2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

키 백업을 위한 IPSec 설계 및 구현

김정범° 박남섭 김태윤
고 려 대 학 교 컴 퓨 터 학 과
{qston, nspark, tykim}@netlab.korea.ac.kr

IPSec Design and Implement for Key Backup

Jeong-Beom Kim, Nam-Sup Park, Tai-Yun Kim
Dept. of Computer Science and Engineering, Korea University

요 약

IETF에서 IP 계층을 위한 보안 구조인 IPSec[1]을 발표한 이래 오픈소스인 리눅스용 IPSec 구현물들이 많이 개발되어왔다. 이러한 구현 프로그램들은 커널 컴파일과 같은 어려운 점과 사용자가 자신의 키를 잃어버린 경우 자신의 데이터에 접속할 수 없는 이유 때문에 IPSec 구현물들의 확장성에 많은 문제가 있었다. 이에 본 논문은 이러한 점을 해결하고자 Ethertap이라는 설정 하나로 커널 컴파일 없이 IPSec을 이용할 수 있도록 하고 또한 사용자가 자신의 키를 백업할 수 있도록 리눅스용 IPSec을 설계 및 구현하였다.

1. 서론

IETF(Internet Engineering Task Force)에서 IPSec(Internet Protocol security)을 발표한 이래 리눅스 환경하에서 많은 IPSec 프로그램이 개발되었다. 하지만 많은 리눅스용 IPSec 제품들이 사용자가 사용하기에는 커널 컴파일과 같은 이유 때문에 많이 쓰이지 않고 있으며 또한 사용자가 이러한 IPSec을 사용하게 될 때 자신의 키를 잃어버려서 자신의 데이터에 접속하지 못하는 사례가 많이 발생하고 있다. 이에 본 논문은 리눅스 커널의 컴파일을 하지 않고도 사용자가 간편하게 설정하여 사용할 수 있도록 하여 IPSec 프로그램의 신장성을 높이고 사용자의 키를 백업할 수 있는 IPSec 프로그램을 개발, 구현하였다. 이를 위해서 본 논문에서는 이러한 해결책을 위해 ethertap을 이용하여 설계하고 여기에다가 KRA(Key Recovery Association)[3]를 이용하여 사용자의 키를 백업할 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 ethertap[2]과 KRA에 대해서 설명하고, 3장에서는 이를 바탕으로 한 IPSec 구현과 이를 가지고 어떻게 VPN(Virtual Private Neetwork)[4]을 구현하는지에 대해 설명한다. 4장에서는 결론 및 향후 연구과제를 제시한다.

2. 관련연구

2.1 IPSec(Internet Protocol Security)

IPSec은 다음과 같은 보안 서비스를 제공한다. 이 서비스는 선택적이며 일반적으로 로컬 보안 정책은 이 서비스 중에서 하나나 그 이상의 서비스를 선택한다.

- 데이터의 비밀성 - IPSec의 송신자는 네트워크로 전송되기 전에 데이터를 암호화하여 보낸다.
- 데이터의 무결성 - IPSec 수신자가 보내는 데이터는 받는 사람의 데이터와 같다.
- 데이터의 인증 - IPSec의 수신자는 패킷을 보낸 송신자의 인증을 확인할 수 있다.
- 재사용성 방지 - IPSec 수신자는 재사용 공격을 방지할 수 있다.

IPSec 동작에는 세 가지 기본 구성요소가 필요하다. 즉, Security Association(SA), Authentication Header(AH), Encapsulating Security Payload(ESP)가 IPSec 동작에 중요한 역할을 한다.

