情 報 保 護 學 會 誌
第 15 卷 第 2 號, 2005. 4

리눅스 시스템 기반의 IPv6 네트워크 보안 방법

임 재 덕*, 김 기 영*

요 약

대부분의 IT 영역이 네트워크화 되어 서로 유기적으로 융합됨에 따라, 인터넷 영역의 범위는 시간이 지날수록 그 범위가 기하급수적으로 넓어지고 있다. 이런 현상에 따라 현재 사용되는 IPv4 환경의 네트워크로는 그 요구를 수용하기에 한계에 이르렀다. IP를 사용하는 시스템의 영역이 일반 가전, 휴대 단말기 등으로 급속히 확대되면서 IPv4 네트워크 영역에서는 이들 시스템을 수용하기 위해 주소 영역을 확장하기 위한 수단으로 NAT 등과 같은 기법 등을 적용하고 있지만 이 역시 조만간 역부족일 것이다. 그리고 IPv4 네트워크 구조는 보안에 염두에 두지 않고 통신에 그 기능을 맞추고 있던 터라 보안 문제가 심각한 화제로 떠오르고 있는 현실에는 부족함이 없지 않다. 따라서 IPv4의 차세대 버전으로 IPv6가 연구되어 왔고, 현재는 대부분의 운영체제에서 IPv6를 지원하고 있다. IPv6는 주소 표현의 특성상 IPv4와는 비교도 되지 않는 많은 주소를 지원하고 프로토콜 자체에 IPsec 이라는 보안 프로토콜을 지원하여 보안 기능을 기본으로 제공한다. 본 논문에서는 리눅스 시스템 기반의 IPv6 네트워크에서 IPsec을 이용하여 네트워크 트래픽을 보호하는 방법에 대해 설명한다. 본 논문에 적용한 IPsec 프로토콜은 USAGI라는 프로젝트에서 제공하는 IPsec 프로토콜을 사용하여 다양한 모드에서의 IPsec 동작 방법을 설명한다.

Ⅰ. 서 론

대부분의 IT 영역이 네트워크화 되어 서로 유기적으로 융합됨에 따라, 인터넷 영역의 범위는 시간이 지날수록 그 범위가 기하급수적으로 넓어지고 있다. 이런 현상에 따라 현재 사용되는 IPv4 환경의 네트워크로는 그 요구를 수용하기에 한계에 이르렀다. 홈네트워크, RFID 및 유비쿼터스 네트워크 등 현재의 다양한 기술들은 IP를 사용하는 시스템의 영역을 일반적인 네트워크 장비로부터 생활 가전, 휴대 단말기 및 대부분의 IT 장비 등으로 급속히 확대시키면서, IPv4의 주소 고갈을 더욱 앞당기고 있다. IPv4 네트워크 영역에서는 주소 고갈 문제를 해결하기 위해 주소 영역을 확장하기 위한 수단으로 NAT 등과 같은 기법 등을 통해 하나의 공인 IP를 이용하여 수십 대의 시스템을 사용할 수 있도록 하고 있지만, 이 역시 조만간 역부족일 것이다. 그리고 초기의 IPv4 네트워크 구조는 보안을 염두에 두지 않고 통신 기능에 초점을 맞추어 설계된 이유로 별도의 보안 대책을 수립하지 않는다면 보안 문제가 심각한 화제로 떠오르고 있는 현실에 적합하지 않다.[1]

1990년대부터 IETF에서는 IPng WG(IP Next Generation Working Group)을 통해 IPv4의 차세대 버전으로 IPv6에 대한 표준개발을 계속 해 오고 있고, 1996년부터는 IPv6 프로토콜의 실험을 위한 대규모 실험망인 6Bone(IPv6 Backbone)이 구축되어 지금까지 운영되고 있다.[2] 6Bone은 IPv6에 대한 새로운 프로토콜과 운영 구조 등을 테스트해 볼 수 있는 실험망으로 2002년 기준으로 전세계 57개국의 1,000 여개 사이트가 연결되어 있다.[3] 현재는 각 나라마다 고유의 IPv6 망을 운영하고 있고, 우리나라도 한국전산원에서 2001년 8월부터 6NGIX/6KANet를 구축하여 IPv6 시범서비스를 실시하고 있다. 6NGIX(IPv6 Next Generation Internet eXchange)는 국내외의 IPv6 망을 상호 연동시키기 위한 IPv6 기반의 인터넷 교환노드이고, 6KANet(IPv6 Korea Advanced Network)은 국내 공공기관 및 연구기관 등에 IPv6 인터넷 서비스를 제공하는 IPv6 가입자망이다.[4] IPv6 기능을 제공하는 운영체제는 리눅스를 비롯하여 FreeBSD, 솔라리스 등의 대부분

* 한국전자통신연구원 정보보호연구단 보안운영체제연구팀 ({jdscol92, kykim}@etri.re.kr)

의 유닉스 시스템과 윈도우 2000 및 XP 등 대부분의 운영체제에서 IPv6 프로토콜을 지원하고 있고, 리눅스는 커널 버전 2.4 이후부터 IPv6를 지원한다.

