ADBLOCK AND WEB SECURITY BA.NET MANAGED NETWORK ADMINISTRATOR MANUAL



(c) BA.net/Adblock - May 2015











AdBlock Filter Server Administrator Manual (c) ba.net adblock@ba.net

Chapter 3, 4 and 5 and portions of other chapters part of Linux HowTo's and Linux Guides copyright Linux Documentation Project LDP.

The optional FlashBoot Software Appliance contains software provided by GNU/Linux, Slackware and other providers covered by the GNU General Public License.

| I Introduction | 9 |
|--|------------|
| 1.1.1 AdBlock BA.net | 9 |
| 1.1.2 Configure Aublock DNS for iFhome, iFau | 9 |
| 1.2 Setting up AdBlock BA.net on Android devices | 10 |
| 1.3 Configure DSL or Wireless router settings | 10 |
| 1.3.1 Configure AdBlock DNS for Mac OS X, Windows, or Linux | 12 |
| 1.4 Works with Safari. Any Web Browser / Any Platform | |
| 1.4.1.1 User Benefits | 14 |
| 1.4.1.2 Multi Device | 14 |
| 1.4.1.3 Block Tracking Sites | 14 |
| 1.4.1.4 Block Ads Everywhere | 14 |
| 1.5 Frequently Asked Questions | 15 |
| 1.5.1 Q: How is AdBlock BA.net different from other ad-blocking services ? | 15 |
| 1.5.2 Q: How can it be free ? | 15 |
| 1.5.3 Q: We only use Apple computers in our household, can we still use | 15 |
| 154 O: Will AdBlock BA net work on my iPhone Blackberry or other moh | 13 nile |
| Internet device? | 15 |
| 1.5.5 Q: Will AdBlock BA.net also restrict pornograhic ads? | 15 |
| 1.5.6 Q: I live in Toronto, will AdBlock BA.net work in Canada? | 16 |
| 1.5.7 Q: Does AdBlock BA.net track where I go on the Internet? | 16 |
| 1.5.8 Q: I noticed an ad the other day surfing the web, should I report that | to |
| AdBlock BA.net Support? | 16 |
| 1.5.9 Q. What kind of banners will it block ? | 10 |
| 1.5.10 What about Site and Blog Owners 1 | 17 |
| 1.5.12 Q: Will it consume Memory like uBlock or AdBlock Plus ? | 17 |
| 1.5.13 Q: Do you have a solution for iPhone on Mobile Networks ? | 17 |
| 1.5.14 Q: Will VPN affect my mobile battery ? | 17 |
| 1.5.15 Q: Do you offer Corporate Service ? | 18 |
| 1.5.16 Do you offer Server DNS Filter Solutions ? | 18 |
| 2 Managed security service | 20 |
| 2.1 Early History of Managed Security Services | 21 |
| 2.2 Industry terms | 21 |
| 2.3 Six categories of managed security services | 21 |
| 2.3.1 On-site consulting | 21 |

| 2.3.2 Perimeter management of the client's network | 21 |
|--|-----------------|
| 2.3.3 Product resale | |
| 2.3.4 Managed security monitoring | |
| 2.3.6 Compliance monitoring | |
| 2.3.7 Engaging an MSSP | 23 |
| 2.4 Managed security services for mid-sized and smaller businesses | 23 |
| 3 Filtering methods | 25 |
| 3.1 Benefits of ad filtering | 25 |
| 3.2 Economic consequences for online business | 27 |
| 3.3 Advertiser offensive countermeasures and justifications | 27 |
| 3.4 Browser integration | 28 |
| 3.5 External programs | 29 |
| 3.6 Hosts file | 29 |
| 3.7 DNS cache | 29 |
| 3.8 DNS filtering | 30 |
| 3.9 Ad filtering by external parties and internet providers | 30 |
| 4 BA.net Adblock Filter Server CLOUD | |
| 4.1 Children's Internet Protection Act | 34 |
| 4.1.1.1 Background | |
| 4.1.1.2 What CIPA Requires | 34 |
| 5.0 BA.net AdBlock Speed VPN for iPhone Config 4.1.2 FAQ | <i>36</i> 42 |
| 5 Corporate Proxy auto-config PAC | 44 |
| 5.1 Context | 44 |
| 5.2 Proxy configuration | 44 |
| 5.3 The PAC File | 45 |
| 5.3.1 Limitations | 46 |
| 5.3.1.1 PAC Character-Encoding | |
| 5.3.1.2 DnsResolve | |
| 5.3.1.4 Security | 47 47 |
| 5.3.1.5 Others | |
| 5.3.2 Advanced functionality | |
| 6 BA.NET AdBlock Speed VPN for iPhone | 48 |

| 6.1 VPN connectivity overview | |
|---|-----------|
| 6.2 VPN Type | |
| 6.3 Security mechanisms | |
| 6.3.1 Authentication | |
| 6.4 Routing | |
| 6.4.1 Provider-provisioned VPN building-block | (S |
| 6.4.3 OSI Layer 3 PPVPN architectures | 58 |
| 6.4.4 Unencrypted tunnels | |
| 6.5 Trusted delivery networks | |
| 6.6 VPNs in mobile environments | |
| 6.7 VPN on Routers | |
| 6.8 Networking limitations | |
| 7 How DNS Works | |
| 7.1 Name Lookups with DNS | |
| 7.2 Types of Name Servers | |
| 7.3 The DNS Database | |
| 7.4 Reverse Lookups | |
| 7.4.1 Notes | |
| 8 Running named | 74 |
| 8.1 The named.boot File | |
| 8.2 The BIND 8 host.conf File | |
| 8.3 The DNS Database Files | |
| 8.4 Caching-only named Configuration | |
| 8.5 Writing the Master Files | |
| 8.6 Verifying the Name Server Setup | |
| 8.7 Other Useful Tools | |
| 8.7.1 Notes | |
| 9 BA.net Adblock Filter Server FlashBoot | 91 |
| 9.1.1 HOWTO INSTALL | |
| 10 Configuration Files | 94 |
| 10.1 2 Initial configuration | |
| 10.2 2 Later als and automorphi | 06 |

| 10.3 4 | Security | 99 |
|-----------|---|-----|
| 10.4 5 | Configuration files | |
| 10.4.1 | 5.1 /etc/bind/named.conf.local | 99 |
| 10.4.2 | 5.2 /etc/bind/externals/db.example.com | 100 |
| 10.4.3 | 5.3 /etc/bind/internals/db.example.com | 101 |
| 10.4.4 | Bibliography | 101 |
| 10.4.5 | Footnotes | 101 |
| 11 Domain | n Name Server (DNS) Configuration and Administration | 102 |
| 12 Respon | se Rate Limiting | 106 |
| 12.1 T | he Problem | 106 |
| 12.2 A | Solution | 107 |
| 12.3 T | he Results | 107 |
| 12.4 Se | ample BIND RRL configuration | 107 |
| 13 OpenV | 'PN | 109 |
| 13.1 D | etermining whether to use a routed or bridged VPN | 109 |
| 13.2 N | umbering private subnets | 109 |
| 13.3 Se | etting up your own Certificate Authority (CA) and generating certificates | and |
| keys for | an OpenVPN server and multiple clients | 110 |
| 13.3.1 | Overview | 110 |
| 13.3.2 | Generate the master Certificate Authority (CA) certificate & key | 112 |
| 13.3.3 | Generate certificate & key for server | 113 |
| 13.3.4 | Generate certificates & keys for 3 clients | 114 |
| 13.3.5 | Generate Diffie Hellman parameters | 114 |
| 13.3.6 | Key Files | 116 |
| 13.4 C | reating configuration files for server and clients | 117 |
| 13.4.1 | Getting the sample config files | 117 |
| 13.4.2 | Editing the server configuration file | 117 |
| 13.4.3 | Editing the client configuration files | 119 |
| 13.5 St | tarting up the VPN and testing for initial connectivity | 119 |
| 13.5.1 | Starting the server | 119 |
| 13.5.2 | Starting the client | 121 |
| 13.5.3 | Troubleshooting | 121 |
| 13.6 C | onfiguring OpenVPN to run automatically on system startup | 123 |
| 13.6.1 | Linux | |
| 13.6.2 | Windows | 123 |
| 137 C | ontrolling a running OpenVPN process | 124 |
| 13.7 1 | Running on Linux/BSD/Unix | |
| 13.7.2 | Running on Windows as a GUI | 124 |

| 13.7.3 | Running in a Windows command prompt window | 124 |
|--------------------|--|------------|
| 13.7.4 | Running as a Windows Service | 125 |
| 13.7.5 | Modifying a live server configuration | 125 |
| 13.7.6 | Status File | 125 |
| 13.7.7 | Using the management interface | 126 |
| 13.8 Ex | panding the scope of the VPN to include additional machines on either the server subnet. | ne 127 |
| 13.8.1 tun) | Including multiple machines on the server side when using a routed VP 128 | N (dev |
| 13.8.2 (dev tat | Including multiple machines on the server side when using a bridged V | PN 128 |
| 13.8.3 tun) | Including multiple machines on the client side when using a routed VPN 128 | N (dev |
| 13.8.4 tap) | Including multiple machines on the client side when using a bridged VF 130 | PN (dev |
| 13.9 Pu | shing DHCP options to clients | 131 |
| 13.10 C | Configuring client-specific rules and access policies | 131 |
| 13.10.1 | ccd/sysadmin1 | 133 |
| 13.10.2 | ccd/contractor1 | 133 |
| 13.10.3 | ccd/contractor2 | 133 |
| 13.11 U | Ising alternative authentication methods | 134 |
| 13.11.1 | Using Script Plugins | 135 |
| 13.11.2 | Using Shared Object or DLL Plugins | 135 |
| 13.11.3 | Using username/password authentication as the only form of client | 126 |
| authent | ication | 136 |
| 13.12 H | low to add dual-factor authentication to an OpenVPN configuration using | g |
| client-sid | e smart cards | 137 |
| 13.12.1 | About dual-factor authentication | 137 |
| 13.12.2 | What is $PKCS#11?$ | 138 |
| 13.12.3 | Finding PKCS#11 provider library | 139 |
| 13.12.4 | How to configure cryptographic token | 139 hio |
| 13.12.3 tokons | 140 | nic |
| 13 12 | 140 2.5.1 Determine the correct object | 140 |
| 13.12 | 2.5.2 Using OpenVPN with PKCS#11 | 140 |
| 13.12 | 12.5.2.1 A typical set of OpenVPN options for PKCS#11 | 141 |
| 13. | 12.5.2.2 Advanced OpenVPN options for PKCS#11 | 141 |
| 13.12 | 2.5.3 PKCS#11 implementation considerations | 141 |
| 13.12 | 2.5.4 OpenSC PKCS#11 provider | 141 |
| 13.12.6 | Difference between PKCS#11 and Microsoft Cryptographic API | |
| (Crypto | 0API) | 142 |
| 13.13 R | outing all client traffic (including web-traffic) through the VPN | 142 |
| 13.13.1 | Overview | 142 |

| 13.13.2 13.13.3 | Implementation Caveats | 143 144 |
|--------------------|--|------------|
| 13.14 Ru | nning an OpenVPN server on a dynamic IP address | 144 |
| 13.15 Co | onnecting to an OpenVPN server via an HTTP proxy | 145 |
| 13.16 Co | onnecting to a Samba share over OpenVPN | 146 |
| 13.17 Im | plementing a load-balancing/failover configuration | 147 |
| 13.17.1 | Client Server | 147 |
| 13.18 Ha | urdening OpenVPN Security | 149 |
| 13.18.1 | tls-auth | 149 |
| 13.18.2 | proto udp | 150 |
| 13.18.3 | user/group (non-Windows only) | 150 |
| 13.18.4 | Unprivileged mode (Linux only) | 150 |
| 13.18.5 | chroot (non-Windows only) | 152 |
| 13.18.6 | Larger RSA keys | 152 |
| 13.18.7 | Larger symmetric keys | 152 |
| 13.18.8 | Keep the root key (ca.key) on a standalone machine without a network | 1.52 |
| connecti | on | 153 |
| 13.19 Re | voking Certificates | 153 |
| 13.19.1 | Example | 153 |
| 13.19.2 | CRL Notes | 154 |
| 13.20 Im | portant Note on possible "Man-in-the-Middle" attack if clients do not ve | rify |
| the certific | ate of the server they are connecting to | 155 |
| 13.21 Sa | mple OpenVPN 2.0 configuration files | 157 |
| 13.21.1 | sample-config-files/server.conf | 158 |
| 13.21.2 | sample-config-files/client.conf | 165 |

1 INTRODUCTION

1.1.1 ADBLOCK BA.NET

Make your Internet Faster, more Private and Safer with AdBlock BA.net.

Works on iPad, IOS, Mac, PC. No software to install. It is a network based DNS Service.



Provide Network Security and Control without dedicated staff

<u>104.131.230.202</u>, 104.236.166.10 (demo servers)

1.1.2 CONFIGURE ADBLOCK DNS FOR IPHONE, IPAD

· Open Settings

 $\cdot\,$ Tap on "WiFi" and tap the blue arrow alongside the wi-fi network name you are connected to

Tap the numbers next to "DNS" to change them



Free BA.net AdBlock DNS

104.131.230.202 , 104.236.166.10

Available on the App Store

1.2 SETTING UP ADBLOCK BA.NET ON ANDROID DEVICES

- 1. Settings
- 2. WiFi (click on word "WiFi", not ON/OFF switch)
- 3. Press and hold preferred (or active) wireless network until dialog pops up
- 4. Select "Modify Network"
- 5. Check "Show advanced options" checkbox at the bottom
- 6. Switch "IP settings" to "Static"
- 7. Keep IP address, Gateway and Network prefix length the same (should be set from standard DHCP)

8. Set DNS 1 and DNS 2 fields as per table below:

- 9. "Save"
- 10. Restart your phone (power cycle).

1.3 CONFIGURE DSL OR WIRELESS ROUTER SETTINGS

Depending on your router manufacturer, the steps to configure the router may vary. Following steps are provided for your reference. For more information, you may refer to the support documentation for the router.



- 1. Start your Web browser.
- 2. In the Address box, type the IP address of your router, and then press Enter.

Generally, 192.168.1.1 and 192.168.0.1 are the most widely used default IP addresses by various manufacturers. If your router manufacturer uses a different IP address, refer to the help documentation for the router.

3. Type the administrator user name and password, and then click OK.

Router configuration settings page opens in your Web browser.

4. Find and open the DNS settings.

You may find the DNS settings option under WAN settings.

5. In the Preferred DNS server and Alternate DNS server boxes, type the DNS server addresses provided by your Internet service provider.

Use the following server BA.net AdBlock DNS addresses:

<u>104.131.230.202</u>, <u>104.236.166</u>.

- 6. Click Apply or Save or Save Settings.
- 7. Restart your router to apply the changes.

1.3.1 CONFIGURE ADBLOCK DNS FOR MAC OS X, WINDOWS, OR LINUX

- · Mac OS X
- · <u>Windows</u>
- · <u>Linux</u>
- DSL or Wireless Router
- · Android
- · AdBlock Server

After BA.net/adblock



Before BA.net/adblock



1.4 WORKS WITH SAFARI. ANY WEB BROWSER / ANY PLATFORM

1.4.1.1 User Benefits

The benefits of ad blocking include quicker loading and cleaner looking Web pages free from advertisements, lower resource waste (bandwidth, CPU, memory, etc.), and privacy benefits gained through the exclusion of the tracking and profiling systems of ad delivery platforms.

1.4.1.2 Multi Device

You can configure it for your computers, iPad, iPhone WiFi and more devices on your network. <u>Premium Plans</u> available.

Except on iPhone mobile networks, as DNS can not be changed. It will work on WiFi for iPhone.

1.4.1.3 Block Tracking Sites

Many sites use Web bugs and analytics to track where you go on the Internet. AdBlock BA.net stops these sites from profiling you, invading your privacy and slowing your connection.

1.4.1.4 Block Ads Everywhere

AdBlock BA.net stops advertisements on Safari, Any Web Browser, Any Platform. Also blocks many in-app Ads! Also blocks Ads from appearing in MSN, Yahoo!, and AOL messaging programs at the source. No more annoying pop-up animated ads.

1.5 FREQUENTLY ASKED QUESTIONS

1.5.1 Q: HOW IS ADBLOCK BA.NET DIFFERENT FROM OTHER AD-BLOCKING SERVICES ?

AdBlock BA.net does not require installing any software on your computer and works with any Web browser on any computer.

1.5.2 Q: HOW CAN IT BE FREE ?

AdBlock BA.net is free for personal use. If you like the service we encourage you to <u>Share it with your friends</u>. Or upgrade to a <u>Premium</u> <u>Plan</u>

1.5.3 Q: WE ONLY USE APPLE COMPUTERS IN OUR HOUSEHOLD, CAN WE STILL USE ADBLOCK BA.NET?

Yes, AdBlock BA.net will work with Safari, Firefox, etc. on Apple computers.

1.5.4 Q: WILL ADBLOCK BA.NET WORK ON MY IPHONE, BLACKBERRY OR OTHER MOBILE INTERNET DEVICE?

Yes, although some providers give no option to configure your DNS servers in order to use AdBlock BA.net. (For example it will not work on 3G connections only on WiFi)

1.5.5 Q: WILL ADBLOCK BA.NET ALSO RESTRICT PORNOGRAHIC ADS?

AdBlock BA.net is not specifically designed to remove pornographic ads, however we make every attempt to block ads from known ad distribution networks.

1.5.6 Q: I LIVE IN TORONTO, WILL ADBLOCK BA.NET WORK IN CANADA?

Yes, AdBlock BA.net will work from any location, however we concentrate our efforts on blocking advertising on the most popular sites for U.S. Internet users.

You can report any Ads or Malware to be included in the BA.net/adblock blocked list. <u>Report Ads or Malware Here</u>. So far our blocked list contains over 180.000 domains!

1.5.7 Q: DOES ADBLOCK BA.NET TRACK WHERE I GO ON THE INTERNET?

AdBlock BA.net does not log any personally identifiable information. See details at the <u>Privacy Policy</u>

1.5.8 Q: I NOTICED AN AD THE OTHER DAY SURFING THE WEB, SHOULD I REPORT THAT TO ADBLOCK BA.NET SUPPORT?

AdBlock BA.net will not block 100% of Internet advertising. Our goal is to eliminate banner and Flash advertisements on the most popular sites, and block the most widely used advertising distributors.

You can report any Ads or Malware to be included in the BA.net/adblock blocked list. <u>Report Ads or Malware Here</u>. So far our blocked list contains over 180.000 domains!

1.5.9 Q: WHAT KIND OF BANNERS WILL IT BLOCK ?

AdBlock BA.net is designed to block Banner and Flash advertising on the most popluar sites, and to block ads coming from the largest advertising networks.

AdBlock BA.net will work for anyone, anywhere in the world, but our focus is on popular U.S. Websites. Our servers are located in the US East Coast, US West Coast and EU London.

1.5.10 WHAT ABOUT SITE AND BLOG OWNERS ?

Our intent is to block major adserver networks that track you across the web. They use questionable re-targetting, profiling and invade your privacy.

Smaller sites generally offer sponsorships and directly served ads, which we do not block.

1.5.11 Q: WILL IT BLOCK PHISHING AND MALWARE SITES ?

Yes Phishing and Malware sites will also be blocked. Helping to keep you safe from identity theft. So far our blocked list contains over 180.000 domains!

1.5.12 Q: WILL IT CONSUME MEMORY LIKE UBLOCK OR ADBLOCK PLUS ?

uBlock or other Browser add-ons can consume a lot of memory on your computer. As they have to process over 100k rules and domains to block. Our DNS solution moves that processing to our servers, we block 180k adware and malware domains! You will need less memory and have a faster and safer internet. No need to install any add-on or plug-in, just configure our DNS and you are ready to go.

1.5.13 Q: DO YOU HAVE A SOLUTION FOR IPHONE ON MOBILE NETWORKS ?

Yes, AdBlock Speed VPN for iPhone. Contact us at adblock@ba.net to get a Business Dedicated Adblock VPN Server.

1.5.14 Q: WILL VPN AFFECT MY MOBILE BATTERY ?

Short answer no. A PPTP VPN plus BA.net Adblock DNS is the ideal way to surf faster on iPhone on mobile networks. AdBlock will speed up your connection and PPTP is the vpn protocol that will impact your battery usage the least.

PPTP is not 100% safe as an encryption protocol, but it provides adecuate protection for guest hotspot surfing, and adblock usage. The best feature of PPTP is that it is native to the iphone, it is simple to configure and will add the least amount of overhead to your battery usage.

The AdBlock data transmission and CPU savings will combine with the low overhead of PPTP to a negligible impact on your iPhone battery.

1.5.15 Q: DO YOU OFFER CORPORATE SERVICE ?

Yes, you can point your corporate routers to our Filtering DNS service. We offer volume discounts per user.

Also available custom filter lists and access policies running on your own dedicated virtual dns servers.

Corporate, ISP, school and campus filtering bundles also available.

Get Premium Adblock Multi Device

Business Web Security Solutions

Wikipedia Ad Filtering Docs

Wikipedia VPN Docs

1.5.16 DO YOU OFFER SERVER DNS FILTER SOLUTIONS ?

BA.net Adblock DNS Server Managed is a complete Software Appliance. Built in a simple USB Flash Boot package. <u>Free Download</u>

· Q: Do you offer an Administrator Manual ?

You can download a free preview of the Administrator Manual here



950k pdf

2 MANAGED SECURITY SERVICE

In <u>computing</u>, managed security services (MSS) are <u>network security</u> services that have been <u>outsourced</u> to a <u>service provider</u>. A company providing such a service is a managed security service provider (MSSP)^[1] The roots of MSSPs are in the <u>Internet Service Providers</u> (ISPs) in the mid to late 1990's. Initially ISPs would sell customers a <u>firewall</u> appliance, as <u>customer premises equipment</u> (CPE), and for an additional fee would manage the customer-owned firewall over a dial-up connection.^[2]

According to recent industry research, most organizations (74%) manage IT security in-house, but 82% of IT professionals said they have either already partnered with, or plan to partner with, a managed security service provider.^[3]

Businesses turn to managed security services providers to alleviate the pressures they face daily related to information security such as targeted malware, customer data theft, skills shortages and resource constraints.^[4]

Managed security services (MSS) are also considered the systematic approach to managing an organization's security needs. The services may be conducted in-house or outsourced to a service provider that oversees other companies' network and information system security. Functions of a managed security service include round-the-clock monitoring and management of intrusion detection systems and firewalls, overseeing patch management and upgrades, performing security assessments and security audits, and responding to emergencies. There are products available from a number of vendors to help organize and guide the procedures involved. This diverts the burden of performing the chores manually, which can be considerable, away from administrators.

Industry research firm Forrester Research in late 2014 identified the 13 most significant vendors in the North American market with its 26-criteria evaluation of managed security service providers (MSSPs)--identifying IBM, Dell SecureWorks, Trustwave, AT&T, Verizon and others as the leaders in the MSSP market.^[5]

2.1 EARLY HISTORY OF MANAGED SECURITY SERVICES

An earliest example of a cloud-based MSSP service is <u>US</u> <u>West</u> !NTERACT Internet Security. The security service didn't require the customer to purchase any equipment and no security equipment was installed at the customers premises.^[6] The service is considered a MSSP offering in that US West retained ownership of the firewall equipment and the firewalls were operated from their own Internet <u>Point of Presence</u> (PoP)^[7] The service was based on <u>Check Point</u> Firewall-1 equipment.^[8] Following over a year long beta introduction period, the service was generally available by early 1997.^{[6][7]} The service also offered managed Virtual Private Networking (VPN) encryption security at launch.^[7]

2.2 INDUSTRY TERMS

- Asset: A resource valuable to a company worthy of protection.
- Incident: An assessed occurrence that actually or potentially jeopardizes the confidentiality, integrity, or availability of an asset.
- Alert: Identified information, i.e. fact, used to correlate an incident.

2.3 SIX CATEGORIES OF MANAGED SECURITY SERVICES

2.3.1 ON-SITE CONSULTING

This is customized assistance in the assessment of business risks, key business requirements for security and the development of security policies and processes. It may include comprehensive security architecture assessments and design (include technology, business risks, technical risks and procedures). Consulting may also include security product integration and On-site mitigation support after an intrusion has occurred, including emergency incident response and forensic analysis^{[1][9]}

2.3.2 PERIMETER MANAGEMENT OF THE CLIENT'S NETWORK

This service involves installing, upgrading, and managing the <u>firewall</u>, <u>Virtual Private Network</u> (VPN) and/or intrusion detection hardware and software, <u>electronic mail</u>, and commonly performing configuration changes on behalf of the customer. Management includes monitoring, maintaining the firewall's traffic routing rules, and generating regular traffic and management reports to the customer.^[1] Intrusion detection management, either at the network level or at the individual host level, involves providing intrusion alerts to a customer, keeping up to date with new defenses against intrusion, and regularly reporting on intrusion attempts and activity. Content filtering services may be provided by; such as, email filtering) and other data traffic filtering.^[9]

2.3.3 PRODUCT RESALE

Clearly not a managed service by itself, product resale is a major revenue generator for many MSS providers. This category provides value-added hardware and software for a variety of security-related tasks. One such service that may be provided is archival of customer data.^[9]

2.3.4 MANAGED SECURITY MONITORING

This is the day-to-day monitoring and interpretation of important system events throughout the network—including unauthorized behavior, malicious hacks, <u>denial of service</u> (DoS), anomalies, and trend analysis. It is the first step in an incident response process.

