned short src_port;
    unsigned int src_ip;

    src_ip = htonl(rand());
    memcpy(ip + 12, &src_ip, 4); // modify source IP

    src_port = htons(rand());
    memcpy(ip + 20, &src_port, 2); // modify source port

```

4.7 엔디안

- 엔디안(Endianness): 주어진 멀티바이트 데이터 항목이 메모리에 저장되는 순서를 나타내는 용어.
  - 리틀 엔디안(Little Endian): 데이터의 최상위 바이트를 가장 높은 주소에 저장
  - 빅 엔디안(Big Endian): 데이터의 최상위 바이트를 가장 낮은 주소에 저장



네트워크 통신에서 엔디안

- 바이트 순서가 서로 다른 컴퓨터는 서로를 "잘못 이해 " 한다.
  - ➲ **해결책: 통신을 위한 공통 순서에 서로 동의**
  - ➲ **이를 "네트워크 순서"라고 하며 빅 엔디안 순서와 동일하다.**

- 모든 컴퓨터는 "호스트 순서"와 "네트워크 순서" 간에 데이터를 변환해야 한다.

매크로	설 명
htons( )	부호 없는 짧은 정수를 호스트 순서에서 네트워크 순서로 변환
htonl( )	부호 없는 정수를 호스트 순서에서 네트워크 순서로 변환
ntohs( )	부호 없는 짧은 정수를 네트워크 순서에서 호스트 순서로 변환
ntohl( )	부호 없는 정수를 네트워크 순서에서 호스트 순서로 변환

4.8 검사합 계산

■ 인터넷 검사합 계산

```
unsigned short in_cksum (unsigned short *buf, int length)
{
unsigned short *w = buf;
int nleft = length;
int sum = 0;
unsigned short temp=0;
/*
* The algorithm uses a 32 bit accumulator (sum), adds
* sequential 16 bit words to it, and at the end, folds back all
* the carry bits from the top 16 bits into the lower 16 bits.
*/
while (nleft > 1) {
sum += *w++;
nleft -= 2;
}
/* treat the odd byte at the end, if any */
if (nleft == 1) {
*(u_char *)(&temp) = *(u_char *)w ;
sum += temp;
}
/* add back carry outs from top 16 bits to low 16 bits */
sum = (sum >> 16) + (sum & 0xffff); // add hi 16 to low 16
sum += (sum >> 16); // add carry
return (unsigned short)(~sum);
}
```

4.8 TCP 검사합 계산

```
/* Psuedo TCP header */
struct pseudo_tcp
{
      unsigned saddr, daddr;
      unsigned char mbz;
      unsigned char ptcl;
      unsigned short tcpl;
      struct tcpheader tcp;
      char payload[1500];
};

unsigned short calculate_tcp_checksum(struct ipheader *ip)
{
  struct tcpheader *tcp = (struct tcpheader *)((u_char *)ip +
                          sizeof(struct ipheader));
  int tcp_len = ntohs(ip->iph_len) - sizeof(struct ipheader);
  /* pseudo tcp header for the checksum computation */
  struct pseudo_tcp p_tcp;
  memset(&p_tcp, 0x0, sizeof(struct pseudo_tcp));

  p_tcp.saddr = ip->iph_sourceip.s_addr;
  p_tcp.daddr = ip->iph_destip.s_addr;
  p_tcp.mbz   = 0;
  p_tcp.ptcl  = IPPROTO_TCP;
  p_tcp.tcpl  = htons(tcp_len);
  memcpy(&p_tcp.tcp, tcp, tcp_len);
  return (unsigned short) in_cksum((unsigned short *)&p_tcp,
                                   tcp_len + 12);
}
```

4.9 요약

- Packet sniffing
  - Using raw socket
  - Using PCAP APIs
- Packet spoofing using raw socket
- Sniffing and the spoofing
- Endianness