通信情報保護學會學會誌
第 9 券 第 4 號, 1999. 12

SOCKS VPN 소개

백 석철*

요 약

인터넷을 통한 전자상거래가 활성화되면서 기업 또는 개인들 사이에서 인터넷을 통한 안전한 데이터 통신에 대한 욕구가 날로 증가하고 있는 실정이다. 이러한 요구 사항을 만족시키기 위하여 가상사설망(VPN) 기술에 대한 관심 또한 지대하다고 할 수 있다. 해당 기술로는 IP 또는 데이터링크 계층에 작동하는 VPN 프로토콜인 IPSEC, PPTP 외에 응용계층에서 작동하는 SOCKS VPN 등이 있다. 본 논문에서는 SOCKS VPN에 대해서 설명하고자 한다. 우선, VPN 기술에 대한 간략한 소개를 한 다음 SOCKS 기능에 대하여 상술하고 SOCKS V.5를 이용한 VPN 구축 방법을 서술한다. SOCKS VPN을 구축하는데 핵심 모듈인 SOCKS V.5의 GSS-API는 수출규제를 받기 때문에 하나의 대안으로 SOCKS 4.3과 SSL을 이용한 VPN 구축 방법을 제안한다.

Ⅰ. 서 론

인터넷의 사용자 수는 1996년 말 집계에 따르면 5천만 정도 되었으나 2000년이 되면 2억 정도에 이를 것이라는 것이 전문가들의 분석이다.[1] 한편, 많은 NSP(Network Service Provider)들은 이러한 예측에 입각하여 많은 투자를 해왔으나 오히려 많은 손실을 보게 되었다. 사실 일반 사용자들의 인터넷 이용 방식은 이들 NSP들의 이윤 창출에 별반 기여를 못하기 때문이었다.

현재 대부분 NSP들은 인터넷을 통한 이익은 기업고객의 확보를 통하여 얻을 수 있다는 판단을 하게 되었다. 기업들도 자사의 제품이나 서비스를 홍보, 판매, 고객 지원, 전자메일을 통한 내, 외부 협력 등을 하면서 인터넷 활용의 중요성을 확신하게 되었다. 특히 기업들은 비용이 많이 드는 사설망에 대한 대체 수단으로서 인터넷을 고려하고 있다. 이러한 요구사항에 부응하기 위하여 요즈음 NSP들은 VPN(Virtual Private Networks)이라는 상품을 앞 다투어 내놓고 있다.

이들이 강력히 추진하고 있는 VPN은 인터넷을 이용하여 기업들이 내부 또는 타 기업과 통신을 비교적 저렴하고 안전하게 할 수 있도록 서비스하는 것을 핵심 목표로 한다.

현재 VPN 프로토콜로 주목받고 있는 것들로는 IP계층에서 작동되는 IPSEC, 데이터링크 계층의 PPTP 등과 응용계층에서 작동하는 SOCKS V.5 가 있다. 본 논문에서는 응용계층의 VPN 표준으로 자리잡아가고 있는 SOCKS V.5에 대하여 설명하기로 한다.

Ⅱ장에서는 SOCKS 프로토콜에 대하여 서술한다. Ⅲ장에서는 SOCKS V.5를 이용하여 VPN을 구축하는 방법에 대하여 소개하기로 한다. Ⅳ장에서는 SOCKS VPN을 이용 방안과 향후 연구과제에 대하여 설명한다.

* (주)시큐어소프트 보안연구소 (scbaek@securesoft.co.kr)

Ⅱ. SOCKS

1. SOCKS 개요

SOCKS[2]는 두 컴퓨터 사이에 안전한 프록시 데이터 채널을 설정하기 위한 메커니즘으로 클라이언트-서버 환경을 위하여 고안된 것이다. 클라이언트 입장에서 보면 SOCKS는 없는 것처럼 작동한다. 즉, 투명성을 보장해준다. 그리고 서버 측에서 SOCKS는 하나의 클라이언트로 간주될 수 있다. SOCKS는 SOCKS 서버와 SOCKS 클라이언트 라이브러리로 구성되어 있다. 그림 1에 예시된 바와 같이 SOCKS 서버는 OSI 7 계층 중에서 응용 계층에 위치하고 있지만 SOCKS 클라이언트 라이브러리는 클라이언트의 응용계층과 수송계층 사이에 존재한다.