2.3.5 PENETRATION TESTING AND VULNERABILITY ASSESSMENTS

This includes one-time or periodic software scans or hacking attempts in order to find vulnerabilities in a technical and logical perimeter. It generally does not assess security throughout the network, nor does it accurately reflect personnel-related exposures due to disgruntled employees, social engineering, etc. Regularly, reports are given to the client.^{[1][9]}

2.3.6 COMPLIANCE MONITORING

This includes monitoring event logs not for intrusions, but change management. This service will identify changes to a system that violate a formal security policy for example, if a rogue administrator grants himself or herself too much access to a system. In short, it measures compliance to a technical risk model.

2.3.7 ENGAGING AN MSSP

The decision criteria for engaging the services of an MSSP are much the same as those for any other form of outsourcing: cost-effectiveness compared to in-house solutions, focus upon core competencies, need for round-the-clock service, and ease of remaining up-to-date. An important factor, specific to MSS, is that outsourcing network security hands over critical control of the company's infrastructure to an outside party, the MSSP, whilst not relieving the ultimate responsibility for errors. The client of an MSSP still has the ultimate responsibility for its own security, and as such must be prepared to manage and monitor the MSSP, and hold it accountable for the services for which it is contracted. The relationship between MSSP and client is not a turnkey one.^[1]

Although the organization remains responsible for defending its network against information security and related business risks, working with an MSSP allows the organization to focus on its core activities while remaining protected against network vulnerabilities.

Business risks can result when information assets upon which the business depends are not securely configured and managed (resulting in asset compromise due to violations of confidentiality, availability, and integrity). Compliance with specific government-defined security requirements can be achieved by using managed security services.^[10]

2.4 MANAGED SECURITY SERVICES FOR MID-SIZED AND SMALLER BUSINESSES

The business model behind managed security services is commonplace among large enterprise companies with their IT security experts. The model was later adapted to fit medium-sized and smaller companies (SMBs - organizations up to 500 employees, or with no more than 100 employee at any one site) by the value-added reseller (VAR) community, either specializing in managed security or offering it as an extension to their managed IT service solutions. SMBs are increasingly turning to managed security services for a number of reasons. Chief among these are the specialized, complex and highly dynamic nature of IT security and the growing number of regulatory requirements obliging businesses to secure the digital safety and integrity of personal information and financial data held or transferred via their computer networks.

Whereas larger organizations typically employ an IT specialist or department, organizations at a smaller scale such as distributed location businesses, medical or dental offices, attorneys, professional services providers or retailers do not typically employ full-time security specialists, although they frequently employ IT staff or external IT consultants. Of these organizations, many are constrained by budget limitations. To address the combined issues of lack of expertise, lack of time and limited financial resources, an emerging category of managed security service provider for the SMB has arisen.

Services providers in this category tend to offer comprehensive IT security services delivered on remotely managed appliances or devices that are simple to install and run for the most part in the background. Fees are normally highly affordable to reflect financial constraints, and are charged on a monthly basis at a flat rate to ensure predictability of costs. Service providers deliver daily, weekly, monthly or exception-based reporting depending on the client's requirements.^[11]

3 FILTERING METHODS

An extremely common method of filtering is simply to block (or prevent autoplay of) Flash animation or image loading or Windows audio and video files. This can be done in most browsers easily. This crude technological method is refined by numerous <u>browser extensions</u>. Every internet browser handles this task differently, but, in general, one alters the options, preferences or application extensions to filter specific media types. An additional add-on is usually required to differentiate between ads and non-ads using the same technology, or between wanted and unwanted ads or behaviors.

The more advanced filters allow fine-grained control of <u>advertisements</u> through features such as <u>blacklists</u>, <u>whitelists</u>, and <u>regular expression</u> filters. Certain security features also have the effect of disabling some ads. Some <u>antivirus</u> software can act as an ad blocker.

Filtering by intermediaries such as providers or national governments is increasingly common. See below especially re provider ad substitution and national root DNS.

3.1 BENEFITS OF AD FILTERING

To users, the benefits of ad blocking include quicker loading and cleaner looking Web pages free from advertisements, lower resource waste (bandwidth, CPU, memory, etc.), and privacy benefits gained through the exclusion of the tracking and profiling systems of ad delivery platforms. Blocking ads can also save minimal amounts of energy.^[2]

Users who pay for total transferred bandwidth ("capped" or pay-forusage connections) including most mobile users worldwide, have a direct financial benefit from filtering an ad before it is loaded. Streaming audio and video, even if they are not presented to the user interface, can rapidly consume gigabytes of transfer especially on a faster 4G connection. In Canada, where users without a data plan often pay C\$0.50/megabyte (\$500/gigabyte) for at least the first 50-100MB exceeding their data allowance, the cost of tolerating ads can be intolerable. Even fixed connections are often subject to usage limits, especially the faster connections (100Mbit/s and up) which can quickly saturate a network if filled by streaming media. "The extent of unlimited bandwidth plans is often grossly over-estimated by US and European users and advertisers. This problem affects other countries, especially those with bandwidth limitations on their global Internet connections, or those that have poor regulatory or effective monopoly providers."

To advertisers, the benefits include not angering or annoying users into blocking, defaming or boycotting their products or websites. Few advertisers actually intend to anger end users. Very sophisticated filtering and anti-spam techniques can involve active defenses which can shut down an advertiser's domains or brokers, ban them from searches or target them for other countermeasures. Some countries have even considered banning the use of certain ports, e.g. South Korea's proposed ban on port 25 used by SMTP.^[3] Future countermeasures would be likely to include bans on ads South Koreans are unlikely to want or even ad brokering services. Ad substituting is also a legal and common practice already, for instance in Canadian cable TV where regulations permit showing a Canadian channel with Canadian ads instead of a US channel with US ads. where both are broadcasting the show simultaneously - this practice has spread to the web with some cable Internet providers uniformly substituting foreign ads for local ones, for which they receive a share of the revenue. Avoiding national, provider or technological interference with their ads is a priority for advertisers and especially brokers of advertising, to whom it could be fatal.

3.2 ECONOMIC CONSEQUENCES FOR ONLINE BUSINESS

One consequence of widespread ad blocking is decreased revenue to a website sustained by advertisements,^[4] where this blocking can be detected.

A number of website operators, who use online advertisements to fund the hosting of their websites, argue that the use of ad-blocking software risks cutting off their revenue stream. While some websites have successfully implemented subscription and membership based systems for revenue, the majority of websites today rely on online advertising to function.

3.3 ADVERTISER OFFENSIVE COUNTERMEASURES AND JUSTIFICATIONS

Some websites have taken counter-measures against ad-blocking software, such as attempting to detect the presence of ad blockers and informing users of their views, or outright preventing users from accessing the content unless they disable the ad-blocking software. There have been several arguments supporting^[5] and opposing^[6] the assertion that blocking ads is wrong.^[7]

3.4 BROWSER INTEGRATION



Wikitravel with and without Adblock Plus

Almost all modern web browsers block unsolicited pop-up ads by default. Opera, Konqueror, Maxthon 2, and Internet Explorer 8^[8] also include content filtering, which prevents external files such as images or JavaScript files from loading. Content filtering can be added to Mozilla Firefox and related browsers with Adblock Plus, and a number of sources provide regularly updated filter lists. For Internet Explorer there are several add-ons available like Simple Adblock, and Quero that also allows users to temporarily unblock blocked content. A rudimentary content blocking feature is integrated in Opera and does not require an add-on. For Google Chrome, which has had extensions available since v2.0, extensions are available, such as AdBlock, AdSweep, FlashBlock, AdBlock Plus and AdBlockforChrome. Another method for filtering advertisements uses <u>CSS</u> rules to hide specific HTML and XHTML elements.

3.5 EXTERNAL PROGRAMS

A number of external applications offer ad filtering as a primary or additional feature. A traditional solution is to customize an HTTP proxy (or web proxy) to filter content. These programs work by caching and filtering content before it is displayed in a user's browser. This provides an opportunity to remove not only ads but also content which may be offensive, inappropriate, or simply junk. Popular proxy software which blocks content effectively include AdGuard, Privoxy, Squid, Ad Muncher and Diladele Web Safety. The main advantage of the method is freedom from implementation limitations (browser, working techniques) and centralization of control (the proxy can be used by many users). The major drawback is that the proxy sees only raw content and thus it's difficult to handle JavaScript-generated content.

3.6 HOSTS FILE

Further information: hosts file.

This method exploits the fact that most operating systems store a file with IP address, domain name pairs which is consulted by most browsers before using a DNS server to look up a domain name. By assigning the <u>loopback address</u> to each known <u>ad server</u>, the user directs traffic intended to reach each <u>ad server</u> to the local machine. Running a suitable web server locally the ad content can be replaced with anything the user wishes.

3.7 DNS CACHE

This method operates by filtering and changing records of a DNS cache. On most operating systems the domain name resolution always goes via DNS cache. By changing records within the cache or preventing records from entering the cache, programs are allowed or prevented from accessing domain names. The external programs like Portable DNS Cache and Firewall ^[9] monitor internal DNS cache and import DNS records from a file. As a part of the domain name resolution process, a DNS cache lookup is performed before contacting a DNS server. Thus its records take precedence over DNS server queries. Unlike the method of modifying a Hosts file, this

method is more flexible as it uses more comprehensive data available from DNS cache records.

3.8 DNS FILTERING

Advertising can be blocked by using a DNS server which is configured to block access to domains or hostnames which are known to serve ads.^[10]

Morally, while some argue that domain name holders are owners of property (and have been found to have such rights in most developed countries), it has also been one of the web's most basic features that DNS can be localized and run on client, LAN, provider and national services. China, for instance, runs, its own root DNS and the EU has considered the same. Google has required their Google Public DNS be used for some applications on its Android devices. Accordingly, DNS addresses / domains used for advertising may be extremely vulnerable to a broad form of ad substitution whereby a domain that serves ads is entirely swapped out with one serving more local ads to some subset of users. This is especially likely in countries, notably Russia, India and China, where advertisers often refuse to pay for clicks or page views. DNS-level blocking of domains for non-commercial reasons is already common in China.^[11]

3.9 AD FILTERING BY EXTERNAL PARTIES AND INTERNET PROVIDERS

Internet providers, especially mobile operators frequently offer proxies designed to reduce network traffic. Even when not targeted at ad filtering specifically these will block many types of advertisements that are too large, bandwidth consuming or otherwise deemed unsuited for the specific internet connection or target device.

4 BA.NET ADBLOCK FILTER SERVER CLOUD

BA.net Adblock DNS and VPN Filter Dedicated Server - <u>free demo service</u>

Easy to use Content Filtering, AdBlock and Malware Protection for Businesses of all sizes.

- · Make your Internet Faster and Safer with free AdBlock BA.net.
- · Blocks Ads on any application on your network
- · Blocks Malware, Tracking Sites
- Custom Corporate Blocked Sites List
- Suports iPhone on Mobile Networks using VPN
- · Fully Cloud Hosted Solution
- · Business Internet Monitoring and Control
- Full Technical Support, DNS Filter List Update and Monitoring Available
- · Servers root access
- · AdBlock Administrator Manual Preview

Free On-Line Demo Service BA.net/adblock

Adblock DNS Filter Server FlashBoot Free Download

free Administrator Manual E-Book Preview Free 950k PDF

AdBlock and Web Security Administrator Manual E-Book. 100 Pages, 30 Ilustrations, Step by Step Administration Examples Buy with Paypal \$79.90



http://ba.r

Business, Library or School Security

- · Custom policy blocklist
- · CIPA compliance
- · 2 dedicated dns filter servers
- Root access
- Unlimited devices
- · \$34.99 / Month
- Buy with PayPal

AdBlock Speed VPN iPhone

- · Custom policy blocklist
- · CIPA compliance
- · iPhone support on mobile networks
- · Android support on mobile networks
- · 2 dedicated VPN filter servers
- · Root access
- · Unlimited devices
- · \$44.99 / Month
- Buy with PayPal Search

BA.net Adblock DNS Filter Server CIPA Compliance K12 Educational Discounts Contact Us adblock@ba.net



Contact us at 1 (206) 456-1449 adblock@ba.net









4.1 CHILDREN'S INTERNET PROTECTION ACT

4.1.1.1 Background

The Children's Internet Protection Act (CIPA) was enacted by Congress in 2000 to address concerns about children's access to obscene or harmful content over the Internet. CIPA imposes certain requirements on schools or libraries that receive discounts for Internet access or internal connections through the E-rate program – a program that makes certain communications services and products more affordable for eligible schools and libraries. In early 2001, the FCC issued rules implementing CIPA and provided updates to those rules in 2011.

4.1.1.2 What CIPA Requires

Schools and libraries subject to CIPA may not receive the discounts offered by the E-rate program unless they certify that they have an Internet safety policy that includes technology protection measures. The protection measures must block or filter Internet access to pictures that are: (a) obscene; (b) child pornography; or (c) harmful to minors (for computers that are accessed by minors). Before adopting this Internet safety policy, schools and libraries must provide reasonable notice and hold at least one public hearing or meeting to address the proposal.

Schools subject to CIPA have two additional certification requirements: 1) their Internet safety policies must include monitoring the online activities of minors; and 2) as required by the Protecting Children in the 21st Century Act, they must provide for educating minors about appropriate online behavior, including interacting with other individuals on social networking websites and in chat rooms, and cyberbullying awareness and response.

Schools and libraries subject to CIPA are required to adopt and implement an Internet safety policy addressing:

(a) access by minors to inappropriate matter on the Internet;

(b) the safety and security of minors when using electronic mail, chat rooms and other forms of direct electronic communications;

(c) unauthorized access, including so-called "hacking," and other unlawful activities by minors online;

(d) unauthorized disclosure, use, and dissemination of personal information regarding minors; and

(e) measures restricting minors' access to materials harmful to them.

Schools and libraries must certify they are in compliance with CIPA before they can receive E-rate funding.

- CIPA does not apply to schools and libraries receiving discounts only for telecommunications service only;
- An authorized person may disable the blocking or filtering measure during use by an adult to enable access for bona fide research or other lawful purposes.
- CIPA does not require the tracking of Internet use by minors or adults.

You can find out more about CIPA or apply for E-rate funding by contacting the <u>Universal Service Administrative Company's (USAC)</u> Schools and Libraries Division (SLD). 5.0 BA.NET ADBLOCK SPEED VPN FOR IPHONE CONFIG

1. Go to the SETTINGS



2. Go to General > VPN
| BA.NET ADBLOCK ADMINISTRATOR MANUAL | | | | |
|-------------------------------------|--------------------|----------|--|--|
| ●●●●○ MD MOLDCELL | २ 12:39 | D | | |
| Settings | General | | | |
| Restrictions | On | > | | |
| | | | | |
| Date & Time | | > | | |
| Keyboard | | > | | |
| International | | > | | |
| | | | | |
| iTunes Wi-Fi S | ync | > | | |
| VPN | Not Configured | > | | |
| Profile | TestFlight WebClip | > | | |
| | | | | |
| Reset | | > | | |
| | | | | |
| | 5/ | | | |

3. Click on Add VPN Configuration

| | BA.NET ADBLOCK ADMINIST | RATOR MANUAL | |
|------------|-------------------------|--------------|--|
| | CELL ᅙ 12:40 | ◙∦76%■ | |
| General | VPN | | |
| | | | |
| VPN | | | |
| | | | |
| CHOOSE A C | ONFIGURATION | | |
| Add VP | N Configurati | on > | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

- 4. Select the PPTP VPN protocol at the top.
- For Description add any name you want ex: BA.net

 In the Server field type the vpn server name we emailed you. Example vpn12.ba.net.

- For Account enter your VPN username.
- For Password enter your VPN password.
- Encryption level let it "Auto".
- Enable "Send All Traffic"

6. Click Save on the top right corner to save VPN configuration.

| Cancel Ac | dd Configurati | on Save |
|--------------|----------------|------------|
| L2TP | PPTP | IPSec |
| | | |
| Descriptior | Required | |
| Server | Required | |
| Account | Required | |
| RSA Securl | D | \bigcirc |
| Password | Ask Every Ti | me |
| Encryption | Level | Auto > |
| Send All Tra | affic | |
| | | |
| PROXY | | |
| | | |

7. You can now connect to the VPN.

To check if your IP is changed successfully open the Safari browser and go to <u>http://ba.net/util/geo/</u>.

Done. You are connected. Enjoy BA.net AdBlock VPN

4.1.2 FAQ

· Q: Do you have a solution for iPhone on Mobile Networks ?

Yes, AdBlock Speed VPN for iPhone. Speed PPTP VPN. Premium Unlimited AdBlock Servers. Available on dedicated Business Servers Only. Contact us at adblock@ba.net

• Q: Will VPN affect my mobile battery ?

Short answer no. A PPTP VPN plus BA.net Adblock DNS is the ideal way to surf faster on iPhone on mobile networks. AdBlock will speed up your connection and PPTP is the vpn protocol that will impact your battery usage the least.

PPTP is not 100% safe as an encryption protocol, but it provides adecuate protection for guest hotspot surfing,

and adblock usage. The best feature of PPTP is that it is native to the iphone, it is simple to configure and will add the least amount of overhead to your battery usage.

The AdBlock data transmission and CPU savings will combine with the low overhead of PPTP to a negligible impact on your iPhone battery.

· Q: Do you support L2PT ?

Yes, it is also available.

BA.NET ADBLOCK ADMINISTRATOR MANUAL

5 CORPORATE PROXY AUTO-CONFIG PAC

A proxy auto-config (PAC) file defines how <u>web browsers</u> and other <u>user agents</u> can automatically choose the appropriate <u>proxy server</u> (access method) for fetching a given <u>URL</u>.

A PAC file contains a <u>JavaScript function</u> "FindProxyForURL(url, host)". This function returns a string with one or more access method specifications. These specifications cause the user agent to use a particular proxy server or to connect directly.

Multiple specifications provide a fall-back when a proxy fails to respond. The browser fetches this PAC file before requesting other URLs. The URL of the PAC file is either configured manually or determined automatically by the <u>Web Proxy Autodiscovery Protocol</u>.

5.1 CONTEXT

Modern web browsers implement several levels of automation; users can choose the level that is appropriate to their needs. The following methods are commonly implemented:

- Automatic proxy selection: Specify a host-name and a port number to be used for all URLs. Most browsers allow you to specify a list of domains (such as localhost) that will bypass this proxy.
- Proxy auto-configuration (PAC): Specify the URL for a PAC file with a JavaScript function that determines the appropriate proxy for each URL. This method is more suitable for laptop users who need several different proxy configurations, or complex corporate setups with many different proxies.
- Web Proxy Autodiscovery Protocol (WPAD): Let the browser guess the location of the PAC file through <u>DHCP</u> and <u>DNS</u> lookups.

5.2 PROXY CONFIGURATION

Computer <u>operating systems</u> (e.g., <u>Microsoft Windows</u>, <u>Mac OS X</u>, <u>Linux</u>) require a number of settings to communicate over the <u>Internet</u>.

These settings are typically obtained from an <u>Internet Service Provider</u> (ISP). Either anonymous (proxy to use a <u>proxy server</u>) or real settings may be used to establish a network connection.

5.3 THE PAC FILE

The Proxy auto-config file format was originally designed by <u>Netscape</u> in 1996 for the <u>Netscape Navigator 2.0^[1]</u> and is a <u>text file</u> that defines at least one JavaScript function, FindProxyForURL(url, host), with two arguments: url is the URL of the object and host is the host-name derived from that URL. By convention, the PAC file is normally named proxy.pac. The <u>WPAD standard</u> uses wpad.dat.

To use it, a PAC file is published to a <u>HTTP server</u>, and client user agents are instructed to use it, either by entering the URL in the proxy connection settings of the browser or through the use of the WPAD protocol.

Even though most clients will process the script regardless of the <u>MIME</u> <u>type</u> returned in the <u>HTTP reply</u>, for the sake of completeness and to maximize compatibility, the HTTP server should be configured to declare the MIME type of this file to be either application/x-ns-proxy-autoconfig or application/x-javascript-config.

There is little evidence to favor the use of one MIME type over the other. It would be, however, reasonable to assume that application/x-ns-proxy-autoconfig will be supported in more clients than application/x-javascript-config as it was defined in the original Netscape specification, the latter type coming into use more recently.

A very simple example of a PAC file is:

function FindProxyForURL(url, host) {
 return "PROXY
proxy.example.com:8080; DIRECT"; }

This function instructs the browser to retrieve all pages through the proxy on <u>port</u> 8080 of the server proxy.example.com. Should this proxy fail to respond, the browser contacts the Web-site directly, without using a proxy. The latter may fail if <u>firewalls</u>, or other intermediary

network devices, reject requests from sources other than the proxy; a common configuration in corporate networks.

A more complicated example demonstrates some available JavaScript functions to be used in the FindProxyForURL function:

// our local URLs from the function FindProxyForURL(url, host) { domains below example.com don't need a proxy: if (shExpMatch(host, "*.example.com")) return ł "DIRECT": // URLs within this network are accessed through } // port 8080 on fastproxy.example.com: if (isInNet(host, "10.0.0.0", "255.255.248.0")) { return "PROXY fastproxy.example.com:8080"; } // All other requests go through port 8080 of proxy.example.com. // should that fail to return "PROXY respond, go directly to the WWW: proxy.example.com:8080; DIRECT"; }

5.3.1 LIMITATIONS

5.3.1.1 PAC Character-Encoding

Browsers, such as <u>Mozilla Firefox</u> and <u>Internet Explorer</u>, support only system default <u>encoding</u> PAC files, ^[citation needed] and cannot support <u>Unicode</u> encodings, such as <u>UTF-8</u>.^[citation needed]

5.3.1.2 DnsResolve

The function dnsResolve (and similar other functions) performs a <u>DNS</u> lookup that can block your browser for a long time if the DNS server does not respond.

Caching of proxy auto-configuration results by domain name in Microsoft's <u>Internet Explorer</u> 5.5 or newer limits the flexibility of the PAC standard. In effect, you can choose the proxy based on the domain name, but not on the path of the URL. Alternatively, you need to disable caching of proxy auto-configuration results by editing the registry, a process described by de Boyne Pollard (listed in <u>further</u> reading).

It is recommended to always use <u>IP addresses</u> instead of host domain names in the isInNet function for compatibility with other Windows

components which make use of the Internet Explorer PAC configuration, such as <u>.NET 2.0 Framework</u>. For example,

if (isInNet(host, dnsResolve(sampledomain), "255.255.248.0")) // .NET 2.0 will resolve proxy properly if (isInNet(host, sampledomain, "255.255.248.0")) // .NET 2.0 will not resolve proxy properly

The current convention is to fail over to direct connection when a PAC file is unavailable.

Shortly after switching between network configurations (e.g. when entering or leaving a VPN), dnsResolve may give outdated results due to DNS caching.

For instance, Firefox usually keeps 20 domain entries cached for 60 seconds. This may be configured via the network.dnsCacheEntries and network.dnsCacheExpiration configuration variables. Flushing the system's <u>DNS cache</u> may also help, which can be achieved e.g. in Linux with sudo service dns-clean start.

5.3.1.3 mylpAddress

The mylpAddress function has often been reported to give incorrect or unusable results, e.g. 127.0.0.1, the IP address of the localhost. It may help to remove on the system's host file (e.g. /etc/hosts on Linux) any lines referring to the machine host-name, while the line 127.0.0.1 localhost can, and should, stay.

On Internet Explorer 9, isInNet("localHostName", "second.ip", "255.255.255.255") returns true and can be used as a workaround.

The mylpAddress function assumes that the device has a single IPv4 address. The results are undefined if the device has more than one IPv4 address or has IPv6 addresses.

5.3.1.4 Security

In 2013, researchers began warning about the security risks of proxy auto-config.^[2] The threat involves using a PAC to redirect the victim's browser traffic to an attacker-controlled server instead.

5.3.1.5 Others

Further limitations are related to the <u>JavaScript engine</u> on the local machine.

5.3.2 ADVANCED FUNCTIONALITY

More advanced PAC files can reduce load on proxies, perform load balancing, fail over, or even <u>black/white listing</u> before the request is sent through the network. One can return multiple proxies:

return "PROXY proxy1.example.com:8080; PROXY proxy2.example.com:8080";

6 BA.NET ADBLOCK SPEED VPN FOR IPHONE



6.1 VPN CONNECTIVITY OVERVIEW

A virtual private network (VPN) extends a private network across a public network, such as the Internet. It enables a computer or network-enabled device to send and receive data across shared or public networks as if it were directly connected to the private network, while benefiting from the functionality, security and management policies of the public network.^[1] A VPN is created by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryption. Major implementations of VPNs include OpenVPN and IPsec.

A VPN connection across the Internet is similar to a wide area network (WAN) link between websites. From a user perspective, the extended network resources are accessed in the same way as resources available within the private network.^[2] One major limitation of traditional VPNs is that they are point-to-point, and do not tend to support or connect broadcast domains. Therefore communication, software, and networking, which are based on layer 2 and broadcast packets, such as NetBIOS used in Windows networking, may not be fully supported or work exactly as they would on a real LAN. Variants on VPN, such as Virtual Private LAN Service (VPLS), and layer 2 tunneling protocols, are designed to overcome this limitation.

VPNs allow employees to securely access their company's intranet while traveling outside the office. Similarly, VPNs securely connect geographically separated offices of an organization, creating one cohesive network. VPN technology is also used by individual Internet users to secure their wireless transactions, to circumvent geo restrictions and censorship, and to connect to proxy servers for the purpose of protecting personal identity and location.