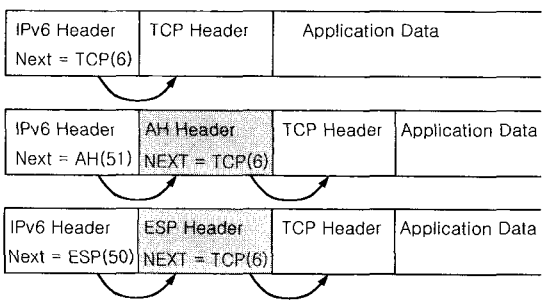
IPv6 프로토콜의 주요 특징으로는 주소 영역의 확장으로 거의 무한개의 주소 제공 기능, 주소 자동 할당 기능, 패킷의 무결성 및 기밀성 제공 기능 등이 있다. 특히, IPv6는 확장 헤더를 통해 패킷의 출처 인증, 데이터 무결성 및 기밀성의 보장 하는 보안 메커니즘을 IPv6 프로토콜 자체로 제공하여 보안 기능을 기본으로 제공한다.[5]

본 논문에서는 IPv6 네트워크에서 IPsec를 제공하는 프로젝트의 종류와 리눅스 시스템 기반의 IPv6 네트워크에서 IPsec을 이용하여 네트워크 트래픽을 보호하는 방법에 대해 설명한다. 본 논문에 적용한 IPsec 프로토콜은 USAGI라는 프로젝트에서 제공하는 IPsec 프로토콜을 사용하여 다양한 모드에서의 IPsec 동작 방법을 설명한다.

II. IPv6 보안

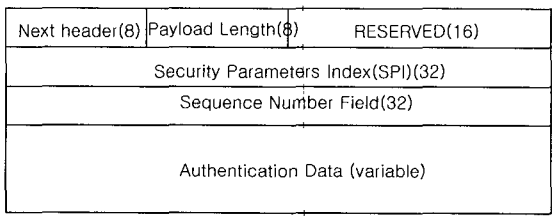
IPv6는 자체적으로 확장 헤더를 통해 패킷의 출처 인증과 패킷의 무결성 및 기밀성을 보장하도록 하고 있으며 확장 헤더가 적용된 IPv6 패킷의 구조는 그림 1과 같다.[6] 이들 헤더는 각각 Authentication Header(AH)와 Encapsulation Security Payload(ESP)로써 RFC 표준 문서로 정의되어 있으며 구조는 그림 2와 그림 3과 같으며 각 헤더 필드의 의미는 다음과 같다.[7,8]

- Next header : AH 혹은 ESP를 따르는 payload 유형 정의
- payload length : AH의 길이를 32비트 워드 단위로 명시
- SPI : 임의의 32비트 값으로 목적지 IP 주소와 보안 프로토콜과 함께 사용되어 이 데이터그램에 적용되는 SA를 명시
- Sequence number : 증가하는 카운터의 값, 재연 공격의 방지에 사용될 수 있다.
- Authentication Data : AH의 경우 해당 패킷을 위한 ICV(initial Check Vector)를 포함. ESP의 경우 Authentication Data를 제외한 부분에 대하여 계산된 ICV 값.
- Payload Data : Next Header 필드에 의해 정의된 데이터를 포함
- Padding : 암호화 알고리즘이 암호화될 데이터가 특정 길이 배수가 되기를 요구할 때 등을 포함한 여러 경우에 사용
- Pad Length : Padding의 길이

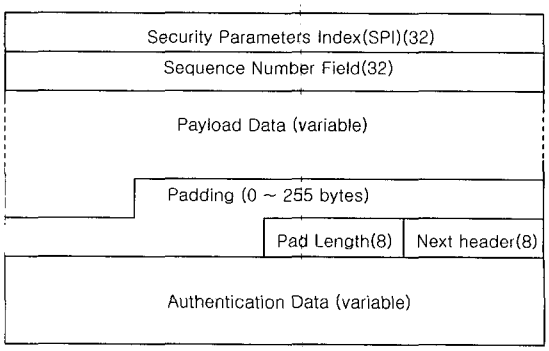


〔그림 1〕 확장 헤더 사용의 예시

이들 헤더는 IPv4에서도 AH 프로토콜과 ESP 프로토콜로서 IPsec 프로토콜로 사용되었다. IPv4에서는 별도의 추가 기능으로 사용해야 했기 때문에 IPsec 사용에 있어 번거로운 작업이 많이 존재했고 상대 시스템에도 IPsec이 지원되는지 확인해야 했다. 하지만, IPv6 에서는 IPsec 프로토콜이 IPv6 프로토콜 규격에 정의되어 있어 IPv6 프로토콜을 사용한다면 이 기능을 기본적으로 이용할 수 있다. 또한, IPv6 장비의 신뢰성 확보 및 기술 보급을 위해 2003년 9월부터 IPv6 포럼에서 시행하고 있는 인증 제도인 IPv6 Ready Logo Program에서는 2005



〔그림 2〕 AH 헤더 구조



〔그림 3〕 ESP 헤더 구조

년 4월부터 두 번째 인증 마크로서 Phase II를 선보였으며 이 인증 기능에는 IPsec 기능도 기본으로 제공하도록 하고 있다.[9]

따라서, 최신의 IPv6 기능 인증을 받기 위해서도 IPsec 프로토콜의 기능 제공이 필수적이며 결론적으로 IPsec 프로토콜의 사용은 IPv6 네트워크에서는 기본이 될 가능성이 크다.

Ⅲ. IPv6 프로젝트

IPv6 프로젝트는 각 나라마다 매우 다양하다. 여기서는 망 위주의 프로젝트 보다 IPsec 구현 위주의 프로젝트에 대해 설명한다. 가장 활발한 프로젝트로는 일본에서 이루어지고 있는 프로젝트로 USAGI와 KEME 프로젝트가 있다. 또한 이들 IPv6 기능 검증을 위한 별도의 기능 검증 프로젝트(TAHI)도 운영하고 있다.

3.1 USAGI Project

처음 USAGI 프로젝트가 시작되었을 당시에도 IPv6 코드가 리눅스 소스 트리에 포함되어 사용 가능하였지만, 당시 리눅스 소스 트리 내의 IPv6 코드는 오래되었고, 제대로 테스트되지 않아 문제점이 종종 발생하였다. 이와 같은 이유로 리눅스 시스템 상의 IPv6 성능을 개선하고 안정화 하기 위해 KAME, WIDE 등의 프로젝트와 같이 시작하였다.[10]

USAGI는 리눅스 커널 내의 IPv6 프로토콜 기능을 개선하고 추가 기능을 구현하는 것과 glibc 라이브러리 내의 IPv6 API 기능을 제공하고 개선하는 것을 목적으로 한다. 최종 목적은 USAGI 프로젝트에 의해 개발된 IPv6 관련 코드를 리눅스 커널과 glibc 소스 트리에 통합하는 것이다.