● Security Association (SA)

데이터 송수신자간에 비밀데이터(인증되었거나, 암호화된 데이터)를 교환할 때 사전에 암호 알고리즘, 키 교환방법, 키 교환 주기 등에 대한 합의가 이루어져야 한다. 데이터 교환 전에 통일되어야 할 이러한 요소들을 IPSec에서는 SA로 정의한다. 데이터 송수신자간의 안전한 통신을 위해서는 적어도 하나의 SA가 필요하다. 그러므로 패킷 인증과 암호를 위해서는 SA가 선행되어야 한다. 동일한 알고리즘이 사용되어도 서로 다른 두 개의 키가 요구될 경우에는 두 개의 SA가 필요하다. SA를 대인간 또는 그룹간, 네트워크간의 네트워크 보안 채널로 생각할 수도 있다. SA는 다중 VPN 구성에 유리하다. 다수의 상대방이 있을 경우 개인별로 SA를 별도로 정의함으로써 서로 다른 VPN을 구성할 수 있다. 또한 SA는 일방향 전송도 가능하다. 하나의 SA가 정의될 경우 송신자는 수신자에게 데이터를 전송할 수 있으나 동일한 SA로 데이터의 수신은 불가능하다. 따라서 양방향 통신을 위해서는 수신자가 전송자에게 보내는 SA가 별도로 정의되어야 한다.

● Authentication Header(AH)

AH는 IPSec에서 IP 데이터에 대한 인증 서비스와 데

이터 무결성 서비스를 제공한다. AH는 패킷의 체크섬 결과를 포함하고 있다. AH는 1)next header field, 2)payload length, 3)security parameter index(SPI), 4)sequence number, 5)authentication data 다섯 가지 필드를 포함한다. 다섯 가지 필드 중에서 SPI는 송신자가 사용하는 보안 프로토콜에 대한 정보를 알려주며, authentication data는 SPI에서 정의된 암호알고리즘으로 패킷의 payload를 암호화하여 얻어진다. IPSec은 인증 기능의 향상을 위해 패킷 체크섬 계산 방법을 기존 MD5 방식 대신 HMAC(Hash-based Message Authentication Code)-MD5와 HMAC-SHA 방식 중 하나를 사용하고 있다.

● Encapsulating Security Payload(ESP)

ESP는 패킷의 암호화 서비스를 제공한다. AH처럼 패킷 처리에 필요한 SA 정보를 수신자에게 알려주기 위해 SPI를 포함하고 있다. ESP는 다양한 암호화 알고리즘을 지원하며 사용자는 상대방에 따라 서로 다른 암호알고리즘을 사용할 수 있다. ESP에서도 인증 서비스를 제공할 수 있으나 이 경우 AH와 달리 IP 헤더에 대한 인증 서비스는 제공하지 못한다.

이외에도 IPSec에서는 키 관리가 중요하다. 키 관리는 IPSec과는 독립적으로 구현되므로 여러 가지 키 관리 프로토콜 중에서 선택하여 사용할 수 있다. IPSec에서 사용 가능한 대표적인 키 관리 프로토콜로는 SKIP, Photuris, ISAKMP 등이 있다. 키 관리 시에는 네트워크 상에서 키의 안전한 교환, 키의 주기적인 변경, 송. 수신자간의 키 협상 등을 고려하여야 한다.

2.2 Ethertap[2]

Ethertap이란 사용자 영역에서 패킷을 송수신하는 역할을 한다. 이것은 네트워크 상에서 패킷을 수신하는 ethernet 디바이스의 기능을 수행하며 그것은 사용자 공간으로부터 패킷을 받는다. 즉, 커널 영역이 아닌 사용자 영역에서 이더넷 프레임들을 ethertap의 영향으로 생긴 가상 디바이스인 tap0를 통해 읽고 쓸 수 있다는 것이다. 하디만 이러한 tap0는 이더넷 장치로 사용할 수 있지만 어떠한 물리적인 LAN에도 연결되지 않는다.

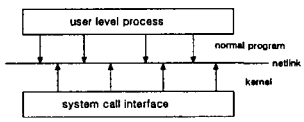


그림1. ethertap 구조

이러한 ethertap은 어떤 특별한 환경에서 사용하는 것이 아닌 AppleTalk에서 IPX까지 두루 적용이 가능하기 때문에 일반 환경에서도 적용 가능하다.

이러한 ethertap 함수의 기본 설정 과정은 다음과 같다.