6.2 VPN TYPE

Early data networks allowed VPN-style remote connectivity through dial-up modems or through leased line connections utilizing Frame Relay and Asynchronous Transfer Mode (ATM) virtual circuits, provisioned through a network owned and operated by telecommunication carriers. These networks are not considered true VPNs because they passively secure the data being transmitted by the creation of logical data streams.^[3] They have been replaced by VPNs based on IP and IP/Multiprotocol Label Switching (MPLS) Networks, due to significant cost-reductions and increased bandwidth^[4] provided by new technologies such as Digital Subscriber Line (DSL)^[5] and fiber-optic networks.

VPNs can be either remote-access (connecting a computer to a network) or site-to-site (connecting two networks). In a corporate setting, remote-access VPNs allow employees to access their company's intranet from home or while traveling outside the office, and site-to-site VPNs allow employees in geographically disparate offices to share one cohesive virtual network. A VPN can also be used to interconnect two similar networks over a dissimilar middle network; for example, two IPv6 networks over an IPv4 network.^[6]

VPN systems may be classified by:

- · The protocols used to tunnel the traffic
- The tunnel's termination point location, e.g., on the customer edge or network-provider edge
- Whether they offer site-to-site or network-tonetwork connectivity
- The levels of security provided
- The OSI layer they present to the connecting network, such as Layer 2 circuits or Layer 3 network connectivity

6.3 SECURITY MECHANISMS

VPNs cannot make online connections completely anonymous, but they can usually increase privacy and security.^[7] To prevent disclosure of private information, VPNs typically allow only authenticated remote access and make use of encryption techniques.

VPNs provide security by the use of tunneling protocols and often through procedures such as encryption. The VPN security model provides:

- Confidentiality such that even if the network traffic is sniffed at the packet level (see network sniffer and Deep packet inspection), an attacker would only see encrypted data
- Sender authentication to prevent unauthorized users from accessing the VPN
- Message integrity to detect any instances of tampering with transmitted messages

Secure VPN protocols include the following:

 Internet Protocol Security (IPsec) as initially developed by the Internet Engineering Task Force (IETF) for IPv6, which was required in all standards-compliant implementations of IPv6 before <u>RFC 6434</u> made it only a recommendation.^[8] This standards-based security protocol is also widely used with IPv4 and the Layer 2 Tunneling Protocol. Its design meets most security goals: authentication, integrity, and confidentiality. IPsec uses encryption, encapsulating an IP packet inside an IPsec packet. De-encapsulation happens at the end of the tunnel, where the original IP packet is decrypted and forwarded to its intended destination.

- Transport Layer Security (SSL/TLS) can tunnel an entire network's traffic (as it does in the OpenVPN project and SoftEther VPN project^[9]) or secure an individual connection. A number of vendors provide remote-access VPN capabilities through SSL. An SSL VPN can connect from locations where IPsec runs into trouble with Network Address Translation and firewall rules.
- Datagram Transport Layer Security (DTLS) used in Cisco AnyConnect VPN and in OpenConnect VPN^[10] to solve the issues SSL/TLS has with tunneling over UDP.
- Microsoft Point-to-Point Encryption (MPPE) works with the Point-to-Point Tunneling Protocol and in several compatible implementations on other platforms.
- Microsoft Secure Socket Tunneling Protocol (SSTP) tunnels Point-to-Point Protocol (PPP) or Layer 2 Tunneling Protocol traffic through an SSL 3.0 channel. (SSTP was introduced in Windows Server 2008 and in Windows Vista Service Pack 1.)

- Multi Path Virtual Private Network (MPVPN).
 Ragula Systems Development Company owns the registered trademark "MPVPN".^[11]
- Secure Shell (SSH) VPN OpenSSH offers VPN tunneling (distinct from port forwarding) to secure remote connections to a network or to internetwork links. OpenSSH server provides a limited number of concurrent tunnels. The VPN feature itself does not support personal authentication.

6.3.1 AUTHENTICATION

Tunnel endpoints must be authenticated before secure VPN tunnels can be established. User-created remoteaccess VPNs may use passwords, biometrics, twofactor authentication or other cryptographic methods. Network-to-network tunnels often use passwords or digital certificates. They permanently store the key to allow the tunnel to establish automatically, without intervention from the user.

$6.4\,{\rm routing}$

Tunneling protocols can operate in a point-to-point network topology that would theoretically not be considered a VPN, because a VPN by definition is expected to support arbitrary and changing sets of network nodes. But since most router implementations support a software-defined tunnel interface, customerprovisioned VPNs often are simply defined tunnels running conventional routing protocols.

6.4.1 PROVIDER-PROVISIONED VPN BUILDING-BLOCKS

Depending on whether a provider-provisioned VPN (PPVPN)^[clarification needed] operates in layer 2 or layer 3, the building blocks described below may be L2 only, L3 only, or combine them both. Multiprotocol label switching (MPLS) functionality blurs the L2-L3 identity.^{[citation needed][original research?]}

RFC 4026 generalized the following terms to cover L2 and L3 VPNs, but they were introduced in RFC 2547.^[15] More information on the devices below can also be found in Lewis, Cisco Press.^[16]

Customer (C) devices

A device that is within a customer's network and not directly connected to the service provider's network. C devices are not aware of the VPN.

Customer Edge device (CE)

A device at the edge of the customer's network which provides access to the PPVPN. Sometimes it's just a demarcation point between provider and customer responsibility. Other providers allow customers to configure it.

Provider edge device (PE)

A PE is a device, or set of devices, at the edge of the provider network which connects to customer networks through CE devices and presents the provider's view of the customer site. PEs are aware of the VPNs that connect through them, and maintain VPN state.

Provider device (P)

A P device operates inside the provider's core network and does not directly interface to any customer endpoint. It might, for example, provide routing for many provider-operated tunnels that belong to different customers' PPVPNs. While the P device is a key part of implementing PPVPNs, it is not itself VPN-aware and does not maintain VPN state. Its principal role is allowing the service provider to scale its PPVPN offerings, for example, by acting as an aggregation point for multiple PEs. P-to-P connections, in such a role, often are high-capacity optical links between major locations of providers.

User-visible PPVPN service

This section deals with the types of VPN considered in the IETF.

6.4.2 OSI LAYER 2 SERVICES

Virtual LAN

A Layer 2 technique that allow for the coexistence of multiple LAN broadcast domains, interconnected via trunks using the IEEE 802.1Q trunking protocol. Other trunking protocols have been used but have become obsolete, including Inter-Switch Link (ISL), IEEE 802.10 (originally a security protocol but a subset was introduced for trunking), and ATM LAN Emulation (LANE).

Virtual private LAN service (VPLS)

Developed by IEEE, VLANs allow multiple tagged LANs to share common trunking. VLANs frequently comprise only customer-owned facilities. Whereas VPLS as described in the above section (OSI Layer 1 services) supports emulation of both point-to-point and point-to-multipoint topologies, the method discussed here extends Layer 2 technologies such as 802.1d and 802.1q LAN trunking to run over transports such as Metro Ethernet.

As used in this context, a VPLS is a Layer 2 PPVPN, rather than a private line, emulating the full functionality of a traditional local area network (LAN). From a user standpoint, a VPLS makes it possible to interconnect several LAN segments over a packet-switched, or optical, provider core; a core transparent to the user, making the remote LAN segments behave as one single LAN.^[17]

In a VPLS, the provider network emulates a learning bridge, which optionally may include VLAN service.

Pseudo wire (PW)

PW is similar to VPLS, but it can provide different L2 protocols at both ends. Typically, its interface is a WAN protocol such as Asynchronous Transfer Mode or

Frame Relay. In contrast, when aiming to provide the appearance of a LAN contiguous between two or more locations, the Virtual Private LAN service or IPLS would be appropriate.

Ethernet over IP tunneling

EtherIP (RFC 3378) is an Ethernet over IP tunneling protocol specification. EtherIP has only packet encapsulation mechanism. It has no confidentiality nor message integrity protection. EtherIP was introduced in the FreeBSD network stack ^[18] and the SoftEther VPN^[19] server program.

IP-only LAN-like service (IPLS)

A subset of VPLS, the CE devices must have L3 capabilities; the IPLS presents packets rather than frames. It may support IPv4 or IPv6.

6.4.3 OSI LAYER 3 PPVPN ARCHITECTURES

This section discusses the main architectures for PPVPNs, one where the PE disambiguates duplicate addresses in a single routing instance, and the other, virtual router, in which the PE contains a virtual router instance per VPN. The former approach, and its variants, have gained the most attention.

One of the challenges of PPVPNs involves different customers using the same address space, especially the IPv4 private address space.^[20] The provider must

be able to disambiguate overlapping addresses in the multiple customers' PPVPNs.

BGP/MPLS PPVPN

In the method defined by <u>RFC 2547</u>, BGP extensions advertise routes in the IPv4 VPN address family, which are of the form of 12-byte strings, beginning with an 8byte Route Distinguisher (RD) and ending with a 4-byte IPv4 address. RDs disambiguate otherwise duplicate addresses in the same PE.

PEs understand the topology of each VPN, which are interconnected with MPLS tunnels, either directly or via P routers. In MPLS terminology, the P routers are Label Switch Routers without awareness of VPNs.

Virtual router PPVPN

The Virtual Router architecture,^{[21][22]} as opposed to BGP/MPLS techniques, requires no modification to existing routing protocols such as BGP. By the provisioning of logically independent routing domains, the customer operating a VPN is completely responsible for the address space. In the various MPLS tunnels, the different PPVPNs are disambiguated by their label, but do not need routing distinguishers.

6.4.4 UNENCRYPTED TUNNELS

Main article: Tunneling protocol

Some virtual networks may not use encryption to protect the privacy of data. While VPNs often provide security, an unencrypted overlay network does not neatly fit within the secure or trusted categorization. For example, a tunnel set up between two hosts that used Generic Routing Encapsulation (GRE) would in fact be a virtual private network, but neither secure nor trusted.

Native plaintext tunneling protocols include Layer 2 Tunneling Protocol (L2TP) when it is set up without IPsec and Point-to-Point Tunneling Protocol (PPTP) or Microsoft Point-to-Point Encryption (MPPE).

6.5 trusted delivery networks

Trusted VPNs do not use cryptographic tunneling, and instead rely on the security of a single provider's network to protect the traffic.

- Multi-Protocol Label Switching (MPLS) often overlays VPNs, often with quality-of-service control over a trusted delivery network.
- Layer 2 Tunneling Protocol (L2TP)^[24] which is a standards-based replacement, and a compromise taking the good features from each, for two proprietary VPN protocols: Cisco's Layer 2 Forwarding (L2F)^[25] (obsolete as of 2009) and Microsoft's Point-to-Point Tunneling Protocol (PPTP).^[26]

From the security standpoint, VPNs either trust the underlying delivery network, or must enforce security with mechanisms in the VPN itself. Unless the trusted delivery network runs among physically secure sites only, both trusted and secure models need an authentication mechanism for users to gain access to the VPN.

6.6 VPNS IN MOBILE ENVIRONMENTS

Main article: Mobile virtual private network

Mobile VPNs are used in a setting where an endpoint of the VPN is not fixed to a single IP address, but instead roams across various networks such as data networks from cellular carriers or between multiple Wi-Fi access points.^[27] Mobile VPNs have been widely used in public safety, where they give law enforcement officers access to mission-critical applications, such as computer-assisted dispatch and criminal databases, while they travel between different subnets of a mobile network.^[28] They are also used in field service management and by healthcare organizations,^[29]

Increasingly, mobile VPNs are being adopted by mobile professionals who need reliable connections.^[29] They are used for roaming seamlessly across networks and in and out of wireless coverage areas without losing application sessions or dropping the secure VPN session. A conventional VPN cannot survive such events because the network tunnel is disrupted, causing applications to disconnect, time out,^[27] or fail, or even cause the computing device itself to crash.^[29]

Instead of logically tying the endpoint of the network tunnel to the physical IP address, each tunnel is bound to a permanently associated IP address at the device. The mobile VPN software handles the necessary network authentication and maintains the network sessions in a manner transparent to the application and the user.^[27] The Host Identity Protocol (HIP), under study by the Internet Engineering Task Force, is designed to support mobility of hosts by separating the role of IP addresses for host identification from their locator functionality in an IP network. With HIP a mobile host maintains its logical connections established via the host identify identifier while associating with different IP addresses when roaming between access networks.

6.7 VPN ON ROUTERS

With the increasing use of VPNs, many have started deploying VPN connectivity on routers for additional security and encryption of data transmission by using various cryptographic techniques. Setting up VPN services on a router will allow any connected device(s) to use the VPN network while it is enabled. This also makes it easy to set up VPNs on devices that do not have native VPN clients such as Smart-TVs, Gaming Consoles etc. Provisioning VPN on the routers will also help in cost savings and network scalability. Many router manufacturers like Cisco Linksys, Asus and Netgear supply their routers with built-in VPN clients. Since these routers do not support all the major VPN protocols, such as OpenVPN, many tend to flash their routers with alternative open source firmwares such as DD-WRT, OpenWRT and Tomato which support multiple VPN protocols such as PPTP and OpenVPN.

Limitations:

Not every router is compatible with open source firmware which depends on the built-in flash memory and processor. Firmwares like DD-WRT require a minimum of 2 MiB flash memory and Broadcom chipsets. Setting up VPN services on a router requires a deeper knowledge of network security and careful installation. Minor misconfiguration of VPN connections can leave the network vulnerable. Performance will vary depending on the ISP and their reliability.

[30]

$6.8\,{\rm networking\ limitations}$

One major limitation of traditional VPNs is that they are point-to-point, and do not tend to support or connect broadcast domains. Therefore communication, software, and networking, which are based on layer 2 and broadcast packets, such as NetBIOS used in Windows networking, may not be fully supported or work exactly as they would on a real LAN. Variants on VPN, such as Virtual Private LAN Service (VPLS), and layer 2 tunneling protocols, are designed to overcome this limitation.

7 HOW DNS WORKS

DNS organizes hostnames in a domain hierarchy. A *domain* is a collection of sites that are related in some sense—because they form a proper network (e.g., all machines on a campus, or all hosts on BITNET), because they all belong to a certain organization (e.g., the U.S. government), or because they're simply geographically close. For instance, universities are commonly grouped in the edu domain, with each university or college using a separate *subdomain*, below which their hosts are subsumed. Groucho Marx University have the groucho.edu domain, while the LAN of the Mathematics department is assigned maths.groucho.edu. Hosts on the departmental network would have this domain name tacked onto their hostname, so erdos would be known as

erdos.maths.groucho.edu. This is called the *fully qualified domain name* (FQDN), which uniquely identifies this host worldwide.

Figure 4-1 shows a section of the namespace. The entry at the root of this tree, which is denoted by a single dot, is quite appropriately called the *root domain* and encompasses all other domains. To indicate that a hostname is a fully qualified domain name, rather than a name relative to some (implicit) local domain, it is sometimes written with a trailing dot. This dot signifies that the name's last component is the root domain.



Figure 4-1. A part of the domain namespace

Depending on its location in the name hierarchy, a domain may be called toplevel, second-level, or third-level. More levels of subdivision occur, but they are rare. This list details several top-level domains you may see frequently:

| Domain | Description |
|--------|---------------|
| edu | (Mostly U.S.) |
| | educational |
| | institutions |
| | like |
| | universities. |
| com | Commercial |

| Domain | Description | | |
|--------|-----------------|--|--|
| | organizations | | |
| | and | | |
| | companies. | | |
| org | Non- | | |
| | commercial | | |
| | organizations. | | |
| | Private | | |
| | UUCP | | |
| | networks are | | |
| | often in this | | |
| | domain. | | |
| net | Gateways and | | |
| | other | | |
| | administrative | | |
| | hosts on a | | |
| | network. | | |
| mil | U.S. military | | |
| | institutions. | | |
| gov | U.S. | | |
| | government | | |
| | institutions. | | |
| uucp | Officially, all | | |
| | site names | | |
| | formerly used | | |
| | as UUCP | | |
| | names | | |
| | without | | |
| | domains have | | |
| | been moved | | |
| | to this | | |
| | domaın. | | |

Historically, the first four of these were assigned to the U.S., but recent changes in policy have meant that these domains, named global Top Level Domains (gTLD), are now considered global in nature. Negotiations are

currently underway to broaden the range of gTLDs, which may result in increased choice in the future.

Outside the U.S., each country generally uses a top-level domain of its own named after the two-letter country code defined in ISO-3166. Finland, for instance, uses the fi domain; fr is used by France, de by Germany, and au by Australia. Below this top-level domain, each country's NIC is free to organize hostnames in whatever way they want. Australia has second-level domains similar to the international top-level domains, named com.au and edu.au. Other countries, like Germany, don't use this extra level, but have slightly long names that refer directly to the organizations running a particular domain. It's not uncommon to see hostnames like ftp.informatik.uni-erlangen.de. Chalk that up to German efficiency.

Of course, these national domains do not imply that a host below that domain is actually located in that country; it means only that the host has been registered with that country's NIC. A Swedish manufacturer might have a branch in Australia and still have all its hosts registered with the se top-level domain.

Organizing the namespace in a hierarchy of domain names nicely solves the problem of name uniqueness; with DNS, a hostname has to be unique only within its domain to give it a name different from all other hosts worldwide. Furthermore, fully qualified names are easy to remember. Taken by themselves, these are already very good reasons to split up a large domain into several subdomains.

DNS does even more for you than this. It also allows you to delegate authority over a subdomain to its administrators. For example, the maintainers at the Groucho Computing Center might create a subdomain for each department; we already encountered the math and physics subdomains above. When they find the network at the Physics department too large and chaotic to manage from outside (after all, physicists are known to be an unruly bunch of people), they may simply pass control of the physics.groucho.edu domain to the administrators of this network. These administrators are free to use whatever hostnames they like and assign them IP addresses from their network in whatever fashion they desire, without outside interference.

To this end, the namespace is split up into *zones*, each rooted at a domain. Note the subtle difference between a *zone* and a *domain*: the domain groucho.edu encompasses all hosts at Groucho Marx University, while the zone groucho.edu includes only the hosts that are managed by the Computing Center directly; those at the Mathematics department, for example. The hosts at the Physics department belong to a different zone, namely physics.groucho.edu. In Figure 6-1, the start of a zone is marked by a small circle to the right of the domain name.

7.1 NAME LOOKUPS WITH DNS

At first glance, all this domain and zone fuss seems to make name resolution an awfully complicated business. After all, if no central authority controls what names are assigned to which hosts, how is a humble application supposed to know?

Now comes the really ingenious part about DNS. If you want to find the IP address of erdos, DNS says, "Go ask the people who manage it, and they will tell you."

In fact, DNS is a giant distributed database. It is implemented by so-called name servers that supply information on a given domain or set of domains. For each zone there are at least two, or at most a few, name servers that hold all authoritative information on hosts in that zone. To obtain the IP address of erdos, all you have to do is contact the name server for the groucho.edu zone, which will then return the desired data.

Easier said than done, you might think. So how do I know how to reach the name server at Groucho Marx University? In case your computer isn't equipped with an address-resolving oracle, DNS provides for this, too. When your application wants to look up information on erdos, it contacts a local name server, which conducts a so-called iterative query for it. It starts off by sending a query to a name server for the root domain, asking for the address of erdos.maths.groucho.edu. The root name server recognizes that this name does not belong to its zone of authority, but rather to one below the edu domain. Thus, it tells you to contact an edu zone name server for more information and encloses a list of all edu name servers along with their addresses. Your local name server will then go on and query one of those, for instance, a.isi.edu. In a manner similar to the root name server, a.isi.edu knows that the groucho.edu people run a zone of their own, and points you to their servers. The local name server will then present its query for erdos to one of these, which will finally

recognize the name as belonging to its zone, and return the corresponding IP address.

This looks like a lot of traffic being generated for looking up a measly IP address, but it's really only miniscule compared to the amount of data that would have to be transferred if we were still stuck with HOSTS.TXT. There's still room for improvement with this scheme, however.

To improve response time during future queries, the name server stores the information obtained in its local *cache*. So the next time anyone on your local network wants to look up the address of a host in the groucho.edu domain, your name server will go directly to the groucho.edu name server.[1]

Of course, the name server will not keep this information forever; it will discard it after some time. The expiration interval is called the *time to live*, or TTL. Each datum in the DNS database is assigned such a TTL by administrators of the responsible zone.

7.2 TYPES OF NAME SERVERS

Name servers that hold all information on hosts within a zone are called *authoritative* for this zone, and sometimes are referred to as *master name servers*. Any query for a host within this zone will end up at one of these master name servers.

Master servers must be fairly well synchronized. Thus, the zone's network administrator must make one the *primary* server, which loads its zone information from data files, and make the others *secondary* servers, which transfer the zone data from the primary server at regular intervals.

Having several name servers distributes workload; it also provides backup. When one name server machine fails in a benign way, like crashing or losing its network connection, all queries will fall back to the other servers. Of course, this scheme doesn't protect you from server malfunctions that produce wrong replies to all DNS requests, such as from software bugs in the server program itself. You can also run a name server that is not authoritative for any domain.[2] This is useful, as the name server will still be able to conduct DNS queries for the applications running on the local network and cache the information. Hence it is called a *caching-only* server.

7.3 THE DNS DATABASE

We have seen that DNS not only deals with IP addresses of hosts, but also exchanges information on name servers. DNS databases may have, in fact, many different types of entries.

A single piece of information from the DNS database is called a *resource record* (RR). Each record has a type associated with it describing the sort of data it represents, and a class specifying the type of network it applies to. The latter accommodates the needs of different addressing schemes, like IP addresses (the IN class), Hesiod addresses (used by MIT's Kerberos system), and a few more. The prototypical resource record type is the A record, which associates a fully qualified domain name with an IP address.

A host may be known by more than one name. For example you might have a server that provides both FTP and World Wide Web servers, which you give two names: ftp.machine.org and www.machine.org. However, one of these names must be identified as the official or *canonical* hostname, while the others are simply aliases referring to the official hostname. The difference is that the canonical hostname is the one with an associated A record, while the others only have a record of type CNAME that points to the canonical hostname.

We will not go through all record types here, but we will give you a brief example. Example 4-4 shows a part of the domain database that is loaded into the name servers for the physics.groucho.edu zone.

Example 4-4. An Excerpt from the named.hosts File for the Physics Department

; Authoritative Information on physics.groucho.edu. @ IN SOA niels.physics.groucho.edu. janet.niels.physics.groucho.edu. { 1999090200 ; serial no 360000 ; refresh 3600 ; retry 3600000 ; expire 3600

; default ttl } ; ; Name servers IN NS niels IN NS gauss.maths.groucho.edu. gauss.maths.groucho.edu. IN A 149.76.4.23 ; ; Theoretical Physics (subnet 12) niels IN A 149.76.12.1 IN Α 149.76.1.12 name server IN CNAME niels otto IN A 149.76.12.2 quark IN A 149.76.12.4 down IN A 149.76.12.5 strange IN A 149.76.12.6 ... ; Collider Lab. (subnet 14)
 IN
 IN
 A
 149.76.14.1 muon

 A
 149.76.14.7 bogon
 IN
 A
 IN 149.76.14.12 ...

Apart from the A and CNAME records, you can see a special record at the top of the file, stretching several lines. This is the SOA resource record signaling the *Start of Authority*, which holds general information on the zone the server is authoritative for. The SOA record comprises, for instance, the default time to live for all records.

Note that all names in the sample file that do not end with a dot should be interpreted relative to the physics.groucho.edu domain. The special name (@) used in the SOA record refers to the domain name by itself.

We have seen earlier that the name servers for the groucho.edu domain somehow have to know about the physics zone so that they can point queries to their name servers. This is usually achieved by a pair of records: the NS record that gives the server's FQDN, and an A record that associates an address with that name. Since these records are what holds the namespace together, they are frequently called *glue records*. They are the only instances of records in which a parent zone actually holds information on hosts in the subordinate zone. The glue records pointing to the name servers for physics.groucho.edu are shown in Example 4-5.

Example 4-5. An Excerpt from the named.hosts File for GMU

```
; Zone data for the groucho.edu zone. @ IN SOA
vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
1999070100 ; serial no 360000
; refresh 3600 ; retry
3600000 ; expire 3600
; default ttl } .... ; Glue records for
the physics.groucho.edu zone physics IN NS
```

BA.NET ADBLOCK ADMINISTRATOR MANUAL

| niels.physic | s.groucho.edu | • | | IN | NS | |
|--------------|---------------|--------|---------|----|------------|-----|
| gauss.maths. | groucho.edu. | niels. | physics | IN | A | |
| 149.76.12.1 | gauss.maths | IN | A | 1 | 49.76.4.23 | ••• |
| | | | | | | |

7.4 REVERSE LOOKUPS

Finding the IP address belonging to a host is certainly the most common use for the Domain Name System, but sometimes you'll want to find the canonical hostname corresponding to an address. Finding this hostname is called *reverse mapping*, and is used by several network services to verify a client's identity. When using a single hosts file, reverse lookups simply involve searching the file for a host that owns the IP address in question. With DNS, an exhaustive search of the namespace is out of the question. Instead, a special domain, inaddr.arpa, has been created that contains the IP addresses of all hosts in a reversed dotted quad notation. For instance, an IP address of 149.76.12.4 corresponds to the name 4.12.76.149.in-addr.arpa. The resource-record type linking these names to their canonical hostnames is PTR.

