Build Your Own Unified Threat Management With pfSense

Introduction

When we last saw Cerberus, the small form factor, low power, high performance IDS firewall, it was chewing through anything the net threw at it. Today's question is: can Cerberus go for the gold and become a full-fledged Unified Threat Management (UTM) Appliance, capable of providing all of the protection required by a home network, let alone an enterprise network?

Cerberus, as the previous article detailed, is an IDS

Firewall built around a mini-ITX 1.8 GHz dual-core Atom and 3 GB of memory, providing three heads of network protection: **pfSense**, a free open source project, providing standard perimeter firewall protection as part of an overall router, and two pfSense packages: **Snort**, the premiere open source Intrusion Detection and Prevention rules engine, and **IP Blocklist**, which uses dynamic categorical lists to block questionable traffic.

To build a capable UTM appliance, we first need to define what Unified Threat Management is. Once we understand that, we'll step through adding and configuring those services to Cerberus, and finally look whether Cerberus can carry the weight or fall short in either functionality or performance.

What is a UTM Appliance?

The concept of Unified Threat Management is straightforward: on the outer reaches of your network perimeter, you install an appliance that stops all possible threats to your network, an über firewall, as it were. The fact of the matter is that UTM hardware is expected to completely overtake separate network protection hardware.
The problem is there is no single definition of the services required in a UTM appliance. For example, one of the foremost makers of UTM appliances for the enterprise, Endian, lists an entire dense page of functionality. In comparison, Untangle, a small organization UTM, lists only about twenty functions.

So what do they have in common? For our purposes, a UTM appliance is something that offers Intrusion Protection Firewall, Anti-Virus, Anti-Spam, and Traffic Control features. Beyond this core protection, a UTM appliance generally includes some enterprise operation capabilities, such as load balancing, fail-over, and network wide caching and monitoring.

pfSense can perform all these functions to some extent. To judge how well pfSense meets these UTM requirements, I've given a subjective grade to each set of UTM function groups. Once we've defined how these functions thwart threats, and how pfSense meets those challenges, we'll upgrade Cerberus, and see how it performs as a UTM.

Intrusion Detection and Prevention (IDS/IPS)

As detailed in the first article, IDS uses a packet inspection engine in conjunction with a standard NAT firewall to recognize patterns in network traffic, either at the packet level or at the stream level. IDS uses dynamic rules to spot these irregularities, such as protocol vulnerabilities, port scans, Denial of service attacks, and alike.
The vast majority of UTM appliances utilize **Snort**, the most widely deployed IDS/IPS rules engine. Snort uses rules that are updated regularly from Snort.org. pfSense has wrapped Snort in an easy to install and administer WebGUI package.
Cerberus is already configured for Snort, so we'll not be covering that as part of the upgrade process. For detailed instructions on how to install and configure Snort, please refer to the previous article.

pfSense Grade: A

Anti-Virus

The ability to block the Internet's malicious flora and fauna from infecting network clients is core to any UTM. This is accomplished by inspecting packets for establish virus signatures and virus meta-patterns.

pfSense includes the HAVP package: HTTP Anti-Virus Proxy, a transparent proxy that scans all HTTP traffic for malware signatures. HAVP utilizes ClamAV, the open source and community anti-virus engine for Linux and BSD distros.
Naturally, the question of effectiveness is raised when using an open source anti-virus solution versus a commercial product. But is difficult to make a clear determination of effectiveness. Some reports place ClamAV in the top five, others in the bottom five.
There is a dirty little secret in anti-virus detection. Most anti-virus programs are good at detecting known malware. But with the preponderance of free Anti-virus solutions, virus writers are able to craft their code to avoid most prevention solutions, they can test their code before it is released into the wild.

This means that anti-malware solutions effectiveness should really be measured in latency, from the point that they are first seen in play, to when they are added to their respective detection databases. Commercial vendors run network scanners, honeypots, and have dedicated personnel associated with finding the newest threats. ClamAV does not have such resources and hence operates at a disadvantage.

HAVP, as the name implies, is also limited to HTTP traffic. This means that viruses imbedded in files transferred via FTP, HTTPS, and other protocols such as P2P are not examined and would not be

detected. Neither are e-mail attachments scanned, which account for one of the largest causes of malware infections.

Because of this, it is important that UTM based anti-virus not be your only malware line of defense. Per client, anti-virus is a critical part of any network's protection. With so many quality products that can be had at little or no cost, there is no excuse not to run anti-virus on each network host.

Additionally, since it is strongly recommended that you run only one anti-virus application per host, HAVP does have significant utility, because HTTP is one of the largest vectors for infection. HAVP gives you two bites at the apple and offers protection against malware that is targeted at closed systems, such as cell phones and Internet-enabled home theater components.

pfSense Grade: C-

Content Filtering

Content filtering is what it sounds like: the ability to block certain and generally NSFW content from your network. Such content is typically porn, gambling, file sharing, and hacking methods, but can extend to bandwidth-consuming audio/video sites and time-consuming social networking, forum, and blog sites.

Most importantly, it can be used to block IP addresses associated with spamming, malware, and addresses deemed to be compromised in some other way. Unless you have kids, this is the category that is of the most interest to home networks.

pfSense excels at content blocking and offers four different packages for controlling what can come in your front door.

