

A tourist's guide to Socket Programmin

Ross Lonstein

June 24, 2002

– Typeset by Foil_ETeX –

What's so interesting about a HTTP Proxy and

What's so interesting about a HTTP Proxy and

- Those fascists are keeping me down!

What's so interesting about a HTTP Proxy and

- Only way out to Internet
- Well defined behavior
- Easy to hide traffic

What's so interesting about a HTTP Proxy and

Other solutions (a.k.a STFW)

- SOCKS
- Randal's tunnel (and others)
- GNU HTTP-Tunnel

Except that none of them work behind an Authenticating

Rolling Your Own

- Something about Sockets
- A little on TCP/IP
- An understanding of HTTP
- Insight into Proxy behavior
- Smattering of Authentication

C Sockets from 60,000 feet

- Introduced in BSD 4.1c circa 1983.
- Rewritten as a generalized “Inter-Process Communication” (IPC) in BSD 4.3 to support varied types of IPC.
- Behaves much like a file descriptor.

C Sockets from 30,000 feet

Socket refers both to the Sockets API and a communication end

- stream (SOCK_STREAM)
- datagram (SOCK_DGRAM)

Domain Communication Domain

- Unix Domain (AF_UNIX)
- Internet Domain (AF_INET)

Server offers its services and “listens” for a request.

Client requests services from the server by initiating a “con
server.

C Sockets from 15,000 feet

- Comparison of OSI and IP stacks.

C Sockets from 7,500 feet

- Basics of IP Addressing

C Sockets from 7,500 feet

```
# from: @(#)services 5.8 (Berkeley) 5/9/91
tcpmux 1/tcp # TCP port service multiplexer
echo 7/tcp
echo 7/udp
discard 9/tcp sink null
discard 9/udp sink null
sysstat 11/tcp users
...
ftp-data 20/tcp # default ftp data port
ftp 21/tcp
ssh 22/tcp
ssh 22/udp
telnet 23/tcp
```

C Sockets from Ground Zero (The Functions)

Lookup	Byte Order	Socket	I/O
<i>getservbyname()</i>	<i>htonl()</i>	<i>bind()</i>	<i>send()</i>
<i>gethostbyname()</i>	<i>ntohl()</i>	<i>connect()</i>	<i>recv()</i>
<i>getservbyaddr()</i>	<i>htons()</i>	<i>listen()</i>	<i>select()</i>
<i>gethostbyaddr()</i>	<i>ntohs()</i>	<i>accept()</i>	
	<i>inet_ntoa()</i>	<i>close()</i>	
	<i>inet_pton()</i>	<i>shutdown()</i>	

C Sockets from Ground Zero (The Structu

```
struct sock_addr {  
    u_short sa_family;  
    char sa_data[14];  
};
```

```
struct in_addr {  
    union {  
10        struct {  
            u_char s_b1, s_b2, s_b3, s_b4;  
        } S_un_b;  
        struct {  
            u_short s_w1, s_w2} S_un_w;  
            u_long S_addr;  
        } S_un;  
    # define s_addr S_un.S_addr  
};
```

20

```
struct sockaddr_in {  
    short sin_family;  
    u_short sin_port;  
    struct in_addr sin_addr;  
    char sin_zero[8];  
  
};
```

```
30 struct servent {  
    char *s_name;  
    char **s_aliases;  
    int s_port;  
    char *s_proto;  
  
};
```

```
    struct hostent {  
        char *h_name;  
40    char **h_aliases;  
        int h_addrtype;  
        int h_length;  
        char **h_addr_list;  
};
```

Simple Socket Client in C (The Hard Wa

```
#include "stdlib.h"
#include "stdio.h"
#include "string.h"
#include <sys/socket.h>
#include <netinet/in.h>
```

```
int main(int argc, char **argv)
{
```

10

```
    const int MAXLINE = 262144;
```

```
    int sockfd, n;
```

```
    char cmd[] = "GET / HTML/1.0\n\n";
```

```
    char recvline[MAXLINE + 1];
```

```
    struct sockaddr_in servaddr;
```

```
    if (argc != 2) {
```

```

        error("must supply IP address!");
20     }

sockfd = socket(AF_INET, SOCK_STREAM, 0);

if (sockfd < 0) {
    error("socket error!");
}

bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
30 servaddr.sin_port = htons(80);
if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr)
    0) {
    printf("inet_pton error for %s", argv[1]);
    exit(EXIT_FAILURE);
}

if (connect

```

```

        (sockfd, (struct sockaddr *) &servaddr,
         sizeof(servaddr)) < 0) {
40     error("connect error!");
    }

    (void) write(sockfd, cmd, sizeof(cmd));

    while ((n = read(sockfd, recvline, MAXLINE)) > 0) {
        recvline[n] = (char) 0x00;
        if (fputs(recvline, stdout) == EOF) {
            error("fputs error!");
50         }
    }

    if (n < 0) {
        error("read error!");
    }