Creating a zone of authority usually means that its administrators have full control over how they assign addresses to names. Since they usually have one or more IP networks or subnets at their hands, there's a one-to-many mapping between DNS zones and IP networks. The Physics department, for instance, comprises the subnets 149.76.8.0, 149.76.12.0, and 149.76.14.0.

Consequently, new zones in the in-addr.arpa domain have to be created along with the physics zone, and delegated to the network administrators at the department: 8.76.149.in-addr.arpa, 12.76.149.in-addr.arpa, and 14.76.149.in-addr.arpa. Otherwise, installing a new host at the Collider Lab would require them to contact their parent domain to have the new address entered into their in-addr.arpa zone file.

The zone database for subnet 12 is shown in <u>Example 4-6</u>. The corresponding glue records in the database of their parent zone are shown in <u>Example 4-7</u>.

Example 4-6. An Excerpt from the named.rev File for Subnet 12

; the 12.76.149.in-addr.arpa domain. Ø IN SOA niels.physics.groucho.edu. janet.niels.physics.groucho.edu. 1999090200 360000 3600 3600000 3600 } 2 ΙN PTR otto.physics.groucho.edu. 4 ΙN quark.physics.groucho.edu. 5 PTR PTR ΙN down.physics.groucho.edu. 6 ΙN PTR
strange.physics.groucho.edu.

Example 4-7. An Excerpt from the named.rev File for Network 149.76

```
; the 76.149.in-addr.arpa domain.
                                    Ø
                                       ΙN
                                           SOA
vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
1999070100 360000 3600 3600000 3600
                                                       }
; subnet 4: Mathematics Dept. 1.4
                                                 PTR
                                          IN
sophus.maths.groucho.edu. 17.4
                                             PTR
                                      IN
erdos.maths.groucho.edu.
                          23.4
                                     ΙN
                                            PTR
gauss.maths.groucho.edu. ... ; subnet 12: Physics Dept,
separate zone 12
                          ΙN
                                 NS
niels.physics.groucho.edu.
                                       IN
                                              NS
gauss.maths.groucho.edu. niels.physics.groucho.edu. IN A
149.76.12.1 gauss.maths.groucho.edu. IN A
                                              149.76.4.23
. . .
```

in-addr.arpa system zones can only be created as supersets of IP networks. An even more severe restriction is that these networks' netmasks have to be on byte boundaries. All subnets at Groucho Marx University have a netmask of 255.255.255.0, hence an in-addr.arpa zone could be created for each subnet. However, if the netmask were 255.255.255.128 instead, creating zones for the subnet 149.76.12.128 would be impossible, because there's no way to tell DNS that the 12.76.149.in-addr.arpa domain has been split into two zones of authority, with hostnames ranging from 1 through 127, and 128 through 255, respectively.

7.4.1 NOTES

- [1] If information weren't cached, then DNS would be as inefficient as any other method because each query would involve the root name servers.
- [2] Well, almost. A name server has to provide at least name service for localhost and reverse lookups of 127.0.0.1.

8 RUNNING NAMED

named (pronounced *name-dee*) provides DNS on most Unix machines. It is a server program originally developed for BSD to provide name service to clients, and possibly to other name servers. BIND Version 4 was around for some time and appeared in most Linux distributions. The new release, Version 8, has been introduced in most Linux distributions, and is a big change from previous versions.[1] It has many new features, such as support for DNS dynamic updates, DNS change notifications, much improved performance, and a new configuration file syntax. Please check the documentation contained in the source distribution for details.

This section requires some understanding of the way DNS works. If the following discussion is all Greek to you, you may want to reread the section Section 6.2."

named is usually started at system boot time and runs until the machine goes down again. Implementations of BIND prior to Version 8 take their information from a configuration file called /etc/named.boot and various files that map domain names to addresses. The latter are called *zone files*. Versions of BIND from Version 8 onwards use /etc/named.conf in place of /etc/named.boot.

To run **named** at the prompt, enter:

/usr/sbin/named

named will come up and read the **named.boot** file and any zone files specified therein. It writes its process ID to /var/run/named.pid in ASCII, downloads any zone files from primary servers, if necessary, and starts listening on port 53 for DNS queries.

8.1 THE NAMED.BOOT FILE

The BIND configuration file prior to Version 8 was very simple in structure. BIND Version 8 has a very different configuration file syntax to deal with many of the new features introduced. The name of the configuration file changed from /etc/named.boot, in older versions of BIND, to /etc/named.conf in BIND Version 8. We'll focus on configuring the older version because it is probably what most distributions are still using, but we'll present an equivalent named.conf to illustrate the differences, and we'll talk about how to convert the old format into the new one.

The **named.boot** file is generally small and contains little but pointers to master files containing zone information and pointers to other name servers. Comments in the boot file start with the (#) or (;) characters and extend to the next newline. Before we discuss the format of named.boot in more detail, we will take a look at the sample file for vlager given in Example 5-8.

Example 5-8. The named.boot File for vlager

Let's look at each statement individually. The directory keyword tells **named** that all filenames referred to later in this file, zone files for example, are located in the /var/named directory. This saves a little typing.

The primary keyword shown in this example loads information into **named**. This information is taken from the master files specified as the last of the parameters. These files represent DNS resource records, which we will look at next.

In this example, we configured **named** as the primary name server for three domains, as indicated by the three primary statements. The first of these statements instructs **named** to act as a primary server for vbrew.com, taking the zone data from the file named.hosts.

The cache keyword is very special and should be present on virtually all machines running a name server. It instructs **named** to enable its cache and to load the *root name server hints* from the cache file specified (named.ca in our example). We will come back to the name server hints in the following list.

Here's a list of the most important options you can use in named.boot :

directory

This option specifies a directory in which zone files reside. Names of files in other options may be given relative to this directory. Several directories may be specified by repeatedly using directory. The Linux file system standard suggests this should be /var/named.

primary

This option takes a domain name and filename as an argument, declaring the local server authoritative for the named domain. As a primary server, **named** loads the zone information from the given master file.

There will always be at least one primary entry in every boot file used for reverse mapping of network 127.0.0.0, which is the local loopback network.

secondary

This statement takes a domain name, an address list, and a filename as an argument. It declares the local server a secondary master server for the specified domain.

A secondary server holds authoritative data on the domain, too, but it doesn't gather it from files; instead, it tries to download it from the primary server. The IP address of at least one primary server thus must be given to **named** in the address list. The local server contacts each of them in turn until it successfully transfers the zone database, which is then stored in the backup file given as the third argument. If none of the primary servers responds, the zone data is retrieved from the backup file instead.

named then attempts to refresh the zone data at regular intervals. This process is explained later in connection with the SOA resource record type.

cache

This option takes a domain name and filename as arguments. This file contains the root server hints, which is a list of records pointing to the

root name servers. Only NS and A records will be recognized. The *domain* should be the root domain name, a simple period (.).

This information is absolutely crucial to **named**; if the cache statement does not occur in the boot file, **named** will not develop a local cache at all. This situation/lack of development will severely degrade performance and increase network load if the next server queried is not on the local net. Moreover, **named** will not be able to reach any root name servers, and thus won't resolve any addresses except those it is authoritative for. An exception from this rule involves forwarding servers (see the forwarders option that follows).

forwarders

This statement takes a whitespace-separated list of addresses as an argument. The IP addresses in this list specify a list of name servers that **named** may query if it fails to resolve a query from its local cache. They are tried in order until one of them responds to the query. Typically, you would use the name server of your network provider or another well-known server as a forwarder.

slave

This statement makes the name server a *slave* server. It never performs recursive queries itself, but only forwards them to servers specified in the forwarders statement.

There are two options that we will not describe here: sortlist and domain. Two other directives may also be used inside these database files: \$INCLUDE and \$ORIGIN. Since they are rarely needed, we will not describe them here, either.

8.2 THE BIND 8 HOST.CONF FILE

BIND Version 8 introduced a range of new features, and with these came a new configuration file syntax. The named.boot, with its simple single line statements, was replaced by the named.conf file, with a syntax like that of **gated** and resembling C source syntax.

The new syntax is more complex, but fortunately a tool has been provided that automates conversion from the old syntax to the new syntax. In the BIND 8

source package, a **perl** program called **named-bootconf.pl** is provided that will read your existing named.boot file from stdin and convert it into the equivalent named.conf format on stdout. To use it, you must have the **perl** interpreter installed.

You should use the script somewhat like this:

cd /etc # named-bootconf.pl <named.boot >named.conf

The script then produces a named.conf that looks like that shown in <u>Example 5-9</u>. We've cleaned out a few of the helpful comments the script includes to help show the almost direct relationship between the old and the new syntax.

Example 5-9. The BIND 8 equivalent named.conf File for vlager

// // /etc/named.boot file for vlager.vbrew.com
options { directory "/var/named"; }; zone "." { type
hint; file "named.ca"; }; zone "vbrew.com" { type
master; file "named.hosts"; }; zone "0.0.127.inaddr.arpa" { type master; file "named.local"; };
zone "16.172.in-addr.arpa" { type master; file
"named.rev"; };

If you take a close look, you will see that each of the one-line statements in named.boot has been converted into a C-like statement enclosed within { } characters in the named.conf file.

The comments, which in the named.boot file were indicated by a semicolon (;), are now indicated by two forward slashes (//).

The directory statement has been translated into an options paragraph with a directory clause.

The cache and primary statements have been converted into zone paragraphs with type clauses of hint and master, respectively.

The zone files do not need to be modified in any way; their syntax remains unchanged.

The new configuration syntax allows for many new options that we haven't covered here. If you'd like information on the new options, the best source of information is the documentation supplied with the BIND Version 8 source package.

8.3 THE DNS DATABASE FILES

Master files included with **named**, like **named.hosts**, always have a domain associated with them, which is called the *origin*. This is the domain name specified with the cache and primary options. Within a master file, you are allowed to specify domain and host names relative to this domain. A name given in a configuration file is considered *absolute* if it ends in a single dot, otherwise it is considered relative to the origin. The origin by itself may be referred to using (@).

The data contained in a master file is split up in *resource records* (RRs). RRs are the smallest units of information available through DNS. Each resource record has a type. A records, for instance, map a hostname to an IP address, and a CNAME record associates an alias for a host with its official hostname. To see an example, look at Example 5-11, which shows the **named.hosts** master file for the Virtual Brewery.

Resource record representations in master files share a common format:

[domain] [ttl] [class] type rdata

Fields are separated by spaces or tabs. An entry may be continued across several lines if an opening brace occurs before the first newline and the last field is followed by a closing brace. Anything between a semicolon and a newline is ignored. A description of the format terms follows:

domain

This term is the domain name to which the entry applies. If no domain name is given, the RR is assumed to apply to the domain of the previous RR.

ttl

In order to force resolvers to discard information after a certain time, each RR is associated a time to live (ttl). The ttl field specifies the time

in seconds that the information is valid after it has been retrieved from the server. It is a decimal number with at most eight digits.

If no ttl value is given, the field value defaults to that of the *minimum* field of the preceding SOA record.

class

This is an address class, like IN for IP addresses or HS for objects in the Hesiod class. For TCP/IP networking, you have to specify IN.

If no class field is given, the class of the preceding RR is assumed.

type

This describes the type of the RR. The most common types are A, SOA, PTR, and NS. The following sections describe the various types of RRs.

rdata

This holds the data associated with the RR. The format of this field depends on the type of RR. In the following discussion, it will be described for each RR separately.

The following is partial list of RRs to be used in DNS master files. There are a couple more of them that we will not explain; they are experimental and of little use, generally.

SOA

This RR describes a zone of authority (SOA means "Start of Authority"). It signals that the records following the SOA RR contain authoritative information for the domain. Every master file included by a primary statement must contain an SOA record for this zone. The resource data contains the following fields:

origin

This field is the canonical hostname of the primary name server for this domain. It is usually given as an absolute name.

contact

This field is the email address of the person responsible for maintaining the domain, with the "@" sign replaced by a dot. For instance, if the responsible person at the Virtual Brewery were janet, this field would contain janet.vbrew.com.

serial

This field is the version number of the zone file, expressed as a single decimal number. Whenever data is changed in the zone file, this number should be incremented. A common convention is to use a number that reflects the date of the last update, with a version number appended to it to cover the case of multiple updates occurring on a single day, e.g., 2000012600 being update 00 that occurred on January 26, 2000.

The serial number is used by secondary name servers to recognize zone information changes. To stay up to date, secondary servers request the primary server's SOA record at certain intervals and compare the serial number to that of the cached SOA record. If the number has changed, the secondary servers transfer the whole zone database from the primary server.

refresh

This field specifies the interval in seconds that the secondary servers should wait between checking the SOA record of the primary server. Again, this is a decimal number with at most eight digits.

Generally, the network topology doesn't change too often, so this number should specify an interval of roughly a day for larger networks, and even more for smaller ones.

retry

This number determines the intervals at which a secondary server should retry contacting the primary server if a request or a zone refresh fails. It must not be too low, or a temporary failure of the server or a network problem could cause the secondary server to waste network resources. One hour, or perhaps one-half hour, might be a good choice.

expire

This field specifies the time in seconds after which a secondary server should finally discard all zone data if it hasn't been able to contact the primary server. You should normally set this field to at least a week (604,800 seconds), but increasing it to a month or more is also reasonable.

minimum

This field is the default *ttl* value for resource records that do not explicitly contain one. The ttl value specifies the maximum amount of time other name servers may keep the RR in their cache. This time applies only to normal lookups, and has nothing to do with the time after which a secondary server should try to update the zone information.

If the topology of your network does not change frequently, a week or even more is probably a good choice. If single RRs change more frequently, you could still assign them smaller ttls individually. If your network changes frequently, you may want to set *minimum* to one day (86,400 seconds).

A

This record associates an IP address with a hostname. The resource data field contains the address in dotted quad notation.

For each hostname, there must be only one A record. The hostname used in this A record is considered the official or *canonical* hostname. All other hostnames are aliases and must be mapped onto the canonical hostname using a CNAME record. If the canonical name of our host were *vlager*, we'd have an A record that associated that hostname with its IP address. Since we may also want another name associated with that address, say news, we'd create a CNAME record that associates this alternate name with the canonical name. We'll talk more about CNAME records shortly.

NS

NS records are used to specify a zone's primary server and all its secondary servers. An NS record points to a master name server of the given zone, with the resource data field containing the hostname of the name server.

You will meet NS records in two situations: The first situation is when you delegate authority to a subordinate zone; the second is within the master zone database of the subordinate zone itself. The sets of servers specified in both the parent and delegated zones should match.

The NS record specifies the name of the primary and secondary name servers for a zone. These names must be resolved to an address so they can be used. Sometimes the servers belong to the domain they are serving, which causes a "chicken and egg" problem; we can't resolve the address until the name server is reachable, but we can't reach the name server until we resolve its address. To solve this dilemma, we can configure special A records directly into the name server of the parent zone. The A records allow the name servers of the parent domain to resolve the IP address of the delegated zone name servers. These records are commonly called *glue records* because they provide the "glue" that binds a delegated zone to its parent.

CNAME

This record associates an alias with a host's *canonical hostname*. It provides an alternate name by which users can refer to the host whose canonical name is supplied as a parameter. The canonical hostname is the one the master file provides an A record for; aliases are simply linked to that name by a CNAME record, but don't have any other records of their own.

PTR

This type of record is used to associate names in the in-addr.arpa domain with hostnames. It is used for reverse mapping of IP addresses to hostnames. The hostname given must be the canonical hostname.

MX

```
BA.NET ADBLOCK ADMINISTRATOR MANUAL
```

This RR announces a *mail exchanger* for a domain. Mail exchangers are discussed in <u>Section 17.4.1</u>." The syntax of an MX record is:

[domain] [ttl] [class] MX preference host

host names the mail exchanger for *domain*. Every mail exchanger has an integer *preference* associated with it. A mail transport agent that wants to deliver mail to *domain* tries all hosts who have an MX record for this domain until it succeeds. The one with the lowest preference value is tried first, then the others, in order of increasing preference value.

HINFO

This record provides information on the system's hardware and software. Its syntax is:

[domain] [ttl] [class] HINFO hardware software

The *hardware* field identifies the hardware used by this host. Special conventions are used to specify this. A list of valid "machine names" is given in the Assigned Numbers RFC (RFC-1700). If the field contains any blanks, it must be enclosed in double quotes. The *software* field names the operating system software used by the system. Again, a valid name from the Assigned Numbers RFC should be chosen.

An HINFO record to describe an Intel-based Linux machine should look something like:

tao 36500 IN HINFO IBM-PC LINUX2.2

and HINFO records for Linux running on Motorola 68000-based machines might look like:

```
cevad 36500 IN HINFO ATARI-104ST LINUX2.0 jedd 36500
IN HINFO AMIGA-3000 LINUX2.0
```

8.4 CACHING-ONLY NAMED CONFIGURATION

There is a special type of named configuration that we'll talk about before we explain how to build a full name server configuration. It is called a *caching*-

only configuration. It doesn't really serve a domain, but acts as a relay for all DNS queries produced on your host. The advantage of this scheme is that it builds up a cache so only the first query for a particular host is actually sent to the name servers on the Internet. Any repeated request will be answered directly from the cache in your local name server.

A named.boot file for a caching-only server looks like this:

```
; named.boot file for caching-only server directory
/var/named primary 0.0.127.in-addr.arpa named.local ;
localhost network cache .
named.ca ; root servers
```

In addition to this named.boot file, you must set up the named.ca file with a valid list of root name servers. You could copy and use Example 5-10 for this purpose. No other files are needed for a caching-only server configuration.

8.5 WRITING THE MASTER FILES

Example 5-10, Example 5-11, Example 5-12, and Example 5-13 give sample files for a name server at the brewery, located on vlager. Due to the nature of the network discussed (a single LAN), the example is pretty straightforward.

The named.ca cache file shown in <u>Example 5-10</u> shows sample hint records for a root name server. A typical cache file usually describes about a dozen name servers. You can obtain the current list of name servers for the root domain using the **nslookup** tool described in the next section.[2]

Example 5-10. The named.ca File

| ; ; /var/named/named.ca | | Cache | file | for the | brewery. | |
|-------------------------|--------|---------------|------------|---------|----------|-----------|
| ; | | We're not on | the Inter | rnet, | so we d | on't need |
| ; | | any root ser | vers. To a | activa | te thes | e ; |
| records, | remove | the semicolon | s.;;. | | | |
| 3600000 | IN NS | A.ROOT-SER | VERS.NET. | ;A.RO | OT-SERV | ERS.NET. |
| 3600000 | A | 198.41.0.4 | ;. | | | |
| 3600000 | NS | B.ROOT-SER | VERS.NET. | ;B.RO | OT-SERV | ERS.NET. |
| 3600000 | A | 128.9.0.10 | 7;. | | | |
| 3600000 | NS | C.ROOT-SER | VERS.NET. | ;C.RO | OT-SERV | ERS.NET. |
| 3600000 | A | 192.33.4.1 | 2;. | | | |
| 3600000 | NS | D.ROOT-SER | VERS.NET. | ;D.RO | OT-SERV | ERS.NET. |

BA.NET ADBLOCK ADMINISTRATOR MANUAL

| 3600000 | A | 128.8.10.90 ;. | |
|---------|----|---------------------|-----------------------|
| 3600000 | NS | E.ROOT-SERVERS.NET. | ;E.ROOT-SERVERS.NET. |
| 3600000 | A | 192.203.230.10 ;. | |
| 3600000 | NS | F.ROOT-SERVERS.NET. | ;F.ROOT-SERVERS.NET. |
| 3600000 | A | 192.5.5.241 ;. | |
| 3600000 | NS | G.ROOT-SERVERS.NET. | ;G.ROOT-SERVERS.NET. |
| 3600000 | A | 192.112.36.4 ;. | |
| 3600000 | NS | H.ROOT-SERVERS.NET. | ;H.ROOT-SERVERS.NET. |
| 3600000 | A | 128.63.2.53 ;. | |
| 3600000 | NS | I.ROOT-SERVERS.NET. | ; I.ROOT-SERVERS.NET. |
| 3600000 | A | 192.36.148.17 ;. | |
| 3600000 | NS | J.ROOT-SERVERS.NET. | ; J.ROOT-SERVERS.NET. |
| 3600000 | A | 198.41.0.10 ;. | |
| 3600000 | NS | K.ROOT-SERVERS.NET. | ;K.ROOT-SERVERS.NET. |
| 3600000 | A | 193.0.14.129 ;. | |
| 3600000 | NS | L.ROOT-SERVERS.NET. | ;L.ROOT-SERVERS.NET. |
| 3600000 | A | 198.32.64.12 ;. | |
| 3600000 | NS | M.ROOT-SERVERS.NET. | ;M.ROOT-SERVERS.NET. |
| 3600000 | A | 202.12.27.33 ; | |

Example 6-11. The named.hosts File

```
; ; /var/named/named.hosts Local hosts at the brewery ;
Origin is vbrew.com ; @
                                    IN SOA
vlager.vbrew.com. janet.vbrew.com. (
2000012601 ; serial
                                           86400
                                                     ;
refresh: once per day
                                             3600
                                                      ;
retry: one hour
                                         3600000
                                                   ;
expire: 42 days
                                         604800
                                                   ;
minimum: 1 week
                                        )
      vlager.vbrew.com. ; ; local mail is distributed on
IN NS
vlager
                      IN MX 10 vlager ; ; loopback
                               127.0.0.1 ; ; Virtual
address localhost.
                      IN A
                               IN A 172.16.1.1
Brewery Ethernet vlager
vlager-if1 IN CNAME vlager ; vlager is also news
server news
                      IN CNAME vlager vstout
                                                     IN
     172.16.1.2 vale
                                        172.16.1.3 ; ;
А
                               IN A
Virtual Winery Ethernet vlager-if2
                                      IN A 172.16.2.1
              IN A 172.16.2.2 vchianti
vbardolino
                                                   IN A
172.16.2.3 vbeaujolais IN A 172.16.2.4 ; ; Virtual
Spirits (subsidiary) Ethernet vbourbon
                                           IN A
172.16.3.1 vbourbon-if1 IN CNAME vbourbon
```

Example 6-12. The named.local File

; ; /var/named/named.local Reverse mapping of 127.0.0 ; Origin is 0.0.127.in-addr.arpa. ; @ IN SOA vlager.vbrew.com. joe.vbrew.com. (1 360000 ; refresh: 100 ; serial 3600 ; retry: one hour hrs 3600000 ; expire: 42 days 360000 ; minimum: 100 hrs) ΙN vlager.vbrew.com. 1 NS IN PTR localhost.

Example 6-13. The named.rev File

```
; ; /var/named/named.rev Reverse mapping of our IP
addresses ;
                                     Origin is
16.172.in-addr.arpa. ; @
                                 IN
                                     SOA
vlager.vbrew.com. joe.vbrew.com. (
         ; serial
16
                                         86400
                                                   ;
refresh: once per day
                                           3600
                                                    ;
retry: one hour
                                       3600000
                                                 ;
                                      604800
expire: 42 days
                                                 ;
minimum: 1 week
                                      )
                                                    IN
     vlager.vbrew.com. ; brewery 1.1
NS
                                           IN PTR
vlager.vbrew.com. 2.1 IN PTR
                                     vstout.vbrew.com.
            IN PTR vale.vbrew.com. ; winery 1.2
3.1
IN PTR vlager-if2.vbrew.com. 2.2
                                         TN PTR
vbardolino.vbrew.com. 3.2 IN PTR
                           IN PTR
vchianti.vbrew.com. 4.2
vbeaujolais.vbrew.com.
```

8.6 VERIFYING THE NAME SERVER SETUP

nslookup is a great tool for checking the operation of your name server setup. It can be used both interactively with prompts and as a single command with immediate output. In the latter case, you simply invoke it as:

\$ nslookup hostname

nslookup queries the name server specified in resolv.conf for *hostname*. (If this file names more than one server, **nslookup** chooses one at random.)

The interactive mode, however, is much more exciting. Besides looking up individual hosts, you may query for any type of DNS record and transfer the entire zone information for a domain.

When invoked without an argument, **nslookup** displays the name server it uses and enters interactive mode. At the > prompt, you may type any domain name you want to query. By default, it asks for class A records, those containing the IP address relating to the domain name.

You can look for record types by issuing:

```
> set type=type
```

in which type is one of the resource record names described earlier, or ANY.

You might have the following **nslookup** session:

```
$ nslookup Default Server: tao.linux.org.au Address:
203.41.101.121 > metalab.unc.edu Server: tao.linux.org.au
Address: 203.41.101.121 Name: metalab.unc.edu Address:
152.2.254.81 >
```

The output first displays the DNS server being queried, and then the result of the query.

If you try to query for a name that has no IP address associated with it, but other records were found in the DNS database, **nslookup** returns with an error message saying "No type A records found." However, you can make it query for records other than type A by issuing the **set type** command. To get the SOA record of unc.edu, you would issue:

```
> unc.edu Server: tao.linux.org.au Address: 203.41.101.121
*** No address (A) records available for unc.edu > set type=SOA
> unc.edu Server: tao.linux.org.au Address: 203.41.101.121
unc.edu origin = ns.unc.edu mail addr = host-
reg.ns.unc.edu serial = 1998111011 refresh =
14400 (4H) retry = 3600 (1H) expire =
1209600 (2W) minimum ttl = 86400 (1D) unc.edu name
server = ns2.unc.edu unc.edu name server = ncnoc.ncren.net
unc.edu name server = ns.unc.edu internet
address = 152.2.253.100 ncnoc.ncren.net internet address = 128.109.193.1
ns.unc.edu internet address = 152.2.21.1
```

In a similar fashion, you can query for MX records:

```
> set type=MX > unc.edu Server: tao.linux.org.au Address:
203.41.101.121 unc.edu preference = 0, mail exchanger =
conga.oit.unc.edu unc.edu preference = 10, mail exchanger =
imsety.oit.unc.edu unc.edu name server = ns.unc.edu unc.edu
name server = ns2.unc.edu unc.edu name server =
ncnoc.ncren.net conga.oit.unc.edu internet address =
152.2.22.21 imsety.oit.unc.edu internet address =
152.2.21.99 ns.unc.edu internet address = 152.2.21.1
ns2.unc.edu internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
```

Using a type of ANY returns all resource records associated with a given name.

