Global Information Assurance Certification Paper

SOHO Remote Access VPN. Easy as Pie, Raspberry Pi…

GIAC (GSEC) Gold Certification

Author: Eric Jodoin, ejodoin@hotmail.com
Advisor: David Shinberg

Accepted: Nov 24th 2013

Abstract

Free, unencrypted wireless access points have proliferated and are now found in various locations such as restaurants, libraries, schools, hotels, airports, etc. In conjunction with the spread of mobile devices, "getting on the Internet" has quickly migrated from the relative confines of offices and homes to locations where a high number of users are sharing the same open access point. While convenient, this situation also increases the likelihood that someone will have the skills and intent to pry into our privacy and collect our Internet traffic for nefarious purposes.

Security professional are always preaching restraint when it comes to using free/public wireless access points. But, when asked for an alternative other than wait to go back home, there is little to offer for small office/home office (SOHO) users that is practical while also remaining free or at least, almost free. This gold paper presents such an alternative in the form of a VPN solution and associated configuration instructions for both server and clients. It leverages the rather inexpensive Raspberry Pi (RPi) hardware in conjunction with the freely distributed OpenVPN Server and client software. The solution requires minimal changes to an existing SOHO network and is simple to configure on a broad variety of mobile platforms including Windows, Linux, OS X, iOS, and Android.

1. Introduction

Free, unencrypted Wireless Access Points (WAPs) have proliferated and are now found in various locations including restaurants, libraries, schools, hotels, airports, etc. In conjunction with the spread of mobile devices, "getting on the Internet" has quickly migrated from the relative confines of offices and homes to locations where a high number of users are sharing the same open access point. Mobile devices connect to numerous WAPs throughout the course of a single day, sharing access with a comparatively high number of unknown and potentially malicious users. This raises the likelihood that someone will have the skills and intent to pry into private communications and collect traffic for nefarious if not outright illegal intent. While more web sites use encryption, there remains a fair number of web and other Internet services that do not offer this protection. In addition, even if the traffic is encrypted, a significant amount of information can be inferred by simply observing traffic and collecting metadata such as the name of sites visited and the amount of encrypted traffic exchanged.

First, this paper presumes a certain level of trust in Internet Service Providers (ISPs) that is routed in the supposition that they comply with their respective country's laws regarding the illegal interception of private communications. Secure in the presumption that ISP's can be trusted, this paper seeks to address protection from "privately" managed WAPs accessed while traveling and the potentially malicious users gravitating around them. Specifically, when traveling to locations and/or jurisdictions where the technical and/or regulatory protection afforded to private communications fall below what you have come to expect from your own ISP.

An obvious solution could be to avoid public networks altogether. If Internet access is required, perhaps a mobile phone with unlimited data plan could suffice. Additional devices could also be tethered to the Internet via that mobile phone. However, this approach suffers from certain limitations. First, there is the issue of signal strength, which may be inadequate. Then, the phone's battery will eventually become an issue, especially if the signal strength is low and power outlets are unavailable. Furthermore, this solution can be expensive in some locations and countries, especially while roaming. Finally, if roaming, especially in a foreign country, the mobile service provider may not afford the same privacy as mobile service providers at home. These limitations can be frustrating to a mobile user when considering that free Internet access is within its reach.

Eric Jodoin, ejodoin@hotmail.com

Other solutions could include the purchase of Virtual Private Network (VPN) or Virtual Private Server (VPS) services. Many VPN service providers offer solutions for a few dollars a month. Some, like vpnbook.com are completely free. Alternatively, a VPS can be rented and configured to run any application, including a VPN server such as Openswan or OpenVPN. However, there will also be a recurring monthly fee incurred. These solutions have the same problem than WAPs regarding the expected level of trust and privacy afforded by the VPN/VPS service provider, especially if their service is located in a different jurisdiction. Take vpnbook.com as an example. Members of Anonymous have been accusing it of sharing metadata with various security services in different countries (Arnote).

Finally, The Onion Routing (TOR) network is another free alternative. Used worldwide by a variety of individuals and organizations, it claims to offer both anonymity and privacy (The Tor Project, Inc.). TOR relies on a worldwide network of entry and exit nodes with multiple layers of encryptions to protect the user's privacy. However, TOR can suffer from speed and performance issues making it difficult to use for long durations (Dingledin & Murdoch). Additionally, TOR exit nodes have been repeatedly used to eavesdrop on communications (Zetter).

The best alternative is to encrypt all data when connected via a free/public WAP and tunnelling all traffic back to a trusted location like a Small Office/Home Office (SOHO) network where it can be further routed to the Internet. Moreover, this will also introduce two additional side benefits. First, sites visited will incorrectly geo-locate the mobile client "at home", providing an additional layer of privacy. Second, it provides an easy reach back into the mobile client's SOHO network resources such as file servers.

There are many VPN solutions currently available on the market that can meet both privacy and reach back requirements. However, the challenge is finding a solution that is economical, easy to configure, simple to use, and compatible with as broad a range of mobile devices. Below is a summary of available VPN solutions separated into six broad categories then considered against three straightforward requirements – cost, simplicity, and compatibility:

1. **Off the Shelf VPN Routers**: By far the easiest solution to deploy for a system administrator. Numerous offerings are available such as the Cisco RV180 or the Linksys WRV54G. However, this is a costly approach. Routers with VPN capabilities are expensive and currently retail online for $150 or more.

2. **Custom Firmware VPN Router**: Some routers can be flashed with custom firmware such as OpenWrt, which natively support VPNs. Some custom firmwares even support OpenVPN such as DD-WRT and Tomato. Unfortunately, not all existing routers are supported which would mean the need to purchase a new router. In that particular case, an off the shelf VPN router would make more sense. However, even if the router on hand is supported using custom firmware will mean flashing the router, which will instantly void the warranty. This also adds a level of complexity to the operation and support of the VPN because custom firmware is experimental in nature. This may make it more difficult to discern the root cause of any problem experienced while trying to establish a VPN tunnel. Finally, the system administrator is at the mercy of code maintained by volunteers who may not get around to patching the firmware to address a security flaw in a timely fashion.

3. **Windows Server**: Windows Server 2012 Standard edition includes IPSec VPN. Windows Home Server (WHS 2.0), a paired down version of Windows server 2003 is also capable of providing IPSec VPN albeit requiring tweaks not supported by Microsoft. Prices range from $800 for Windows Server 2012 down to $100 for WHS 2.0. In addition, a pricey, power hungry PC is required. This may be a good solution to consolidate multiple services along with VPN such as backups, file sharing, and web server. However, it is overkill and over complicated if all that is required is a VPN Server.

4. **SSH Client/Server**: This is the most simplistic approach when it comes to establishing a secure tunnel and it comes pre-configured on Raspberry Pi (RPi). However, every device requires a different app and/or configuration changes unique to its OS's (Android, iOS, Windows, etc.). Because configuration of clients is complicated and problems while traveling are likely, this approach is impractical for all but the savviest mobile users.

5. **Open Source VPNs (openswan/strongswan)**: There are numerous free implementation of IPSec for VPN tunnelling that can run on minimal hardware like a RPi and therefore make viable candidates. However, a search on the Internet quickly highlights flaws in the implementation of client software spanning numerous OS's which forces configuration changes and/or tweaks on both server and clients. This significantly raises the level of complexity and risks of malfunctions. A month long attempt at configuring openswan L2TP/IPSec to work simultaneously with Windows 7, Android and iOS was unsuccessful.

Because some configuration options were required to allow Windows 7 to connect, it consequently prevented Android and iOS devices from connecting, and vice versa.

6. **OpenVPN**: This is the other free offering supported by the RPi. James Yonan, the creator of OpenVPN, indicated in an interview back in 2003 that it was his answer to divergent IPsec implementations that suffered from robustness and usability deficiencies (Dunston). In essence, it was his answer to the problems of IPSec VPNs discussed in the previous paragraph and, which continue to frustrate even a decade later. Although this solution requires the download of both server and client software, setup is straightforward. Once the server is configured, mobile client users only have to search for the term "OpenVPN client" in their respective App store, download, install, and load the client configuration file provided via secure means by the system administrator. Then, the OpenVPN client takes care of the rest.

	Cost	Compatibility	Simplicity of Configuration	
			Server	Clients
Off the Shelf VPN Router				
Custom Firmware VPN Router				
Windows Server				
SSH Client/Server				
Openswan VPN on Linux				
OpenVPN				

Table 1 – VPN Solution Comparison

This paper presents the principles supporting the proposed OpenVPN solution. You may be tempted to jump straight into the step-by-step instructions in Appendix A. However, to take full advantage of this solution, it is highly recommended you read the paper at least once to understand the concepts supporting the proposed approach. Doing so will make the deployment, and most especially any troubleshooting, a lot easier.