```

```
        exit(EXIT_SUCCESS);
    }
60
void error(char *msg[])
{
    printf("ERROR: %s\n", msg);
    exit(EXIT_FAILURE);
}
```

Simple Socket Client in Perl

```
#!/usr/bin/perl

use Socket;

5 my $peer = sockaddr_in(80,inet_aton('127.0.0.1'));
  socket( S, PF_INET, SOCK_STREAM, getprotobyname('tcp') );
  connect( S, $peer ) or die "Couldn't connect!";
  select(S);
  $|=1;
10 print S "GET / HTML/1.0\n\n";
  select STDOUT;
  print <S>;
  close(S);
```

Simple Socket Client in Perl- Using IO::Socket

```
#!/usr/bin/perl
# IO::Socket::INET client #1

use IO::Socket::INET;

5
my $socket = IO::Socket::INET->new( PeerAddr => '127.0.0.1',
                                     PeerPort => '80',
                                     Proto    => 'tcp',
                                     Type     => SOCK_STREAM )
10     or die "Couldn't connect! $!";

print $socket "GET / HTML/1.0\n\n";
print <$socket>;
close($socket);
```

Simple Socket Client in Perl- Using IO::Socket

```
#!/usr/bin/perl
# IO::Socket::INET client #2

use IO::Socket::INET;

5
my $socket = IO::Socket::INET->new('127.0.0.1:80') or d
print $socket "GET / HTML/1.0\n\n";
print <$socket>;
close($socket);
```

Simple Server in Perl- Using IO::Socket::INET

```
#!/usr/bin/perl

use IO::Socket::INET;

5 my $s = IO::Socket::INET->new( LocalPort => 2222,
                                Type       => SOCK_STREAM,
                                Reuse     => 1,
                                Listen    => SOMAXCON )
    or die "$!";
10
   while ($c=$s->accept()) {
       print $c "Hello, World!\n";
       #sleep 100;
       close($c);
15 }
   close($s);
```

“One Shot” Port Forwarder

```
#!/usr/bin/perl

use IO::Socket::INET;
use IO::Select;
5 $| = 1;

my $s = IO::Socket::INET->new(
    LocalPort => 2222,
    Type      => SOCK_STREAM,
10  Reuse     => 1,
    Listen    => SOMAXCON
)
or die "$!";

15 while ( $c = $s->accept() ){
    my $r = IO::Socket::INET->new(
        PeerAddr => '127.0.0.1',
        PeerPort => '80',
```

```

        Proto    => 'tcp',
20      Type      => SOCK_STREAM
    )
    or die "Couldn't connect! $!";

my ( $bytes, $fh );
25 my $select = IO::Select->new( $c, $r );
LOOP: while (1) {
    foreach $fh ( $select->can_read(10) ){
        last LOOP unless ( defined( $bytes = sysread
            , 4096 ));
        last LOOP if ( $bytes == 0 );
30 last LOOP
        unless (
            defined(
                syswrite(
                    ( ( fileno($fh) == fileno($c) )?
                    , $bytes
35                )
            )

```

```

)
);
}
}
40  close($r);
     close($c);

}
close($s);
```

Forking Port Forwarder

```
#!/usr/bin/perl

use IO::Socket::INET;
$| = 1;

5
my $s = IO::Socket::INET->new(
    LocalPort => 2222,
    Type      => SOCK_STREAM,
    Reuse     => 1,
10    Listen  => SOMAXCON
)
or die "$!";

while ( $c = $s->accept() ){
15    my $r = IO::Socket::INET->new(
        PeerAddr => '127.0.0.1',
        PeerPort => '80',
        Proto    => 'tcp',
```

```

        Type    => SOCK_STREAM
20    )
        or die "Couldn't connect! $!";

my $child = fork();
if ($child) {
25    while (<$c>) { print $r $_ }
    }
    else {
        close($s); # server socket not needed by child
        while (<$r>) { print $c $_; }
30    }
    $c->shutdown(0); $r->shutdown(0);
    close($c); close($r);
}
close($s);

```

Welcome to the Application Layer!

- So what is a “Proxy”?
- How does that differ from a “port forwarder”?

HTTP & Authentication: It's all in my head

```
$ nc orwell.bfcorp.com 8080
GET www.google.com HTTP/1.0
<cr>
<cr>
```

HTTP & Authentication: It's all in my head

```
$ nc orwell.bfcorp.com 8080
GET www.google.com HTTP/1.0
<cr>
<cr>
HTTP/1.0 407 Proxy authorization required
Proxy-agent: Netscape-Proxy/3.52
Date: Thu, 15 Mar 2001 13:00:06 GMT
Proxy-authenticate: basic realm="unspecified"
Content-type: text/html Content-length: 271
```

Authentication Schemes

RFC 1945

“Basic“

Userid/Password

Realm

Base64 Encoding

RFC 2069

“Digest“

Userid/Password

Realm

Nonce

Opaque

Domain

MD5

Base64 Encoding

Microsoft

“NTLM“

Userid/Password

Domain

Nonce

“5-way handshake“

SMB Hash (RC4)

Closer Look at Basic Authentication

- Consists of User ID and Password concatenated with a Realm and Base64 encoded.
- Realm is an identifier for the particular authority of the URI.
- Very easy but insecure.