현재까지 제공하는 IPv6 기능에는 다음과 같다.

- ☐ Catching up RFC2292, RFC2292bis APIs,
- ☐ Catching up RFC2553, RFC2553bis APIs,
- ☐ Implementing ICMPv6 NI Queries,
- ☐ Improving buggy NDP and autoconf stack,
- ☐ Implementing general tunnel device,
- ☐ Implement extension header functions,
- ☐ Implement IPv6 multicast routing,
- ☐ Improving SNMP statistics,
- ☐ IPv6 capable ATM driver and tools,
- ☐ IPsec for IPv4 and IPv6, and
- ☐ Mobile IPv6.

가장 최신의 안정화 버전은 2004년 1월에 릴리즈된 STABLE Release 5로써 리눅스 커널 버전 2.4.21을 기반으로 개발되었다. USAGI 프로젝트는 이 버전을 끝으로 커널 버전 2.4 대에서의 개발을 중단하고 커널 버전 2.6 대에서의 개발을 진행 중이라고 밝히고 있다. 현재 배포된 커널 버전 2.6에서의 IPv6 소스 코드는 USAGI 프로젝트에서 개발한 코드가 포함되어 있다.

3.2 KAME Project

KEME 프로젝트는 1998년 4월에 시작된 과제로 초기에는 2년 과제였으나, 이후 2년씩 2번 더 연장되어, 2004년 3월까지 계속 진행되었다. 프로젝트 기간은 끝이 났지만, 지금도 계속 활동하며 BSD 계열에 IPv6 기술을 적용하고 있다. 핵심 참여 연구원은 후지츠, 히타치, NEC, 도시바, 요코가와 일렉트릭, 케이오 대학 등의 연구원들이고, KAME 코드는 WIDE 프로젝트의 Hydrangea IPv6/IPsec 스택에 기반하고 있다.[11]

KEME는 다양한 BSD 계열에 대해서 IPv6, IPsec (IPv4와 IPv6 모두에 대해) 그리고, Advanced packet queuing, ATM, mobility 등과 같은 internetworking 기술에 free reference implementation 을 제공하는 것을 목적으로 한다.

현재 배포되고 있는 BSD에 통합된 코드는 일반 사용자를 위한 것으로 안정화 테스트를 거쳐 어느 정도 사용에 안정성을 가지고 있다. 이들 코드는 FreeBSD 4.0이후 버전, NetBSD 1.5 이후 버전, Open/BSD 2.7 이후 버전, BSD/OS 4.2 이후 버전에 통합되어 있다.

현재 제공되는 기능은 크게 커널 레벨 기능과 응용 레벨 기능으로 나누어진다. 커널 레벨에서 제공되는 기능은 IPv6 기본 프로토콜을 중심으로 IPsec, 각종 터널링 기술 및 모바일 기술 등이 제공되며 응용 레벨에서의 기능은 telnet, ftp, SMTP, 아파치 웹서버 및 웹 클라이언트 등 IPv4 환경에서 사용할 수 있는 대부분의 기본 네트워크 응용 프로그램을 비롯하여 IPv6용 라우팅 프로토콜, IPv6 상의 X 환경 등이 제공된다.

3.3 TAHI Project

TAHI는 IPv6 프로토콜에 대해 유효성 및 상호 운용성 검사를 시행하는 프로젝트이다. TAHI 프로젝트는 KAME와 USAGI 프로젝트에서 자신들이 개발

한 IPv6 기능의 유효성을 검사하기 위해 별도의 시험 프로젝트를 의뢰하여 시작되었으며, 주관기관은 동경대와 요코가와 전기회사이다. 프로젝트에서 제공하는 시험 툴은 공개 툴로서 웹이 공개되어 있고, KAME 및 USAGI의 IPv6 규격 및 상호 운용성 시험을 해주고 있다. 현재 IPv6 Ready Logo phase-2를 위한 테스트 슈트로써 Release-1.2.11 버전이 2005년 3월 11일 배포되었다.[12]

테스트 항목은 기능별로 큰 부류로 정해지고, 이들 부류 내에 또다시 세부 테스트 항목이 나오며 이는 RFC 규격을 따른다. 테스트 항목은 호스트이나 라우터이냐에 따라 차이가 있다.

Ⅳ. IPv6 환경에서의 IPsec 프로토콜 운용

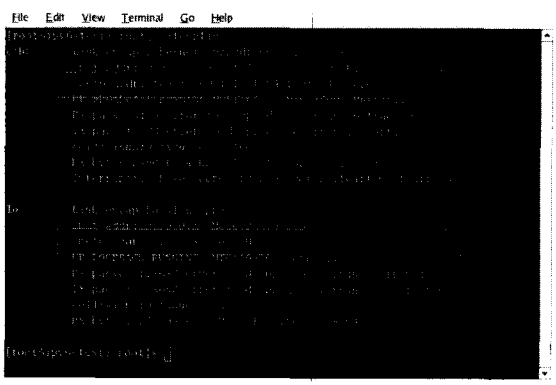
이번 절에서는 리눅스 시스템에서 IPv6 보안 프로토콜을 운용하는 방법에 대해 설명한다. 사용할 프로토콜은 USAGI 프로젝트에서 제공하는 안정화 버전 STABLE Release 5을 사용한다. 시스템은 RedHat Linux 9.0을 운영체제로 하고 리눅스 커널 버전은 2.4.21이다.[10,13,14]