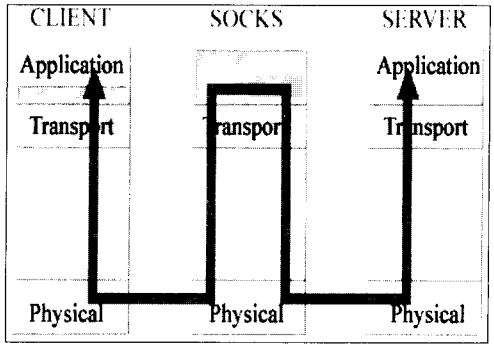

그림1. SOCKS의 OSI 계층 내 위치

이러한 단순성 때문에 SOCKS는 circuit-level 방화벽으로 널리 쓰이고 있다. 그리고 버전 4에서 5로 발전하면서 SOCKS의 기능이 많이 보강되었다. 특히, 인증 기능의 보강 및 UDP 프로토콜의 지원이 추가되었다.

2. SOCKS 작동원리

이러한 SOCKS의 역할은 다음과 같이 요약될 수 있다. 그림 2에 예시된 것처럼 네트웍 내부의 클라이언트가 인터넷상에 존재하는 특정 서버와 연결을 원할 때, 우선 응용 게이트웨이 상에 존재하는 프락시 서버인 SOCKS의 sockd와 연결을 맺어야 한다. 이 후에 클라이언트는 프락시 서버에게 실제로 연결을 하고자 하는 인터넷상의 서버 주소를 SOCKS자체 프로토콜을 이용하여 알려준다. sockd는 넘겨받은 주소와의 연결 시도를 내부 클라이언트를 대신하여 수행한다. 이러한 작업은 일반적인 클라이언트-서버 연결 프로토콜을 이용하여 수행된다. 이 연결 작업이 성공리에 끝나면 sockd는 내부 클라이언트와 외부 서버 사이에서 데이터를 relay해주는 역할을 수행하게 된다. SOCKS는 Circuit-Level 방화벽으로 널리 사용되고 있다. SOCKS 서버는 private 네트웍과 public 네트웍 사이에 위치하고 있으면서, private 네트웍으로부터 public network 의 access를 가능케 한다. 그러나, public 네트웍으로부터 private 네트웍의 access는 차단한다. 그러므로 SOCKS는 private 및 public 네트웍 상의 모든 사용자들에게 있어서 중앙 접점의 역할을 하게 된다.

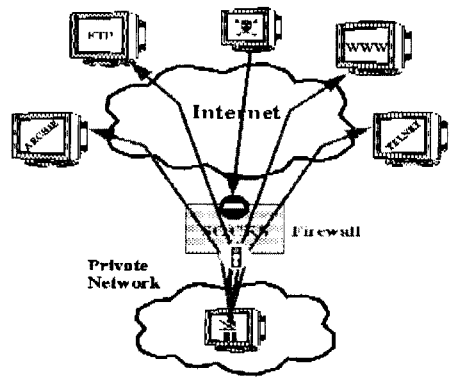

그림2. SOCKS의 역할

그러나 SOCKS의 경우는 TIS Firewall Toolkit와는 다르게 클라이언트 프로그램을 sockd와 통신이 가능하도록 변경시켜야 하는 불편이 있다. 이러한 작업은 클라이언트 프로그램 소스 코드를 요하며 이들이 사용하고 있는 Berkeley Socket API 중에서 몇몇 함수들을 SOCKS가 제공하는 libsocks.a 라는 라이브러리를 이용하여 교체하여야 한다. 이를 Socksify한다고 한다. 교체되어야 하는 함수로는 다음과 같은 것들이 있다.