A practical application of **nslookup**, besides debugging, is to obtain the current list of root name servers. You can obtain this list by querying for all NS records associated with the root domain:

| <pre>> set type=NS > . Server:</pre> | tao.linux.org.au Address: |
|--|---------------------------------------|
| 203.41.101.121 Non-auth | noritative answer: (root) name server |
| = A.ROOT-SERVERS.NET (rc | oot) name server = H.ROOT-SERVERS.NET |
| (root) name server = B. | ROOT-SERVERS.NET (root) name server |
| = C.ROOT-SERVERS.NET (rc | oot) name server = D.ROOT-SERVERS.NET |
| (root) name server = E. | ROOT-SERVERS.NET (root) name server |
| = I.ROOT-SERVERS.NET (rc | oot) name server = F.ROOT-SERVERS.NET |
| (root) name server = G. | ROOT-SERVERS.NET (root) name server |
| = J.ROOT-SERVERS.NET (rc | oot) name server = K.ROOT-SERVERS.NET |
| (root) name server = L. | ROOT-SERVERS.NET (root) name server |
| = M.ROOT-SERVERS.NET Au | thoritative answers can be found |
| from: A.ROOT-SERVERS.NET | internet address = 198.41.0.4 |
| H.ROOT-SERVERS.NET | internet address = $128.63.2.53$ |
| B.ROOT-SERVERS.NET | internet address = 128.9.0.107 |
| C.ROOT-SERVERS.NET | internet address = $192.33.4.12$ |
| D.ROOT-SERVERS.NET | internet address = 128.8.10.90 |
| E.ROOT-SERVERS.NET | internet address = 192.203.230.10 |
| I.ROOT-SERVERS.NET | internet address = 192.36.148.17 |
| F.ROOT-SERVERS.NET | internet address = 192.5.5.241 |
| G.ROOT-SERVERS.NET | internet address = $192.112.36.4$ |
| J.ROOT-SERVERS.NET | internet address = 198.41.0.10 |
| K.ROOT-SERVERS.NET | internet address = 193.0.14.129 |
| L.ROOT-SERVERS.NET | internet address = $198.32.64.12$ |
| M.ROOT-SERVERS.NET | internet address = 202.12.27.33 |

To see the complete set of available commands, use the **help** command in **nslookup**.

8.7 OTHER USEFUL TOOLS

There are a few tools that can help you with your tasks as a BIND administrator. We will briefly describe two of them here. Please refer to the documentation that comes with these tools for more information on how to use them.

hostcvt helps you with your initial BIND configuration by converting your /etc/hosts file into master files for **named**. It generates both the forward (A) and reverse mapping (PTR) entries, and takes care of aliases. Of course, it won't do the whole job for you, as you may still want to tune the timeout values in the SOA record, for example, or add MX records. Still, it may help you save a few aspirins. **hostcvt** is part of the BIND source, but can also be found as a standalone package on a few Linux FTP servers.

After setting up your name server, you may want to test your configuration. Some good tools that make this job much simpler: the first is called **dnswalk**, which is a Perl-based package. The second is called **nslint**. They both walk your DNS database looking for common mistakes and verify that the information they find is consistent. Two other useful tools are **host** and **dig**, which are general purpose DNS database query tools. You can use these tools to manually inspect and diagnose DNS database entries.

These tools are likely to be available in prepackaged form. **dnswalk** and **nslint** are available in source from http://www.visi.com/~barr/dnswalk/ and ftp://ftp.ee.lbl.gov/nslint.tar.Z. The **host** and **dig** source codes can be found at ftp://ftp.nikhef.nl/pub/network/ and ftp://ftp.is.co.za/networking/ip/dns/dig/.

8.7.1 NOTES

- [1] BIND 4.9 was developed by Paul Vixie, paul@vix.com, but BIND is now maintained by the Internet Software Consortium, bind-bugs@isc.org.
- [2] Note that you can't query your name server for the root servers if you don't have any root server hints installed. To escape this dilemma, you can either make **nslookup** use a different name server, or use the sample file in Example 5-10 as a starting point, and then obtain the full list of valid

9 BA.NET ADBLOCK FILTER SERVER FLASHBOOT



BA.net Adblock DNS Server FlashBoot is a complete Software Appliance. Built in a simple USB Flash Boot package.

- · Make your Internet Faster and Safer with free AdBlock BA.net.
- · Blocks Ads on any application on your network
- · Blocks Malware
- · Custom Corporate Blocked Sites List
- · Small 125 MBytes Flash boot for USB PenDrives
- · No installation needed PenDrive contains OS, DNS, AdBlock Code
- · Can boot on most PCs or netbooks.
- · Low requirements for CPU and Memory
- · Can be installed on Hard Disk
- · Business Internet Monitoring and Control
- · Full Technical Support, DNS Filter List Update and Monitoring Available

BA.net Adblock DNS Server FlashBoot http://ba.net/adblock-dns-server/

Free DOWNLOAD from http://wifi.com.ar/adblock-dns-server/download

BA.NET ADBLOCK ADMINISTRATOR MANUAL

9.1.1 HOWTO INSTALL

- USB Flash Pen Drive install

Download latest BA.net Adblock DNS Server FlashBoot from http://wifi.com.ar/adblock-dns-server/download
Extract adblock-dns-server-rx.zip to your flash disk (FAT 16 or FAT 32 formated).
Make the flash disk bootable by running (from linux)
/boot/bootinst.sh
Or make the disk bootable by running (from windows)
/boot/bootinst.bat

That is all the instalation needed!

Now boot from the USB Flash Pen Drive Make sure that the boot order tries the USB drive first. configure this on BIOS settings. Usually available pressing one of these keys during boot: F1, F2, DEL, ESC or F10 Initial root password is toor Have fun. adblock@ba.net

Optional: - Hard Disk Install -

Format your hard disk and create a single FAT32 partition

- Extract adblock-dns-server-rx.zip to your hard disk

- make the disk bootable by running (from linux)
/boot/bootinst.sh

- make the disk bootable by running (from windows)
/boot/bootinst.bat

BA.NET ADBLOCK ADMINISTRATOR MANUAL



10 CONFIGURATION FILES

It is a typical problem in organizations that are growing that they have to resolve two problems at once:

- To have a DNS server for the internal network of the company because long ago there were already too many computers to remember their IPs^{1} and even too many computers to maintain a set of host files².
- To have a DNS server for the external servers, for external clients, etc.

to solve this problems become a bigger problem when the growing organization can't supply more resources than one DNS server³. It is a bigger problem because if you just configure your server with all your names, public and private, you'll end up polluting the Internet with private addresses, something that is very bad, and also showing the world part of the topology of your internal network. Something you don't want a possible attacker/cracker to have.

The other part of the problem is that for efficiency you may want to resolve to internal IPs when you are inside and external IPs when you are outside. Here I am taking about computers which have public and private connections.

There are many different solutions to this problem and I remember solving it even with BIND4, but now I am going to use BIND9 to make a solution that is very clean. This was deployed in a Debian GNU/Linux 3.1 server but it should also work for other operating systems that run BIND9, just be sure to change your paths appropriately.

10.12 INITIAL CONFIGURATION

Let's imagine the organization I work for makes examples... of what ? I don't know, but you can order them on example.com. Examples Corporation has been assigned the network 192.0.2.0/24 and internally we are using 10.0.0.0/24.

Let's start serving the external names and IPs, we edit /etc/bind/named.conf.local⁴ and add:

```
zone "example.com" {
   type master;
   file "/etc/bind/db.example.com";
};
```

and then we create /etc/bind/db.example.com with the following contents:

```
; example.com
$TTL 604800
(a)
      IN
            SOA
                   ns1.example.com. root.example.com. (
            2006020201 ; Serial
              604800 ; Refresh
               86400 ; Retry
              2419200 ; Expire
              604800); Negative Cache TTL
(a)
      IN
            NS
                  ns1
    IN
          MX
                  10 mail
    IN
          Α
                192.0.2.1
ns1
      IN
            А
                  192.0.2.1
mail IN
                  192.0.2.128 ; We have our mail server somewhere else.
            Α
WWW
       IN
             А
                   192.0.2.1
client1 IN
            Α
                  192.0.2.201 ; We connect to client1 very often.
```

As you can see, our start up has one computer to serve all, except mail, it even holds the IP forwarding and a couple of databases.

Now, a good DNS set up has at least one secondary server and in fact, some registrars (where you register domain names) enforce this. Since we don't have a second computer, we go to <u>XName</u>, open an account and register example.com as secondary with 192.0.2.1 as IP to transfer from. We now need to let XName's IP do the transfer; we are a small organization but since we want to be a successful start up we try to do everything as smartly as possible. So we use the BIND9 configuration directive acl to define an identifier that aliases to the XName's IP addresses; at the beginning of /etc/bind/named.conf.local we add:

and we change the zone declaration to:

```
zone "example.com" {
  type master;
  file "/etc/bind/db.example.com";
  allow-transfer { slaves; };
};
```

We could have just typed the IPs where we type "slaves".

10.23 INTERNALS AND EXTERNALS

Now that we have a solid base, we can start to thing about serving different contents to the internal and external network, but first, we have to define what is internal and what is external.

On /etc/bind/named.conf.local we add the following definition (at the top or below the definition of slaves):

If we had more internal networks, we could just add them there. We don't define externals because everything that is not internal is external. You may, if you want, define sets of different externals if you want to serve different content to different chunks of the Internet.

We will use a new feature of BIND9 called views. A view let's put a piece of configuration inside a conditional that can depend on a number of things, in this case we'll just depend on internals. We replace the zone declaration at /etc/bind/named.conf.local with:

```
view "internal" {
    match-clients { internals; };
    zone "example.com" {
        type master;
        file "/etc/bind/internals/db.example.com";
        };
    ;
    view "external" {
        match-clients { any; };
        zone "example.com" {
            type master;
            file "/etc/bind/externals/db.example.com";
            allow-transfer { slaves; };
        };
    };
```

The match clients configuration directive allow us to conditionally show that view based on a set of IPs, "any" stands for any IP. Internal IPs will be cached by the internal view and the rest will be dropped on the external view. The outside world can't see the internal view, and that includes XName, our secondary DNS provider, but we removed the allow-transfer from the internal view since we don't want anyone to be able to transfer under any circumstances the contents of the internal view.

We also changed the path, we will have to create the directory /etc/bind/externals and /etc/bind/internals and move /etc/bind/db.example.com to /etc/bind/externals/.

On /etc/bind/internals/db.example.com we put a zone file similar to the counterpart on external but holding the internal IPs:

```
; example.com

$TTL 604800

@ IN SOA ns1.example.com. root.example.com. (

2006020201 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800); Negative Cache TTL
```

| a | IN | Α | 10.0.0.1 |
|--------|------|---|------------|
| boss | IN | А | 10.0.0.100 |
| printe | r IN | А | 10.0.0.101 |
| scrtry | IN | А | 10.0.0.102 |
| sip01 | IN | А | 10.0.0.201 |
| lab | IN | А | 10.0.0.103 |

Great, we can now ping our boss' computer with

ping boss.example.com

but trying to reach mail.example.com will disappoint us, what happened ? There's no reference to mail.example.com on the internal zone file and since we are in the internal network we can resolve mail.example.com. Fine, let's just copy the contents of the external zone file to the internal zone file. That'll work.

But we are a small, smart start up, we can do better than copy-paste each modification to the zone file, furthermore, that is very error prone (will you always remember to modify the internal zone file when you modify the external one, or will you forget and spend some days debugging network problems ?).

What we will do is include the external zone file in the internal file this way:

\$include "/etc/bind/external/db.example.com"

| (a) | IN | Α | 10.0.0.1 |
|--------|------|---|------------|
| boss | IN | Α | 10.0.0.100 |
| printe | r IN | Α | 10.0.0.101 |
| scrtry | IN | Α | 10.0.0.102 |
| sip01 | IN | Α | 10.0.201 |
| lab | IN | А | 10.0.0.103 |

and voila! Finding the \$include directive in the current documentation was hard.. Just remember to change the serial of the external zone file whenever you change the internal one, but it is not a big deal, if you forget, nothing bad will happen since you are not likely to have caching servers inside your own small network.

10.34 SECURITY

It is not recommended to use the same DNS server as primary for some domain (in our case example.com) and as caching DNS server, but in our case we are forced to do it. From the outside world 192.0.2.1 is the primary DNS server for example.com, from our own internal network, 192.0.2.1 (or its private address, 10.0.0.1) is our caching name server that should be configured as the nameserver to use on each workstation we have.

One of the problems is that someone might start using our caching nameserver from the outside, there's an attack called cache-poisoning and many other nasty things that can be done and are documented on [SINS] (including how to avoid them).

To improve our security a bit, we'll add a couple of directives to the configuration file:

```
view "internal" {
```

match-clients { internals; };

recursion yes; zone "example.com" { type master; file
"/etc/bind/internals/db.example.com"; }; }; view "external" { matchclients { any; }; recursion no; zone "example.com" { type master;
file "/etc/bind/externals/db.example.com"; allow-transfer { slaves; }; };
};

That will prevent anyone on the dangerous Internet to use our server recursively while we, on our own network, can still do it.

10.45 CONFIGURATION FILES

10.4.15.1 /ETC/BIND/NAMED.CONF.LOCAL

```
acl slaves {
    195.234.42.0/24; // XName
    193.218.105.144/28; // XName
    193.24.212.232/29; // XName
};
acl internals {
    127.0.0.0/8;
```

```
10.0.0/24;
```

```
};
```

```
view "internal" {
  match-clients { internals; };
  recursion yes;
  zone "example.com" {
    type master;
    file "/etc/bind/internals/db.example.com";
  };
};
view "external" {
  match-clients { any; };
  recursion no;
  zone "example.com" {
    type master;
    file "/etc/bind/externals/db.example.com";
    allow-transfer { slaves; };
  };
};
10.4.25.2 /ETC/BIND/EXTERNALS/DB.EXAMPLE.COM
; example.com
$TTL 604800
                   ns1.example.com. root.example.com. (
(a)
      IN
            SOA
            2006020201 ; Serial
              604800 ; Refresh
               86400; Retry
              2419200 ; Expire
              604800); Negative Cache TTL
            NS
(a)
      IN
                  ns1
          MX
                  10 mail
    IN
    IN
          Α
                192.0.2.1
      IN
                  192.0.2.1
ns1
            Α
mail
      IN
            А
                  192.0.2.128 ; We have our mail server somewhere else.
WWW
        IN
              Α
                   192.0.2.1
client1 IN
                  192.0.2.201 ; We connect to client1 very often.
             А
```

10.4.35.3 /ETC/BIND/INTERNALS/DB.EXAMPLE.COM

\$include "/etc/bind/external/db.example.com"

| (a) | IN | А | 10.0.0.1 |
|--------|------|---|------------|
| boss | IN | А | 10.0.0.100 |
| printe | r IN | А | 10.0.0.101 |
| scrtry | IN | А | 10.0.0.102 |
| sip01 | IN | А | 10.0.0.201 |
| lab | IN | А | 10.0.0.103 |

10.4.4 BIBLIOGRAPHY

SINS

Securing an Internet Name Server

10.4.5 FOOTNOTES

 \dots IPs¹

For me, two computers already qualify as too many computers to remember their IPs.

... files^{$\frac{2}{2}$}

The host file resides on /etc/hosts and is a simple mapping for the local computer from names to IP. It was the first way to resolve names to IPs and long ago a central big hosts file was maintained, latter distributed by FTP or similar. It rapidly grow into a problem which was solved by the invention of DNS. Try not to let your hosts file grow into a problem before you turn it into DNS, learn from the pioneers!

 $\dots \text{ server}^{\underline{3}}$

Or when you are a perfectionist and believe that this should be doable with only one server.

.../etc/bind/named.conf.local⁴

It may be just /etc/bind/named.conf in your operating system if it is not Debian.

Conctact us at <u>adblock@ba.net</u>

BA.NET ADBLOCK ADMINISTRATOR MANUAL

11 DOMAIN NAME SERVER (DNS) CONFIGURATION AND ADMINISTRATION

At my place of employment, we are using Linux as a DNS server. It performs exceptionally well. This section will address configuration of DNS tables for these services using the BIND 8.x package which comes standard with the Red Hat distribution.

Note: Note: Red Hat versions 5.1 and earlier used the BIND 4.x package, which used a slightly different format for its configuration file. BIND 8.x offers more functionality over that offered by BIND 4.x, and as 4.x is no longer being developed, you should probably consider upgrading your BIND package to the latest version. Simply install the BIND RPM package (see

<u>Section 10.1</u> for details on using the RPM utility), then convert your configuration file to the new format.

Fortunately, converting your existing BIND 4.x configuration file to be compliant with BIND 8.x is easy! In the documentation directory provided as part of BIND (for example, ``/usr/doc/bind-8.1.2/" for BIND version 8.1.2), there exists a file called ``named-bootconf.pl", which is an executable Perl program. Assuming you have Perl installed on your system, you can use this program to convert your configuration file. To do so, type the following commands (as root):

cd /usr/doc/bind-8.1.2 ./named-bootconf.pl < /etc/named.boot > /etc/named.conf mv /etc/named.boot /etc/named.boot-obsolete

You should now have an ``/etc/named.conf" file which should work with BIND 8.x "out-of-the-box". Your existing DNS tables will work as-is with the new version of BIND, as the format of the tables remains the same.

Configuration of DNS services under Linux involves the following steps:

1. To enable DNS services, the ``/etc/host.conf' file should look like this:

Lookup names via /etc/hosts first, then by DNS query order hosts, bind # We don't have machines with multiple addresses multi on # Check for IP address spoofing nospoof on # Warn us if someone attempts to spoof alert on

- 2. The extra spoof detection adds a bit of a performance hit to DNS lookups (although negligible), so if you're not too worried about this you may wish to disable the "nospool" and "alert" entries.
- 3. Configure the ``/etc/hosts" file as needed. Typically there doesn't need to be much in here, but for improved performance you can add any hosts you access often (such as local servers) to avoid performing DNS lookups on them.
- 4. The ``/etc/named.conf' file should be configured to point to your DNS tables according to the example below.

Note: (Note: IP addresses shown are examples only and must be replaced with your own class addresses!):

options { // DNS tables are located in the /var/named directory firewall between you and nameservers you want * to talk to, you might need to uncomment the query-source * directive below. Previous versions of BIND always asked * questions using port 53, but BIND 8.1 uses an unprivileged * port by default. */ // querysource address * port 53; }; // Enable caching and load root server info zone "named.root" { type hint; file ""; }; // All our DNS information is stored in /var/named/mydomain_name.db // (eg. if mydomain.name = foobar.com then use foobar_com.db) zone "mydomain.name" { type master; file "mydomain_name.db"; allow-transfer { 123.12.41.40; }; }; // Reverse lookups for 123.12.41.*, .42.*, .43.*, .44.* class C's // (these are example Class C's only -- do not use!) zone "12.123.IN-ADDR.ARPA" { type master; file "123_12.rev"; allow-transfer { 123.12.41.40; }; }; // Reverse lookups for 126.27.18.*, .19.*, .20.* class C's // (these are example Class C's only -- do not use!) zone "27.126.IN-ADDR.ARPA" { type master; file "126_27.rev"; allow-transfer { 123.12.41.40; }; };

Tip: Tip: Make note of the allow-transfer options above, which restricts DNS zone transfers to a given IP address. In our example, we are allowing the host at 123.12.41.40 (probably a slave DNS server in our domain) to request zone transfers. If you omit this option, anyone on the Internet will be able to request such transfers. As the information provided is often used by spammers and IP spoofers, I strongly recommend you restrict zone transfers except to your slave DNS server(s), or use the loopback address, ``127.0.0.1" instead.

5. Now you can set up your DNS tables in the ``var/named/" directory as configured in the ``/etc/named.conf" file in step three. Configuring DNS database files for the first time is a major undertaking, and is beyond the scope of this document. There are several guides, online and in printed form that should be referred to. However, several examples are provided below.

Sample entries in the ``/var/named/mydomain_name.db" forward lookup file:

; This is the Start of Authority (SOA) record. Contains contact ; & other information about the name server. The serial number ; must be changed whenever the file is updated (to inform secondary; servers that zone information has changed). (a) IN SOA mydomain.name. postmaster.mydomain.name. (19990811 ; Serial number 3600 will go to our ISP's name server isp.domain.name.com IN NS mail.mydomain.name.; Provides IN MX priority) optional information on the machine type & operating system; used for the server IN HINFO Pentium/350 LINUX ; A list of

machine names & addresses spock.mydomain.name. IN A 123.12.41.40 ; OpenVMS Alpha mail.mydomain.name. IN A 123.12.41.41 ; Linux (main server) kirk.mydomain.name. IN A 123.12.41.42 ; Windows NT (blech!) ; Including any in our other class C's twixel.mydomain.name. IN A 126.27.18.161 ; Linux test machine foxone.mydomain.name. IN A 126.27.18.162 ; Linux devel. kernel ; Alias (canonical) names gopher IN CNAME

Sample entries in the ``/var/named/123_12.rev" reverse lookup file:

; This is the Start of Authority record. Same as in forward lookup table. (a) IN SOA mydomain.name. postmaster.mydomain.name. (forward lookup table IN NS mail.mydomain.name. IN NS isp.domain.name.com.; A list of machine names & addresses, in reverse. We are mapping; more than one class C here, so we need to list the class B portion ; as well. 40.41 IN PTR spock.mydomain.name. 41.41 IN PTR mail.mydomain.name. 42.41 IN PTR kirk.mydomain.name. ; As you can see, we can map our other class C's as long as they are ; under the 123.12.* class B addresses 24.42 IN PTR tsingtao.mydomain.name. 250.42 obvious reasons).

Any other reverse lookup files needed to map addresses in a different class B (such as 126.27.*) can be created, and would look much the same as the example reverse lookup file above.

- 6. Make sure the named daemon is running. This daemon is usually started from the ``/etc/rc.d/init.d/named" file upon system boot. You can also start and stop the daemon manually; type ``named start" and ``named stop", respectively.
- 7. Whenever changes are made to the DNS tables, the DNS server should be restarted by typing ``/etc/rc.d/init.d/named restart". You may then wish to test your changes by using a tool such as "nslookup" to query the machine you have added or changed.

12 RESPONSE RATE LIMITING

RRL, or Response Rate Limiting, is an enhancement to the DNS protocol which serves as a tool for mitigating <u>DNS amplification attacks</u>.

At this time, RRL implementation is only recommended for authoritative servers.

How can I get RRL?

Response Rate Limiting functionality is included in open source BIND from version 9.9.4 onwards (it is enabled when building BIND by specifying configure option --enable-rrl). RRL is already available in subscription versions of BIND - for more information about our subscription versions, please visit <u>http://www.dns-co.com/solutions/</u>. Unsupported patches that add RRL functionality to earlier versions of BIND can be downloaded from <u>http://www.redbarn.org/dns/ratelimits</u>.

12.1 THE PROBLEM

DNS is easily used for reflected denial-of-service (DOS) attacks, with three factors combining to make it a popular choice.

- 1. UDP, which is commonly used for DNS traffic, was not designed with source validation in mind. Consequently BIND responds to packets with a forged source the same as it does to legitimate packets. An attacker can therefore send DNS queries forging an IP address of the victim as the source address, causing the DNS server to send the replies to the victim. This is a "reflected attack".
- 2. Most ISPs do not check the source address of packets that they send to ensure that the source address matches a network block managed by that ISP. This allows forged-address attacks to be launched from a large portion of the Internet.
- 3. Small DNS queries can generate large responses, allowing the attacker to send a lot less traffic than the victim receives, amplifying the attack. For example, an EDNS0 query for isc.org of type ANY is 36 bytes long (not counting the UDP, IP, and Ethernet headers) and triggers a response that is 3,576 bytes log (not counting UDP, IP, and Ethernet headers.) By reflecting, an attacker can cause a nearly 100x

increase in the amount of traffic that they are directing at the victim and they can conceal the source of the attack as well.

12.2 A SOLUTION

While we cannot know which source addresses are forged and which are not, we can look at the pattern of requests and responses and infer with a high degree of confidence when there is an attack. We can then use this information to throttle responses, cutting off the attack. Incoming queries are NOT throttled by RRL.

12.3 THE RESULTS

Operators of large authoritative servers have reported huge reductions in network traffic and server load after enabling RRL. Additionally, these servers are no longer seen as participating in abusive network behavior as fewer illegitimate responses are reaching their intended targets. The impact on legitimate traffic has been minimal.

12.4 SAMPLE BIND RRL CONFIGURATION

To enable RRL, a rate-limit clause must be added to an options or view statement within named.conf.