Closer Look at Digest Authentication

- Attempt at security– User ID and Password concatenated with nonce and realm but run through the MD5 hash function before Base64 encoding. The digest is MD5(userid:password:nonce:opaque)).
- Server optionally generates a unique string, the “nonce”, which the client must include in reply.
- Server optionally includes an “opaque”, a short string of data which the client must also include in reply.
- Server optionally includes a “domain”, a list of URIs to which the authentication applies.

Tunnel-Auth

```
#!/usr/bin/perl

my $VERSION=0.03;
use IO::Socket::INET;
5 use IO::Select;
use Getopt::Std;
use POSIX qw(:sys_wait_h);
use strict;
use diagnostics;
10
if ( $^O =~ /win32/i ){
    require 5.6.0;
}

15 my ( $dport,    $dhost,    $proxyhost, $proxyport );
    my ( $remotehost, $remoteport, $auth, $useragent );
```

```

my ( $fhin,      $fhout,      $server,  $proxy, $child );

if ( eval { 'use Digest::MD5;' } ){
20   use Digest::MD5;
      print STDERR "Using MD5.\n";
    }
my $md5avail = !$@;

25 my %opts;
    getopt( 'a:l:p:r:u:', \%opts );
    usage('Missing remote host:port') unless $opts{'r'};
    usage('Missing local host:port') unless $opts{'l'};
    usage('Missing proxy host:port') unless $opts{'p'};
30
    ( $proxyhost, $proxyport )= split ( ':', $opts{'p'} );
    ( $remotehost, $remoteport )= split ( ':', $opts{'r'} )

    if ( $opts{'l'} =~ /:/ ){
35   ( $dhost, $dport )= split ( ':', $opts{'l'} );

```

```

    }
    else {
        $dport = $opts{'l'};
        $dhost = '127.0.0.1';
40 }

    $auth = $opts{'a'};

    $useragent = $opts{'u'} || 'Mozilla/4.0 (compatible; MSIE
        Windows NT)';
45
    $server = IO::Socket::INET->new(
        Proto    => 'tcp',
        LocalAddr => $dhost,
        LocalPort => $dport,
50 Listen     => SOMAXCONN,
        Type      => SOCK_STREAM,
        Reuse     => 1
    )

```

```

    || die "Error creating daemon socket: $!";
55
my %socket_hash = (
    PeerAddr => $proxyhost,
    PeerPort => $proxyport,
    Proto    => 'tcp',
60    Type     => SOCK_STREAM
    );

while ( $fhin = $server->accept() ){
    next if $child = fork();
65    die "fork failed: $!" unless defined $child;
    $fhout = $fhin;

    $proxy = getPeerSocket( \%socket_hash );

70    foreach ( \*$proxy, $fhin, $fhout ){
        select($_);
        $| = 1;

```

```

    }

75  print $proxy "CONNECT $remotehost:$remoteport HTTP/1.1\r\n"
    print $proxy "User-Agent: $useragent\r\n" if $useragent
    print $proxy "\r\n";

    my $status;
80  ($status) = ( split ( /\s+/, <$proxy> ) )[1];

    if ( $status == 407 && $auth ){

        $_ = <$proxy> while ( $_ = !/Proxy-authenticate:
85  print STDERR "Proxy authentication required...";

        $proxy->close() || die ("Error closing proxy connection");
        ;
        print STDERR "Closed proxy.\n Reconnecting...";
        $proxy = getPeerSocket( \%socket_hash );
90  print $proxy "CONNECT $remotehost:$remoteport HTTP/1.1\r\n"

```

```

print $proxy "User-Agent: $useragent\r\n" if $us

CASE: {
    auth_basic(), last CASE if $1 =~ /basic/i;
95     auth_digest(), last CASE if $1 =~ /digest/i

    auth_unsupt(), last CASE;
}

100     print $proxy "\r\n";

    ($status) = ( split ( /\s+/, <$proxy> )) [1];
}

105 die "Bad status code \"$status\" from proxy server."
    if ( int( $status / 100 ) != 2 );

1 until ( <$proxy> =~ /^[\r\n]+$/ );