- 커널 설정 안에서 우선 Ethertap을 활성화 시킨다.
- ethertap 디바이스 파일 디스크립터를 생성한다.

```
- /dev/tap0 -> /dev/tap15.
- mknod /dev/tap* c 36 16 (17 18 19 20 for
tap1,2,3,4)
```

그림2. ethertap 활성화 과정

2.3 KRA(Key Recovery Association)[3]

그림3은 키 정보를 사용하여 암호 통신을 하는 두 사용자 단말 장치 사이의 상호 작용을 나타내고 있다.

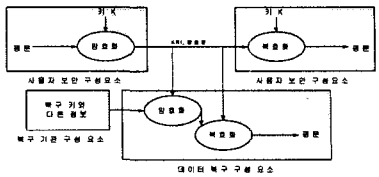


그림3. 키 복구 구조

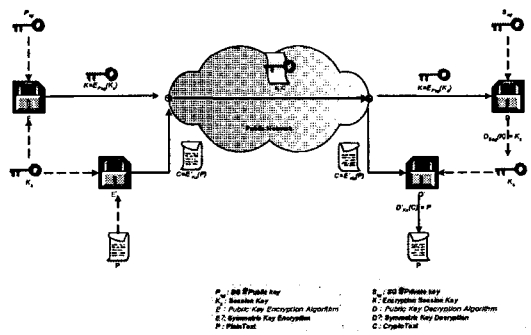
여기서 먼저 목적키를 복구 가능하게 하기 위해서 사용자 단말 장치 내에서 키 복구 정보 생성 기능은 목적키에 대응하는 KRI(Key Recovery Information)를 생성하여 캡슐화 한 후, 상대편 사용자 단말 장치로 암호문과 함께 전송한다. 그러면 받은 사용자는 그 키를 백업해두게 된다.

3. 본문

3.1 제안한 IPSec 구현

먼저 구현 환경을 설명하면 운영체제는 리눅스이고 커널 버전은 2.2.16이다. IPSec에서 쓰일 암호화 방법은 OpenSSL을 이용하였다.

사용자가 자신의 키를 백업하게 되는 과정은 다음과 같다.



이렇게 키를 백업할수 있게 해주는 모듈의 동작원리는 다음과 같다.

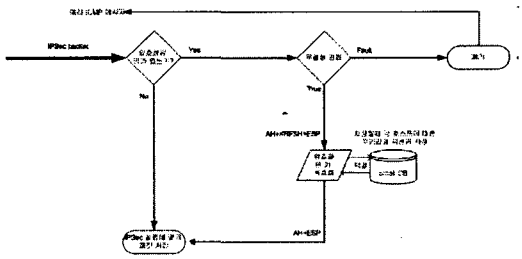


그림5. 프로그램내에서 백업 과정

전체적으로 각각의 모듈을 사용하는 IPSec 프로그램

의 함수 다이어그램은 다음과 같다.

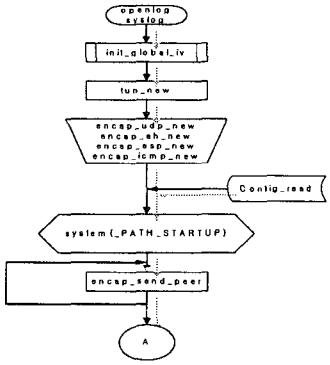


그림 6 나가는 패킷 처리

그림을 설명하면, 먼저 시스템 로그 파일을 연후에 프로그램 시작 로그를 남긴다.

동시에 IV(Initialization Vector)를 초기화시키고 ethertap 자료구조를 저장할 수 있는 영역을 할당하여 초기화한다. UDP(User Datagram Protocol), AH(authentiaction Header)[5], ESP(Encapsulation Header)[6], ICMP(Internet Control Message Protocol) 처리 루틴을 초기화하고 설정 파일을 읽어들여서 메모리에 적재한다. 그러고나서 초기화 파일을 실행시키면 설정 파일에서 peer로 설정된 호스트로 패킷을 보낸다.

들어오는 패킷 처리는 다음과 같다.