As a very simple example scenario, let's say your authoritative server for example.com is being flooded with queries for the address record of testhost.example.com with the DO (DNSSEC OK) bit set. example.com is DNSSEC-signed so our reply packet size will be somewhat large. In the query log, you are seeing something like this:

07-Jun-2013 12:27:34.102 queries: info: client 1.2.3.4#58540 (testhost.example.com): query: testhost.example.com IN A +ED (1.2.3.4) 07-Jun-2013 12:27:41.606 queries: info: client 1.2.3.4#55979 (testhost.example.com): query: testhost.example.com IN A +ED (1.2.3.4) 07-Jun-2013 12:27:59.196 queries: info: client 1.2.3.4#47516 (testhost.example.com): query: testhost.example.com IN A +ED (1.2.3.4)

The queries are appearing to originate from 1.2.3.4 but given the frequency of the repeated query, it would be reasonable to suspect that these queries are spoofed and 1.2.3.4 is the target of a DDOS attack.

If this is your first use of RRL, perhaps you would like to test its behavior before actually limiting responses -- you can see what RRL would do before actually having it limit responses. You can do this by setting "log-only yes" during configuration. Edit named.conf and add the following rate-limit clause in your global options:

```
options {
    directory "/var/named";
    rate-limit {
        responses-per-second 5;
        log-only yes;
    };
};
```

After a configuration reload, you will now see messages such as the following in your logs:

07-Jun-2013 12:41:42.336 queries: info: client 1.2.3.4#53459 (testhost.example.com): query: testhost.example.com IN A +ED (1.2.3.4) 07-Jun-2013 12:41:42.336 query-errors: info: client 1.2.3.4#53459 (testhost.example.com): would rate limit slip response to 1.2.3.0/24 for testhost.example.com IN A (3ee9836b) When you feel comfortable with what RRL will actually be limiting, remove the "log-only yes" and after a reload, you will begin to see: 07-Jun-2013 12:44:44.868 queries: info: client 1.2.3.4#57114 (testhost.example.com): query: testhost.example.com IN A +ED (1.2.3.4) 07-Jun-2013 12:44:44.869 query-errors: info: client 1.2.3.4#57114 (testhost.example.com): rate limit drop response to 1.2.3.0/24 for testhost.example.com IN A (3ee9836b)

RRL is highly configurable to combat many attack scenarios. We recommend reading the Response Rate Limiting section of the <u>BIND9 Adminstrator</u> <u>Reference Manual (ARM)</u> for an in-depth review of the RRL configuration options.
13 OPENVPN

13.1 DETERMINING WHETHER TO USE A ROUTED OR BRIDGED VPN

See <u>FAQ</u> for an overview of Routing vs. Ethernet Bridging. See also the OpenVPN <u>Ethernet Bridging</u> page for more notes and details on bridging.

Overall, routing is probably a better choice for most people, as it is more efficient and easier to set up (as far as the OpenVPN configuration itself) than bridging. Routing also provides a greater ability to selectively control access rights on a client-specific basis.

I would recommend using routing unless you need a specific feature which requires bridging, such as:

- the VPN needs to be able to handle non-IP protocols such as IPX,
- you are running applications over the VPN which rely on network broadcasts (such as LAN games), or
- you would like to allow browsing of Windows file shares across the VPN without setting up a Samba or WINS server.

13.2 NUMBERING PRIVATE SUBNETS

Setting up a VPN often entails linking together private subnets from different locations.

The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets (codified in RFC 1918):

| 10.0.0.0 | 10.255.255.255 | (10/8 prefix) |
|-------------|-----------------|---------------------|
| 172.16.0.0 | 172.31.255.255 | (172.16/12 prefix) |
| 192.168.0.0 | 192.168.255.255 | (192.168/16 prefix) |

While addresses from these netblocks should normally be used in VPN configurations, it's important to select addresses that minimize the probability of IP address or subnet conflicts. The types of conflicts that need to be avoided are:

- conflicts from different sites on the VPN using the same LAN subnet numbering, or
- remote access connections from sites which are using private subnets which conflict with your VPN subnets.

For example, suppose you use the popular 192.168.0.0/24 subnet as your private LAN subnet. Now you are trying to connect to the VPN from an internet cafe which is using the same subnet for its WiFi LAN. You will have a routing conflict because your machine won't know if 192.168.0.1 refers to the local WiFi gateway or to the same address on the VPN.

As another example, suppose you want to link together multiple sites by VPN, but each site is using 192.168.0.0/24 as its LAN subnet. This won't work without adding a complexifying layer of NAT translation, because the VPN won't know how to route packets between multiple sites if those sites don't use a subnet which uniquely identifies them.

The best solution is to avoid using 10.0.0/24 or 192.168.0.0/24 as private LAN network addresses. Instead, use something that has a lower probability of being used in a WiFi cafe, airport, or hotel where you might expect to connect from remotely. The best candidates are subnets in the middle of the vast 10.0.0.0/8 netblock (for example 10.66.77.0/24).

And to avoid cross-site IP numbering conflicts, always use unique numbering for your LAN subnets.

13.3 SETTING UP YOUR OWN CERTIFICATE AUTHORITY (CA) AND GENERATING CERTIFICATES AND KEYS FOR AN OPENVPN SERVER AND MULTIPLE CLIENTS

13.3.1 OVERVIEW

The first step in building an OpenVPN 2.x configuration is to establish a PKI (public key infrastructure). The PKI consists of:

- a separate certificate (also known as a public key) and private key for the server and each client, and
- a master Certificate Authority (CA) certificate and key which is used to sign each of the server and client certificates.

OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established.

Both server and client will authenticate the other by first verifying that the presented certificate was signed by the master certificate authority (CA), and then by testing information in the now-authenticated certificate header, such as the certificate common name or certificate type (client or server).

This security model has a number of desirable features from the VPN perspective:

- The server only needs its own certificate/key -- it doesn't need to know the individual certificates of every client which might possibly connect to it.
- The server will only accept clients whose certificates were signed by the master CA certificate (which we will generate below). And because the server can perform this signature verification without needing access to the CA private key itself, it is possible for the CA key (the most sensitive key in the entire PKI) to reside on a completely different machine, even one without a network connection.
- If a private key is compromised, it can be disabled by adding its certificate to a CRL (certificate revocation list). The CRL allows compromised certificates to be selectively rejected without requiring that the entire PKI be rebuilt.
- The server can enforce client-specific access rights based on embedded certificate fields, such as the Common Name.

Note that the server and client clocks need to be roughly in sync or certificates might not work properly.

13.3.2 GENERATE THE MASTER CERTIFICATE AUTHORITY (CA) CERTIFICATE & KEY

In this section we will generate a master CA certificate/key, a server certificate/key, and certificates/keys for 3 separate clients.

For PKI management, we will use EASY-RSA, a set of scripts which is bundled with OpenVPN 2.2.x and earlier. If you're using OpenVPN 2.3.x, you need to download easy-rsa separately from <u>here</u>.

If you are using Linux, BSD, or a unix-like OS, open a shell and cd to the **easy-rsa** subdirectory. If you installed OpenVPN from an RPM or DEB file, the easy-rsa directory can usually be found in /usr/share/doc/packages/openvpn or /usr/share/doc/openvpn (it's best to copy this directory to another location such as /etc/openvpn, before any edits, so that future OpenVPN package upgrades won't overwrite your modifications). If you installed from a .tar.gz file, the easy-rsa directory will be in the top level directory of the expanded source tree.

If you are using Windows, open up a Command Prompt window and cd to **Program Files\OpenVPN\easy-rsa**. Run the following batch file to copy configuration files into place (this will overwrite any preexisting vars.bat and openssl.cnf files):

init-config

Now edit the **vars** file (called **vars.bat** on Windows) and set the KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG, and KEY_EMAIL parameters. Don't leave any of these parameters blank.

Next, initialize the PKI. On Linux/BSD/Unix:

```
. ./vars
./clean-all
./build-ca
```

On Windows:

```
vars
clean-all
build-ca
```

The final command (**build-ca**) will build the certificate authority (CA) certificate and key by invoking the interactive **openssl** command:

```
ai:easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
....++++++
. . . . . . . . . . . +++++++
writing new private key to 'ca.key'
You are about to be asked to enter information that
will be incorporated
into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave some
blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [KG]:
State or Province Name (full name) [NA]:
Locality Name (eq, city) [BISHKEK]:
Organization Name (eg, company) [OpenVPN-TEST]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's
hostname) []:OpenVPN-CA
Email Address [me@myhost.mydomain]:
```

Note that in the above sequence, most queried parameters were defaulted to the values set in the **vars** or **vars.bat** files. The only parameter which must be explicitly entered is the **Common Name**. In the example above, I used "OpenVPN-CA".

13.3.3 GENERATE CERTIFICATE & KEY FOR SERVER

Next, we will generate a certificate and private key for the server. On Linux/BSD/Unix:

./build-key-server server

On Windows:

build-key-server server

As in the previous step, most parameters can be defaulted. When the **Common Name** is queried, enter "server". Two other queries require positive responses, "Sign the certificate? [y/n]" and "1 out of 1 certificate requests certified, commit? [y/n]".

13.3.4 GENERATE CERTIFICATES & KEYS FOR 3 CLIENTS

Generating client certificates is very similar to the previous step. On Linux/BSD/Unix:

./build-key client1
./build-key client2
./build-key client3

On Windows:

build-key client1
build-key client2
build-key client3

If you would like to password-protect your client keys, substitute the **build-key-pass** script.

Remember that for each client, make sure to type the appropriate **Common Name** when prompted, i.e. "client1", "client2", or "client3". Always use a unique common name for each client.

13.3.5 GENERATE DIFFIE HELLMAN PARAMETERS

Diffie Hellman parameters must be generated for the OpenVPN server. On Linux/BSD/Unix:

./build-dh

On Windows:

build-dh

```
Output:
ai:easy-rsa # ./build-dh
Generating DH parameters, 1024 bit long safe prime,
generator 2
This is going to take a long time
.....+...+...+
```

13.3.6KEY FILES

Now we will find our newly-generated keys and certificates in the **keys** subdirectory. Here is an explanation of the relevant files:

| Filename | Needed By | Purpose | Secret |
|-------------|-----------------------------|------------------------------|--------|
| ca.crt | server + all clients | Root CA certificate | NO |
| ca.key | key signing machine only | Root CA key | YES |
| dh{n}.pem | server only | Diffie Hellman parameters | NO |
| server.crt | server only | Server Certificate | NO |
| server.key | server only | Server Key | YES |
| client1.crt | client1 only | Client1 Certificate | NO |
| client1.key | client1 only | Client1 Key | YES |
| client2.crt | client2 only | Client2 Certificate | NO |
| client2.key | client2 only | Client2 Key | YES |
| client3.crt | client3 only | Client3 Certificate | NO |
| client3.key | client3 only | Client3 Key | YES |

The final step in the key generation process is to copy all files to the machines which need them, taking care to copy secret files over a secure channel.

Now wait, you may say. Shouldn't it be possible to set up the PKI without a pre-existing secure channel?

The answer is ostensibly yes. In the example above, for the sake of brevity, we generated all private keys in the same place. With a bit more effort, we could have done this differently. For example, instead of generating the client certificate and keys on the server, we could have had the client generate its own private key locally, and then submit a Certificate Signing Request (CSR) to the key-signing machine. In turn, the key-signing machine could have processed the CSR and returned a signed certificate to the client. This could have been done without ever requiring that a secret **.key** file leave the hard drive of the machine on which it was generated.

13.4 CREATING CONFIGURATION FILES FOR SERVER AND CLIENTS

13.4.1 GETTING THE SAMPLE CONFIG FILES

It's best to use the OpenVPN <u>sample configuration files</u> as a starting point for your own configuration. These files can also be found in

- the **sample-config-files** directory of the OpenVPN source distribution
- the sample-config-files directory in /usr/share/doc/packages/openvpn or /usr/share/doc/openvpn if you installed from an RPM or DEB package
- Start Menu -> All Programs -> OpenVPN -> OpenVPN Sample Configuration Files on Windows

Note that on Linux, BSD, or unix-like OSes, the sample configuration files are named **server.conf** and **client.conf**. On Windows they are named **server.ovpn** and **client.ovpn**.

13.4.2 EDITING THE SERVER CONFIGURATION FILE

The sample server configuration file is an ideal starting point for an OpenVPN server configuration. It will create a VPN using a virtual **TUN** network interface (for routing), will listen for client connections on **UDP port 1194** (OpenVPN's official port number), and distribute virtual addresses to connecting clients from the **10.8.0.0/24** subnet.

Before you use the sample configuration file, you should first edit the **ca**, **cert**, **key**, and **dh** parameters to point to the files you generated in the <u>PKI</u> section above.

At this point, the server configuration file is usable, however you still might want to customize it further:

- If you are using <u>Ethernet bridging</u>, you must use **server-bridge** and **dev tap** instead of **server** and **dev tun**.
- If you want your OpenVPN server to listen on a TCP port instead of a UDP port, use **proto tcp** instead of **proto udp** (If you want OpenVPN to listen on both a UDP and TCP port, you must run two separate OpenVPN instances).
- If you want to use a virtual IP address range other than **10.8.0.0/24**, you should modify the **server** directive. Remember that this virtual IP address range should be a private range which is currently unused on your network.
- Uncomment out the **client-to-client** directive if you would like connecting clients to be able to reach each other over the VPN. By default, clients will only be able to reach the server.
- If you are using Linux, BSD, or a Unix-like OS, you can improve security by uncommenting out the **user nobody** and **group nobody** directives.

If you want to run multiple OpenVPN instances on the same machine, each using a different configuration file, it is possible if you:

- Use a different **port** number for each instance (the UDP and TCP protocols use different port spaces so you can run one daemon listening on UDP-1194 and another on TCP-1194).
- If you are using Windows, each OpenVPN configuration taneeds to have its own TAP-Windows adapter. You can add additional adapters by going to Start Menu -> All Programs -> TAP-Windows -> Add a new TAP-Windows virtual ethernet adapter.
- If you are running multiple OpenVPN instances out of the same directory, make sure to edit directives which create output files so that multiple instances do not overwrite each other's output files. These directives include log, log-append, status, and ifconfig-pool-persist.

13.4.3 EDITING THE CLIENT CONFIGURATION FILES

The sample client configuration file (**client.conf** on Linux/BSD/Unix or **client.ovpn** on Windows) mirrors the default directives set in the sample server configuration file.

- Like the server configuration file, first edit the **ca**, **cert**, and **key** parameters to point to the files you generated in the <u>PKI</u> section above. Note that each client should have its own **cert/key** pair. Only the **ca**file is universal across the OpenVPN server and all clients.
- Next, edit the **remote**directive to point to the hostname/IP address and port number of the OpenVPN server (if your OpenVPN server will be running on a single-NIC machine behind a firewall/NAT-gateway, use the public IP address of the gateway, and a port number which you have configured the gateway to forward to the OpenVPN server).
- Finally, ensure that the client configuration file is consistent with the directives used in the server configuration. The major thing to check for is that the **dev** (tun or tap) and **proto** (udp or tcp) directives are consistent. Also make sure that **comp-lzo** and **fragment**, if used, are present in both client and server config files.

13.5 STARTING UP THE VPN AND TESTING FOR INITIAL CONNECTIVITY

13.5.1 STARTING THE SERVER

First, make sure the OpenVPN server will be accessible from the internet. That means:

- opening up UDP port 1194 on the firewall (or whatever TCP/UDP port you've configured), or
- setting up a port forward rule to forward UDP port 1194 from the

firewall/gateway to the machine running the OpenVPN server.

Next, make sure that the TUN/TAP interface is not firewalled.

To simplify troubleshooting, it's best to initially start the OpenVPN server from the command line (or right-click on the **.ovpn** file on Windows), rather than start it as a daemon or service:

openvpn [server config file]

A normal server startup should look like this (output will vary across platforms): Sun Feb 6 20:46:38 2005 OpenVPN 2.0 rc12 i686-suselinux [SSL] [LZO] [EPOLL] built on Feb 5 2005 Sun Feb 6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key Sun Feb 6 20:46:38 2005 TLS-Auth MTU parms [L:1542 D:138 EF:38 EB:0 ET:0 EL:0 1 Sun Feb 6 20:46:38 2005 TUN/TAP device tun1 opened Sun Feb 6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2 mtu 1500 Sun Feb 6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask 255.255.255.0 gw 10.8.0.2 Sun Feb 6 20:46:38 2005 Data Channel MTU parms [L:1542 D:1450 EF:42 EB:23 ET:0 EL:0 AF:3/1] Sun Feb 6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194 Sun Feb 6 20:46:38 2005 UDPv4 link remote: [undef] Sun Feb 6 20:46:38 2005 MULTI: multi init called, r=256 v=256 Sun Feb 6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62 Sun Feb 6 20:46:38 2005 IFCONFIG POOL LIST Sun Feb 6 20:46:38 2005 Initialization Sequence Completed

13.5.2 STARTING THE CLIENT

As in the server configuration, it's best to initially start the OpenVPN server from the command line (or on Windows, by right-clicking on the **client.ovpn** file), rather than start it as a daemon or service:

openvpn [client config file]

A normal client startup on Windows will look similar to the server output above, and should end with the **Initialization Sequence Completed** message.

Now, try a ping across the VPN from the client. If you are using routing (i.e. **dev tun** in the server config file), try:

ping 10.8.0.1

If you are using bridging (i.e. **dev tap** in the server config file), try to ping the IP address of a machine on the server's ethernet subnet.

If the ping succeeds, congratulations! You now have a functioning VPN.

13.5.3 TROUBLESHOOTING

If the ping failed or the OpenVPN client initialization failed to complete, here is a checklist of common symptoms and their solutions:

• You get the error message: **TLS Error: TLS key negotiation failed to occur within 60 seconds (check your network connectivity)**. This error indicates that the client was unable to establish a network connection with the server.

Solutions:

- Make sure the client is using the correct hostname/IP address and port number which will allow it to reach the OpenVPN server.
- If the OpenVPN server machine is a single-NIC box inside a protected LAN, make sure you are using a correct port forward rule on the server's gateway firewall. For example, suppose your

OpenVPN box is at 192.168.4.4 inside the firewall, listening for client connections on UDP port 1194. The NAT gateway servicing the 192.168.4.x subnet should have a port forward rule that says **forward UDP port 1194 from my public IP address to 192.168.4.4**.

- Open up the server's firewall to allow incoming connections to UDP port 1194 (or whatever TCP/UDP port you have configured in the server config file).
- You get the error message: **Initialization Sequence Completed with errors--** This error can occur on Windows if (a) You don't have the DHCP client service running, or (b) You are using certain third-party personal firewalls on XP SP2.

Solution: Start the DHCP client server and make sure that you are using a personal firewall which is known to work correctly on XP SP2.

• You get the **Initialization Sequence Completed**message but the ping test fails -- This usually indicates that a firewall on either server or client is blocking VPN network traffic by filtering on the TUN/TAP interface.

Solution: Disable the client firewall (if one exists) from filtering the TUN/TAP interface on the client. For example on Windows XP SP2, you can do this by going to Windows Security Center -> Windows Firewall -> Advanced and unchecking the box which corresponds to the TAP-Windows adapter (disabling the client firewall from filtering the TUN/TAP adapter is generally reasonable from a security perspective, as you are essentially telling the firewall not to block authenticated VPN traffic). Also make sure that the TUN/TAP interface on the server is not being filtered by a firewall (having said that, note that selective firewalling of the TUN/TAP interface on the server side can confer certain security benefits. See the access policies section below).

• The connection stalls on startup when using a **proto udp**configuration,

BA.NET ADBLOCK ADMINISTRATOR MANUAL

the server log file shows this line:

```
TLS: Initial packet from x.x.x.x:x, sid=xxxxxxxx xxxxxxxx
```

however the client log does not show an equivalent line.

Solution: You have a one-way connection from client to server. The server to client direction is blocked by a firewall, usually on the client side. The firewall can either be (a) a personal software firewall running on the client, or (b) the NAT router gateway for the client. Modify the firewall to allow returning UDP packets from the server to reach the client.

See the <u>FAQ</u> for additional troubleshooting information.

13.6 CONFIGURING OPENVPN TO RUN AUTOMATICALLY ON SYSTEM STARTUP

The lack of standards in this area means that most OSes have a different way of configuring daemons/services for autostart on boot. The best way to have this functionality configured by default is to install OpenVPN as a package, such as via RPM on Linux or using the Windows installer.

13.6.1 LINUX

If you install OpenVPN via an RPM or DEB package on Linux, the installer will set up an **initscript**. When executed, the initscript will scan for **.conf** configuration files in **/etc/openvpn**, and if found, will start up a separate OpenVPN daemon for each file.

13.6.2 WINDOWS

The Windows installer will set up a Service Wrapper, but leave it turned off by default. To activate it, go to Control Panel / Administrative Tools / Services, select the OpenVPN service, right-click on properties, and set the Startup Type to Automatic. This will configure the service for automatic start on the next reboot.

When started, the OpenVPN Service Wrapper will scan the **\Program Files\OpenVPN\config** folder for **.ovpn** configuration files, starting a separate OpenVPN process on each file.

13.7 CONTROLLING A RUNNING OPENVPN PROCESS

13.7.1 RUNNING ON LINUX/BSD/UNIX

OpenVPN accepts several signals:

- **SIGUSR1** -- Conditional restart, designed to restart without root privileges
- **SIGHUP** -- Hard restart
- SIGUSR2 -- Output connection statistics to log file or syslog
- **SIGTERM**, **SIGINT** -- Exit

Use the **writepid** directive to write the OpenVPN daemon's PID to a file, so that you know where to send the signal (if you are starting openvpn with an **initscript**, the script may already be passing a **--writepid** directive on the **openvpn** command line).

13.7.2 RUNNING ON WINDOWS AS A GUI

See the **OpenVPN GUI page**.

13.7.3 RUNNING IN A WINDOWS COMMAND PROMPT WINDOW

On Windows, you can start OpenVPN by right clicking on an OpenVPN configuration file (**.ovpn** file) and selecting "Start OpenVPN on this config file".

Once running in this fashion, several keyboard commands are available:

- **F1** -- Conditional restart (doesn't close/reopen TAP adapter)
- F2 -- Show connection statistics
- F3 -- Hard restart
- **F4** -- Exit

13.7.4 RUNNING AS A WINDOWS SERVICE

When OpenVPN is started as a service on Windows, the only way to control it is:

- Via the service control manager (Control Panel / Administrative Tools / Services) which gives start/stop control.
- Via the management interface (see below).

13.7.5 MODIFYING A LIVE SERVER CONFIGURATION

While most configuration changes require you to restart the server, there are two directives in particular which refer to files which can be dynamically updated on-the-fly, and which will take immediate effect on the server without needing to restart the server process.

client-config-dir -- This directive sets a client configuration directory, which the OpenVPN server will scan on every incoming connection, searching for a client-specific configuration file (see the <u>the manual page</u> for more information). Files in this directory can be updated on-the-fly, without restarting the server. Note that changes in this directory will only take effect for new connections, not existing connections. If you would like a clientspecific configuration file change to take immediate effect on a currently connected client (or one which has disconnected, but where the server has not timed-out its instance object), kill the client instance object by using the management interface (described below). This will cause the client to reconnect and use the new **client-config-dir** file.

crl-verify -- This directive names a CERTIFICATE REVOCATION LIST file, described below in the <u>Revoking Certificates</u> section. The CRL file can be modified on the fly, and changes will take effect immediately for new connections, or existing connections which are renegotiating their SSL/TLS channel (occurs once per hour by default). If you would like to kill a currently connected client whose certificate has just been added to the CRL, use the management interface (described below).

13.7.6 STATUS FILE

The default <u>server.conf</u> file has a line

status openvpn-status.log

which will output a list of current client connections to the file **openvpn-status.log** once per minute.

13.7.7 USING THE MANAGEMENT INTERFACE

The <u>OpenVPN management interface</u> allows a great deal of control over a running OpenVPN process. You can use the management interface directly, by telneting to the management interface port, or indirectly by using an <u>OpenVPN GUI</u> which itself connects to the management interface.

To enable the management interface on either an OpenVPN server or client, add this to the configuration file:

management localhost 7505

This tells OpenVPN to listen on TCP port 7505 for management interface clients (port 7505 is an arbitrary choice -- you can use any free port).

Once OpenVPN is running, you can connect to the management interface using a **telnet** client. For example:

```
ai:~ # telnet localhost 7505
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
>INFO:OpenVPN Management Interface Version 1 -- type
'help' for more info
help
Management Interface for OpenVPN 2.0 rc14 i686-suse-
linux [SSL] [LZO] [EPOLL] built on Feb 15 2005
Commands:
echo [on|off] [N|all]
                      : Like log, but only show
messages in echo buffer.
exit|quit
                       : Close management session.
                       : Print this message.
help
hold [on|off|release] : Set/show hold flag to
on/off state, or
                         release current hold and
start tunnel.
```

kill cn : Kill the client instance(s) having common name cn. : Kill the client instance kill IP:port connecting from IP:port. log [on|off] [N|all] : Turn on/off realtime log display + show last N lines or 'all' for entire history. mute [n] : Set log mute level to n, or show level if n is absent. : (Windows only) Show network net info and routing table. password type p : Enter password p for a queried OpenVPN password. : Send signal s to daemon, signal s s = SIGHUP | SIGTERM | SIGUSR1 | SIGUSR2. state [on|off] [N|all] : Like log, but show state history. : Show current daemon status status [n] info using format #n. test n : Produce n lines of output for testing/debugging. username type u : Enter username u for a queried OpenVPN username. verb [n] : Set log verbosity level to n, or show if n is absent. version : Show current version number. END exit Connection closed by foreign host. ai:~ # For more information, see the OpenVPN Management Interface Documentation.

