**[ Protocol 확인 ]**

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

* 빨간 박스 부분이 WireShark 속 Protocol을 확인할 수 있다.
* NTP Protocol : 네트워크 사의 컴퓨터들이 시간을 동기화하기 위해 사용하는 프로토콜
* DNS : 도메인 이름을 IP 주소로 변환하는 시스템
* Mdns : 로컬 네트워크에서 DNS 기능을 제공하는 프로토콜 주로 네트워크 내의 기기들이 서로 쉽게 발견할 수 있도록 해줌
* ARP : 네트워크 상에서 IP주소를 물리적 MAC 주소로 변환하는 프로토콜
* TCP : 신뢰성 있는 데이터 전송을 보장하는 프로토콜
* HTTP : 웹 브라우저와 웹 서버 간의 통신에 사용되는 프로토콜

위 WireShark에서 나타난 프로토콜 이 외에도 다양한 프로토콜이 있다.

**[ Mac Address 확인하기 ]**

* **Ubuntu Server Mac Address : 00-0C-29-FC-87-6E**
* **Ubuntu Server IP Address : 192.168.100.131**

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

MAC 주소는 컴퓨터 네트워킹 내 이더넷 프레임, 특히 스위칭 환경에서 중요한 역할을 한다. 데이터 패킷을 의도한 대상으로 적절하게 전달함으로써 네트워크 고유성과 효율성을 보장하는데 필수적이다.

이러한 MAC주소는 방금 설명 했듯이 이더넷 프레임에서 존재하기 때문에 WireShark에서도 이더넷 프레임을 확인해서 MAC주소를 확인한다.

[ TCP 연결을 위해 전송 된 SYN 패킷 ]

Time Source Destination Protocol Length Info



* Ubuntu Server의 IP 주소인 192.168.100.131에서 시작하여 185.125.190.98 IP에 도착하고 있음을 확인할 수 있다.

[ SYN패킷 속 Ethernet 프레임 ]

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

* Source VMware를 확인하면 위에서 확인한 Ubuntu Server의 MAC Address와 동일한 것을 확인할 수 있다.

[ **00-0C-29-FC-87-6E**의 Mac Address를 가지는 패킷 필터링 **]**

* **00-0C-29-FC-87-6E**의 Mac Address를 가지는 패킷 필터링 구문



텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

[ SYN패킷 ]

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

* 위 SYN 패킷은 51342번 포트에서 시작하여 TCP 연결이기에 80번 포트로 전송되고 있는 것을 확인할 수 있다. ( HTTP서버는 80번 포트를 사용한다.)
* CF) 자주 사용되는 Port 번호

|  |  |  |  |
| --- | --- | --- | --- |
| 번호 | 프로토콜 | 통신 프로토콜 | 설명 |
| 80 | HTTP | TCP | 웹 서버 접속 |
| 443 | HTTPS | TCP | 웹 서버 접속(SSL) |
| 110 | POP3 | TCP | 메일읽기 |
| 25 | SMTP | TCP | 메일 서버간 메일 전송 |
| 22 | SSH | TCP | 컴퓨터 원격 로그인 |
| 53 | DNS | UDP | DNS 질의 |
| 123 | NTP | TCP | 시간 동기화 |
| 20 | FTP | TCP | 데이터 전송 |
| 21 | FTP | TCP | FTP 제어 |

[ 위 SYN패킷 속 Option필드 ]

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* TCP Header는 기본 20byte이다. TCP Option을 포함하면 TCP Header의 크기는 최대 60byte이다.

따라서 TCP Header 길이가 20byte이상이라면 Option 필드를 체크해야 한다.

**[ 패킷 속에서 원하는 Data문자열 검색 ]**

1. 편집
2. 패킷 찾기

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

1. 원하는 Data검색 구문을 입력

* Find Data라는 검색 구문을 입력하였지만 좌측 하단을 확인하면 Find Data라는 문자열이 포함 된 패킷이 없어 검색에 실패한 모습을 확인할 수 있다.