4.1 IPv6 환경 구축

IPv6 환경의 네트워크 사용하기 위해서는 IPv6 프로토콜 스택을 설치하여야 한다. 본 논문에서는 Linux 시스템에서 IPv6 환경을 구축할 것이다. 기본적으로 Linux 커널 버전 2.4 대는 IPv6 프로토콜 스택을 포함하고 있지만, - 사용하기 위해서는 IPv6 옵션을 선택하여 커널을 컴파일해야 한다. - 안정성에 문제가 있어 USAGI 프로젝트에서 배포하는 IPv6 프로토콜 스택을 사용한다. 패키지에는 IPv6 기반의 응용 프로그램 즉, ping6, telent6, ftp6 등을 포함하고 있으며, IPv4 뿐만 아니라 IPv6 환경에서 운용할 수 있는 IPsec 프로토콜도 포함하고 있다. 패키지에는 Linux 커널 버전 2.4.21을 기반으로 하고 패키지에 이 버전의 커널이 포함되어 있어 별도로 커널을 구하지 않아도 된다.

4.2 IPv6 네트워크 확인

IPv6가 지원 여부는 /proc 파일시스템을 통해 확인 가능하다. 즉, IPv6 프로토콜이 동작하고 있다면 '/proc/net/if_inet6' 의 파일을 볼 수 있다. 패키지를 제대로 설치하였다면 IPv6 프로토콜은 모듈로 동작하므로 만약 IPv6 프로토콜이 동작하고 있지 않다면 IPv6 모듈을 적재함으로써 IPv6 프로토콜을 동작시킬 수 있다. IPv6 모듈이 제대로 동작하고 있다면 시스템이 가진 인터페이스는 그림 4와 같이 IPv6 주소를 할당 받는다. 운용 환경에서는 시스템들간에 링크로 연결되어 있고 라우터가 존재하지 않으므로 링크 주소만 할당되어 진다. 주소 할당은 자동 주소 할당 방법에 의해 얻어지며 이는 stateless autoconfiguration 문서를 참조하면 된다.[15] 두 시스템이 IPv6 주소로 통신이 가능한지의 확인은 간단한 IPv6 응용인 ping6 프로그램을 통해 가능하다.



〔그림 4〕 IPv6 주소 확인

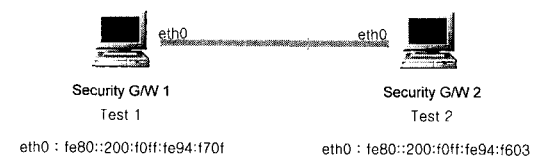
4.3 IPv6 IPsec 프로토콜 운용

IPsec 운용은 transport mode와 tunnel mode에서 운용하였고, 각 모드에 대해 manual key 방식과 auto key 방식으로 키를 교환하여 운용하였다.

4.3.1 Transport Mode

Transport Mode는 호스트에서만 구현이 가능하며 종단 대 종단에 위치한 두 호스트 간에 이용되며 IP 패킷의 페이로드, 즉, TCP와 UDP와 같은 상위 계층 프로토콜 데이터만을 보호하게 된다. 이럴 경우 보안 헤더가 원래 헤더와 데이터 사이에 위치하여 원래 헤더가 보호 영역 범위를 벗어남으로 송신자와 수신자 주소가 노출로 인해 패킷 흐름이 노출된다.

운용은 그림 5와 같은 환경에서 이루어진다.



〔그림 5〕 Tranport Mode 운용을 위한 시스템 구성

4.3.1.1 Manual Key 운용

매뉴얼 키 운용은 두 호스트 간에 미리 협상된 SP 및 SA를 자신의 호스트에 설정하여 바로 IPsec 통신을 가능하게 하는 것이다. 사용되는 명령어는 'pfkey'이다. 우선 AH 운용을 설명한다. 설정할 SA는 다음과 같다.

- mode : transport AH
- AH auth algo : hmac-md5
  (key 0x0123456789abcdef0123456789abcdef)
- SPI : TEST1 -> TEST2 AH:0x1234,
  TEST2 -> TEST1 AH:0x9abc
- 터널을 통해 사용할 Protocol : ICMP

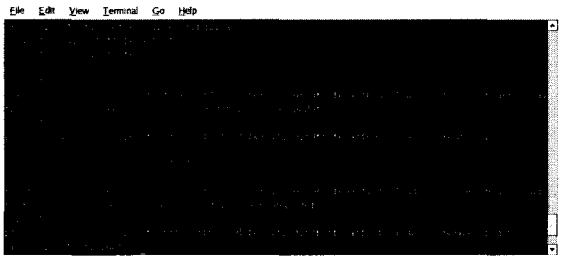
SA(Security Association)와 SP(Security Policy)는 그림 6과 같이 적용하며, 제시된 명령어는 Test1 시스템에서의 명령어이다. Test 2 시스템에서는 주소 영역만 수정하여 동일하게 적용하면 된다. SA는 패킷에 적용할 알고리즘 프로토콜 등의 정보이며, SP는 어떤 패킷에 적용할 지에 대한 정책이다. 터널 생성 확인은 생성된 터널을 통해 ping6를 수행한 후, tcpdump를 통해 가능하며, 그림 7은 AH가 적용되었음을 보여준다.

AH 프로토콜은 패킷에 대한 무결성만 제공하므로 패킷 데이터를 확인할 수 있어 AH가 보호하는 프로토콜이 무엇인지 확인이 가능하며 그림 7은 이를 보여준다.

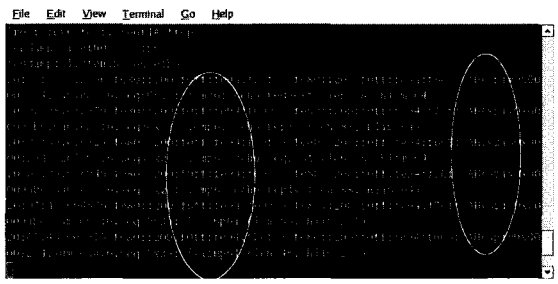
ESP 운용은 AH와 유사하며 기밀성 보장을 위해 암호화 알고리즘이 추가된다. 설정할 SA는 다음과 같다.