```

```

110  shuffle_packets(
        {
            input => $fhin,
            output => $fhout,
            proxy => $proxy,
115      }
    );
    $proxy->close() || die ("Error closing proxy socket");
    $server->close() || die ("Error closing server socket");
    exit;
120 }
    continue {
        close $fhin or die ("Error in parent closing accept
            !");
    }

125 sub REAP {
    1 until ( -1 == waitpid( -1, WNOHANG ) );
    $SIG{CHLD} = \&REAP;

```

```

    }

130 sub shuffle_packets {
    my $href = shift || die ("Incomplete parameters: $

    my $in  = $$href{'input'};
    my $out = $$href{'output'};
135 my $proxy = $$href{'proxy'};

    my $s = IO::Select->new( $in, \*$proxy );
    my $num;
    LOOP: for ( ; ; ){
140     foreach my $fh ( $s->can_read(10) ){
        last LOOP unless ( defined( $num = sysrea
            , 4096 )));
        last LOOP if $num == 0;
        last LOOP
            unless (
145             defined(

```

```

                                syswrite(
                                    ( ( fileno($fh) == fileno($in) )
                                        $out ),
                                    $_,
                                    $num
150                                )
                                )
                                );
                                }
                                }
155 }

```

```

sub auth_basic {
    print STDERR "Using BASIC authentication.\n";
    print $proxy "Proxy-Authorization: Basic ", encode_base64($pass)
        , "\r\n";
160 }

```

```

sub auth_digest {

```

```

print STDERR "Using DIGEST authentication.\n";

165  $_ = <$proxy> while ( $_ =~ /digest/i );
my ($challenge) = ( split ( ':', $_ ))[1];

my ( $user, $pass )= split ( ':', $auth );
my $response = md5_base64("$user:$pass:$challenge");
170  print $proxy "Proxy-Authorization:$response\r\n";
    }

sub auth_unsupt {
    print STDERR "Unsupported authentication type '$1' :
175 }

sub getPeerSocket {
    my $hr = shift || die ("No parameters: $!");
    my $socket = IO::Socket::INET->new(
180     PeerAddr => $$hr{'PeerAddr'},
        PeerPort => $$hr{'PeerPort'},

```

```

        Proto => $$hr{'Proto'} || 'tcp',
        Type  => $$hr{'Type'}  || SOCK_STREAM
    )
185     || die "Socket setup failed: $!";
    return $socket;
}

sub encode_base64 {
190     use integer;
    my $res = "";
    pos( $_[0] )= 0;
    while ( $_[0] =~ /(.{1,45})/gs ){
        $res .= substr( pack( 'u', $1 ), 1 );
195     chop($res);
    }
    $res =~ tr |` -_|AA-Za-z0-9+//|;
    my $padding = ( 3 - length( $_[0] )% 3 )% 3;
    $res =~ s/.{ $padding}$/'=' x $padding/e if $padding;
200     return $res;
}

```

```

}

sub usage {
    print "\n!! ", @_, " !!\n";
205    print <<USAGE;

Usage $0 -p <proxy>:<port> -l [<local>:]<port> -r <remote>
    <proxyid:password>] [-u <user-agent>]
Connect to a remote host through a proxy supporting CONNECT.
    <proxy>:<port>      -- ip or hostname and port of your proxy
210    <local>:<port>     -- port to listen on. ip/hostname
    <remote>:<port>    -- remote host and port (likely port 80)
    <proxyid>:<password> -- userid/password for authentication
    <user-agent>      -- header string send to proxy. Default is
                        'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT)'
```

215 Example:

```

$0 -p proxy.example.com:8080 -l 2222 -r myhost.nowhere.com
    -a joe:h4ckm3 -u 'Mozilla/4.76 [en] (WinNT; U)'
```

```
        USAGE
220      exit 1;
      }
```

Required Reading

The texts:

Unix Network Programming 2nd Ed., W. Richard Stevens, 1998.

TCP/IP Illustrated Vol. 1 & 2, W. Richard Stevens, Addison 1995.

The Design and Implementation of the 4.4 BSD Operating System, Marshall Kirk McKusick, Addison-Wesley, 1996.

Programming Perl 3rd Ed., L. Wall, T. Christiansen & J. O'Keefe, Addison-Wesley Assoc., 2000.

Perl Cookbook, T. Christiansen & N. Torkington, O'Reilly Assoc., 1998.

Required Reading

Papers & Articles:

An Advanced 4.3BSD Interprocess Communication Tutorial,
Joy, et. al.

Beej's Guide to Network Programming, Brian "Beej" Hall.

BSD Sockets: A Quick & Dirty Primer, J. Frost.

Required Reading

The RFCs:

1945 Hypertext Transfer Protocol – HTTP/1.0

2068 Hypertext Transfer Protocol – HTTP/1.1

2069 An Extension to HTTP : Digest Access Authentication

2616 Hypertext Transfer Protocol – HTTP/1.1

2617 HTTP Authentication: Basic and Digest Access Authen