2. Building a Raspberry Pi VPN Server using OpenVPN

Figure 1 represents a typical Remote-Access VPN configuration where one or more client devices connect back to a private (aka SOHO) network. Once authenticated, the VPN server and the client establish an encrypted tunnel that will securely carry all traffic through the Internet back to the trusted network where it will be routed to its final destination in the same manner as if the client had been connected directly into the SOHO network itself. The diagram below will be useful throughout this paper to visualize the various components of the proposed RPi OpenVPN architecture and how they interact with each other.

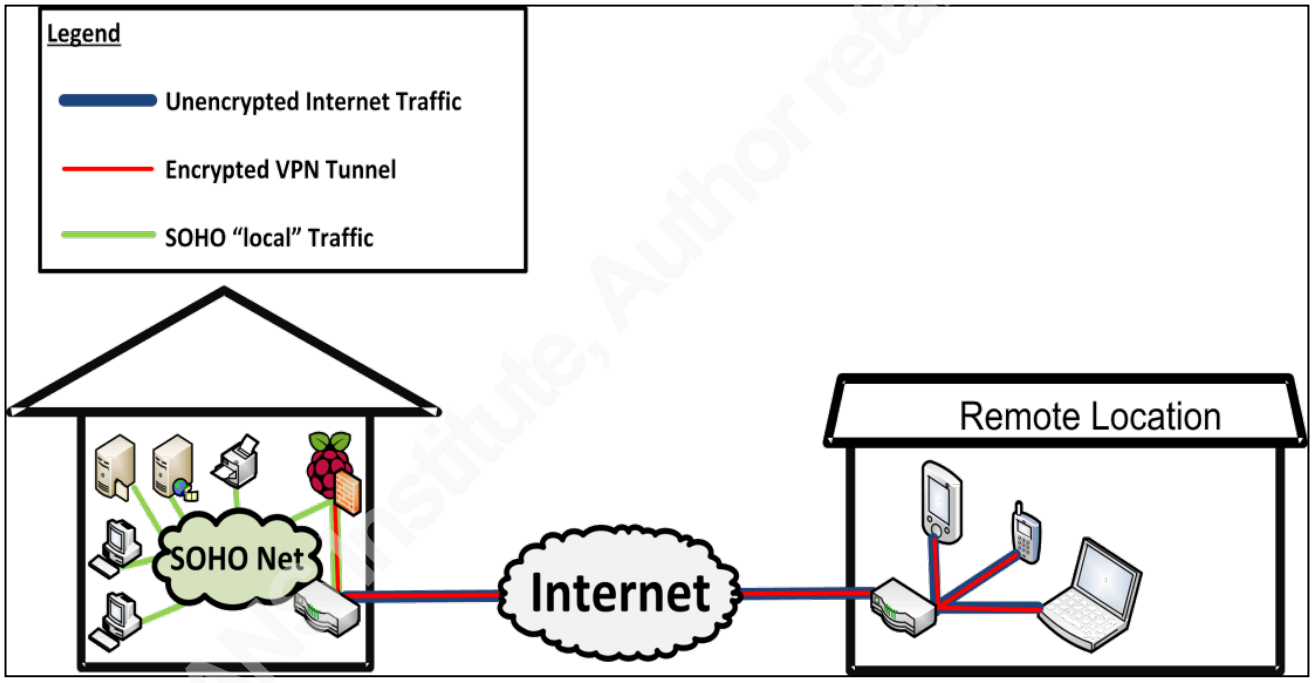

Figure 1- Raspberry Pi VPN Architecture

2.1. OpenVPN Features

OpenVPN is an open source VPN solution and the brainchild of James Yonan. It came about when Yonan was looking for a VPN solution to meet his privacy requirements while traveling through Asia in the days prior to 9/11. Unsatisfied with the VPN implementations available at the time, he set out to develop a solution that would rival IPsec's security but without some of its complexity (Dunston). OpenVPN became a highly customizable solution, supporting a broad range of options and capabilities, the five most relevant described below.

2.1.1. Privacy through authentication and encryption

Because OpenVPN was created for privacy, authentication and encryption, this is a very comprehensive topic. In fact, it is the subject of an entire gold paper written by Charlie Hosner titled "*OpenVPN and the SSL VPN Revolution*". This paper was used extensively to summarise key OpenVPN SSL/TLS concepts presented in this section and which are required to securely deploy OpenVPN.

OpenVPN offers two encryption modes: Pre-shared Key and TLS. Once the OpenVPN tunnel is established, both methods leverage the OpenSSL library to offer the same cryptologic protection. Therefore, the main differences between the two modes are in the use of the symmetric key and the way clients and server authenticate to establish that secure tunnel.

The Pre-shared key method is the easiest and quickest way to setup an OpenVPN Server but it is not very secure. For starters, a single set of static keys is generated and used by all devices to authenticate, establish, and encrypt the secure VPN tunnel. If the keys are compromised, either during the transfer or recovered from a mobile device, all traffic (past and present) is assumed to be disclosed and new keys must be generated and securely re-distributed to every single client (Feilner & Graf). Furthermore, because every client has the same set of keys, there is no way to differentiate and track login activity from individual clients or devices. Finally, only a single client can connect at once, making the pre-shared key method impractical for most environments (openvpn.net).

The TLS method leverages the SSL/TLS protocol with private/public key pairs to authenticate then establish a VPN tunnel using a process called TLS handshake. Client and servers exchange their respective certificates, which contain the public key, and validate it through a Certification Authority (CA) in the standard X.509 Public Key Infrastructure (PKI) approach. Once both sides are satisfied that their counterpart's certificate is valid, they exchange the data required to create an ephemeral symmetric key that will be used to encrypt the VPN Tunnel for that specific session (Dierks & Rescorla).

CA's provide a third party trusted service whose primary purpose is to digitally sign and publish public keys. Some large organizations have the resources to deploy their own enterprise CA infrastructure but most rely on a few commercial service providers. A detailed discussion of the PKI infrastructure is beyond the scope of this paper, but suffice to say that CAs are used to prove that hosts trying to connect really are whom they says they are when they exchange their certificates. This is particularly important when two unrelated hosts are trying to establish a secure connection while

preventing a MitM attack. However, because of the infrastructure and tight security required to process and secure the certificate information, this service can cost several hundreds of dollars a year. For a detailed description of PKI, refer to Chapter 2 of "*Beginning OpenVPN 2.0.9 Build and Integrate Virtual Private Networks Using OpenVPN*" by Markus Feilner & Norbert Graf.

A variation to CA signed PKI certificates are self-signed certificates. In this instance, you take on the role of the CA. Using a highly trusted host you control, such as your RPi OpenVPN Server, you generate and sign all certificates. These certificates will not be trusted anywhere else on the Internet. However, since you control the entire certificate creation process, including the CA's private keys used to sign the client certificate they are suitable for this purpose. In addition, the certificates are only used to connect back to your own OpenVPN Server. Thus, only the clients to whom you have distributed key pairs will have the private and signed public keys required to connect to the server, making the validation through a commercial CA superfluous. However, you must use the option "--ns-cert-type server" in the OpenVPN configuration file to prevent a MitM attack (Yonan).

Regardless of a certificate being CA signed or self-signed, the private key always remains the single most important piece of information stored on a mobile client and must be safeguarded accordingly. Although protected by a password, do not rely solely on this feature to protect the private key and by extension, access to the VPN. Programs such as Phrasen|Drescher[1] can brute force the password protecting the private key. The permanent solution to mitigate a lost private key, which should be carried out soonest after discovery of potential compromise, is to revoke the certificate, as it will prevent any unauthorized user from accessing the VPN via the stolen key pair. This service is provided by all CA's. As explained at Annex D, it can also be enabled with self-signed certificates using the option "--crl-verify" (Burgsma).

In conclusion, TLS Mode is the only recommended method to consider as it provides a significantly greater level of security. And, unless third party authentication is necessary, self-signed certificates will afford adequate security provided the practices as described above are followed. Therefore, this paper will present how to configure an RPi using self-signed certificates. However, if the CA signed certificate approach is required, the information and instructions included in this paper still apply. The only configuration difference between the CA signed and self-signed certificates is the means by which the certificates are generated.

[1] http://www.leidecker.info/projects/phrasendrescher/

2.1.2. Additional Authentication and Protection against DoS and OpenVPN 0-day vulnerabilities

OpenVPN includes a form of Denial of Service (DoS) and buffer overflow protection through a creative use of Hash-based Message Authentication Code (HMAC). Under normal circumstances, HMAC key is generated during the TLS Handshake then used to validate data integrity and authenticate incoming encrypted data packets before they are processed by the decryption routine. Any packet with an incorrect HMAC hash code is expeditiously dropped, saving space in stacks and CPU cycles. In addition, it prevents maliciously crafted packets from reaching and exploiting a buffer overflow vulnerability in the OpenVPN or OpenSSL libraries.