- mode : transport ESP
- ESP auth algo: hmac-md5
  (key 0x0123456789abcdef0123456789abcdef)
- ESP enc algo: 3des-cbc
  (key
  0xa7a36ebd91863edfba763fa7edcba64d89
  123ace6359 eba7)
- SPI : TEST1 -> TEST2 ESP:0x5678,
  TEST2 -> TEST1 ESP:0xdef0
- 터널을 통해 사용할 Protocol : ICMP

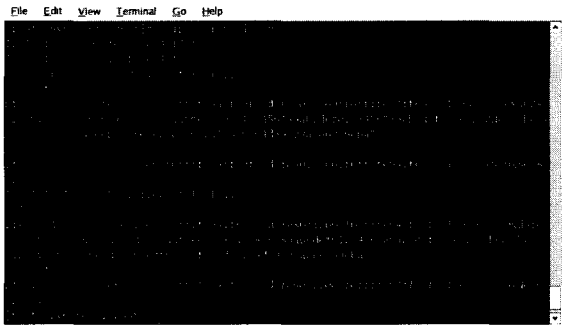
SA와 SP는 그림 8과 같이 적용하며, AH와 마찬가지로 터널 생성 확인은 ping6 프로그램을 수행한 후, tcpdump를 통해 확인한다. 그림 9는 ping6 패킷에 ESP가 적용되었음을 보여준다. AH와 달리 ESP는 기밀성을 제공하므로 ESP가 보호하는 프로토콜이 무엇



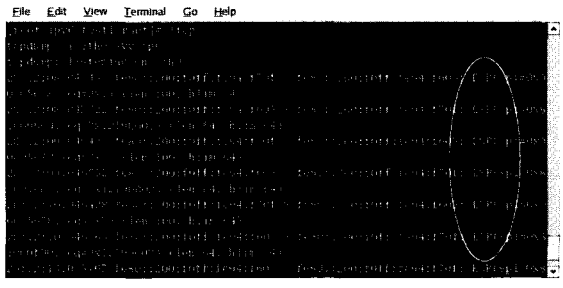
(그림 6) Tranport Mode에서 Manual Key 방식으로 AH 적용



(그림 7) Tcpdump를 이용한 AH 적용 확인



(그림 8) Tranport Mode에서 Manual Key 방식으로 ESP 적용



(그림 9) Tcpdump를 이용한 ESP 적용 확인

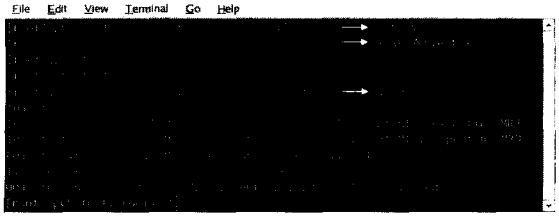
인지 알 수 없고 이는 그림 9에서도 확인된다.

4.3.1.2 Auto Key 운용

Manual Key 운용에서는 패킷에 적용할 키 및 SA 등의 정보를 상대 호스트와 미리 약속된 내용으로 별도의 협상과정 없이 설정하여 사용하였지만, Auto Key 운용에서는 터널 생성이 필요한 시점에 직접 네트워크 상에서 터널을 생성할 상대 호스트와의 협상을 통해 결정된다. 즉, Auto Key 운용에서는 Manual Key 운용 시 수동으로 입력한 SPI, Key, SA 등의 정보를 상대 호스트와의 협상을 통해 자동으로 설정한다. Auto Key 운용 시 한 호스트는 자신이 제공할 수 있는 임의의 SP 및 SA를 상대 호스트로 보내 상대 호스트가 제공가능한지를 협상한다. 상대 호스트가 수신한 SP 및 SA를 제공할 수 있으면 수신한 SP 및 SA를 수용하고 상대 호스트에게도 알린다. 이 때 각각의 호스트는 상대 호스트를 인증하는 기능도 포함하고 있다. 이에 대한 교환 프로토콜은 RFC 규격으로 ISAKMP(Internet Security Association and Key Management Procotol) 프로토콜과 IKE (Internet Key Exchange) 프로토콜 등으로 정의되어 있다.[16,17]

본 논문에서 운용한 USAGI 패키지에서는 'pluto'라는 데몬을 통해 SP 및 SA 협상이 이루어지며, 데몬의 세부 제어는 해당 문서를 참조한다.[14]

Auto Key 운용을 하기 전에 협상할 호스트는 상대 호스트가 협상할 호스트가 맞는지 인증하는 과정을 거쳐야 한다. 협상할 상대 호스트를 인증하는 방법은 RFC 규격서에 4가지로 정의하고 있으나, 이들 중 pre-shared key 방법과 인증서를 이용한 RSA 서명 방법이 주로 사용된다. Pre-shared Key 방법은 미리 두 호스트가 약속한 일종의 password를 가지고 있다가 상대 호스트의 메시지를 이 키를 이용하여 인증함으로써 상대 호스트를 인증하는 방법이다. 이를 위해 USAGI에서는 기본 디렉토리(/usr/local/v6/etc)에 'ipsec.secrets' 라는 파일 내에 설정한다. 그리고 난 후, 'pluto' 데몬이 이 키를 인식할 수 있도록 해주어야 한다. 이 외에 협상할 SA 및 SP는 같은 디렉토리 내의 'ipsec.conf' 파일에 명시하면 된다.



