목차

[1. FreeRadius 2](#_Toc69582064)

[**1-1-1.** **FreeRadius 사전 준비** 2](#_Toc69582065)

[**1-1-2.** **FreeRadius 실행** 2](#_Toc69582066)

[**1-1-3.** **Wireshark로 캡쳐한 결과** 2](#_Toc69582067)

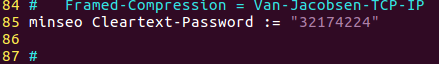
[**2.** **TESLA 및 Group Key Management 정리** 3](#_Toc69582068)

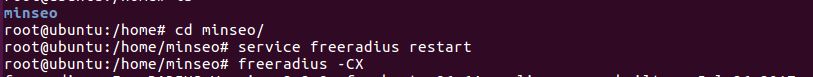
[**2-1. TESLA 정리** 3](#_Toc69582069)

[**2-2. Group Key Management 정리** 4](#_Toc69582070)

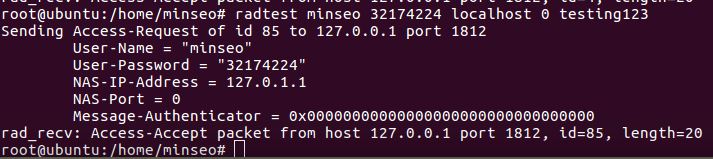
1. FreeRadius
   * 1. **FreeRadius 사전 준비**

etc/freeradius/users 파일 끝에 추가한 내용

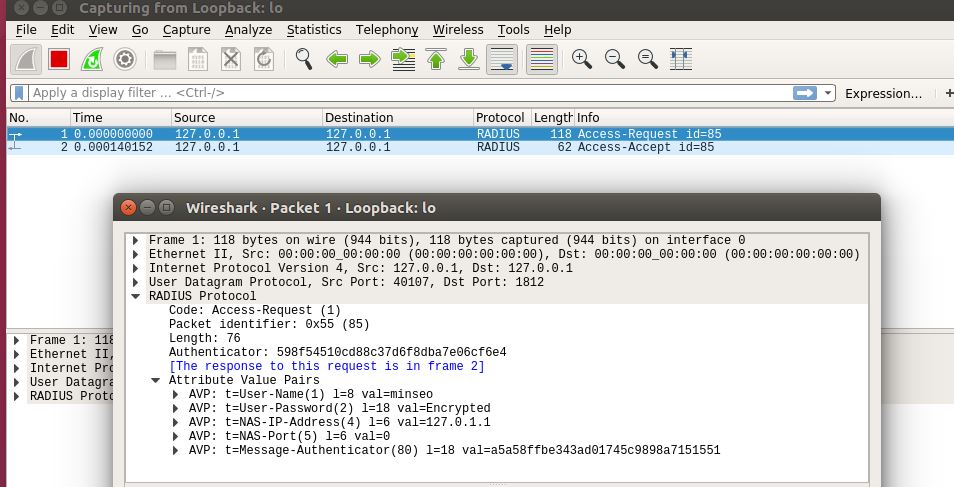




* + 1. **FreeRadius 실행**



* + 1. **Wireshark로 캡쳐한 결과**



# **TESLA 및 Group Key Management 정리**

**2-1. TESLA 정리**

TELSA는

Timed -> 송수신자간 느슨한 시간 동기화 요구

Efficient -> 서명 방식과 MAC 방식 둘 다 사용 단, 서명은 한번하는데 오래 걸려서 맨 처음 multicast packet 1개에서만 적용

Stream -> online방식 = 실시간으로 multicast packet이 생성

Loss-Tolerant 일부 multicast packet들이 전송도중 분실되어도 나머지에는 영향이 없음

의 약자다.

여기서 사용되는 MAC 방식은 사전에 대칭키를 공유하는 방식이 아닌 지연 인증 방식인 다음 packet에 암호화에 사용된 대칭키를 전송하는 방식으로 인증한다.

그래서 주의해야하는 점이 5가지가 존재한다.