The "traditional" HMAC works well to protect an established TLS connection but what about protecting the connection initiation and TLS handshake itself? Couldn't the server falter because it is overwhelmed by a flood of TLS Handshake initiations, even if they would eventually be dropped because the authentication failed? What about maliciously crafted packets sent during the TLS Handshake? Couldn't they compromise the OpenVPN or OpenSSL libraries? The answer to both hypothetical scenarios is yes. But, because the TLS protocol was designed to establish a secure connection between two unknown parties, there is nothing to protect against these kinds of threat.

This is where OpenVPN implements its own variant of HMAC protection to defend against DoS and Buffer Overflow attempts from the very first packet sent to initiate the connection. In this case, OpenVPN uses a static pre-shared HMAC Key generated during the OpenVPN Server configuration. The OpenVPN documentation refers to this feature as "tls-auth". However, to avoid confusion with the HMAC that is part of TLS, this paper will refer to this pre-computed key as the OpenVPN-HMAC Key.

The OpenVPN-HMAC Key provides an additional layer of security that addresses both hypothetical scenarios because a server would immediately drop any packet lacking the authentication code computed from the pre-shared OpenVPN-HMAC Key. Furthermore, only clients with the pre-shared OpenVPN-HMAC Key would be able to exchange certificates. Therefore, any attempts at buffer overflow through malicious packet injection or MitM using fake certificates would be defeated as the attacker would be unable to compute a valid authentication code without the OpenVPN-HMAC Key.

The OpenVPN-HMAC keys are manually generated during the server configuration and must be enabled using the "tls-auth" option. It is then included in the OpenVPN configuration file alongside the client public/private key pair and distributed securely to the client device.

While the OpenVPN-HMAC Key concept conveniently introduces an additional layer of protection, it comes at a price. This key can only be pre-shared and is identical for every client. Therefore, if it is compromised, and even if using a commercial CA, all mobile clients will require a new OpenVPN-HMAC Key to be re-distributed securely. At least, there will be no need to revoke certificates unless they are also suspected of being compromised.

Instructions to enable OpenVPN-HMAC in TLS mode are included at Section 2.5.5.

2.1.3. Additional privacy and protection through Proxy

A VPN connection back to a SOHO network provides an additional layer of privacy to the user of mobile devices. Since all traffic appears to be coming out of the SOHO network, any connections made from a mobile device to a server on the Internet will appear to have originated from the SOHO network itself. For example, sites such as Facebook, Google, or Amazon will not be able to establish the VPN client location solely based on the IP address, as it will appear to originate from the edge of the SOHO Network. And, because some services alter the content delivered based on IP location, this will help ensure the content delivered remains consistent whether the user is connecting from inside the SOHO network or from any other location. Note that if the mobile device authorizes its browser or any other app to access GPS data, this particular privacy protection will have been overcome.

There may be another benefit of having a mobile device Internet traffic routed through the SOHO network. If additional boundary security is deployed on the SOHO network, for example an IPS, the mobile devices will also benefit from the added security while connected through the VPN.

2.1.4. SOHO network access

A key feature of any VPN service is the ability to securely access resources hosted on the SOHO network without having to punch more holes (ports) through the firewall. This opens up a world of possibilities for mobile users:

a. Access to files hosted on network shares and Network-Attached Storage (NAS);
b. Print documents;
c. Remote desktop into a host;
d. Access an IP Based camera or surveillance system.

2.1.5. Broad device support

OpenVPN has been around since 2001 (Dunston). Support continually expands to include popular platforms. As of this writing, clients are available for Windows, Mac OSX, Linux, iOS 5.0+, and Android 4.0+.

2.2. Internet Requirements

There are a few things required from the SOHO Internet connection to allow clients to connect to an OpenVPN Server. Unfortunately, not all ISPs are created equal; some may be more or less suited than others for hosting a VPN. In addition, some functionality deployed on Free/Public WAPs may affect the establishment of VPN Tunnels unless precautions are taken. Finally and while technically feasible, it will be important to bear in mind acceptable use policies when considering the use of VPN.

2.2.1. SOHO Network Internet Upload/Download Speeds

In order to ensure a usable connection speed between the OpenVPN Server and the mobile clients, the upload and download speeds provisioned by the ISP to the SOHO network must be carefully considered. Usually, the download speed of the SOHO Internet connection will not be the problem. Referring to figure 1, note that mobile clients will "download" traffic from the SOHO Network. This means that the VPN server will "upload" data to the client. Therefore, the bottleneck will most likely be the "upload" speed imposed by the ISP providing Internet access to the SOHO network.

Do not let low upload speed offered by many ISPs act as a deterrent. Testing conducted for this paper demonstrated that 1Mbps upload speed was adequate for web browsing and emails. 3Mbps was sufficient for more network intensive tasks such as remote desktop, file transfer and even video streaming. Beside, unless users pay a premium while on the road, they are likely to be limited to low download speeds on their end. In North America, free WAPs download speed commonly hovers between 256 kbps and 1 Mbps while hotels and other paid WAPs download speed range between 1Mbps and 5Mbps.

Finally, another factor that will affect throughput is the number of clients connecting simultaneously to an OpenVPN Server. Greater upload speed will significantly increase the number of concurrent connections possible.

2.2.2. Internet Facing IP Address / Internet hostname

In order to connect, mobile clients will need to know the server's IP address. The easiest solution is to hard code the SOHO Network Internet facing IP Address in the OpenVPN client configuration file. Some ISP will be able to provide a static IP address, usually for a small additional fee. Otherwise, ISPs will switch their client's Internet IP addresses for various reasons on an unpredictable schedule. To resolve this issue, some dynamic DNS services providers offer persistent hostname to IP address resolution for free. This is usually accomplished by running a small daemon program in the server's background that checks on a regular basis to see if the Internet IP address assigned to the network has changed. When the Internet IP does change, the daemon automatically updates the DNS record with the new IP address. See section 2.5.9 for details on how to configure a dynamic DNS daemon on the RPi.

2.2.3. Usage Allowance

Some ISPs have placed Usage Allowance restrictions in their Internet packages that limit the quantity of data a customer is allowed to consume before incurring excess charges. Therefore, it is important to carefully consider mobile client's traffic volume through the VPN when selecting a plan with monthly allowance restrictions. As a rule of thumb, a factor of 2.5 to 1 is adequate to estimate the traffic volume through the SOHO. For example, for every 1MB of Internet data a mobile client downloads through the VPN, the ISP will record the consumption of 2.5MB of data. This accounts for 1MB of data coming in from the Internet to the VPN Server and then, another 1MB for that same data to traverse the VPN tunnel out to the client. The remaining ½MB accounts for the encrypted packet encapsulation and the TLS control data.

2.2.4. Allowed/Blocked Ports & Acceptable Use Policies

For various reasons, ISPs may block certain incoming ports such as TCP 80 (HTTP Web Server), TCP 443 (HTTPS Web Server), or even UDP 1194 (OpenVPN's default port). However, several solutions exist. First, it may be possible to contact the ISP and request a specific port to be unblocked. If they refuse, review the ISP's acceptable use policy to ensure that nothing else you try will contravene it.

Next, alternate ports may be open by default; including UDP and TCP Port 53 which are normally reserved for DNS and are very rarely blocked inbound. TCP Port 20, which is used in active FTP, is also more likely to be open. Finally, try a high port such as UDP 60000. However, if these ports are also blocked, it is time to consider changing ISP or at least switching Internet plan. Because, the ISP clearly wishes to block services hosted by their clients under the current plan and any other solution is likely going to be against the ISP's acceptable use policy.

Nevertheless, an ISP blocking incoming ports is not as common as it once was. On the other hand, free/public WAPs will probably have many outbound ports blocked. These WAP services, especially the free ones, usually try to manage the volume of traffic by limiting users to certain Internet services such as web and email. OpenVPN can be configured to use any TCP or UDP Port. Therefore, it is easy to setup OpenVPN to listen on TCP Port 80 or 443. In addition, OpenVPN offers an innovative solution to concurrently use TCP port 443 for both OpenVPN and HTTPS web server. It is called Port Sharing. This option enables OpenVPN to inspect incoming packets bound for TCP port 443 and differentiate between OpenVPN traffic and HTTPS traffic, which it automatically forwards to the HTTPS server (Keijser). Finally, free/public WAPs may also have restrictions included in their acceptable use policy which mobile user will have to respect before initiating a VPN connection.