[그림 10] Auto Key 운용을 통한 SA 및 SP 협상

Auto key 운용을 이용한 협상은 상대 인증을 위한 키를 데몬에게 알려주는 과정(키초기화)을 시작으로, 상대 호스트와 협상을 진행할 SA 및 SP을 데몬에게 알려주고(SP와 SA rule 추가), 마지막으로 협상 요청(협상 시작)을 통해 상대 호스트와 SP와 SA를 협상하는 것으로 이루어진다. 이 때 협상 요청은 사용자에 의한 요청과 상대 호스트의 협상 요청 등이 될 수 있다.

그림 10은 데몬을 이용하여 키협상을 진행하는 과정을 보여준다. 협상 과정에서 보이는 메시지 중 ISAKMP SA는 Phase 1 협상의 결과이고, IPsec SA는 Phase 2 협상의 결과이다. ISAKMP SA는 Phase 2 협상 메시지들에게 적용되고, IPsec SA는 실제 호스트를 통과하는 패킷들에게 적용된다. 두 SA 모두 유효 시간을 가지고 있어 유효 시간이 지나기 전에 새로운 SA를 협상해야 한다.

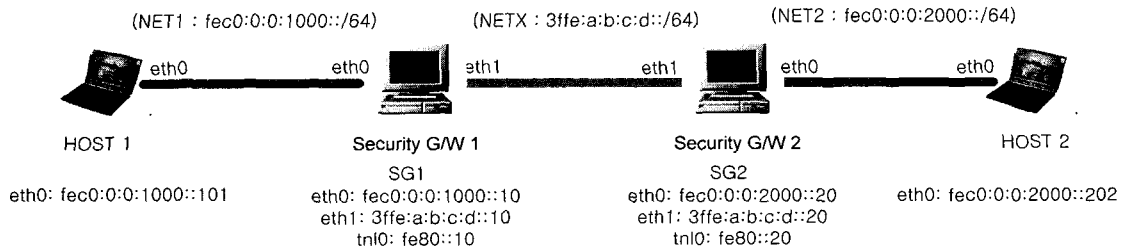
협상이 끝난 후, IPsec SA가 생성되었다는 메시지가 보이면 정상적으로 터널이 생성되었음을 나타낸다. 이 경우 생성된 터널의 생성 확인 역시 Manual Key 운용에서와 같이 ping6과 같은 프로그램을 통해 확인할 수 있다.

4.3.2 Tunnel Mode

Tunnel Mode는 호스트나 보안 게이트웨이에서 사용될 수 있으며, AH나 ESP가 보안 게이트웨이에 구현된 경우 반드시 tunnel mode가 사용되어야 한다. Tunnel mode가 적용되는 패킷은 터널링되는 시스템에서 전혀 새로운 패킷으로 재생성되며 이 때 재생성되는 IP의 출발지 목적지 주소는 터널링이 이루어지는 시스템과 터널링이 해제되는 시스템의 주소가 된다. 또한 본래의 출발지와 목적지 주소는 터널링 헤더 내부에 보호되므로 터널링이 해제되는 시스템에서 그 정보를 확인할 수 있을 뿐 아니라, 터널링 구간에서는 외부에 그 정보가 노출되지 않아 transport mode에서의 패킷보다 안전하다.

본 논문에서는 그림 11과 같은 네트워크 구성에서 tunnel mode를 운용하며, 사용할 SA는 다음과 같다. 다음 SA는 manual key 운용 및 auto key 운용에 모두 사용된다.

- mode : tunnel AH/ESP



[그림 11] Tunnel Mode 운용을 위한 네트워크 구성

- AH algo : hmac-md5
  (key 0x0123456789abcdef0123456789abcdef)
- ESP auth algo : hmac-md5
  (key 0x0123456789abcdef0123456789abcdef)
- ESP enc algo : 3des-cbc
  (key
  0xa7a36ebd91863edfba763fa7edcba64d89
  123ace6359eba7)
- SPI : SG1 -> SG2 AH:0x1234 ESP:0x5678,
  SG2 -> SG1 AH:0x9abc ESP:0xdef0

4.3.2.1 Manual Key 운용

매뉴얼 키를 운용하기 위해서는 각 노드 즉, HOST1과 HOST2의 인터페이스를 그림 11과 같이 설정하고, SG1과 SG2를 라우터로 설정하여야 한다. 만약 HOST1과 HOST2의 인터페이스 주소를 SG1과 SG2로부터 Router Advertisement에 의해 설정하고자 한다면 다음과 같은 수동적인 설정은 필요없다.

- HOST1 설정

```
# ip addr add fec0:0:0:1000::101/64 dev eth0
```

- HOST2 설정

```
# ip addr add fec0:0:0:2000::202/64 dev eth0
```

- SG1 라우터 설정

```
# ip addr add fec0:0:0:1000::10/64 dev eth0
# ip addr add 3ffe:a:b:c:d::10/64 dev eth1
# sysctl -w net/ipv6/conf/all/forwarding=1
# sysctl -w net/ipv6/conf/all/accept_ra=0
```

- SG2 라우터 설정

```
# ip addr add fec0:0:0:2000::20/64 dev eth0
# ip addr add 3ffe:a:b:c:d::20/64 dev eth1
# sysctl -w net/ipv6/conf/all/forwarding=1
# sysctl -w net/ipv6/conf/all/accept_ra=0
```

그러고 난 후, SG1과 SG2 사이 즉, NETX 구간에 터널 디바이스를 셋팅해야 하고, 라우팅을 설정해야 한다. 터널 설정을 확인하려면 아래 설정을 마친 후 'ipv6tunnel show tnl0' 로 확인하면 된다.