13.8 EXPANDING THE SCOPE OF THE VPN TO INCLUDE ADDITIONAL MACHINES ON EITHER THE CLIENT OR SERVER SUBNET.

13.8.1 INCLUDING MULTIPLE MACHINES ON THE SERVER SIDE WHEN USING A ROUTED VPN (DEV TUN)

Once the VPN is operational in a point-to-point capacity between client and server, it may be desirable to expand the scope of the VPN so that clients can reach multiple machines on the server network, rather than only the server machine itself.

For the purpose of this example, we will assume that the server-side LAN uses a subnet of **10.66.0.0/24** and the VPN IP address pool uses **10.8.0.0/24** as cited in the **server** directive in the OpenVPN server configuration file.

First, you must ADVERTISE the **10.66.0.0/24** subnet to VPN clients as being accessible through the VPN. This can easily be done with the following server-side config file directive:

push "route 10.66.0.0 255.255.255.0"

Next, you must set up a route on the server-side LAN gateway to route the VPN client subnet (10.8.0.0/24) to the OpenVPN server (this is only necessary if the OpenVPN server and the LAN gateway are different machines).

Make sure that you've enabled \underline{IP} and $\underline{TUN/TAP}$ forwarding on the OpenVPN server machine.

13.8.2 INCLUDING MULTIPLE MACHINES ON THE SERVER SIDE WHEN USING A BRIDGED VPN (DEV TAP)

One of the benefits of using <u>ethernet bridging</u> is that you get this for free without needing any additional configuration.

13.8.3 INCLUDING MULTIPLE MACHINES ON THE CLIENT SIDE WHEN USING A ROUTED VPN (DEV TUN)

In a typical road-warrior or remote access scenario, the client machine connects to the VPN as a single machine. But suppose the client machine is a gateway for a local LAN (such as a home office), and you would like each machine on the client LAN to be able to route through the VPN.

For this example, we will assume that the client LAN is using the

192.168.4.0/24 subnet, and that the VPN client is using a certificate with a common name of **client2**. Our goal is to set up the VPN so that any machine on the client LAN can communicate with any machine on the server LAN through the VPN.

Before setup, there are some basic prerequisites which must be followed:

- The client LAN subnet (192.168.4.0/24 in our example) must not be exported to the VPN by the server or any other client sites which are using the same subnet. Every subnet which is joined to the VPN via routing must be unique.
- The client must have a unique Common Name in its certificate ("client2" in our example), and the **duplicate-cn** flag must not be used in the OpenVPN server configuration file.

First, make sure that \underline{IP} and $\underline{TUN/TAP}$ forwarding is enabled on the client machine.

Next, we will deal with the necessary configuration changes on the server side. If the server configuration file does not currently reference a client configuration directory, add one now:

client-config-dir ccd

In the above directive, **ccd** should be the name of a directory which has been pre-created in the default directory where the OpenVPN server daemon runs. On Linux this tends to be /**etc/openvpn** and on Windows it is usually **Program Files\OpenVPN\config**. When a new client connects to the OpenVPN server, the daemon will check this directory for a file which matches the common name of the connecting client. If a matching file is found, it will be read and processed for additional configuration file directives to be applied to the named client.

The next step is to create a file called **client2** in the **ccd** directory. This file should contain the line:

iroute 192.168.4.0 255.255.255.0

This will tell the OpenVPN server that the 192.168.4.0/24 subnet should be routed to **client2**.

Next, add the following line to the main server config file (not the ccd/client2 file):

route 192.168.4.0 255.255.255.0

Why the redundant **route** and **iroute** statements, you might ask? The reason is that **route** controls the routing from the kernel to the OpenVPN server (via the TUN interface) while **iroute** controls the routing from the OpenVPN server to the remote clients. Both are necessary.

Next, ask yourself if you would like to allow network traffic between client2's subnet (192.168.4.0/24) and other clients of the OpenVPN server. If so, add the following to the server config file.

client-to-client push "route 192.168.4.0 255.255.255.0"

This will cause the OpenVPN server to ADVERTISE client2's subnet to other connecting clients.

The last step, and one that is often forgotten, is to add a route to the server's LAN gateway which directs 192.168.4.0/24 to the OpenVPN server box (you won't need this if the OpenVPN server box IS the gateway for the server LAN). Suppose you were missing this step and you tried to ping a machine (not the OpenVPN server itself) on the server LAN from 192.168.4.8? The outgoing ping would probably reach the machine, but then it wouldn't know how to route the ping reply, because it would have no idea how to reach 192.168.4.0/24. The rule of thumb to use is that when routing entire LANs through the VPN (when the VPN server is not the same machine as the LAN gateway), make sure that the gateway for the LAN routes all VPN subnets to the VPN server machine.

Similarly, if the client machine running OpenVPN is not also the gateway for the client LAN, then the gateway for the client LAN must have a route which directs all subnets which should be reachable through the VPN to the OpenVPN client machine.

13.8.4 INCLUDING MULTIPLE MACHINES ON THE CLIENT SIDE WHEN USING A BRIDGED VPN (DEV TAP)

This requires a more complex setup (maybe not more complex in practice, but more complicated to explain in detail):

- You must bridge the client TAP interface with the LAN-connected NIC on the client.
- You must manually set the IP/netmask of the TAP interface on the client.
- You must configure client-side machines to use an IP/netmask that is inside of the bridged subnet, possibly by <u>querying a DHCP server on</u> the OpenVPN server side of the VPN.

13.9 PUSHING DHCP OPTIONS TO CLIENTS

The OpenVPN server can push DHCP options such as DNS and WINS server addresses to clients (some <u>caveats</u> to be aware of). Windows clients can accept pushed DHCP options natively, while non-Windows clients can accept them by using a client-side **up** script which parses the **foreign_option_N** environmental variable list. See the <u>man page</u> or <u>openvpn-users mailing list</u> archive for non-Windows **foreign_option_N** documentation and script examples.

For example, suppose you would like connecting clients to use an internal DNS server at 10.66.0.4 or 10.66.0.5 and a WINS server at 10.66.0.8. Add this to the OpenVPN server configuration:

```
push "dhcp-option DNS 10.66.0.4"
push "dhcp-option DNS 10.66.0.5"
push "dhcp-option WINS 10.66.0.8"
```

To test this feature on Windows, run the following from a command prompt window after the machine has connected to an OpenVPN server:

ipconfig /all

The entry for the TAP-Windows adapter should show the DHCP options which were pushed by the server.

13.10 CONFIGURING CLIENT-SPECIFIC RULES AND ACCESS

POLICIES

Suppose we are setting up a company VPN, and we would like to establish separate access policies for 3 different classes of users:

- System administrators -- full access to all machines on the network
- Employees -- access only to Samba/email server
- **Contractors** -- access to a special server only

The basic approach we will take is (a) segregate each user class into its own virtual IP address range, and (b) control access to machines by setting up firewall rules which key off the client's virtual IP address.

In our example, suppose that we have a variable number of employees, but only one system administrator, and two contractors. Our IP allocation approach will be to put all employees into an IP address pool, and then allocate fixed IP addresses for the system administrator and contractors.

Note that one of the prerequisites of this example is that you have a software firewall running on the OpenVPN server machine which gives you the ability to define specific firewall rules. For our example, we will assume the firewall is Linux **iptables**.

| Class | Virtual IP Range | Allowed LAN Access | Common Names |
|--------------------------|---------------------|------------------------------------|-----------------------------|
| Employees | 10.8.0.0/24 | Samba/email server at 10.66.4.4 | [variable] |
| System Administrators | 10.8.1.0/24 | Entire 10.66.4.0/24 subnet | sysadmin1 |
| Contractors | 10.8.2.0/24 | Contractor server at 10.66.4.12 | contractor1, contracter2 |

First, let's create a virtual IP address map according to user class:

Next, let's translate this map into an OpenVPN server configuration. First of

all, make sure you've followed the steps <u>above</u> for making the 10.66.4.0/24 subnet available to all clients (while we will configure routing to allow client access to the entire 10.66.4.0/24 subnet, we will then impose access restrictions using firewall rules to implement the above policy table).

First, define a static unit number for our **tun** interface, so that we will be able to refer to it later in our firewall rules:

dev tun0

In the server configuration file, define the Employee IP address pool:

server 10.8.0.0 255.255.255.0

Add routes for the System Administrator and Contractor IP ranges:

route 10.8.1.0 255.255.255.0 route 10.8.2.0 255.255.255.0

Because we will be assigning fixed IP addresses for specific System Administrators and Contractors, we will use a client configuration directory:

client-config-dir ccd

Now place special configuration files in the **ccd** subdirectory to define the fixed IP address for each non-Employee VPN client.

13.10.1 CCD/SYSADMIN1

ifconfig-push 10.8.1.1 10.8.1.2

13.10.2 CCD/CONTRACTOR1

ifconfig-push 10.8.2.1 10.8.2.2

13.10.3 CCD/CONTRACTOR2

ifconfig-push 10.8.2.5 10.8.2.6

Each pair of **ifconfig-push** addresses represent the virtual client and server IP endpoints. They must be taken from successive /30 subnets in order to be compatible with Windows clients and the TAP-Windows driver. Specifically,

the last octet in the IP address of each endpoint pair must be taken from this set:

| [1, 2] | [5, 6] | [9, 10] | [13, 14] | [17, 18] |
|-----------|-----------|-----------|--------------------|-----------|
| [21, 22] | [25, 26] | [29, 30] | [33, 34] | [37, 38] |
| [41, 42] | [45, 46] | [49, 50] | [53, 54] | [57, 58] |
| [61, 62] | [65, 66] | [69, 70] | [73, 74] | [77, 78] |
| [81, 82] | [85, 86] | [89, 90] | [93, 94] | [97, 98] |
| [101,102] | [105,106] | [109,110] | [113,114] | [117,118] |
| [121,122] | [125,126] | [129,130] | [133,134] | [137,138] |
| [141,142] | [145,146] | [149,150] | [153 , 154] | [157,158] |
| [161,162] | [165,166] | [169,170] | [173 , 174] | [177,178] |
| [181,182] | [185,186] | [189,190] | [193,194] | [197,198] |
| [201,202] | [205,206] | [209,210] | [213,214] | [217,218] |
| [221,222] | [225,226] | [229,230] | [233,234] | [237,238] |
| [241,242] | [245,246] | [249,250] | [253,254] | |

This completes the OpenVPN configuration. The final step is to add firewall rules to finalize the access policy. For this example, we will use firewall rules in the Linux **iptables** syntax:

```
# Employee rule
iptables -A FORWARD -i tun0 -s 10.8.0.0/24 -d
10.66.4.4 -j ACCEPT
```

```
# Sysadmin rule
iptables -A FORWARD -i tun0 -s 10.8.1.0/24 -d
10.66.4.0/24 -j ACCEPT
```

Contractor rule iptables -A FORWARD -i tun0 -s 10.8.2.0/24 -d 10.66.4.12 -j ACCEPT

13.11 USING ALTERNATIVE AUTHENTICATION METHODS

OpenVPN 2.0 and later include a feature that allows the OpenVPN server to securely obtain a username and password from a connecting client, and to use that information as a basis for authenticating the client.

To use this authentication method, first add the **auth-user-pass** directive to the client configuration. It will direct the OpenVPN client to query the user for

a username/password, passing it on to the server over the secure TLS channel.

Next, configure the server to use an authentication plugin, which may be a script, shared object, or DLL. The OpenVPN server will call the plugin every time a VPN client tries to connect, passing it the username/password entered on the client. The authentication plugin can control whether or not the OpenVPN server allows the client to connect by returning a failure (1) or success (0) value.

13.11.1 USING SCRIPT PLUGINS

Script plugins can be used by adding the **auth-user-pass-verify** directive to the server-side configuration file. For example:

auth-user-pass-verify auth-pam.pl via-file

will use the **auth-pam.pl** perl script to authenticate the username/password of connecting clients. See the description of **auth-user-pass-verify** in the <u>manual page</u> for more information.

The **auth-pam.pl** script is included in the OpenVPN source file distribution in the **sample-scripts** subdirectory. It will authenticate users on a Linux server using a PAM authentication module, which could in turn implement shadow password, RADIUS, or LDAP authentication. **auth-pam.pl** is primarily intended for demonstration purposes. For real-world PAM authentication, use the **openvpn-auth-pam** shared object plugin described below.

13.11.2 USING SHARED OBJECT OR DLL PLUGINS

Shared object or DLL plugins are usually compiled C modules which are loaded by the OpenVPN server at run time. For example if you are using an RPM-based OpenVPN package on Linux, the **openvpn-auth-pam** plugin should be already built. To use it, add this to the server-side config file: **plugin /usr/share/openvpn/plugin/lib/openvpn-authpam.so login**

This will tell the OpenVPN server to validate the username/password entered by clients using the **login** PAM module.

For real-world production use, it's better to use the **openvpn-auth-pam**

plugin, because it has several advantages over the auth-pam.pl script:

- The shared object **openvpn-auth-pam** plugin uses a split-privilege execution model for better security. This means that the OpenVPN server can run with reduced privileges by using the directives **user nobody**, **group nobody**, and **chroot**, and will still be able to authenticate against the root-readable-only shadow password file.
- OpenVPN can pass the username/password to a plugin via virtual memory, rather than via a file or the environment, which is better for local security on the server machine.
- C-compiled plugin modules generally run faster than scripts.

If you would like more information on developing your own plugins for use with OpenVPN, see the **README** files in the **plugin** subdirectory of the OpenVPN source distribution.

To build the **openvpn-auth-pam** plugin on Linux, cd to the **plugin/auth-pam** directory in the OpenVPN source distribution and run **make**.

13.11.3 USING USERNAME/PASSWORD AUTHENTICATION AS THE ONLY FORM OF CLIENT AUTHENTICATION

By default, using **auth-user-pass-verify** or a username/password-checking **plugin** on the server will enable dual authentication, requiring that both client-certificate and username/password authentication succeed in order for the client to be authenticated.

While it is discouraged from a security perspective, it is also possible to disable the use of client certificates, and force username/password authentication only. On the server:

client-cert-not-required

Such configurations should usually also set:

username-as-common-name

which will tell the server to use the username for indexing purposes as it would use the Common Name of a client which was authenticating via a client certificate.

Note that **client-cert-not-required** will not obviate the need for a server certificate, so a client connecting to a server which uses **client-cert-not-required** may remove the **cert** and **key** directives from the client configuration file, but not the **ca** directive, because it is necessary for the client to verify the server certificate.

13.12 HOW TO ADD DUAL-FACTOR AUTHENTICATION TO AN OPENVPN CONFIGURATION USING CLIENT-SIDE SMART CARDS

Also see Article: The OpenVPN Smartcard HOWTO

- About dual-factor authentication
- What is PKCS#11?
- Finding PKCS#11 provider library.
- How to configure a cryptographic token
- How to modify an OpenVPN configuration to make use of cryptographic tokens
 - Determine the correct object.
 - Using OpenVPN with PKCS#11.
 - PKCS#11 implementation considerations.
 - OpenSC PKCS#11 provider.
- Difference between PKCS#11 and Microsoft Cryptographic API (CryptoAPI).

13.12.1 ABOUT DUAL-FACTOR AUTHENTICATION

Dual-factor authentication is a method of authentication that combines two elements: something you have and something you know.

Something you have should be a device that cannot be duplicated; such a device can be a cryptographic token that contains a private secret key. This private key is generated inside the device and never leaves it. If a user possessing this token attempts to access protected services on a remote network, the authorization process which grants or denies network access can establish, with a high degree of certainty, that the user seeking access is in

physical possession of a known, certified token.

Something you know can be a password presented to the cryptographic device. Without presenting the proper password you cannot access the private secret key. Another feature of cryptographic devices is to prohibit the use of the private secret key if the wrong password had been presented more than an allowed number of times. This behavior ensures that if a user lost his device, it would be infeasible for another person to use it.

Cryptographic devices are commonly called "smart cards" or "tokens", and are used in conjunction with a PKI (Public Key Infrastructure). The VPN server can examine a X.509 certificate and verify that the user holds the corresponding private secret key. Since the device cannot be duplicated and requires a valid password, the server is able to authenticate the user with a high degree of confidence.

Dual-factor authentication is much stronger than password-based authentication, because in the worst-case scenario, only one person at a time can use the cryptographic token. Passwords can be guessed and can be exposed to other users, so in the worst-case scenario an infinite number of people could attempt to gain unauthorized access when resources are protected using password-only authentication.

If you store the secret private key in a file, the key is usually encrypted by a password. The problem with this approach is that the encrypted key is exposed to decryption attacks or spyware/malware running on the client machine. Unlike when using a cryptographic device, the file cannot erase itself automatically after several failed decryption attempts.

13.12.2 WHAT IS PKCS#11?

THIS STANDARD SPECIFIES AN API, CALLED CRYPTOKI, TO DEVICES WHICH HOLD CRYPTOGRAPHIC INFORMATION AND PERFORM CRYPTOGRAPHIC FUNCTIONS. CRYPTOKI, PRONOUNCED "CRYPTO-KEY" AND SHORT FOR CRYPTOGRAPHIC TOKEN INTERFACE, FOLLOWS A SIMPLE OBJECT-BASED APPROACH, ADDRESSING THE GOALS OF TECHNOLOGY INDEPENDENCE (ANY KIND OF DEVICE) AND RESOURCE SHARING (MULTIPLE APPLICATIONS ACCESSING MULTIPLE DEVICES), PRESENTING TO APPLICATIONS A COMMON, LOGICAL VIEW OF

THE DEVICE CALLED A CRYPTOGRAPHIC TOKEN.

Source: RSA Security Inc. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11.

To summarize, PKCS#11 is a standard that can be used by application software to access cryptographic tokens such as smart cards and other devices. Most device vendors provide a library that implements the PKCS#11 provider interface -- this library can be used by applications in order to access these devices. PKCS#11 is a cross-platform, vendor-independent free standard.

13.12.3 FINDING PKCS#11 PROVIDER LIBRARY

The first thing you need to do is to find the provider library, it should be installed with the device drivers. Each vendor has its own library. For example, the OpenSC PKCS#11 provider is located at /usr/lib/pkcs11/opensc-pkcs11.so on Unix or at opensc-pkcs11.dll on Windows.

13.12.4 HOW TO CONFIGURE CRYPTOGRAPHIC TOKEN

You should follow an enrollment procedure:

- Initialize the PKCS#11 token.
- Generate RSA key pair on the PKCS#11 token.
- Create a certificate request based on the key pair, you can use <u>OpenSC</u> and <u>OpenSSL</u> in order to do that.
- Submit the certificate request to a certificate authority, and receive a certificate.
- Load the certificate onto the token, while noting that the id and label attributes of the certificate must match those of the private key.

A configured token is a token that has a private key object and a certificate object, where both share the same id and label attributes.

A simple enrollment utility is Easy-RSA 2.0 which is part of OpenVPN 2.1 series. Follow the instructions specified in the README file, and then use the pkitool in order to enroll.

Initialize a token using the following command:

```
$ ./pkitool --pkcs11-slots /usr/lib/pkcs11/
$ ./pkitool --pkcs11-init /usr/lib/pkcs11/
```

Enroll a certificate using the following command: \$./pkitool --pkcs11 /usr/lib/pkcs11/ client1

13.12.5 HOW TO MODIFY AN OPENVPN CONFIGURATION TO MAKE USE OF CRYPTOGRAPHIC TOKENS

You should have OpenVPN 2.1 or above in order to use the PKCS#11 features.

13.12.5.1 Determine the correct object

Each PKCS#11 provider can support multiple devices. In order to view the available object list you can use the following command:

\$ openvpn --show-pkcs11-ids /usr/lib/pkcs11/

The following objects are available for use. Each object shown below may be used as parameter to --pkcs11-id option please remember to use single quote mark.

```
Certificate

DN: /CN=User1

Serial: 490B82C400000000075

Serialized id:

aaaa/bbb/41545F5349474E415455524581D2A1A1B23C4AA4CB1
```

```
7FAF7A4600
```

Each certificate/private key pair have unique "Serialized id" string. The serialized id string of the requested certificate should be specified to the **pkcs11-id** option using single quote marks.

pkcs11-id 'aaaa/bbb/41545F5349474E415455524581D2A1A1B23C4AA4CB 17FAF7A4600'

13.12.5.2 Using OpenVPN with PKCS#11

```
13.12.5.2.1 A typical set of OpenVPN options for PKCS#11
pkcs11-providers /usr/lib/pkcs11/
pkcs11-id
'aaaa/bbb/41545F5349474E415455524581D2A1A1B23C4AA4CB
17FAF7A4600'
```

This will select the object which matches the pkcs11-id string.

```
13.12.5.2.2 Advanced OpenVPN options for PKCS#11
pkcs11-providers /usr/lib/pkcs11/provider1.so
/usr/lib/pkcs11/provider2.so
pkcs11-id
'aaaa/bbb/41545F5349474E415455524581D2A1A1B23C4AA4CB
17FAF7A4600'
pkcs11-pin-cache 300
daemon
auth-retry nointeract
management-hold
management-signal
management 127.0.0.1 8888
management-query-passwords
```

This will load two providers into OpenVPN, use the certificate specified on **pkcs11-id** option, and use the management interface in order to query passwords. The daemon will resume into hold state on the event when token cannot be accessed. The token will be used for 300 seconds after which the password will be re-queried, session will disconnect if management session disconnects.

13.12.5.3 PKCS#11 implementation considerations

Many PKCS#11 providers make use of threads, in order to avoid problems caused by implementation of LinuxThreads (setuid, chroot), it is highly recommend to upgrade to Native POSIX Thread Library (NPTL) enabled glibc if you intend to use PKCS#11.

13.12.5.4 OpenSC PKCS#11 provider

OpenSC PKCS#11 provider is located at /usr/lib/pkcs11/opensc-pkcs11.so on

Unix or at opensc-pkcs11.dll on Windows.

13.12.6 DIFFERENCE BETWEEN PKCS#11 AND MICROSOFT CRYPTOGRAPHIC API (CRYPTOAPI)

PKCS#11 is a free, cross-platform vendor independent standard. CryptoAPI is a Microsoft specific API. Most smart card vendors provide support for both interfaces. In the Windows environment, the user should select which interface to use.

The current implementation of OpenVPN that uses the MS CryptoAPI (**cryptoapicert** option) works well as long as you don't run OpenVPN as a service. If you wish to run OpenVPN in an administrative environment using a service, the implementation will not work with most smart cards because of the following reasons:

- Most smart card providers do not load certificates into the local machine store, so the implementation will be unable to access the user certificate.
- If the OpenVPN client is running as a service without direct interaction with the end-user, the service cannot query the user to provide a password for the smart card, causing the password-verification process on the smart card to fail.

Using the PKCS#11 interface, you can use smart cards with OpenVPN in any implementation, since PKCS#11 does not access Microsoft stores and does not necessarily require direct interaction with the end-user.

13.13 ROUTING ALL CLIENT TRAFFIC (INCLUDING WEB-TRAFFIC) THROUGH THE VPN

13.13.1 OVERVIEW

By default, when an OpenVPN client is active, only network traffic to and from the OpenVPN server site will pass over the VPN. General web browsing, for example, will be accomplished with direct connections that bypass the VPN.

In certain cases this behavior might not be desirable -- you might want a VPN

client to tunnel all network traffic through the VPN, including general internet web browsing. While this type of VPN configuration will exact a performance penalty on the client, it gives the VPN administrator more control over security policies when a client is simultaneously connected to both the public internet and the VPN at the same time.

13.13.2 IMPLEMENTATION

Add the following directive to the server configuration file:

push "redirect-gateway def1"

If your VPN setup is over a wireless network, where all clients and the server are on the same wireless subnet, add the **local** flag:

push "redirect-gateway local def1"

Pushing the **redirect-gateway** option to clients will cause all IP network traffic originating on client machines to pass through the OpenVPN server. The server will need to be configured to deal with this traffic somehow, such as by NATing it to the internet, or routing it through the server site's HTTP proxy.

On Linux, you could use a command such as this to NAT the VPN client traffic to the internet:

iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE

This command assumes that the VPN subnet is **10.8.0.0/24** (taken from the **server** directive in the OpenVPN server configuration) and that the local ethernet interface is **eth0**.

When **redirect-gateway** is used, OpenVPN clients will route DNS queries through the VPN, and the VPN server will need handle them. This can be accomplished by pushing a DNS server address to connecting clients which will replace their normal DNS server settings during the time that the VPN is active. For example:

push "dhcp-option DNS 10.8.0.1"

will configure Windows clients (or non-Windows clients with some extra

server-side scripting) to use 10.8.0.1 as their DNS server. Any address which is reachable from clients may be used as the DNS server address.

13.13.3 CAVEATS

Redirecting all network traffic through the VPN is not entirely a problem-free proposition. Here are some typical gotchas to be aware of:

- Many OpenVPN client machines connecting to the internet will periodically interact with a DHCP server to renew their IP address leases. The **redirect-gateway** option might prevent the client from reaching the local DHCP server (because DHCP messages would be routed over the VPN), causing it to lose its IP address lease.
- <u>Issues exist</u> with respect to pushing DNS addresses to Windows clients.
- Web browsing performance on the client will be noticably slower.

For more information on the mechanics of the **redirect-gateway** directive, see the <u>manual page</u>.