In the end, the selection of the port and protocol will be an educated guess of the most likely port to be opened both inbound at the SOHO's network edge with the ISP and outbound from the most commonly used WAPs[2]. There is nothing wrong with trying UDP port 1194 first and conducting some tests. It is always possible and relatively easy to reconfigure the server and clients later to a different port and/or protocol. Finally, the acceptable use policies of both the SOHO ISP and the WAP must be respected to avoid unnecessary legal complications.

2.3. Hardware Requirements

2.3.1. Router

Most modern routers will allow incoming connections for services such as OpenVPN through relatively minor configuration changes. All that is needed is some form of Port Forwarding (aka Virtual Server) functionality. Meyers describe port forwarding as "a mean for servers to work behind a NAT

[2] Based on personal experience writing this gold paper, I found that approximately one quarter of all free WAPs I tested blocked UDP port 1194. None of the paid WAPs I tried blocked it

router". In essence, a port is open on the router's external interface and any incoming connection on that specific port is routed to a pre-determined internal IP address where a server will be listening for incoming connections.

You may be tempted to take a short cut by assigning the OpenVPN server to the Demilitarized Zone (DMZ) instead of configuring port forwarding. Don't! A host located in the DMZ, while inside the network, is not protected by the router's firewall. All traffic, for which the router does not have a NAT Table entry or port-forwarding rule, will be sent to the host in the DMZ. A host in the DMZ should be considered compromised at all times (White, Conklin, Cothren, & Davis). Thus, a DMZ will significantly increase the risks that the OpenVPN server will be compromised by being virtually fully exposed to the Internet.

2.3.2. Raspberry Pi Model B

The RPi is the brainchild of Eben Upton, Rob Mullins, Jack Lang and Alan Mycroft, from the University of Cambridge's Computer Laboratory (Raspberry Pi Foundation). They set out to design a small, affordable computer to stem their perceived decline in student's computer skill levels. They established the Raspberry Pi Foundation as a UK registered charity and entered into manufacturing deals with element 14/Premier Farnell and RS Electronics. Model A suffered from significant shortcomings including a single USB port and lack of an Ethernet port. However, it was so well received by the enthusiast's community that it inspired the development of Model B, which sold over one million units within its 1st year.

The RPi Model B costs approximately $35 and comes with a 700MHz ARM CPU, 512MB RAM (since Oct 2012), Composite RCA and HDMI Video output, 10/100 wired Ethernet and two USB ports (elinux.org). It also comes with a Secure Digital (SD) / MMC / SDIO card slot used to hold the OS and provide rewritable storage.

The RPi requires a 5v 700mA power source provided via Micro USB. Use of a PC's USB port to power the RPi is not recommended. USB 2.0 specification sets the maximum power draw at 500mA. Therefore, although the RPi may appear to work properly, it may malfunction in unpredictable ways (elinux.org). Not to mention running the risks of damaging the PC's USB chipset overtime. An adequate power source can be found for less than $10. Moreover, many old Cell Phone chargers with a Micro USB connector will do the trick as long as it supplies 5v with a minimum of 700mA.

Also, the RPi does not come with a case. While not absolutely necessary, it is a good idea to prevent accidental short-circuits during operation that could permanently damage the RPi. Cases can be found online for as little as $5. Alternatively, Legos plans[3,4,5] for RPi abound on the Internet and have been extensively used as a free solution for Lego owners.

Finally, while humble in its specifications, the RPi has enough processing power to run Linux with OpenVPN Server.

2.4. Operating System Requirement

According to elinux.org, there are 28 (and counting) OS distributions available for the RPi. Each distribution has its own advantages and disadvantages based on its intended purpose. However, to minimize problems and have access to a large community support group, Raspbian is recommended. It is a port from the Debian Linux Distribution and includes over 35,000 packages, pre-compiled software bundles, including OpenVPN (raspbian.org).

2.5. Server Configuration

OpenVPN offers a multitude of functionality and is highly configurable. The following instructions will result in an OpenVPN server in TLS mode listening on UDP port 1194. It will use self-signed certificates generated using the Easy RSA management package included with OpenVPN. Finally, it will also use "OpenVPN-HMAC" (See section 2.1.2) to authenticate packets.

A step-by-step OpenVPN Configuration guide is located at Appendix A.

2.5.1. OS Installation & Configuration

The latest Raspbian image can be downloaded from the Raspberry Pi Foundation Download Page[6]. A 4GB SD Card is required to store the OS and provide a writable file system. Installation instructions to load the raw image on the SD card are also be located on the download page.

[3] http://www.instructables.com/id/Lego-Raspberry-Pi-Case/?ALLSTEPS
[4] http://www.raspberrypi.org/archives/1354
[5] http://anoved.net/2013/06/raspberry-pi-lego-case/
[6] http://www.raspberrypi.org/downloads

Initially, a monitor and keyboard is required to interact with the RPi. On its very first boot, it will start the Raspberry Pi Configuration Tool automatically. This is a good opportunity to complete the following tasks:

a. Update the Raspberry Pi Configuration Tool;
b. Change the user ("pi") password;
c. Set the RPi Hostname; and,
d. Enable SSH.

Once done, the RPi will reboot. Log in as username "*pi*". The next step is to update Raspbian using the following commands:

```
a. sudo apt-get update
b. sudo apt-get upgrade
c. sudo apt-get dist-upgrade
```

Finally, record the RPi MAC Address using the command "**`ifconfig eth0`**". Once done, shut down the RPi using the command "**`sudo poweroff`**"

2.5.2. Router Subnet Configuration

First, it is important to avoid subnet conflicts (openvpn.net). Imagine a SOHO Network using the subnet 192.168.1.0/24. Now imagine a mobile client connected to a WAP in some remote hotel that is also using the subnet 192.168.1.0/24. Once the mobile client connects to the OpenVPN Server, there will be routing conflicts and the mobile device cannot be trusted to route all traffic reliably through the VPN tunnel. To reduce the likelihood of this problem occurring, it is recommended that the SOHO Network be assigned a private subnet that is rarely seen. For this purpose, these instructions will use the subnet 192.168.200.0/24.

2.5.3. RPi's static IP

Using the previously recorded MAC address of the RPi, it is time to allocate a static IP Address on the SOHO LAN to supplement port forwarding in section 2.5.8. This paper will assign to the RPi the IP address of 192.168.200.200.

Network administrators have two options. The RPi can be assigned a static IP Address via the host configuration file "**`/etc/network/interfaces`**"[7]. Alternatively, the IP Address can be reserved then dynamically assigned by leveraging the home router's Dynamic Host Configuration Protocol (DHCP).

Note that any future change to the RPi's IP Address will require equivalent changes to the OpenVPN Server Configuration file (section 3.5.6) and iptables rule (section 2.5.7).

2.5.4. VPN Installation

The RPi is now ready for the installation and configuration of the OpenVPN Service. At this point, the monitor and keyboard is no longer required. The RPi can be accessed via SSH from any other PC. OpenVPN is installed using the command "**`sudo apt-get install openvpn`**".

2.5.5. Key Creation

OpenVPN requires Server and Clients key pairs. They are generated using the Easy_RSA example included with OpenVPN under "**`/usr/share/doc/openvpn/examples/easy-rsa/2.0/`**". A copy must be made to "**`/etc/openvpn/easy-rsa/`**" and the environment variable of EASY_RSA within the file "vars" must be changed to this new directory.

Next, Root Certificate and the root CA Key are built using the commands:

a. **`./source vars`**: Sets required environment variables and default certificate values;
b. **`./clean-all`**: Removes any previous attempts at creating Root Cert & CA Keys. Should not be required but recommended to avoid problems; and,
c. **`./build-ca`**: Builds the CA certificate and key by invoking the interactive openssl command.

Then, the OpenVPN Server key pair is built using the command "**`./build-key-server [Server_Name]`**". The Common Name must be identical to the value used for [Server_Name]. The challenge password must remain blank and the certificate must be signed.

Next, as many Client key-pairs as needed are generated using the command "**`./build-key-pass [User_Name]`**". As with the server certificate, the Common Name must remain identical to the

[7] http://elinux.org/RPi_Setting_up_a_static_IP_in_Debian

value used for [User_Name]. A PEM pass phrase the user will remember must be set but the challenge password shall remain blank. The client certificates must also be signed. Normally, this should mark the end of the key pair creation process. However, there is a problem with OpenSSL version 1 used by Easy_RSA which prevents the OpenVPN Connect Clients for both Android and iOS from successfully parsing the certificates (Wolfs). To ensure that the keys will work on all clients, the following command must be run: "**`openssl rsa -in [User_Name].key -des3 -out [User_Name].3des.key`**".