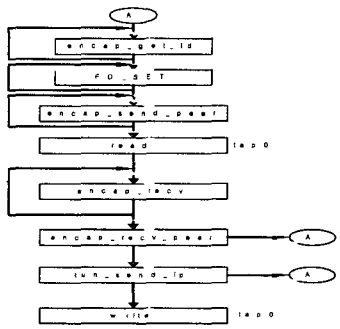


그림 7 들어오는 패킷 처리

그림을 보면, 먼저 tap0~15로 들어오는 패킷이 있는지 polling한 후들어오는 패킷이 있는 경우 FD(File Descriptor)를 설정하여 설정 파일에서 peer로 설정된 호스트로 IPSec 루틴을 거친 패킷을 보낸다. tap0~15로 들어오는 실제 패킷을 버퍼로 읽어들이며 들어오는 패킷의 종류에 따른 수신 루틴을 실행한다. 패킷의 종류에 따라 IPSec 루틴을 거친 후 실제 패킷을 얻어낸다. 해당 어드레스로 실제 패킷을 보낸다.

이러한 IPSec 프로그램의 전체 모듈 구성과 동작은 다음과 같다.

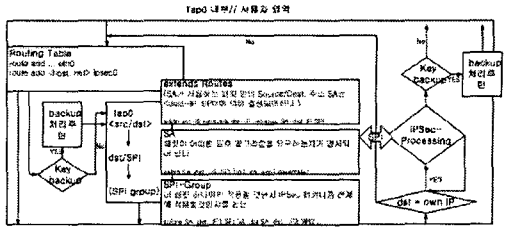


그림8. 전체 모듈 동작 구성

3.2 제안한 VPN[4] 구현

이렇게 구현된 IPSec을 이용한 전체 VPN 구조는 다음과 같다.

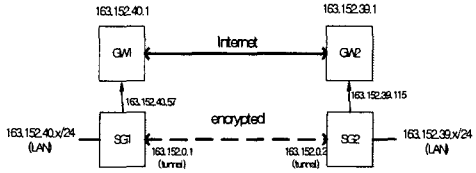


그림9. 실험 환경

그림과 같이 163,152,40.x/24 라인으로 연결된 LAN A와 163.152.39.x/24 라인으로 연결된 LAN B를 중간 노드로 SG(Security Gateway) A와 SG B를 이용해서 구현하였다. SG A와 SG Ben 노드는 PPP(Point-to-Point)방식을 이용하여 연결한다. 이를 위해서 앞에서 언급한 ethertap과 netlink 모듈을 사용한다. 또한 tap0와 실제 패킷을 전송하는 eth0와의 연동은 그림10과 같은 startup이라는 설정 파일에서 해주게 되는데 이것의 역할은 tap0의 가상 IP를 통하여 SG1과 SG2 사이를 가상 링크로 연계시켜주는 역할을 한다.

```
#!/bin/sh
ifconfig tap0 hw ether fe:fe:0:0:0:0
ifconfig tap0 163.152.0.1 pointopoint 163.152.0.2 netmask 255.255.255.0
arp
ip route add 163.152.0.2 mtu 1500 dev tap0
```

그림10. starup shell program

터널링에 따른 AH나 ESP 중 IPSec 터널에서 어떠한 방식을 이용할 것이며 암호화 방식을 구별해주는 SPI는 무엇인지 등의 SA와 관련된 내용은 그림11과 같은 config 파일에서 설정한다.

```
sa ipesp spi=1000 enc=blowfish_cbc ekey=f1f2f3f4f5f6f7f8f9fafbfcfdfeff
dest=163.152.40.57
sa ipesp spi=1000 enc=blowfish_cbc ekey=d00db00fd00d00d00db00fd00dc00e
if /dev/tap0 local_spi=1000 remote_spi=1000
```

그림11. config 설정

3.3 구현 결과

이러한 IPSec 프로그램을 통한 패킷을 스니핑해서 패킷을 보내기 전과 받은 다음 복호화 하기 전 그리고 복호화 된 후 보내기 전과 같은지를 비교하였다.