1. Packet을 전송할 때는 다음 packet에 대한 힌트를 탑재해서 전송해야 한다.

* 그러지 않음 메시지를 탈취한 후 MAC 함수 값을 공격자가 본인이 원하는 값으로 변조할 수 있기 때문이다.

1. 첫 packet은 꼭 전자 서명을 해야 한다.

* 왜냐하면 악의적인 사용자가 첫 packet을 탈취하고 첫 packet부터 변조를 하면 인증할 방법이 없기 때문에 맨 처음 packet은 전자서명을 해서 보내야한다.

1. 만약 중간에 packet이 분실되면 그 다음 packet을 인증할 수 없으므로 n 번째 packet에는 n+1번째 packet에 대한 힌트를 n+1의 key를 hash한 값으로 넣어둔다.

* 이 힌트를 n+1번째 key를 해쉬 값으로 하는 이유는 그 다음 키 값이므로 중간에 분실이 되어도 추측이 가능하게 한다.

1. 송신자는 메시지가 모든 수신자들에게 전송된 다음에 전송을 해야 한다.

* 왜냐하면 다음 packet에는 현재 packet의 암호화에 사용된 키가 담겨있기 때문에 만약 아직 모든 사용자들에게 packet이 도착하지 않은 채 다음packet을 전송하면 packet의 내용이 노출될 뿐만 아니라 메시지 변조가 가능하다.

1. TELSA는 느슨한 시간 동기화를 하기 때문에 시간 동기화가 중요한 일이다.

* 2가지 알고리즘이 존재하는데
* 하나는 Cristian’s Algorithm 인데 이는 서버시간에 무조건 동기화 하는 방법이다,
* 다른 하나는 Berkeley Algorithm으로 모든 사용자들과 서버시간의 평균으로 계산하는 방법이다.

**2-2. Group Key Management 정리**

그룹 키 관리는 누군가가 join을 하면 이전 내용을 보호하기 위해서 key를 바꾸고, 누군가가 leave를 하면 이후 내용을 보호하기 위해서 key를 갱신한다.

그룹키 관리에는 5가지 정도가 있는데

1. Star key Graph 기반 rekeying : 이는 새로운 그룹키(Kg’)를 모든 사용자의 대칭키로 하나하나 다 암호화해서 전송하는 방식으로,

* 사람이 늘면 message 양도 늘어난다.

1. Complete key Graph기반 rekeying : 이는 message가 늘어나는걸 방지하기 위해서 key들의 부분집합들을 만들어서 그 부분 집합에 해당하는 걸로 message를 보내는 건데

* 이는 할당되는 개인키가 많아서 관리하는 게 힘들다. (2^n- 1)개

1. Tree key Graph : 개별키와 message를 최소화 하기 위해서 tree로 만들어서 각각의 사람이 가지는 키를 log 2 n개로 줄였다. (그룹키 제외)

* 그렇지만 만약 한명이 새로 들어오거나 나가면 들어오거나 나간 부분이 해당하는 부모 키를 전부 갱신을 해야해서 조금은 복잡함

1. Tree key Graph 에서 Join프로토콜 개선 : 새롭게 키를 할당하는 것이 아닌, 기존의 key들을 hash해서 전송!

* 전송할 필요 없이 그냥 메시지를 전송하면 다들 갱신을 진행한다.

1. OFT (One-Way Function Tree) 기반의Rekeying) : 이는 자신부터 root(group) 까지 존재하는 node key를 알고 있고, 경로상의 sibling의 blind node key를 알고 있다는 전제하에 진행된다.

* 여기서의 blind node key는 내 node key를 hash한 것이고,
* 내 부모의 node key는 나와 내 sibling의 blind node키를 hash를 취한 것이다.
* 만약 내가 단말 노드 1개 였는데 새로운 사람이 등록하면, 나도 새로 할당받고, 새로 등록한 사람도 할당받는다.
* 만약 2명이 존재했는데 탈퇴를 하면 내가 내 부모의 node가 된다.