Then, the Diffie-Hellman keys that are required by the server as part of the TLS Handshake with clients must be generated using the command:

a. **`./build-dh`**: Builds the Diffie-Hellman parameters for the server side.

Finally, the OpenVPN-HMAC protection mechanism described in Section 2.1.2 is generated using the command:

a. **`Openvpn –genkey –secret keys/ta.key`**: Creates the OpenVPN-HMAC Keys.

2.5.6. VPN Configuration

The file "server.conf" is used to dictate the behaviour of OpenVPN. Create and open "**`/etc/openvpn/server.conf`**" in a text editor such as nano. This section will summarize the most important entries required based on the OpenVPN man page (Yonan) and the "How-To" page (openvpn.net). The recommended configuration file for this paper is included in its entirety at Appendix A.

- **`local [host]`**: Binds OpenVPN to the RPi's Static IP. It must be identical to the IP assigned in section 2.5.3.
- **`proto udp & port 1194`**: Tells OpenVPN Server which port and which protocol to listen to for incoming client connections. The default is UDP Port 1194 but it can be set to any port. TCP works equally well but will provide a slower connection due to the additional overhead required by the protocol.
- **`dev tun`**: Establish a virtual point-to-point Layer 3 IP link.
- **`ca / cert / key / dh / tls-auth`**: Options used to point OpenVPN to the various Certificates and Keys required.

- **`Server 10.8.0.0 255.255.255.0`**: Configures server mode and VPN subnet. The server will be assigned 10.8.0.1 and the rest will be assigned to clients as they connect.
- **`ifconfig 10.8.0.1 10.8.0.2`**: Tunnel adapter parameters where 10.8.0.1 is the IP Address of the local VPN endpoint and 10.8.0.2 is the IP address of the remote endpoint. The two private IP addresses cannot be a part of any existing subnet in use in the SOHO Network or WAP or the routing problems described in section 2.5.2 will surface.
- **`Push`**: Pushes configuration options to the client. Used to enable routing between the clients and the SOHO Network through the OpenVPN Server and, to push the SOHO DNS Server IP Address.
- **`Client-to-client`**: Allow OpenVPN to route packets between remote clients.
- **`Cipher AES-128-CBC`**[8]: Tells OpenVPN which algorithm to use. The command "**`openssl speed`**" will test the throughput speed of all algorithms. The command "**`openvpn --show-ciphers`**" will show which one is supported by OpenVPN.
- **`User nobody & group nobody`**: Drop privileges of OpenVPN. Security feature in case an OpenVPN 0-day is exploited to gain foothold on the VPN Server.

2.5.7. Routing traffic through the RPi between mobile clients and the SOHO network using iptables

iptables is the command used to configure the Rapsbian Network Packet Processing subsystem (Purdy). Also referred to as the Linux Kernel firewall, iptables is the secret sauce that defines rules used to affect or monitor packets flow. For the purpose of the RPi OpenVPN Server, a rule must be added to enable the RPi to provide NAT services to connected clients.

First, IP forwarding, which is disabled by default as a security measure, needs be enabled. "**`net.ipv4.ip_forward = 1`**" must be uncommented/set in "**`/etc/sysctl.conf`**" (Purdy)[9]. The change is applied using the command "**`sysctl -p`**".

[8] AES-128-CBC was selected because it provided the best throughput when tested using openssl speed on a RPi.
[9] A detailed description of iptables is beyond the scope of this paper. However, Gregor N. Purdy wrote a great book titled "*Linux iptables pocket guide*" which is exceptional both in its simplicity and in thoroughness at explaining iptables.

Then, remember from section 2.5.6 that clients connected to the OpenVPN server will be assigned an IP Address in the 10.8.0.0/24 subnet for their tunnel. However, the SOHO Subnet is 192.168.200.0/24. Using the command below, the RPi will provide NAT between the OpenVPN Client Subnet (10.8.0.0/24) and the SOHO Subnet (192.168.200.2/24).

```
iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j SNAT --to-
source 192.168.200.200
```

Lastly, iptables commands entered manually will not persist across reboot. The easiest way to remedy this situation is to include them in a batch file as demonstrated at Appendix A.

2.5.8. Punching a hole in the router's firewall

It is time to setup the router's port forwarding rules to enable clients to connect from the Internet. In section 2.5.6, the OpenVPN server was configured to listen on a specific protocol and port. Using this information, navigate the router's configuration page and enter the port forwarding information. In this paper's example, the router must pass incoming UDP packets destined for port 1194 (OpenVPN Default) to a listening server with an IP address of 192.168.200.200.

2.5.9. Dynamic DNS Configuration

Because most Internet plans do not include a static Internet IP, mobile clients will need an alternate mean of connecting back to the SOHO network. Dynamic DNS service providers such as dnsdynamic.org[10] can provide a static Internet hostname that can be resolved to the SOHO IP from anywhere on the Internet. A small application can be installed on the RPi that regularly verifies if the IP address has changed and automatically update the DNS A record hosted by the Dynamic DNS Service provider. A quick tutorial on how to setup a Dynamic DNS Internet hostname using dnsdynamic.org is included at Sub-Appendix A-1.

2.6. Client Configuration

The RPi is now fully configured and after one last reboot, will be ready to receive incoming client connections. All that is left is the creation of an OpenVPN (.ovpn) configuration file that can be

[10] http://www.dnsdynamic.org/

securely uploaded to client devices. The configuration file greatly simplifies the actions required by the end user to start using the newly created VPN Service and can be resumed in three simple steps:

a. Create the .ovpn configuration file;
b. Find and install the OpenVPN Client Software on a mobile client; and,
c. Securely transfer the .ovpn configuration file to the client and test the connection to the Server.

2.6.1. Creating the .ovpn configuration file

This file sets all the options required by an OpenVPN Client to successfully connect to an OpenVPN Server. This section will summarize the most important entries required based on the OpenVPN man page (Yonan) and the "How-To" page (openvpn.net). The recommended client configuration file for this paper is included in its entirety at Appendix A.

- **`remote [host] [port]`**: [host] tells the client where to connect to. It can be a static Internet IP Address or preferably the host name setup in section 2.5.9. As for [port], it must be identical to the port setup in the server configuration file (section 2.5.6) and allowed through the router's firewall (sections 2.5.8)
- **`proto udp`**: This option tells the client which protocol to use and should be identical to the protocol setup in the server configuration file (section 2.5.6) and allowed through the router's firewall (sections 2.5.8).
- **`ns-cert-type server`**: Security precaution that protect against MitM attacks by blocking clients from connecting to any presumed server that lack the nsCertType=server designation in its certificate.
- **`<ca>`**: Contains the Certification Authority's Public Certificate. Will be used to verify the signature of the OpenVPN server's certificate.
- **`<cert>`**: Contains the Client's Public Certificate.
- **`<key>`**: Contains the password protected Client Private Key (*[User_Name].3des.key*).
- **`<tls-auth>`**: Contains the static key for OpenVPN-HMAC.
- **`key-direction 1`**: Tells client that OpenVPN-HMAC (tls-auth) is enabled and will be initiated from the other end.

2.6.2. Installing OpenVPN Clients

OpenVPN Clients installation is self-evident. Simply download and install the application following the normal process for the specific device. Below is a table of recommended OpenVPN clients for the more popular operating systems.

Supported OS	Recommended OpenVPN Client
Windows (XP, Vista, 7, 8)	OpenVPN GUI[11]
Mac OS X	Tunnelblick[12]
Ubuntu/Debian	apt-get install openvpn
Fedora/CentOS/RedHat	yum install openvpn
IOS 5.0+	OpenVPN Connect[13] by OpenVPN Technologies
Android 4.0+	OpenVPN Connect[14] by OpenVPN Technologies

2.6.3. Transferring the .ovpn configuration file to the client & testing the connection to the OpenVPN Server

With an OpenVPN Client installed and running, the mobile user is almost ready to connect. But first, the .ovpn file must be securely transferred to the mobile device. Although the private key held within it is secured by a password, the .ovpn file should never be placed in a publicly available location. After all, the key is only as strong as the password protecting it and is susceptible to password cracking as described in section 2.1.1. Instead, move the .ovpn file to a removable media and transfer the file directly to the mobile device or through a trusted PC via cable. "`scp`" is also a viable option.

Once transferred, open the .ovpn client configuration file in the OpenVPN Client application and hit connect. When asked, supply the password to decrypt the client private key. Provided all is set correctly, it will successfully connect to the OpenVPN Server.

If the SOHO has wireless networking enabled and the user is connected to it, try connecting from within the network. A successful connection and the client's ability to browse the web will validate the setup.