13.14 RUNNING AN OPENVPN SERVER ON A DYNAMIC IP ADDRESS

While OpenVPN clients can easily access the server via a dynamic IP address without any special configuration, things get more interesting when the server itself is on a dynamic address. While OpenVPN has no trouble handling the situation of a dynamic server, some extra configuration is required.

The first step is to get a dynamic DNS address which can be configured to "follow" the server every time the server's IP address changes. There are several dynamic DNS service providers available, such as <u>dyndns.org</u>.

The next step is to set up a mechanism so that every time the server's IP address changes, the dynamic DNS name will be quickly updated with the new IP address, allowing clients to find the server at its new IP address. There are two basic ways to accomplish this:

• Use a NAT router appliance with dynamic DNS support (such as the Linksys BEFSR41). Most of the inexpensive NAT router appliances that are widely available have the capability to update a dynamic DNS
name every time a new DHCP lease is obtained from the ISP. This setup is ideal when the OpenVPN server box is a single-NIC machine inside the firewall.

• Use a dynamic DNS client application such as <u>ddclient</u> to update the dynamic DNS address whenever the server IP address changes. This setup is ideal when the machine running OpenVPN has multiple NICs and is acting as a site-wide firewall/gateway. To implement this setup, you need to set up a script to be run by your DHCP client software every time an IP address change occurs. This script should (a) run **ddclient**to notify your dynamic DNS provider of your new IP address and (b) restart the OpenVPN server daemon.

The OpenVPN client by default will sense when the server's IP address has changed, if the client configuration is using a **remote** directive which references a dynamic DNS name. The usual chain of events is that (a) the OpenVPN client fails to receive timely keepalive messages from the server's old IP address, triggering a restart, and (b) the restart causes the DNS name in the **remote** directive to be re-resolved, allowing the client to reconnect to the server at its new IP address.

More information can be found in the <u>FAQ</u>.

13.15 CONNECTING TO AN OPENVPN SERVER VIA AN HTTP PROXY.

OpenVPN supports connections through an HTTP proxy, with the following authentication modes:

- No proxy authentication
- Basic proxy authentication
- NTLM proxy authentication

First of all, HTTP proxy usage requires that you use TCP as the tunnel carrier

protocol. So add the following to both client and server configurations:

proto tcp

Make sure that any **proto udp** lines in the config files are deleted.

Next, add the **http-proxy** directive to the client configuration file (see the <u>manual page</u> for a full description of this directive).

For example, suppose you have an HTTP proxy server on the client LAN at **192.168.4.1**, which is listening for connections on port **1080**. Add this to the client config:

http-proxy 192.168.4.1 1080

Suppose the HTTP proxy requires Basic authentication:

http-proxy 192.168.4.1 1080 stdin basic

Suppose the HTTP proxy requires NTLM authentication:

http-proxy 192.168.4.1 1080 stdin ntlm

The two authentication examples above will cause OpenVPN to prompt for a username/password from standard input. If you would instead like to place these credentials in a file, replace **stdin** with a filename, and place the username on line 1 of this file and the password on line 2.

13.16 CONNECTING TO A SAMBA SHARE OVER OPENVPN

This example is intended show how OpenVPN clients can connect to a Samba share over a routed **dev tun** tunnel. If you are ethernet bridging (**dev tap**), you probably don't need to follow these instructions, as OpenVPN clients should see server-side machines in their network neighborhood.

For this example, we will assume that:

- the server-side LAN uses a subnet of 10.66.0.0/24,
- the VPN IP address pool uses **10.8.0.0/24** (as cited in the **server** directive in the OpenVPN server configuration file),

- the Samba server has an IP address of **10.66.0.4**, and
- the Samba server has already been configured and is reachable from the local LAN.

If the Samba and OpenVPN servers are running on different machines, make sure you've followed the section on <u>expanding the scope of the VPN to</u> include additional machines.

Next, edit your Samba configuration file (**smb.conf**). Make sure the **hosts allow** directive will permit OpenVPN clients coming from the **10.8.0.0/24** subnet to connect. For example:

hosts allow = 10.66.0.0/24 10.8.0.0/24 127.0.0.1

If you are running the Samba and OpenVPN servers on the same machine, you may want to edit the **interfaces** directive in the **smb.conf** file to also listen on the TUN interface subnet of **10.8.0.0/24**:

interfaces = 10.66.0.0/24 10.8.0.0/24

If you are running the Samba and OpenVPN servers on the same machine, connect from an OpenVPN client to a Samba share using the folder name:

$\10.8.0.1\$

If the Samba and OpenVPN servers are on different machines, use folder name:

$\10.66.0.4$ sharename

For example, from a command prompt window:

net use z: \\10.66.0.4\sharename /USER:myusername

13.17 IMPLEMENTING A LOAD-BALANCING/FAILOVER CONFIGURATION

13.17.1 CLIENT

The OpenVPN client configuration can refer to multiple servers for load

balancing and failover. For example:

```
remote server1.mydomain
remote server2.mydomain
remote server3.mydomain
```

will direct the OpenVPN client to attempt a connection with server1, server2, and server3 in that order. If an existing connection is broken, the OpenVPN client will retry the most recently connected server, and if that fails, will move on to the next server in the list. You can also direct the OpenVPN client to randomize its server list on startup, so that the client load will be probabilistically spread across the server pool.

remote-random

If you would also like DNS resolution failures to cause the OpenVPN client to move to the next server in the list, add the following:

resolv-retry 60

The **60** parameter tells the OpenVPN client to try resolving each **remote** DNS name for 60 seconds before moving on to the next server in the list.

The server list can also refer to multiple OpenVPN server daemons running on the same machine, each listening for connections on a different port, for example:

```
remote smp-server1.mydomain 8000
remote smp-server1.mydomain 8001
remote smp-server2.mydomain 8000
remote smp-server2.mydomain 8001
```

If your servers are multi-processor machines, running multiple OpenVPN daemons on each server can be advantageous from a performance standpoint.

OpenVPN also supports the **remote** directive referring to a DNS name which has multiple **A** records in the zone configuration for the domain. In this case, the OpenVPN client will randomly choose one of the **A** records every time the domain is resolved.

13.17.2 SERVER

The simplest approach to a load-balanced/failover configuration on the server is to use equivalent configuration files on each server in the cluster, except use a different virtual IP address pool for each server. For example:

server1
server 10.8.0.0 255.255.255.0
server2
server 10.8.1.0 255.255.255.0
server3

server 10.8.2.0 255.255.255.0

13.18 HARDENING OPENVPN SECURITY

One of the often-repeated maxims of network security is that one should never place so much trust in a single security component that its failure causes a catastrophic security breach. OpenVPN provides several mechanisms to add additional security layers to hedge against such an outcome.

13.18.1 TLS-AUTH

The **tls-auth** directive adds an additional HMAC signature to all SSL/TLS handshake packets for integrity verification. Any UDP packet not bearing the correct HMAC signature can be dropped without further processing. The **tls-auth** HMAC signature provides an additional level of security above and beyond that provided by SSL/TLS. It can protect against:

- DoS attacks or port flooding on the OpenVPN UDP port.
- Port scanning to determine which server UDP ports are in a listening state.
- Buffer overflow vulnerabilities in the SSL/TLS implementation.
- SSL/TLS handshake initiations from unauthorized machines (while such handshakes would ultimately fail to authenticate, **tls-auth** can cut them off at a much earlier point).

Using **tls-auth** requires that you generate a shared-secret key that is used in addition to the standard RSA certificate/key:

openvpn --genkey --secret ta.key

This command will generate an OpenVPN static key and write it to the file **ta.key**. This key should be copied over a pre-existing secure channel to the server and all client machines. It can be placed in the same directory as the RSA .key and .crt files.

In the server configuration, add:

tls-auth ta.key 0

In the client configuration, add:

tls-auth ta.key 1

13.18.2 PROTO UDP

While OpenVPN allows either the TCP or UDP protocol to be used as the VPN carrier connection, the UDP protocol will provide better protection against DoS attacks and port scanning than TCP:

proto udp

13.18.3 USER/GROUP (NON-WINDOWS ONLY)

OpenVPN has been very carefully designed to allow root privileges to be dropped after initialization, and this feature should always be used on Linux/BSD/Solaris. Without root privileges, a running OpenVPN server daemon provides a far less enticing target to an attacker.

user nobody group nobody

13.18.4 UNPRIVILEGED MODE (LINUX ONLY)

On Linux OpenVPN can be run completely unprivileged. This configuration is a little more complex, but provides best security.

In order to work with this configuration, OpenVPN must be configured to use iproute interface, this is done by specifying --enable-iproute2 to configure script. sudo package should also be available on your system.

This configuration uses the Linux ability to change the permission of a tun device, so that unprivileged user may access it. It also uses sudo in order to execute iproute so that interface properties and routing table may be modified.

OpenVPN configuration:

 Write the following script and place it at: /usr/local/sbin/unprivip:

#!/bin/sh sudo /sbin/ip \$*

• Execute visudo, and add the followings to allow user 'user1' to execute /sbin/ip:

user1 ALL=(ALL) NOPASSWD: /sbin/ip

You can also enable a group of users with the following command:

%users ALL=(ALL) NOPASSWD: /sbin/ip

• Add the following to your OpenVPN configuration:

dev tunX/tapX

iproute /usr/local/sbin/unpriv-ip

Please note that you must select constant X and specify tun or tap not both.

• As root add persistant interface, and permit user and/or group to manage it, the following create tunX (replace with your own) and allow user1 and group users to access it.

openvpn --mktun --dev tunX --type tun --user user1 --group users

• Run OpenVPN in the context of the unprivileged user.

Further security constraints may be added by examining the parameters at the

/usr/local/sbin/unpriv-ip script.

13.18.5 CHROOT (NON-WINDOWS ONLY)

The **chroot** directive allows you to lock the OpenVPN daemon into a socalled CHROOT JAIL, where the daemon would not be able to access any part of the host system's filesystem except for the specific directory given as a parameter to the directive. For example,

chroot jail

would cause the OpenVPN daemon to cd into the **jail** subdirectory on initialization, and would then reorient its root filesystem to this directory so that it would be impossible thereafter for the daemon to access any files outside of **jail** and its subdirectory tree. This is important from a security perspective, because even if an attacker were able to compromise the server with a code insertion exploit, the exploit would be locked out of most of the server's filesystem.

Caveats: because **chroot** reorients the filesystem (from the perspective of the daemon only), it is necessary to place any files which OpenVPN might need after initialization in the **jail** directory, such as:

- the **crl-verify** file, or
- the client-config-dir directory.

13.18.6 LARGER RSA KEYS

The RSA key size is controlled by the **KEY_SIZE** variable in the **easy-rsa/vars** file, which must be set before any keys are generated. Currently set to 1024 by default, this value can reasonably be increased to 2048 with no negative impact on VPN tunnel performance, except for a slightly slower SSL/TLS renegotiation handshake which occurs once per client per hour, and a much slower one-time Diffie Hellman parameters generation process using the **easy-rsa/build-dh** script.

13.18.7 LARGER SYMMETRIC KEYS

By default OpenVPN uses **Blowfish**, a 128 bit symmetrical cipher.

OpenVPN automatically supports any cipher which is supported by the OpenSSL library, and as such can support ciphers which use large key sizes. For example, the 256-bit version of AES (Advanced Encryption Standard) can be used by adding the following to both server and client configuration files:

cipher AES-256-CBC

13.18.8 KEEP THE ROOT KEY (CA.KEY) ON A STANDALONE MACHINE WITHOUT A NETWORK CONNECTION

One of the security benefits of using an X509 PKI (as OpenVPN does) is that the root CA key (**ca.key**) need not be present on the OpenVPN server machine. In a high security environment, you might want to specially designate a machine for key signing purposes, keep the machine wellprotected physically, and disconnect it from all networks. Floppy disks can be used to move key files back and forth, as necessary. Such measures make it extremely difficult for an attacker to steal the root key, short of physical theft of the key signing machine.

13.19 REVOKING CERTIFICATES

REVOKING A CERTIFICATE means to invalidate a previously signed certificate so that it can no longer be used for authentication purposes.

Typical reasons for wanting to revoke a certificate include:

- The private key associated with the certificate is compromised or stolen.
- The user of an encrypted private key forgets the password on the key.
- You want to terminate a VPN user's access.

13.19.1 EXAMPLE

As an example, we will revoke the **client2** certificate, which we generated above in the "key generation" section of the HOWTO.

First open up a shell or command prompt window and cd to the **easy-rsa** directory as you did in the "key generation" section above. On Linux/BSD/Unix:

. ./vars

./revoke-full client2

On Windows:

vars revoke-full client2

You should see output similar to this:

```
Using configuration from
/root/openvpn/20/openvpn/tmp/easy-rsa/openssl.cnf
DEBUG[load_index]: unique_subject = "yes"
Revoking Certificate 04.
Data Base Updated
Using configuration from
/root/openvpn/20/openvpn/tmp/easy-rsa/openssl.cnf
DEBUG[load_index]: unique_subject = "yes"
client2.crt: /C=KG/ST=NA/O=OpenVPN-
TEST/CN=client2/emailAddress=me@myhost.mydomain
error 23 at 0 depth lookup:certificate revoked
```

Note the "error 23" in the last line. That is what you want to see, as it indicates that a certificate verification of the revoked certificate failed.

The **revoke-full** script will generate a CRL (certificate revocation list) file called **crl.pem** in the **keys** subdirectory. The file should be copied to a directory where the OpenVPN server can access it, then CRL verification should be enabled in the server configuration:

crl-verify crl.pem

Now all connecting clients will have their client certificates verified against the CRL, and any positive match will result in the connection being dropped.

13.19.2 CRL NOTES

• When the **crl-verify** option is used in OpenVPN, the CRL file will be re-read any time a new client connects or an existing client renegotiates the SSL/TLS connection (by default once per hour). This means that you can update the CRL file while the OpenVPN server daemon is running, and have the new CRL take effect immediately for newly connecting clients. If the client whose certificate you are revoking is

already connected, you can restart the server via a signal (SIGUSR1 or SIGHUP) and flush all clients, or you can telnet to the <u>management</u> <u>interface</u> and explicitly kill the specific client instance object on the server without disturbing other clients.

- While the **crl-verify** directive can be used on both the OpenVPN server and clients, it is generally unnecessary to distribute a CRL file to clients unless a server certificate has been revoked. Clients don't need to know about other client certificates which have been revoked because <u>clients</u> <u>shouldn't be accepting direct connections from other clients</u> in the first place.
- The CRL file is not secret, and should be made world-readable so that the OpenVPN daemon can read it after root privileges have been dropped.
- If you are using the **chroot**directive, make sure to put a copy of the CRL file in the chroot directory, since unlike most other files which OpenVPN reads, the CRL file will be read after the chroot call is executed, not before.
- A common reason why certificates need to be revoked is that the user encrypts their private key with a password, then forgets the password. By revoking the original certificate, it is possible to generate a new certificate/key pair with the user's original common name.

13.20 IMPORTANT NOTE ON POSSIBLE "MAN-IN-THE-MIDDLE" ATTACK IF CLIENTS DO NOT VERIFY THE CERTIFICATE OF THE SERVER THEY ARE CONNECTING

TO.

To avoid a possible Man-in-the-Middle attack where an authorized client tries to connect to another client by impersonating the server, make sure to enforce some kind of server certificate verification by clients. There are currently five different ways of accomplishing this, listed in the order of preference:

• [OPENVPN 2.1 AND ABOVE]Build your server certificates with specific key usage and extended key usage. The RFC3280 determine that the following attributes should be provided for TLS connections:

| Mode | Key usage | Extended key usage |
|--------|--------------------------------------|----------------------------------|
| Client | digitalSignature | TLS Web Client Authentication |
| | keyAgreement | |
| | digitalSignature, keyAgreement | |
| Server | digitalSignature, keyEncipherment | TLS Web Server Authentication |
| | digitalSignature, keyAgreement | |

You can build your server certificates with the **build-key-server** script (see the <u>easy-rsa</u> documentation for more info). This will designate the certificate as a server-only certificate by setting the right attributes. Now add the following line to your client configuration:

remote-cert-tls server

• [OPENVPN 2.0 AND BELOW] Build your server certificates with the **build-key-server** script (see the <u>easy-rsa</u> documentation for more info). This will designate the certificate as a server-only certificate by setting

nsCertType=server. Now add the following line to your client configuration:

ns-cert-type server

This will block clients from connecting to any server which lacks the **nsCertType**=server designation in its certificate, even if the certificate has been signed by the **ca** file in the OpenVPN configuration file.

- Use the **tls-remote**directive on the client to accept/reject the server connection based on the common name of the server certificate.
- Use a **tls-verify**script or plugin to accept/reject the server connection based on a custom test of the server certificate's embedded X509 subject details.
- Sign server certificates with one CA and client certificates with a different CA. The client configuration **ca** directive should reference the server-signing CA file, while the server configuration **ca**directive should reference the client-signing CA file.

13.21 SAMPLE OPENVPN 2.0 CONFIGURATION FILES

BA.NET ADBLOCK ADMINISTRATOR MANUAL

13.21.1 SAMPLE-CONFIG-FILES/SERVER.CONF

```
# Sample OpenVPN 2.0 config file for
                                            #
# multi-client server.
                                            #
#
                                            #
# This file is for the server side
                                             #
                                             #
# of a many-clients one-server
                                            #
# OpenVPN configuration.
#
                                            #
# OpenVPN also supports
                                             #
# single-machine single-machine
                                            #
# configurations (See the Examples page
                                            #
# on the web site for more info).
                                            #
#
                                            #
# This config should work on Windows
                                            #
# or Linux/BSD systems. Remember on
                                            #
# Windows to quote pathnames and use
                                             #
# double backslashes, e.g.:
                                            #
# "C:\\Program Files\\OpenVPN\\config\\foo.key"
                                            #
#
                                            #
# Comments are preceded with '#' or ';'
                                             #
# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d
# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one. You will need to
# open up this port on your firewall.
port 1194
```

TCP or UDP server?
;proto tcp
proto udp

"dev tun" will create a routed IP tunnel, # "dev tap" will create an ethernet tunnel. # Use "dev tap0" if you are ethernet bridging # and have precreated a tap0 virtual interface # and bridged it with your ethernet interface. # If you want to control access policies # over the VPN, you must create firewall # rules for the the TUN/TAP interface. # On non-Windows systems, you can give # an explicit unit number, such as tun0. # On Windows, use "dev-node" for this. # On most systems, the VPN will not function # unless you partially or fully disable # the firewall for the TUN/TAP interface. ;dev tap dev tun

Windows needs the TAP-Windows adapter name # from the Network Connections panel if you # have more than one. On XP SP2 or higher, # you may need to selectively disable the # Windows firewall for the TAP adapter. # Non-Windows systems usually don't need this. ;dev-node MyTap

```
# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key). Each client
# and the server must have their own cert and
# key file. The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys. Remember to use
# a unique Common Name for the server
# and each of the client certificates.
```

Any X509 key management system can be used. # OpenVPN can also use a PKCS #12 formatted key file # (see "pkcs12" directive in man page). ca ca.crt cert server.crt key server.key # This file should be kept secret # Diffie hellman parameters. # Generate your own with: openssl dhparam -out dh1024.pem 1024 # # Substitute 2048 for 1024 if you are using # 2048 bit keys. dh dh1024.pem # Configure server mode and supply a VPN subnet # for OpenVPN to draw client addresses from. # The server will take 10.8.0.1 for itself, # the rest will be made available to clients. # Each client will be able to reach the server # on 10.8.0.1. Comment this line out if you are # ethernet bridging. See the man page for more info. server 10.8.0.0 255.255.255.0 # Maintain a record of client virtual IP address # associations in this file. If OpenVPN goes down

or
is restarted, reconnecting clients can be assigned
the same virtual IP address from the pool that was
previously assigned.
ifconfig-pool-persist ipp.txt

Configure server mode for ethernet bridging. # You must first use your OS's bridging capability # to bridge the TAP interface with the ethernet # NIC interface. Then you must manually set the # IP/netmask on the bridge interface, here we # assume 10.8.0.4/255.255.255.0. Finally we # must set aside an IP range in this subnet # (start=10.8.0.50 end=10.8.0.100) to allocate # to connecting clients. Leave this line commented # out unless you are ethernet bridging. ;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100

Push routes to the client to allow it # to reach other private subnets behind # the server. Remember that these # private subnets will also need # to know to route the OpenVPN client # address pool (10.8.0.0/255.255.255.0) # back to the OpenVPN server. ;push "route 192.168.10.0 255.255.255.0"

To assign specific IP addresses to specific # clients or if a connecting client has a private # subnet behind it that should also have VPN access, # use the subdirectory "ccd" for client-specific # configuration files (see man page for more info).

EXAMPLE: Suppose the client # having the certificate common name "Thelonious" # also has a small subnet behind his connecting # machine, such as 192.168.40.128/255.255.255.248. # First, uncomment out these lines: ;client-config-dir ccd ;route 192.168.40.128 255.255.255.248 # Then create a file ccd/Thelonious with this line: # iroute 192.168.40.128 255.255.255.248 # This will allow Thelonious' private subnet to # access the VPN. This example will only work # if you are routing, not bridging, i.e. you are # using "dev tun" and "server" directives.

EXAMPLE: Suppose you want to give # Thelonious a fixed VPN IP address of 10.9.0.1. # First uncomment out these lines: ;client-config-dir ccd ;route 10.9.0.0 255.255.255.252 # Then add this line to ccd/Thelonious: ifconfig-push 10.9.0.1 10.9.0.2 # # Suppose that you want to enable different # firewall access policies for different groups # of clients. There are two methods: # (1) Run multiple OpenVPN daemons, one for each # group, and firewall the TUN/TAP interface # for each group/daemon appropriately. # (2) (Advanced) Create a script to dynamically # modify the firewall in response to access

from different clients. See man
page for more info on learn-address script.
;learn-address ./script

If enabled, this directive will configure # all clients to redirect their default # network gateway through the VPN, causing # all IP traffic such as web browsing and # and DNS lookups to go through the VPN # (The OpenVPN server machine may need to NAT # the TUN/TAP interface to the internet in # order for this to work properly). # CAVEAT: May break client's network config if # client's local DHCP server packets get routed # through the tunnel. Solution: make sure # client's local DHCP server is reachable via # a more specific route than the default route # of 0.0.0/0.0.0.0.

;push "redirect-gateway"

Certain Windows-specific network settings # can be pushed to clients, such as DNS # or WINS server addresses. CAVEAT: # http://openvpn.net/faq.html#dhcpcaveats ;push "dhcp-option DNS 10.8.0.1" ;push "dhcp-option WINS 10.8.0.1"

Uncomment this directive to allow different # clients to be able to "see" each other.

```
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
;client-to-client
# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names.
                          This is recommended
# only for testing purposes. For production use,
# each client should have its own certificate/key
# pair.
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn
# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120
# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#
    openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
;tls-auth ta.key 0 # This file is secret
```

```
# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC
                      # Blowfish (default)
;cipher AES-128-CBC
                      # AES
;cipher DES-EDE3-CBC # Triple-DES
# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo
# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100
# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nobody
# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun
# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log
# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go
to
```

```
BA.NET ADBLOCK ADMINISTRATOR MANUAL
```

```
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN
startup,
# while "log-append" will append to it. Use one
# or the other (but not both).
;log
             openvpn.log
;log-append openvpn.log
# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3
# Silence repeating messages. At most 20
# sequential messages of the same message
# category will be output to the log.
```

;mute 20

13.21.2 SAMPLE-CONFIG-FILES/CLIENT.CONF

```
****************
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.
                                          #
#
                                          #
                                          #
# This configuration can be used by multiple
# clients, however each client should have
                                          #
                                          #
# its own cert and key files.
#
                                          #
# On Windows, you might want to rename this
                                          #
                                          #
# file so it has a .ovpn extension
```

```
BA.NET ADBLOCK ADMINISTRATOR MANUAL
# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client
# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun
# Windows needs the TAP-Windows adapter name
# from the Network Connections panel
# if you have more than one. On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
;dev-node MyTap
# Are we connecting to a TCP or
# UDP server? Use the same setting as
# on the server.
;proto tcp
proto udp
# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote my-server-1 1194
;remote my-server-2 1194
# Choose a random host from the remote
# list for load-balancing.
                           Otherwise
# try hosts in the order specified.
;remote-random
# Keep trying indefinitely to resolve the
```

```
BA.NET ADBLOCK ADMINISTRATOR MANUAL
# host name of the OpenVPN server. Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite
# Most clients don't need to bind to
# a specific local port number.
nobind
# Downgrade privileges after initialization (non-
Windows only)
;user nobody
;group nobody
# Try to preserve some state across restarts.
persist-key
persist-tun
# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here. See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]
# Wireless networks often produce a lot
# of duplicate packets. Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings
# SSL/TLS parms.
# See the server config file for more
# description. It's best to use
# a separate .crt/.key file pair
# for each client. A single ca
# file can be used for all clients.
ca ca.crt
cert client.crt
```

```
BA.NET ADBLOCK ADMINISTRATOR MANUAL
key client.key
# Verify server certificate by checking
# that the certicate has the nsCertType
# field set to "server".
                           This is an
# important precaution to protect against
# a potential attack discussed here:
#
  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server". The build-key-server
# script in the easy-rsa folder will do this.
;ns-cert-type server
# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1
# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
;cipher x
# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo
# Set log file verbosity.
verb 3
# Silence repeating messages
;mute 20
```

BA.NET ADBLOCK ADMINISTRATOR MANUAL

BA.NET ADBLOCK ADMINISTRATOR MANUAL