Functions	Parameters
connect => **R**connect	(int socket, struct sockaddr *name, int namelen)
bind => **R**bind	(int socket, struct sockaddr *name, int namelen, ***struct sockaddr *remote***)
listen => **R**listen	(int socket, int backlog)
getsockname => **R**getsockname	(int socket, struct sockaddr *name, int *namelen)
accept => **R**accept	(int socket, struct sockaddr *addr, int *addrlen)

표1 SOCKS Library Routines

표 1에서 알 수 있는 것처럼 일반적인 소켓 콜 함수 앞에 **R**을 추가한 것과 Rbind( ) 함수에서 bind( ) 함수보다 ***struct sockaddr *remote*** 라는 파라미터가 추가된 것 이외에는 별 다른 변화가 없음을 알 수 있다. 여기서 Rbind( )에 추가된 파라미터는 sockd가 연결을 시도하거나 연결을 허락하지 않을 인터넷상에 존재하는 호스트의 주소이다.

SOCKS클라이언트와 방화벽의 응용 게이트웨이 상에서 수행되는 sockd 서버 사이에 통용되는 프로토콜은 Rconnect와 Rbind로 구성된다. 이들의 작동 상황은 그림3과 4로 요약될 수 있다.

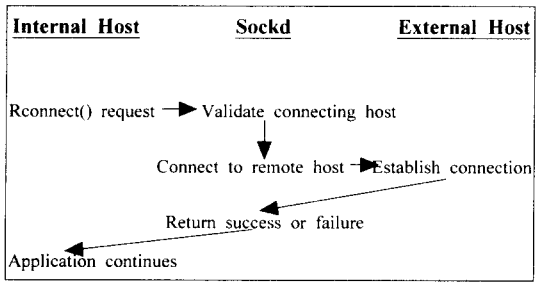

그림 3 CONNECT Request

그림 4에서 보는 바와 같이 BIND request는 CONNECT request보다 복잡한 양상을 띄고 있다. 그러나 같은 원리로 작동됨을 알 수 있다. 이상 CONNECT와 BIND 두 명령어에 의해 클라이언트 시스템 내에 있는 특정 소켓에 대한 모든 READ, WRITE행위가 SOCKS데몬(sockd)이 설치되어있는 방화벽 시스템을 통과할 수 있는 상태가 된다.

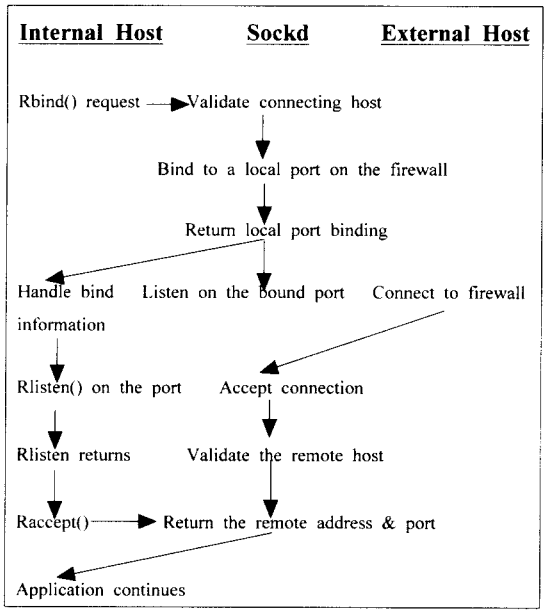

그림 4 BIND Request

한편, SOCKS가 설치되어있는 방화벽 시스템에는 sockd.conf라고 하는 설정 파일이 존재한다. 이 파일은 sockd에 의해 방화벽 시스템에 연결을 시도하는 내부 또는 외부 호스트들을 허락하거나 거부하는데 사용된다.

이 파일에 입력되는 리스트들은 다음과 같은 syntax에 따라 작성된다.

{permit | deny} {source-host} {mask} [{dest-host} {mask} [{operator} {port}]]

다음 그림은 이들을 이용한 샘플 sockd.conf 파일의 내용을 예시한 것이다.

```
# Deny all host to every host whois service
 deny  0.0.0.0  0.0.0.0  0.0.0.0  0.0.0.0  eq
 whois
# Let lloyd.mips.com only use finger service
# to sgi.com
 permit  lloyd.mips.com 255.255.255.255 sgi.com
 255.255.255.255  eq  finger
 deny  lloyd.mips.com 255.255.255.255 sgi.com
 255.255.255.255
# Allow all hosts on the 130.62 network access
# to the world
 permit  130.62.0.0  255.255.0.0
# Deny all hosts which do not match anything
# in this file(i.e. All hosts coming in from
# the Internet)
```

그림 5 A Sample Configuration File

이외에도 SOCKS 데몬(sockd)은 UNIX syslog 인터페이스를 이용하여 다음과 같은 세 종류의 메시지를 기록할 수 있는 logging 기능을 갖고 있다.