[11] http://openvpn-gui.sourceforge.net/
[12] http://code.google.com/p/tunnelblick/
[13] https://play.google.com/store/apps/details?id=net.openvpn.openvpn
[14] https://itunes.apple.com/app/openvpn-connect/id590379981

Next, the user should try connecting from a distant network. Home networks are preferred as they usually have fewer restrictions and blocks that may interfere with an OpenVPN connection attempt. If the connection succeeds then the mobile user will have proven that connectivity can be established from the Internet, through the SOHO router/firewall and reach the OpenVPN Server inside.

Then, it will be a matter of trying to connect through a free/public WAP. Mobile users will be wise to only connect to WAPs that allow, both technically and through the acceptable use policy, the connection to be established.

Finally, mobile users should be reminded that most of the time when they connect to a free/public WAP, their devices would usually not be allowed to connect to the Internet, let alone connect to an OpenVPN Server, until they have visited a web page where they may be asked to authenticate and/or agree with the acceptable use policy. Obviously, the OpenVPN Client will not be able to connect until the user has been authorized access to the Internet.

2.7. Basic Troubleshooting

In cases where the mobile client(s) are unable to connect to the OpenVPN server, the following leads can be used to troubleshoot the problem:

a. Verify the OpenVPN Service started successfully;

b. Verify RPi's IP Address and connectivity with the router;

c. Double check the firewall (port forwarding/virtual server) entry on the router and iptables entry on the RPi;

d. If unable to connect with any client, verify server.conf then each client .ovpn configuration file;

e. If a single client is unable to connect, verify the client's .ovpn configuration file;

f. Increase logging by inserting/updating the option "**`verb 6`**" in both server and client configuration files; and,

g. Read the OpenVPN Logs. Server logs are located in "**`/var/log/openvpn.log`**" and "**`/var/log/openvpn-status.log`**".

3. Conclusion

For security professionals, online privacy has always been a concern. In fact, if asked by family, friends or other usually cash strapped entities such as charities and small businesses, most of us would recommend against using free/public WAPs to access any kind of sensitive or private information. Yet, we must also be cognizant of the unrelenting drive users feel to access their information where and whenever they need to. In fact, one could argue that this need/desire has emerged as an inextricable aspect of our culture as we became increasingly connected. In addition, even the most security conscious user will admit to making exceptions in an "emergency". As a result, the desire and expectations of staying connected have increased to the point where telling users not to use free/public WAP has become an unrealistic proposition for almost everyone.

Yet, there is little to offer in terms of a useful and affordable solution that addresses both privacy and access needs. Hopefully, this paper provides a basic understanding of VPN configuration requirements and a practical solution that can be easily deployed in the service of netizens inhabiting SOHO networks near you.

4. References

Arnote, P. (2013, September Issue 80). *OpenVPN: Other VPN Services.* Retrieved October 14, 2013, from The PCLinuxOS Magazine: http://pclosmag.com/html/Issues/201309/links.html

Burgsma, R. (2013, Feb 27). *Improving OpenVPN security by revoking unneeded certificates.* Retrieved July 14, 2013, from blog.remibergsma.com: http://blog.remibergsma.com/2013/02/27/improving-openvpn-security-by-revoking-unneeded-certificates/

Dierks, T., & Rescorla, E. (2008, Aug). *RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2.* Retrieved August 17, 2013, from ietf.org: http://tools.ietf.org/html/rfc5246

Dingledin, R., & Murdoch, S. J. (2009, Mar 11). *Performance Improvments on Tor or, Why Tor is slow and what we're going to do about it.*

Dunston, D. (2003, Nov 10). *OpenVPN: An Introduction and Interview with Founder, James Yonan.* Retrieved 07 August, 2013, from LinuxSecurity.com: http://www.linuxsecurity.com/content/view/117363/171/

elinux.org. (2013, Jun 24). *RPi -Hardware*. Retrieved June 24, 2013, from elinux.org: http://elinux.org/RPi_Hardware

Feilner, M., & Graf, N. (2009). *Beginning OpenVPN 2.0.9 Build and Integrate Virtual Private Networks Using OpenVPN.* Birmingham, UK: Packt Publishing.

Eric Jodoin, ejodoin@hotmail.com

Keijser, J. J. (2011). *OpenVPN 2 Cookbook: 100 simple and incredibly effective recipes for harnessing the power of OpenVPN 2 network.* Birmingham, U.K.: Packt Publishing.

McMillan, R. (2005, Jun 23). *Experts split on port 445 security risk*. Retrieved September 14, 2013, from Security Central - InfoWorld: http://www.infoworld.com/d/security-central/experts-split-port-445-security-risk-630

Meyers, M. (2009). *CompTIA Network+* (4th ed.). New York: McGraw-Hill.

openvpn.net. (2013, Oct 14). *HOWTO - Sample OpenVPN 2.0 configuration files*. (X, Editor) Retrieved October 14, 2013, from openvpn.net: http://openvpn.net/index.php/open-source/documentation/howto.html

openvpn.net. (2013, Aug 14). *Static- Key Mini-HOWTO*. Retrieved August 14, 2013, from openvpn.net: http://openvpn.net/index.php/open-source/documentation/miscellaneous/78-static-key-mini-howto.html

Purdy, G. N. (2004). *Linux iptables Pocket Reference.* Sebastopol, CA: O'Reilly.

Raspberry Pi Foundation. (2013, Aug 23). *About us | Raspberry Pi*. Retrieved August 23, 2013, from raspberrypi.org: http://www.raspberrypi.org/about

raspbian.org. (2013, Aug 24). *Home Page*. Retrieved August 24, 2013, from www.raspbian.org: http://www.raspbian.org/

The Tor Project, Inc. (2013, Oct 14). *Tor project: Overview*. Retrieved October 14, 2013, from The Onion Router (TOR): https://www.torproject.org/about/overview.html.en

White, G., Conklin, A. W., Cothren, C., & Davis, R. L. (2009). *CompTIA security+* (2nd ed.). New York: McGraw-Hill.

Wolfs, D. (2013, Feb 12). *PolarSSL: error parsing config private key [Msg 9]*. Retrieved July 29, 2013, from Message posted to https://forums.openvpn.net/topic12035-15.html

Yonan, J. (2013, Jul 12). *OpenVPN 2.3 Man Page.* Retrieved July 21, 2013, from OpenVPN Community: https://community.openvpn.net/openvpn/wiki/Openvpn23ManPage

Zetter, K. (2007, Sep 10). *Rogue Nodes Turn Tor Anonymizer Into Eavesdropper's Paradise*. Retrieved October 14, 2013, from Wired.com: http://www.wired.com/politics/security/news/2007/09/embassy_hacks?currentPage=all

Eric Jodoin, ejodoin@hotmail.com

OpenVPN step-by-step Configuration

This appendix is meant to expedite the configuration of your OpenVPN service. Commands are described in details in sections 2.5 and 2.6 of the main document. Any part of a command or configuration file highlighted in yellow requires amendments using information specific to your particular network.

Raspberry Pi Configuration (Req. Keyboard & Monitor)

1. Download and install Rasbian "Wheezy"[15] on SD Card from: http://www.raspberrypi.org/downloads
2. Boot the Raspberry Pi (RPi)
3. Once inside the Raspberry Pi Configuration Tool, select the following Setup Options:
   a. **8 Advanced Options -> A5 Update**
   b. **2 Change User Password** (The user name is "pi")
   c. **3 Enable Boot to Desktop -> Console Text console, (default)**
   d. (Optional) **1 Expand File System**
   e. (Optional) **8 Advanced Options -> A2 Hostname -> OpenVPN-Server**
   f. (Optional) **8 Advanced Options -> A3 Memory Split -> 16**
   g. (Optional) **8 Advanced Options -> A4 SSH -> Enable**
   h. **<finish> ->** Select **"Yes"** to Reboot
4. Login using user name "pi" and password set in step 3c.
5. Update the RPi using the following commands:
   a. **sudo apt-get update**
   b. **sudo apt-get upgrade**
   c. **sudo apt-get dist-upgrade**
6. Record the MAC Address using the command: **ifconfig eth0**

```
pi@OpenVPN-Server ~ $ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr b8:27:eb:a1:c4:88
          inet addr:192.168.200.200  Bcast:192.168.200.255
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50556 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10641 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:67745129 (64.6 MiB)  TX bytes:998895 (975.4 KiB)
```

7. Shutdown the RPi using the command: **sudo poweroff**

[15] This tutorial was created using the "2013-09-25-wheezy-raspbian.zip" image.

Router Configuration[16]

1. Set the DHCP Lease IP Address Range between 192.168.200.10 and 192.168.200.200
2. Reserve IP Address 192.168.200.200 for the RPi
3. Enable Virtual Server/Port Forwarding to the RPi on Ports 1194 UDP

OpenVPN Configuration

The means of access to RPi to complete the configuration is a personal choice. The steps below can be done locally using a keyboard and monitor or remotely via SSH, meaning the keyboard and monitor are no longer necessary and can be removed from the RPi. Regardless of which method chosen, all changes and saves will require "root" privileges.