- SG1 설정

```
# modprobe ipv6_tunnel
# ipv6tunnel add tnl0 —tunnel-local-packets
  encaplimit 0 \ remote 3ffe:a:b:c:d::20
  local 3ffe:a:b:c:d::10
# ip link set tnl0 up
# ip addr add fe80::10 dev tnl0
# ip route add fec0:0:0:2000::/64 dev tnl0
  (in IPv4 case)
# ip tunnel add tnl0 mode ipip local
  192.168.1.1 \ remote 192.168.2.1
# ip addr add 10.0.254.1/30 dev tnl0
# ip link set tnl0 up
# ip route add 10.0.2.0/24 dev tnl0
```

- SG2 설정

```
# modprobe ipv6_tunnel
# ipv6tunnel add tnl0 —tunnel-local-packets
  encaplimit 0 \ remote 3ffe:a:b:c:d::10 lo-
  cal 3ffe:a:b:c:d::20
# ip link set tnl0 up
# ip addr add fe80::20 dev tnl0
# ip route add fec0:0:0:1000::/64 dev tnl0
  (in IPv4 case)
# ip tunnel add tnl0 mode ipip local
  192.168.2.1 \ remote 192.168.1.1
# ip addr add 10.0.254.2/30 dev tnl0
```

```
# ip link set tnl0 up
# ip route add 10.0.1.0/24 dev tnl0
```

위 상태로 설정한 후, ping 프로그램을 통해 HOST1과 HOST2 사이의 동작을 확인하고, 만약 제대로 되지 않을 경우 위 설정을 다시 한 번 검사한다.

이제, SG1과 SG2 두 시스템 모두에서 SA와 SPD를 다음과 같이 설정한다.

- SG1 □ SG2 에 관한 설정

```
# pfkey -A sa -s 3ffe:a:b:c:d::10 -d 3ffe:a:b:c:d::20 \ -T ah -S 0x1234 --auth hmac-md5 \ --authkey 0x0123456789abcdef0123456789abcdef
# pfkey -A sa -s 3ffe:a:b:c:d::10 -d 3ffe:a:b:c:d::20 \ -T esp -S 0x5678 --esp 3des-cbc --espkey \ 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7 \ --auth hmac-md5 --authkey \ x0123456789abcdef0123456789abcdef
# pfkey -A sp -s fec0:0:0:1000::/64 -d fec0:0:0:2000::/64 \ -T ah -S 0x1234 --tunnel --sad 3ffe:a:b:c:d::20
# pfkey -A sp -s fec0:0:0:1000::/64 -d fec0:0:0:2000::/64 \ -T esp -S 0x5678 --tunnel --sad 3ffe:a:b:c:d::20
```

- SG2 □ SG1에 관한 설정

```
# pfkey -A sa -s 3ffe:a:b:c:d::20 -d 3ffe:a:b:c:d::10 -T ah \ -S 0x9abc --auth hmac-md5 \ --authkey 0x0123456789abcdef0123456789abcdef
# pfkey -A sa -s 3ffe:a:b:c:d::20 -d 3ffe:a:b:c:d::10 -T esp \ -S 0xdef0 --esp 3des-cbc \ --espkey \ 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7 \ --auth hmac-md5 --authkey \ 0x0123456789abcdef0123456789abcdef
# pfkey -A sp -s fec0:0:0:2000::/64 -d fec0:0:0:1000::/64 \ -T ah -S 0x9abc --tunnel -sad 3ffe:a:b:c:d::10
# pfkey -A sp -s fec0:0:0:2000::/64 -d fec0:0:0:1000::/64 \ -T esp -S 0xdef0 --tunnel -sad 3ffe:a:b:c:d::10
```

이제 터널이 생성되었으며, HOST1과 HOST2 사이에 ping 등의 프로그램으로 IPsec 프로토콜이 적용되었는지 확인하면 된다.

4.3.2.2 Auto Key 운용

Tunnel Mode도 Transport Mode와 마찬가지로 'pluto' 라는 키 협상 데몬을 통해 터널 모드 협상을 하며, 운영 방법도 Transport Mode와 동일하다. 단지 SA 및 SP를 설정하는 과정에서 tunnel mode를 명시해야 한다. Auto keu 운용을 위한 설정은 두 개의 파일을 통해 이루어진다. 하나는 Pre-shared Key와 RSA 공개키 및 비밀키 쌍의 정보를 가지는 'ipsec.secret' 키 파일이고, 다른 하나는 SP 및 SA 정보를 가지는 'ipsec.conf' 설정 파일이다.

키 파일에 저장되는 키 정보 중 Pre-shared key 정보는 다음과 같은 방법으로 저장한다. 아래의 설정은 지금까지 설명한 운용에 적용된다.

```
fe80::200:f0ff:fe94:f70ffe80::200:f0ff:fe94:f603 : PSK "1234567890qwertyuiop"
10.2.1.5 10.2.1.1 : PSK"1234567890qwertyuiop"
```

윗줄은 IPv6 주소에 대한 설정이고 아랫줄은 IPv4 주소에 대한 설정이다. 즉,

```
host1_addr host2_addr : PSK "your pre-shared key"
```

형식으로 키를 저장한다.

설정 파일에 저장되는 SA 및 SP는 다음과 같은 형식으로 저장되며, 지금까지 설명된 운용에 적용된다. 저장된 SA 및 SP는 각각 제 ID(connection name)를 가지며 이를 통해 키협상 데몬에 의해 제어가 된다.

```
- Transport Mode
conn testv6
        af=inet6
        type=transport
        authby=secret
        left=fe80::200:f0ff:fe94:f70f
        right=fe80::200:f0ff:fe94:f603
```

```
        esp=3des-md5-96
        ah=hmac-md5-96

- Tunnel Mode
conn tunv6
        af=inet6
        authby=secret
        left=3ffe:a:b:c:d::10
        leftsubnet=fec0:0:0:1000::/64
        leftnexthop=3ffe:a:b:c:d::20
        right=3ffe:a:b:c:d::20
        rightsubnet=fec0:0:0:2000::/64
        rightnexthop=3ffe:a:b:c:d::10
        esp=3des-md5-96
        ah=hmac-md5-96
```

Tunnel Mode는 터널 엔드에서 보호하려는 호스트들을 명시하는 부분이 추가된다.