- o Access Denied
- o Successful Connection
- o Resource failure, (e.g. Out of File Descriptors, No More Processes)

이들 중에서 첫번째, 두 번째 메시지들이 의미가 있다. 이 메시지들은 연결을 시도했던 호스트 및 사용자 이름 뿐만 아니라 데몬에 요구했던 요청 사항(type of requests, Connect or Bind)등도 포함하고 있다. 이러한 정보들의 기록은 요청(Requests)이 데몬에 요구될 때마다 이루어진다. 이상에서 언급한 SOCKS에 관한 내용은 주로 버전 4.x에 바탕을 둔 것이다.

현재는 SOCKS V5.0이 미국의 NEC연구소를 중심으로 개발되어 있는 상태이다. 버전 5.0에서는 4.x 에서 지원하지 안았던 UDP프로토콜도 지원하고 있다.

그러나 UDP는 보안상의 취약점 때문에 대부분의 경우 방화벽에서 UDP 패킷들은 차단하는 것이 상례이다.

3. SOCKS V5

SOCKS는 원래 TCP 프로토콜을 기반으로 하는 클라이언트-서버 응용 프로그램들을 중간에서 서로 연결해주기 위하여 설계되었다. 그러므로 클라이언트가 서버와 통신하기 위해서는 우선 SOCKS에게 서비스 요청을 하여야 한다. 이 서비스 요청 안에는 연결하고자 하는 서버의 주소, 연결 타입(active or passive), 사용자 id 등이 들어 있어야 한다. 이러한 요청을 받은 SOCKS는 클라이언트를 대신하여 해당 서버와 커뮤니케이션 채널을 설정하게 된다. 이와 같이 프록시 회로(Proxy Circuit)가 설정이 되면 SOCKS는 클라이언트와 서버 사이에서 단순히 데이터를 전달하게된다. 그리고 프록시 회로가 설정되어 있는 동안에 SOCKS는 인증, 메시지 보안 수준 협상, 권한 부여 등의 기능도 수행한다.

SOCKS는 Connection request, Proxy circuit setup, Application data relay 기능들로 구성되어있으며 IETF Network Working Group 에서 RFC1928로 확정된 SOCKS V5[3] 는 인증 기능이 부가되어있다. 다음 그림은 SOCKS 기능들의 연관 관계 및 흐름도를 나타낸 것이다. 점선으로 된 박스 안에 있는 것은 SOCKS V.4[4] 기능을 나타낸 것이다.

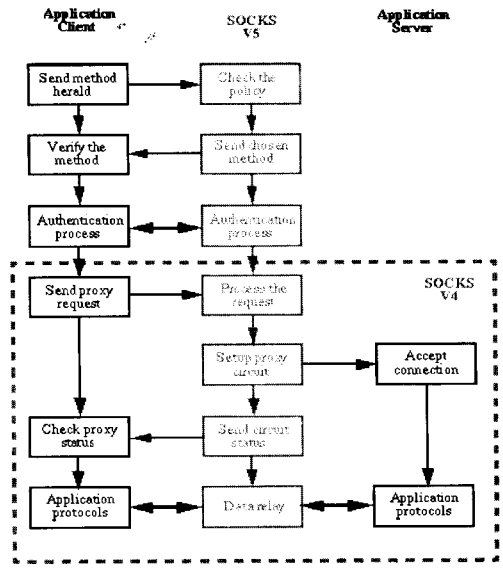

그림6. SOCKS 기능 흐름도

그리고 그림 6 전체는 SOCKS V5 작동 원리 전부를 의미한다. 이 그림에서 SOCKS V5의 인증 기능이 어떻게 SOCKS V4에 부가되어 있는지 파악할 수 있다. SOCKS V4와 비교해 볼 때, SOCKS V5는 다음과 같은 새로운 기능들이 부가되어있다.