먼저 그림12는 보내기전 패킷의 일부를 printk()로 찍어보았다.

```
[tun_send_ip : After writing TAP0]
Before sending ip packet:
 fb ff ff ff ff ff ff ff fe fd 00 00 00 00 08 06
 00 01 08 00 06 04 00 01 fe fd 00 00 00 00 a3 98
 00 02 00 00 00 00 00 00 a3 98 27 76 01 02 02 04
```

그림12. 암호화해서 전송하기 전 패킷 내용

그림13은 패킷을 받은 후 복호화 하기 전의 패킷을 역시 printk() 함수를 써서 찍었다.

```
[main function:after encap_recv function]
After encap_recv ip packet:
encap_recv src address is 163.152.39.115
encap_recv packet is 84
receiving ESP packet (before decrypt):
 12 cd aa 8b f0 a7 e2 b0 2a fc e2 40 35 ec b7 17
 70 90 f4 1a f0 bd bd 98 6c 0b 47 79 4e a3 dd 72
 f1 7a e0 e1 9f 40 8b 41 06 a3 91 6e 9d 3f 63 d7
```

그림13. 수신받은 패킷의 복호화 하기전 내용

그림14는 패킷을 복호화한 후 printk() 함수를 써서 패킷의 내용을 캡쳐했다.

```
receiving ESP packet (after decrypt 48):
 fb ff ff ff ff ff ff ff fe fd 00 00 00 00 08 06
 00 01 08 00 06 04 00 01 fe fd 00 00 00 00 a3 98
 00 02 00 00 00 00 00 00 a3 98 27 76 01 02 02 04
pad len: 2, next_header: 4
```

그림 14. 복호화한 후의 패킷 내용

그림13과 그림14를 비교해보면 패킷의 내용이 같음을 알 수 있었으며 실험결과 암호화와 복호화가 잘 운영된다는 것을 알 수 있었다.

4. 결론 및 향후 연구 과제

본 연구에서는 VPN을 구성하는데 있어 핵심 기술이라 할 수 있는 IPSec을 새로운 커널 컴파일 없이 사용할 수 있게 구현하였다. IPSec의 개발시 여러 가지 구현 방법이 있지만 본 연구에서는 서브넷 간을 연결하는데 중점을 두었다. 서브넷의 임의의 호스트에 개발 모듈을 설치하고 그 호스트를 SG로 설정하였고 같은 방식으로 다른 서브넷에서도 이러한 환경을 제공하였다. 결구 이러한 SG 간에서 일어나는 가상적인 터널을 통해 두 서브넷은 가상적인 하나의 서브넷을 구성할 수 있었다.

향후 연구과제로는 제안된 모델에서는 클라이언트들 간이 트랜스포트 모드를 지원하지 않고 있으며 사용되는 암호 알고리즘을 오픈소스인 openSSl을 사용하고 있어 향후 보안상의 공격가능성이 있다고 할 것이다. 또한 따로 데이터베이스를 운영해야 하므로 이에 따른 정책 역시 개발되어야 할 것으로 보인다. 향후 트랜스포트 모드 지원을 통해 호스트간의 터널링 역시 지원될 수 있도록 연구가 지속되어야 하고 독자적으로 개발된 암호 알고리즘을 사용하여 보안에 있어서 좀 더 깊은 연구가 진행되어야 할 것이다. 또한 현재 사용한 리눅스 커널 버전이 2.2.x 버전이며 이러한 커널의 모듈 중 ethertap 모듈을 이용했는데 커널이 2,4.x에서는 이러한 모듈을 통합한 형태인 netfilter 모듈을 지원한다. 따라서 이를 이용한 연구가 계속 진행되어야 할 것이다.

5. 참고 문헌

[1] Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, NRL,

[2] Atkinson, R., "IP Authentication Header", RFC 2401 August 1998.

[3] Atkinson, R., "IP Encapsulating Security Payload", RFC 1827, NRL, August 1995.

[4] Internet Security Association and Key ManagementProtocol(ISAKMP),Douglas Maughan, Mark Schertler, Mark Schneider, Jeff Turner, INTERNET-DRAFT draft-ietf-ipsec-isakmp-08.txt,.ps,July26, 1997.

[5] Tailer, James S. "IPSec Virtual Private Networks"

[6] Dan harkins, "The New Security Standard for the Internet Intranets, and Virtual Private Networks"