1. Boot the RPi and login.

2. Root Shell access is required. Use the command:
   `sudo bash`

3. Install a Dynamic DNS service if required (See Appendix B).

4. Download and install the packages required for VPN setup using the command:
   `apt-get install openvpn`

5. Copy the default Easy RSA configuration files using the command:
   `cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0 /etc/openvpn/easy-rsa`

6. Next, go to the newly copied directory using the command:
   `cd /etc/openvpn/easy-rsa`

7. Open the file **`/etc/openvpn/easy-rsa/vars`** for editing. (For example: **`nano vars`**)

   a. Find and change the EASY_RSA variable to:

   `export EASY_RSA="/etc/openvpn/easy-rsa"`

   b. (Optional) At the bottom of the file, update the default values used to create the certificates:

```
export KEY_COUNTRY="CA"
export KEY_PROVINCE="Ontario"
export KEY_CITY="Ottawa"
export KEY_ORG="GOLD1"
export KEY_EMAIL="ejodoin@hotmail.com"
```

[16] It is recommended to select a rarely used subnet to avoid routing conflicts. See Section 2.5.2.

8. Build the CA Root Certificate and root CA key using the commands:
    a. `source vars`
    b. `./clean-all`
    c. `./build-ca`
        i. Enter anything when asked for input.
9. Build the OpenVPN Server key pair using the command:
    a. `./build-key-server OpenVPN-Server`
        i. Enter anything when asked for input except for the following:
        ii. `Common Name [OpenVPN-Server]:` <- Must Match: OpenVPN-Server
        iii. `A challenge password? []:` <- Must be left blank!
        iv. `Sign the certificate? [y/n]: y`
        v. `1 out of 1 certificate requests certified, commit? [y/n] y`
10. Build the Clients key pairs using the commands:
    a. `./build-key-pass UserName`
        i. When prompted, provide a PEM password that the user will have to remember.
        ii. For the rest of the input, enter anything with the following exceptions:
        iii. `Common Name [UserName]:` <- Must Match: UserName
        iv. `PEM Pass Phrase? []:` <- Password User will remember
        v. `A challenge password? []:` <- Must be left blank!
        vi. `Sign the certificate? [y/n]: y`
        vii. `1 out of 1 certificate requests certified, commit? [y/n] y`
    b. `cd keys`
    c. `openssl rsa -in UserName.key -des3 -out UserName.3des.key`
    d. `cd ..`
    e. Repeat steps a to d for every client.

Eric Jodoin, ejodoin@hotmail.com

11. Return to the easy_rsa directory using the command:
    cd /etc/openvpn/easy-rsa/

12. Generate the Diffie-Hellman using the command:
    ./build-dh

13. Generate the OpenVPN-HMAC Key using the command:
    openvpn --genkey --secret keys/ta.key

14. Create and Open **/etc/openvpn/server.conf** for editing, enter the following and save.

WARNING: Cut and Paste from this PDF document **WILL** substitute some characters. Carefully Review all your code for errors.

```
local 192.168.200.200
dev tun
proto udp
port 1194
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/OpenVPN-Server.crt
key /etc/openvpn/easy-rsa/keys/OpenVPN-Server.key
dh /etc/openvpn/easy-rsa/keys/dh1024.pem
server 10.8.0.0 255.255.255.0
# server and remote endpoints
ifconfig 10.8.0.1 10.8.0.2
# Add route to Client routing table for the OpenVPN Server
push "route 10.8.0.1 255.255.255.255"
# Add route to Client routing table for the OpenVPN Subnet
push "route 10.8.0.0 255.255.255.0"
# your local subnet
push "route 192.168.200.0 255.255.255.0"
# Set primary domain name server address to the SOHO Router
# If your router does not do DNS, you can use Google DNS 8.8.8.8
push "dhcp-option DNS 192.168.200.1"
# Override the Client default gateway by using 0.0.0.0/1 and
# 128.0.0.0/1 rather than 0.0.0.0/0. This has the benefit of
# overriding but not wiping out the original default gateway.
push "redirect-gateway def1"
client-to-client
duplicate-cn
keepalive 10 120
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
cipher AES-128-CBC
comp-lzo
user nobody
group nogroup
persist-key
persist-tun
status /var/log/openvpn-status.log 20
log /var/log/openvpn.log
verb 1
```

15. Open **/etc/sysctl.conf** for editing.

    a. Find and uncomment the following entry:

    ```
    net.ipv4.ip_forward=1
    ```

    b. Save and close.

    c. Apply the changes using the command: **sysctl -p**

16. Create **/etc/firewall-openvpn-rules.sh** for editing, enter the following and save.

    ```
    #!/bin/sh

    iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j SNAT --to-
    source 192.168.200.200
    ```

17. Change permissions and ownership of **/etc/firewall-vpn-rules.sh** to root using the commands:

    a. **chmod 700 /etc/firewall-Openvpn-rules.sh**

    b. **chown root /etc/firewall-Openvpn-rules.sh**

18. Open **/etc/network/interfaces** for editing.

    a. Find the *iface* command for eth0 , insert immediately below(indented):

    ```
    iface eth0 inet dhcp
        pre-up /etc/firewall-openvpn-rules.sh
    ```

19. Reboot using the command: **sudo reboot**

OpenVPN Client Configuration

Now that the VPN server is up and running, it is time to build and deploy client configuration files.

1. For each client, create and open **/etc/openvpn/UserName.ovpn** for editing, enter the following and save[17].

WARNING: Cut and Paste from this PDF document **WILL** substitute some characters. Carefully Review all your code for errors.

```
client
dev tun
proto udp
remote <server Hostname or IP> 1194
resolv-retry infinite
nobind
persist-key
persist-tun
mute-replay-warnings
ns-cert-type server
key-direction 1
cipher AES-128-CBC
comp-lzo
verb 1
mute 20

<ca>
-----BEGIN CERTIFICATE-----
(Copy and insert content of ca.crt) …
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
(Copy and insert content of UserName.crt) …
-----END CERTIFICATE-----
</cert>
<key>
-----BEGIN RSA PRIVATE KEY-----
(Copy and insert content of UserName.3des.key) …
-----END RSA PRIVATE KEY-----
</key>
<tls-auth>
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
(Copy and insert content of ta.key) …
-----END OpenVPN Static key V1-----
</tls-auth>
```

[17] Use the script included in Sub-Appendix A-1 to automate this task and avoid manual entry errors.

2. Securely transfer .ovpn file to client.

3. Remove the following files from the RPi OpenVPN server and store securely offline.

Filename	Purpose
ca.key	Root CA key
UserName.key	UserName Key
UserName.3des.key	Re-encrypted UserName Key
UserName.ovpn	OpenVPN Configuration file used on client devices

Automated .ovpn Creation Script

Creating .ovpn client configuration files by hand can be tedious if there is going to be multiple clients. This also increases the risks of manual input errors. To remediate these problems, use the small script below that merges the "default" configuration settings with all the keys a specific user needs. When run, the script will ask for the user name and automatically generate the .ovpn file.

Defaults Text File

This file contains the default configuration options which will be identical for every client. Simply create a file in the **`/etc/openvpn/easy-rsa/keys/`** folder and name it "**`Defaults.txt`**". Open it for editing, enter the following and save.

WARNING: Cut and Paste from this PDF document **WILL** substitute some characters. Carefully Review all your code for errors.

```
client
dev tun
proto udp
remote <server Hostname or IP> 1194
resolv-retry infinite
nobind
persist-key
persist-tun
mute-replay-warnings
ns-cert-type server
key-direction 1
cipher AES-128-CBC
comp-lzo
verb 1
mute 20
```

Script File

The second file is a shell script that will create a .ovpn file based on the client name used when creating the client key pair. Start by creating a file in the **/etc/openvpn/easy-rsa/keys/** folder and name it "**MakeOVPN.sh**".

Open it for editing, enter the script shown in the following pages below and save. Next, make the script executable using the command: "**chmod 700 MakeOVPN.sh**".