첫 번째, 두 개의 메시지를 이용한 인증 방법 협상 및 두 가지 인증 기능을 제공한다. 첫번째 메시지는 클라이언트에 의해 SOCKS 서버로 보내지는데 그 내용은 클라이언트가 지원할 수 있는 인증 방법들로 되어있다. 두 번째 메시지

는 SOCKS 서버가 클라이언트로 보내게 되는데 그 내용은 클라이언트가 사용해야 될 인증 방법이 들어 있다. SOCKS 서버에 의하여 결정된 인증 방법은 SOCKS 서버 설정에 정의된 보안정책에 기초를 둔 것이다. 그러므로 클라이언트가 첫 번째 메시지에서 보낸 인증 방법들이 SOCKS 서버의 보안요구조건을 만족시키는 것이 아닐 경우, SOCKS 서버는 클라이언트와의 통신을 중지하게 된다. 한편, 인증 방법이 성공적으로 결정되면 클라이언트와 SOCKS 서버 사이에서는 인증 작업이 시작되는데 SOCKS V.5 는 Username/Password 인증[5]과 Kerberos 5에 입각한 GSS-API 인증[6] 방법을 이용하여 클라이언트를 인증하게 된다.

두번째, UDP 패킷들을 통과할 수 있게 한다. 기존의 SOCKS V4는 TCP 패킷만을 위하여 프록시 데이터 릴레이 기능만을 했으나 버전 5에서는 UDP패킷도 서비스하고 있다. UDP 패킷을 전달하기 위한 프록시 회로는 기존의 TCP 패킷을 전달하기 위해서 설정되는 TCP 프록시 회로와는 다음과 같은 두 가지 점에서 다르다. 즉, SOCKS 서버에게 보내지는 UDP 패킷은 다음과 같은 구조를 이루고 있다. 즉, UDP를 위한 프록시 회로는 데이터그램을 보내고 받는 클라이언트-서버 양단을 위한 한 쌍의 주소로 되어있다. 그리고 응용 데이터(Application Data)는 해당 데이터그램의 목적지 주소가 포함되어있는 UDP 프록시 헤더로 포장되어있다.

이상으로 SOCKS V.5에 대한 설명을 마치기로 하고 다음 장에서는 SOCKS V.5를 이용한 VPN 구축에 대하여 설명하기로 한다.

Ⅲ. SOCKS VPN 구축

이제부터는 SOCKS V.5를 이용하여 VPN을 구축하는 방법을 설명하기로 한다. 그림 7과 같은 네트워 환경에서 VPN를 구축해보기로 한다. 이때 적용할 보안 정책은 다음과 같다.

즉, 네트워 A에 있는 모든 클라이언트들이 인터넷 B를 통하여 다른 네트워 C에 안전한 연결을 할 수 있도록 한다.

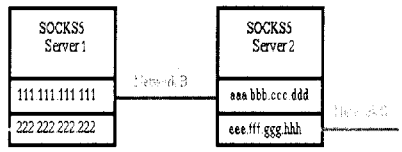

그림 7. VPN 네트워 구성도

그리고 서버1과 서버2 사이 통신이 안전하게 하기 위해서 GSS-API (i.e. kerberos)를 이용하여야 한다. 왜냐하면 GSS-API 가 두 서버 사이에 인증과 데이터 암호화를 가능하게 해주기 때문이다. 이제부터 VPN을 위하여 필요한 조치들을 설명하기로 한다. 우선 네트워 A에 있는 클라이언트들의 해당 파일(/etc/libsocks5.conf)을 설정하기로 한다.

```
noproxy - 222.222.222. - -

socks5 - - - - 222.222.222.222
```

그림 8. Clients on Network A (/etc/libsocks5.conf)

네트워 A 상에 있는 모든 클라이언트들은 SOCKS5 서버 2를 전혀 의식할 필요가 없으며 단지 SOCKS5 서버 1이 SOCKS5 서버 2를 이용할 수 있게 자신의 설정 파일(/etc/socks5.conf)을 다음과 같이 설정해주면 된다.

```
auth - - -

interface 222.222.222. - 222.222.222.222

interface - - 111.111.111.111

SOCKS5 eee.fff.ggg - aaa.bbb.ccc.ddd

noproxy - - -

permit - - 222.222.222. - - -
```