텍스트, 전자제품, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

**[ 패킷 필터링 구문 ]**

* 도착IP가 192.198.100.131인 패킷 필터링
* Ip.dst == 192.168.100.131

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* TCP 연결, TCP 연결 해제, TCP 서버 포트가 닫혀 있을 때 서버 응답
* Tcp.flags.syn ==1, rcp.flags.fin ==1 tcp.flags.rest == 1

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* Data 길이 필터링
* data.len > 0

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

* Data 특정 영영게 특정 값을 가지는 패킷 필터링
* data[4] == 0x11 and data[10] != 0x01

텍스트, 소프트웨어, 컴퓨터 아이콘, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

* 필터링 구문에 해당하는 패킷이 없어 빈 화면을 출력한다.
* Data에 특정 문자열 값을 포함하는 패킷 필터링
* Data contains”\”hello\””

스크린샷, 소프트웨어, 텍스트, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

* 필터링 구문에 해당하는 패킷이 없어 빈 화면을 출력한다.

**[ TCP 연결 수립 ( 3-Way-Handshaking ) ]**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* 전송계층에서 사용되는 프로토콜인 TCP는 ‘연결지향적’인 특징을 가지고 있어서 데이터를 주고 받기 전에 연결을 진행 한다.
* Sequence Number
* Segment에 있는 첫 번째 바이트의 바이트 스트림 번호

Ex) ACK 패킷이 어떤 패킷에 대한 ACK 패킷인지 확인이 필요하기 때문에 Sequence Number와 ACK Number를 사용하게 된다.

* Squence Number는 TCP 세그먼트의 연속된 데이터 번호이다.
* TCP 연결, 종료 시에는 Sequence Number를 임의의 랜덤 값으로 설정
* Sequence Number가 노출되면 공격자가 외조 패킷을 보낼 수 있어 보안을 위해 랜덤 값으로 설정 한다.

**< Port 상태 정보 >**

* LISTEN : 포트가 열린 상태로 연결 요청을 대기하는 상태
* SYN-SENT: 연결 요청을 하고 Server의 ACK를 기다리는 상태
* SYN-RCVD: 연결 요청에 응답/연결을 요청하고 Client의 응답을 기다리는 상태
* ESTABLISHED: 연결 완료 상태

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 클라이언트가 서버에게 연결을 요청하는 SYN Segment를 전송한다 .

* Sequence Number는 클라이언트의 최초 순서 번호로 설정한다.
* **SYN Segment를 전송한 후 SYNSENT 상태로 서버의 ACK Segment를 기다린다.**

1. 서버가 클라이언트의 SYN Segment에 대한 ACK Segment를 전송한다.

* Acknowledge Number는 client\_isn + 1로 설정한다.
* Sequence Number는 서버의 최초 순서 번호로 설정한다.

동시에 서버가 클라이언트에게 연결을 요청하는 SYN Segment를 전송한다.

* SYN bit를 1로 설정한다.
* **SYN/ACK Segment를 전송하고 SYN RCVD의 상태로 클라이언트의 ACK를 기다린다**

클라이언트는 ACK Segment을 받고 연결이 완료된 ESTAB 상태가 된다.

**// 2번째 단계는 2개의 Handshake에 해당한다. 하지만 이 두 과정은 서로에게 영향을 끼치지 않고 독립적이다.**

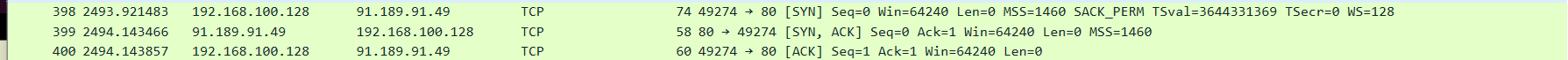
1. 클라이언트가 서버에게 ACK 패킷을 전송한다.