V. 결 론

현재 IT의 발전 방향을 보면 모든 시스템이 하나의 네트워크로 묶여 유기적으로 운용되는 것이 추세이다. 이는 비단 IT 분야에만 국한된 것이 아니라 자동차, 건설 등의 비 IT 분야에까지 그 영역이 확대되고 있다. 자동차 산업에 텔레메틱스 기술이 적용되는 현상과 건설 산업에 홈네트워크 기술이 적용되고 있는 것은 대표적인 예이다. 이는 제어가 필요한 모든 기기는 네트워크 즉, 인터넷에 연결되어야 하고 이는 모든 기기가 IP 주소를 가져야 함을 의미한다. 하지만, 현재의 IPv4 구조로는 이들의 요구를 수용할 수 없다. 이미 현재의 상황으로도 IPv4가 제공할 수 있는 주소 할당 능력은 거의 다 소진되었다고 볼 수 있다. 이에 따라 차세대 버전인 IPv6가 대두되었고, 이는 무한대에 가까운 주소 할당 능력, 자동 주소 할당 기능, 프로토콜 자체에 포함된 보안 기능 등의 제공으로 차세대 네트워크에 적용될 것이다. 현재는 필요한 기본 기술들이 대부분 구현되어 있으나 이미 구축되어 있는 IPv4 환경이 너무 방대하여 IPv6 환경으로의 전환이 더디어 지고 있다. 하지만, 각국의 IPv6 적용 노력이 활발하고, 이미 IPv6 기술이 적용된 장비들이 많이 등장하여 머지않아 IPv6 환경이 정착될 것이라 본다.

현재까지는 패킷의 빠른 전달에 네트워크 기술이 집중되었다면, 미래는 패킷의 안전한 전달이 주목적이 될 것이다. 이는 점차 심각해지고 있는 네트워크 침해로부터 네트워크 사용자의 정보를 얼마나 안전하게 보호하면서 전달할 수 있나 라는 문제에 해당한다. 즉, 안전한 네트워크 환경의 구축이 앞으로의 화두가 될 것이다. 안전한 네트워크 환경 구축을 위한 기술로는 방화벽, IPS 등 많은 기술을 적용할 수 있지만, IPv6 프로토콜 레벨에서 자체로 제공되는 패킷의 무결성, 기밀성 기술도 무척이나 중요하다.

본 논문은 이미 배포된 많은 IPv6 프로토콜 구현 중 리눅스에 적용 가능한 구현을 이용하여 패킷에 무결성과 기밀성을 제공할 수 있는 다양한 방법을 제시하였다. 다양한 방법만큼이나 다양한 설정의 노력이 필요한 만큼 네트워크 관리자라면 설정 방법을 숙지하여 보다 안전한 네트워크 환경을 구축하는데 기여해야 한다.

참 고 문 헌

[1] Salvia Hagen, *IPv6 Essentials*, O'REILLY, pp.1-16.

[2] IP Version 6 Working Group (ipv6), http://www.ietf.org/html.charters/ipv6-charter.html.

[3] 6Bone, testbed for deployment of IPv6, http://www.6bine.net.

[4] IPv6 in NCA, 한국전산원 IPv6 서비스, http://www.ngix.ne.kr

[5] IPv6 Status Report 2004, "IPv6 동향 2004," 한국전산원, NCAII-RER-04144, pp18-25, 2004.

[6] S. Deering, R. Hinden, "*Internet Protocol, Version 6 (IPv6) Specification*," RFC 1883, Dec 1998.

[7] S. Kent, R. Atkinson, "*IP Authentication Header*," RFC 2402, Nov 1998.

[8] S. Kent, R. Atkinson, "*IP Encapsulating Security Payload (ESP)*," RFC 2406, Nov 1998.

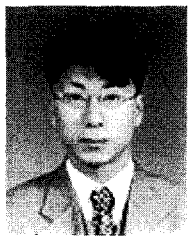
[9] IPv6 Ready Logo Program, http://ipv6ready.org

[10] USAGI(UniverSAl playGround for Ipv6) project - Linux IPv6 Development Project, http://www.linux-ipv6.org

[11] KAME Project, http://www.kame.net

[12] TAHI project, Test and Verification for IPv6, http://www.tahi.org
[13] Linux IPv6 HOWTO, 2004.08.30, http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Linux+IPv6-HOWTO.html
[14] 임재덕, "*IPv6 네트워크 환경에서의 IPsec 프로토콜의 운용*," TM-200403673, 한국전자통신연구원, Oct 2004.
[15] S. Thomson, T. Narten, "*IPv6 Stateless Address Autoconfiguration*," RFC 2462, Dec 1998.
[16] D. Maughan, M. Schertler, M. Schneider, J. Turner, "*Internet Security Association and Key Management Protocol (ISAKMP)*," RFC 2408, Nov 1998.
[17] D. Harkins, D. Carrel, "*The Internet Key Exchange (IKE)*," RFC 2409, Nov 1998.

〈著 者 紹 介〉



임 재 덕 (Lim Jae Deok)
정회원

1999년 : 경북대학교 전자공학과 졸업(학사)
2001년 : 경북대학교 대학원 전자공학과 졸업(석사)
2000년~현재 : 한국전자통신연구원 연구원
〈관심분야〉 시스템 보안, 네트워크 보안, 운영체제, IPv6 보안기술 등

김 기 영 (Kim Ki Young)
정회원

1988년 : 전남대학교 전산통계학과 졸업(이학사)
1993년 : 전남대학교 대학원 전산통계학과 졸업(이학석사)
2002년 : 충북대학교 대학원 전자계산학과 졸업(이학박사)
1988년 2월~현재 : 한국전자통신연구원 보안운영체제 연구팀 팀장
〈관심분야〉 네트워크 보안, 고성능 네트워크 침입탐지 및 대응기술, IPv6 보안기술 등