그림 9. SOCKS5 서버 1

SOCKS5 서버 2는 SOCKS5 서버 1에게 GSS-API (i.e. kerberos)를 이용하도록 요구해야 한다. 즉,

SOCKS5 서버 2는 다음과 같이 설정 파일을 작성해야 한다.

```
auth 111.111.111.111 - k

interface eee.fff.ggg. - eee.fff.ggg.hhh

interface - - aaa.bbb.ccc.ddd

permit k - 222.222.222. eee.fff.ggg - -
```

그림 10. SOCKS5 서버 2 (/etc/socks5.conf)

이상의 방법으로 SOCKS V.5를 이용하여 응용계층에서 VPN을 구축할 수 있다.

Ⅳ. 결 론

SOCKS V.5를 이용한 VPN 구축은 기존의 IPSEC 프로토콜과 상호 보완적으로 사용될 수 있다. 특히, 윈도우 계통의 OS는 소스가 공개되어있지 않아 IPSEC을 구현하는 것이 사실상 불가능하다고 할 수 있으며 현재와 같이 주로 소프트웨어적으로 VPN을 구축하는 경우 IPSEC에 의하여 모든 패킷을 포장하는 것은 네트웍 처리 속도의 상당한 저하를 초래할 수 있다.

따라서 꼭 필요한 서비스들만을 SOCKS VPN을 이용하여 처리하는 것이 훨씬 효율적일 때가 있다. 특히, 외부에 출장 나간 일반 사용자들이 자신의 회사 내부에 있는 시스템에 접근할 필요가 생겼을 때 SOCKS VPN이 설치된 방화벽을 자신이 소지하고 있는 노트북에 설치된 SOCKS VPN 클라이언트를 이용하면 내부에 있는 시스템에 안전하게 접근할 수 있게 된다. 즉, 서버 대 서버가 아닌 클라이언트 대 서버 사이에 VPN을 구축할 수 있게 되는 것이다.

한편, SOCKS V.5에 들어있는 GSS-API는 수출용 버전에는 들어있지 않아 국내에서 이를 이용하는 것은 사실 상 불가능하다고 할 수 있다. 따라서 이에 대한 대안으로 Socksified SOCKS 서버인 Rsockd를 지원하는 SOCKS 4.3 버전에 SSL을 연결하여 TCP 프로토콜에 대한 VPN 기능을 구현할 수 있을 것이다.

참 고 문 헌

[1] Asend Communications, Inc. "Virtual Private Networks Resource Guide", pp. 1, 1997.

[2] David Koblas and Michelle R. Koblas. SOCKS

[3] M. Leech, M. Gania, Y. Lee, R. Kurig, D. Koblas, L. Jones, Socks Protocol Version5 IETF Network Working Group RFC 1928, March 1996.

[4] Marcus. Leech, Socks Protocol Version4. IETF Network Working Group INTERNET-DRAFT, Jan, 5 1994.

[5] M. Leech, Username/Password Authentication for SOCKS V5. IETF Network Working Group RFC 1929, March 1996.

[6] P. McMahon, GSS-API Authentication Method for SOCKS Version 5. IETF Network Working Group RFC 1961, June 1996.

著者紹介

백 석 철(Seok-Chul Baek)

1982년 2월 : 서울대학교 물리교육학과 졸업(학사)
1983~1985년 2월 : KAIST 물리학과 석사과정 졸업(석사)
1985~1991년 2월 : 한국통신 연구개발본부 전임연구원
1991~1995년 8월: KAIST 물리학과 박사과정 졸업(박사)
1995~1998년 7월: 한국통신 멀티미디어연구소 인터넷 보안연구실장
1999년 현재: 시큐어소프트 보안연구소 소장, 리눅스 시큐리티랩 대표이사, 숭실대 정보과학대학원 겸임 교수
<관심분야> 정보보호(PKI, Firewall, IDS), 정보처리