* 클라이언트는 서버의 SYN Segment에 대한 ACK Segment를 전송한다.
* 연결 요청은 아니다.
* 서버는 ACK Segment를 받고 연결이 완료 된 ESTAB 상태가 된다.

1. 최종적으로 클라이언트와 서버가 서로 데이터를 주고 받는다.

**[ WireShark에서 TCP연결이 오고 갔던 패킷을 가져 옴 ]**

Time Source Destination Protocol Length Info



Ubuntu 서버의 IP가 ‘192.168.100.128’ 이라 Source IP가 192.168.100.128에서 시작하고 있는 것을 알 수 있다

.

[ **클라이언트가 서버에 처음 전송하는 SYN-Sagment ]**

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

* Sequence Number(raw)

Sequcne Number(raw)가 실제 TCP 세그먼트 데이터 번호이며 Sequnce Number는 보기 편하게 가공 된 번호이다. 통신이 시작되면 난수인 ISN가 결정되고 해당 번호부터 순서가 시작되어 Sequence Number(raw)가 된다.

* Sequence Number를 확인하면 0부터 시작해서 1까지의 TCP 세그먼트 데이터를 전송하고 있어 Sequnce Number는 0으로 설정되었다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* 6개의 Segment 상태를 나타내기 위한 공간이다. 0과 1로 표기
* ACK패킷
* Acknowledgement Number가 유효한지를 나태낸다.
* 최초 연결의 첫 번째 Sagement를 제외한 모든 Segement의 ACK 비트는 1로 설정한다.
* 최초 연결의 첫 번째 Handsahke과정에서는 응답 요청이 없기 때문이다.
* SYN패킷
* 연결을 요청할 때 SYN bit사용

Ex) SYN bit = 1 일 경우 TCP 연결을 요청

* 다른 모든 경우에는 SYN bit를 0으로 설정
* 연결 요청에 응답하는 마지막 Handsake의 Segment에서도 SYN bit를 0으로 설정한다.

**[ 서버가 클라이언트의 SYN Segment에 대한 ACK Segment를 전송한다,**

**서버가 클라이언트에게 연결을 요청하는 SYN Segment를 전송한다. ]**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* Sequence Number를 확인하면 1부터 시작해서 TCP 세그먼트 데이터를 전송하고 있어 Sequnce Number는 1으로 설정되었다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* Acknowledgement 패킷의 경우 최초 연결의 첫 번째 Sagement를 제외한 모든 Segement의 ACK 비트는 1로 설정하기 때문에 Wireshark에서 1로 설정되어 있는 것을 확인할 수 있다.
* SYN 패킷의 경우는 서버가 클라이언트에게 연결을 요청하고 있기 때문에 1로 설정 되어있는 것을 확인 할 수 있다.

[ 클라이언트가 서버에게 ACK 패킷을 전송한다. ]

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

* Sequence Number를 확인하면 1부터 시작해서 TCP 세그먼트 데이터를 전송하고 있어 Sequnce Number는 1으로 설정되었다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* Acknowledgement 패킷의 경우 최초 연결의 첫 번째 Sagement를 제외한 모든 Segement의 ACK 비트는 1로 설정하기 때문에 Wireshark에서 1로 설정되어 있는 것을 확인할 수 있다.
* SYN 패킷의 경우 더 이상의 연결 요청이 없기 때문에 0으로 설정되어 있는 것을 확인할 수 있다.

**[ TCP 연결 종료 ( 4-Hand-Shaking) ]**

**[ Ubuntu Server에서 KT DNS Server로 접속하여 통신하는 과정을 Wireshark 패킷으로 캡처 ]**

**< Port 상태 정보 >**

**FIN\_WAIT1 : Close()를 호출한 측의 소켓이 진입하는 상태, FIN을 보낸다.**

**CLOSE\_WAIT : Close()를 받으면 CLOSE\_WAIT 상태로 진입함, ACK을 보낸다.**

**FIN\_WAIT2 : ACK신호를 받은 소켓은 FIN\_WAIT1 > FIN\_WAIT2 상태로 변경된다.**

**LAST\_WAIT: Close() 호출 후 진입하는 상태, FIN을 보낸다.**

**TIME\_WAIT : Close()를 받으면 진입하는 상태, ACK을 보낸다,**

**CLOSED: 연결종료**

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 클라이언트가 서버에 연결 종료 요청하는 FIN 패킷을 전송한다.