Finally, execute the script using the command: "**./MakeOVPN.sh**"

WARNING: Cut and Paste from this PDF document **WILL** substitute some characters. Carefully Review all your code for errors.

```
#!/bin/bash

# Default Variable Declarations
DEFAULT="Defaults.txt"
FILEEXT=".ovpn"
CRT=".crt"
KEY=".3des.key"
CA="ca.crt"
TA="ta.key"

#Ask for a Client name
echo "Please enter an existing Client Name:"
read NAME


#1st Verify that client's Public Key Exists
if [ ! -f $NAME$CRT ]; then
    echo "[ERROR]: Client Public Key Certificate not found: $NAME$CRT"
    exit
fi
echo "Client's cert found: $NAME$CR"


#Then, verify that there is a private key for that client
if [ ! -f $NAME$KEY ]; then
    echo "[ERROR]: Client 3des Private Key not found: $NAME$KEY"
    exit
fi
echo "Client's Private Key found: $NAME$KEY"
```

Script continues on the next page…

Eric Jodoin, ejodoin@hotmail.com

WARNING: Cut and Paste from this PDF document **WILL** substitute some characters. Carefully Review all your code for errors.

```
#Confirm the CA public key exists
if [ ! -f $CA ]; then
    echo "[ERROR]: CA Public Key not found: $CA"
    exit
fi
echo "CA public Key found: $CA"

#Confirm the tls-auth ta key file exists
if [ ! -f $TA ]; then
    echo "[ERROR]: tls-auth Key not found: $TA"
    exit
fi
echo "tls-auth Private Key found: $TA"

#Ready to make a new .opvn file - Start by populating with the default file
cat $DEFAULT > $NAME$FILEEXT

#Now, append the CA Public Cert
echo  "<ca>" >> $NAME$FILEEXT
cat  $CA >> $NAME$FILEEXT
echo  "</ca>" >> $NAME$FILEEXT

#Next append the client Public Cert
echo  "<cert>" >> $NAME$FILEEXT
cat $NAME$CRT | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >> $NAME$FILEEXT
echo  "</cert>" >> $NAME$FILEEXT

#Then, append the client Private Key
echo  "<key>" >> $NAME$FILEEXT
cat  $NAME$KEY >> $NAME$FILEEXT
echo  "</key>" >> $NAME$FILEEXT

#Finally, append the TA Private Key
echo  "<tls-auth>" >> $NAME$FILEEXT
cat  $TA >> $NAME$FILEEXT
echo  "</tls-auth>" >> $NAME$FILEEXT

echo "Done! $NAME$FILEEXT Successfully Created."
```

Eric Jodoin, ejodoin@hotmail.com

Dynamic DNS

This appendix will explain how to setup a permanent Internet hostname pointing to your SOHO Network Internet facing IP address using Dynamic DNS.

There are numerous Dynamic DNS Service providers. You can choose whichever one you wish. This example uses DNSdynamic.com because it is both free and compatible with ddclient which runs on Raspbian.

Registration

1. Register online: https://www.dnsdynamic.org/signup.php.
    a. Make sure to use a unique password.
2. Complete the registration process by checking your email.
3. Choose your Host and Domain Name.
4. Record the following information for future use:
    a. Email:_____
    b. Password:_____
    c. Host and Domain Name selected:_____

ddclient Configuration

1. Install ddclient on the RPi using the command:
   `sudo apt-get install ddclient`

2. During installation, the configuration routine will start automatically. When asked to select your Dynamic DNS Service Provider, select: "**`4. Other`**".

3. Dynamic DNS Server: **`www.dnsdynamic.org`**

4. Dynamic DNS Protocol: **`dyndns2`**

5. Username: [Email address used for registration - See Registration 4a.]

6. Password: [Password used for Registration - See Registration 4b.]

7. Network Interface: **`eth0`**

8. Fully Qualified Domain Name: [See Registration 4c.]

9. ddclient assumes it is connected directly to the Internet. Because the RPi is sitting inside your SOHO network, "**/etc/ddclient.conf**" will need to be manually edited. Find the line "**use=if, if=eth0**" and replace it with "**use=web, if=myip.dnsdynamic.org**". Edit the file using the command: **sudo nano /etc/ddclient.conf**

```
# Configuration file for ddclient generated by debconf
#
# /etc/ddclient.conf

protocol=dyndns2
use=if, if=eth0
use=web, if=myip.dnsdynamic.org
server=www.dnsdynamic.org
login=ejodoin@hotmail.com
password=''
MyPiVPN.dnsdynamic.com
```

10. Finally, restart the ddclient service using the command:

    a. **sudo service ddclient restart**

Additional Notes

1. If at any point the ddclient configuration routine needs to be re-executed, use the command: **sudo dpkg-reconfigure ddclient**

2. ddclient supports Internet IP Address discovery directly with routers. For a comprehensive list of routers and instructions, see ddclient documentation[18].

[18] http://sourceforge.net/p/ddclient/wiki/routers/

Self-Signed Certificate Revocation

Revoking a certificate will prevent any device using that certificate to authenticate and use the OpenVPN Server. This is especially useful if a user's private key has been compromised or to block a user that is no longer authorized access. It uses a Certificate Revocation List (CRL) that contains the particulars of every revoked certificate. Whenever a client attempts to connect, OpenVPN reads the CRL and verifies if the client's certificate has been revoked before allowing the connection.

It is important to note that it will be impossible to revoke a certificate without a copy being present in "**/etc/openvpn/easy-rsa/keys**". Also, a new entry will be required in "**/etc/openvpn/server.conf**". It was not included in the initial configuration because of a known bug, which crashes OpenVPN when the CRL is empty (rram). Finally, the CRL must be located in a folder with access rights that are "World Readable" since OpenVPN runs as user "nobody".

Certificate Revocation

1. Log in the RPi as "pi".

2. Root Shell access is required. Use the command:
   sudo bash

3. Go to the easy-rsa directory using the command:
   cd /etc/openvpn/easy-rsa

4. Revoke the certificate using the command: **./revoke-full-pass UserName**

   a. The script will display status information as it progress. Look specifically for "error 23". This indicates that the certificate was successfully revoked.

```
Write out database with 1 new entries
Data Base Updated
root@OpenVPN-Server:/etc/openvpn/easy-rsa# ./revoke-full MrGone
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Revoking Certificate 06.
Data Base Updated
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
MrGone.crt: C = CA, ST = Ontario, L = Ottawa, O = GOLD1, OU =
changeme, CN = UserName, name = I.M. Gone, emailAddress =
MrGone@jodoin.ca
error 23 at 0 depth lookup:certificate revoked
```

5. Verify using the command: **`openssl crl -text -noout -in crl.pem`**

    a. Confirm that **`UserName`** 's certificate has been added to the CRL .

```
Certificate Revocation List (CRL):
        Version 1 (0x0)
    Signature Algorithm: md5WithRSAEncryption
        Issuer:
/C=CA/ST=Ontario/L=Ottawa/O=GOLD1/OU=changeme/CN=UserName/name=I.M.
Gone/emailAddress=MrGone@jodoin.ca
        Last Update: Nov  4 06:16:38 2013 GMT
        Next Update: Dec  4 06:16:38 2013 GMT
Revoked Certificates:
    Serial Number: 06
        Revocation Date: Nov  4 06:16:38 2013 GMT
    Signature Algorithm: md5WithRSAEncryption
         26:60:e7:5a:db:d0:3d:fa:d9:63:d0:52:8e:b0:6e:5d:8e:f3:
         c9:aa:ca:4e:5d:20:71:24:33:7c:77:f6:09:6f:bb:d6:13:5a:
         08:68:4a:d1:39:ab:b7:73:e6:aa:70:75:1d:a6:f9:be:2a:70:
         d0:48:a6:46:9a:36:cb:32:cf:53:78:b5:ec:e9:07:89:a7:8a:
         e7:58:d5:21:01:b7:b6:44:33:36:69:72:02:fc:e7:09:e8:d7:
         93:97:41:13:23:ad:a4:a4:74:7e:38:de:9f:e2:92:67:ab:ac:
         04:8d:18:a0:fa:eb:d1:30:7e:29:39:a6:02:ba:1a:29:55:c2:
         fe:b8
```

Copying to OpenVPN folder

1. Copy the CRL to the OpenVPN folder using the command: **`cp /etc/openvpn/easy-rsa/crl.pem /etc/openvpn/crl.pem`**

2. Make sure the file is world readable using the command: **`chmod 644 /etc/openvpn/crl.pem`**

Configuration change to OpenVPN Server[19]

1. Open "**`/etc/openvpn/server.conf`**" and enter this single line at the bottom:

```
crl-verify /etc/openvpn/crl.pem
```

Activating the new CRL[20] (Optional)

1. Restart the OpenVPN server using the command: **`service openvpn restart`**

[19] This step is only required for the very 1st certificate revoked. It will then persist across reboots. AND, restarting the OpenVPN Server as described in the next section is absolutely required for OpenVPN to start checking the CRL.
[20] Optional Step. OpenVPN will immediately start consulting the new CRL for any new connections. However, a user currently logged in and whose certificate is revoked will not be kicked off unless the OpenVPN service is restarted.

Eric Jodoin, ejodoin@hotmail.com