* 클라이언트는 더 이상 데이터를 전송하지 않을 것을 알린다.
* FIN 패킷을 전송 후 Port 상태는 FIN\_WAIT1 상태가 된다.

1. 서버가 클라이언트로부터 받은 FIN 패킷에 대해서 응답하는 ACK 패킷을 전송한다.

* ACK 패킷을 전송 후 Port 상태는 CLOSE\_WAIT가 된다.

1. 서버가 통신을 끝낼 준비가 완료되면 클라이언트에서 FIN 패킷을 전송한다.

* FIN 패킷을 전송 후 Port 상태는 LAST\_WAIT가 된다.

1. 클라이언트는 서버로부터 받은 FIN 패킷에 대해서 응답하는 ACK 패킷을 전송한다.

* ACK 패킷을 전송한 후 Port 상태는 TIME\_WAIT 상태가 된다.
* TIME\_WAIT
* 연결을 누가 먼저 종료하는지에 따라서 연결 종료 후 도달하는 Port 상태에 해당한다.
* **네트워크에 남아 있을 수 있는 종료된 연결의 패킷이 완전히 제거될 수 있을 때 까지 대기**
* **이후 생성되는 새로운 연결에 영향을 미치지 않기 위한 용도로 이용되는 Port 상태**

**[ ‘telnet’을 사용하여 KT DNS Server에 접속하는 과정 ]**

‘telnet’을 사용하여 KT DNS Server에 접속

1. ‘telnet’이 설치되어 있지 않아 ‘apt-get’명령어로 ‘telnet’설치

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**#sudo apt-get update**

1. KT DNS서버(192.126.63.1)와 포트 53에 ‘telnet’으로 접속

(정상 접속이 됐다가 후에 ctrl + c로 연결을 종료한 모습을 확인할 수 있다.)

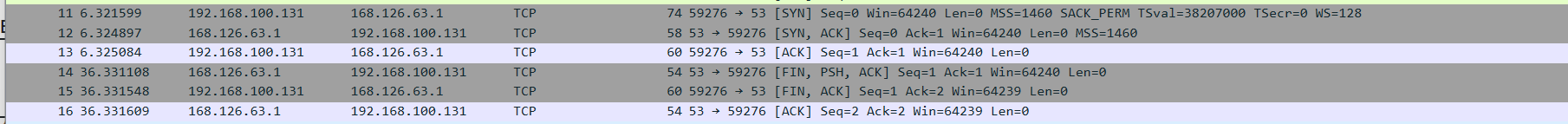
**#sudo telent 168.126.63.1 53**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[ Wireshark로 4-hand-shaking 패킷 분석]

Time Source Destination Protocol Length Info



* Ubuntu Server의 IP인 192.168.100.131에서 KT DNS Server인 168.126.63.1로 패킷이 전송되고 있음을 확인할 수 있다.
* Info부분을 확인하면 처음 패킷 전송이 임의의 포트에서 시작하여 53포트에 전송되고 있음을 확인할 수 있다.
* Ubuntu Server에서 KT DNS Server에 FIN 패킷 전송한다.
* PSH : Telnet과 같이 상호작용이 중요한 프로토콜의 경우 빠른 응답이 중요한데 이때 받은 데이터를 즉시 목적지인 OSI 7계층으로 전송하도록 하는 Flag
* 아래 전송된 Port번호를 확인하면 53번 Port로 패킷이 전송되고 있음을 확인할 수 있다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

* KT DNS Server에서 Ubuntu Server에 FIN 패킷을 전달 받았다는 의미인 ACK 패킷을 전송한다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명