Content Blocking Packages		
DNS Blacklist	Included functionality uses a static category list	Domain blocking by category
Country Block	Add-on Package	Block entire country access
Squid Guard	Add-on Package, works in conjunction with Squid Caching Proxy Server	Full Featured URL filter
IP Blocklist	Add-on Package, uses frequently updated categorical address lists from IBlocklist.com	Block IP Addresses based on diverse set of lists

Both Country Block and DNS Blacklist are simple. DNS Blacklist, which use a simple list of categories, is a real grab bag and allows the standard blocking of adult and gambling sites, but also astrology, and for

some reason, French educational institutes sites (?!?).

IP Blocklist, which had its origins in the P2P peer blocking arena, blocks hosts that perform IP tracking for media companies and associations like the RIAA and the MPAA. It has grown to allow the blocking of spammers, advertising, malware, and other compromised sites. The lists differ significantly in quality; some are excellent, with spot-on targeting, while others seem ill-maintained, and hence have unintentional causalities - for example, one of the adware lists blocks all of CNet.

The real star here is **Squid Guard**, which works with the caching proxy server Squid. Squid Guard allows for Access Control Lists for specific IPs, with scheduling and user-defined redirect pages. It comes with a built-in blacklist, but also allows the use of community-maintained categorical blacklists. Squid Guard is an ideal solution for café hotspots, schools and libraries.

pfSense Grade: B

Anti-Spam

Unless you are running a domain out of your home, there is not a lot of call for anti-spam. However, for folks who run a domain's mailserver, spam is a real problem. The current estimate is that over 75% of all e-mail traversing the net is spam.

Spam traffic is a burden on any network, and as previously stated, e-mail accounts for one of the largest vectors for malware infection, either as attachments or through referred malicious web-sites.

pfSense does not currently provide an anti-spam solution. For that solution, you need to drop to the underlying operating system, FREEBSD, which offers numerous packages. There are two significant open source projects for controlling spam: SpamD and SpamAssassin. Notably, in the next release of pfSense, version 2.0, support for SpamAssassin is planned.

The Perl-based **SpamAssassin** is a complex spam filtering tool, analyzing the e-mail stream for tell-tale indications that the mail being received isn't legit. This includes the use of White and Blacklist to vet the e-mail. Beyond filtering, it also can be configured to use ClamAV for malware scanning of the e-mail payload. Depending on your e-mail load, this can be processor intensive.

SpamD takes a much simpler, but clever approach to thwarting Spam. It pretends to be a sendmail-like daemon for mail processing, analyzing the sender against three lists: a white list of approved senders, a black list of known spammers, and a grey list of yet-to-be verified senders.

If on a whitelist, it passes the connection on to the proper mail processing daemon behind the firewall. If it doesn't know the sender, it responds with a "Please Send Later" message, deferring delivery and adding the sender to the grey list. If the mail is actually resent later, the sender is added to the whitelist,

and the mail connection passed on for delivery.

If the sender has been black listed, SpamD tarpits the connection, very slowly and repeatedly asking for details, like a brain-damaged sendmail.

The grey list process counts on the fact that most spam is delivered by hit and run bots, and if delivery fails, the process will just move on. The black list process just screws with the process, slowing down or stopping the ultimate delivery of spam to recipients.

Notably, when it comes to threats, pfSense creates an overlapping field of fire approach with many packages working in conjunction to avert the success of a threat. With spam, Snort provides a set of spam/phishing rules. Country Block content filtering provides a list of the countries most responsible for spam (I personally don't see a lot of correspondence from Korea, the number one source of spam).

IP Blocklist and DNS Blacklist both provide lists for blocking spammers. This is also true of content management where Snort has a set of rules defining inappropriate content. Phrases like "XXX Teen" and other more colorful words can trigger the source address to be blocked.

pfSense Grade: D

Traffic Control

Part of threat management is the ability to control traffic on your network. This includes Quality of Service (QOS) and protocol/application blocking such as P2P, IM, and Gaming or Tor proxy traffic. pfSense doesn't provide a single point of traffic control. Snort provides protocol blocking – a set of rules that block specific traffic, like P2P.

QOS, the allotting of particular levels of bandwidth to specific applications/hosts or protocols, is accomplished through a Traffic Shaping Wizard that allows you to both prioritize and limit different types or destinations of traffic. The Wizard is very good at simplifying a complex problem, but does not allow a high degree of fine tuning. Additionally, the current version of traffic is limited to single-WAN/LAN prioritization. Version 2.0 of pfSense, now in beta, allows for Multi-WAN/LAN configurations.

The pfSense traffic shaping wizard uses your real world speed to allocate bandwidth, and steps you through a series of pages that allow you to "Shape" specific traffic. These include VOIP, P2P, Gaming, and other application traffic such as HTTP, Instant Messengers, VPN, and Multimedia traffic. You are also allowed to penalize (limit) bandwidth for either a single IP or a Single set of IPs.

The Squid Package is a tunable caching proxy server, which provides both a high speed cache, and the ability to throttle traffic. You can throttle all HTTP traffic, per host traffic, specific traffic by category such as binary or multimedia, or by specific user defined extensions, say avi, mp3, and zip extensions. You can also set maximum upload and download sizes to further limit bandwidth usage..

Another aspect of Traffic Control is the ability to encrypt traffic via a **VPN**.

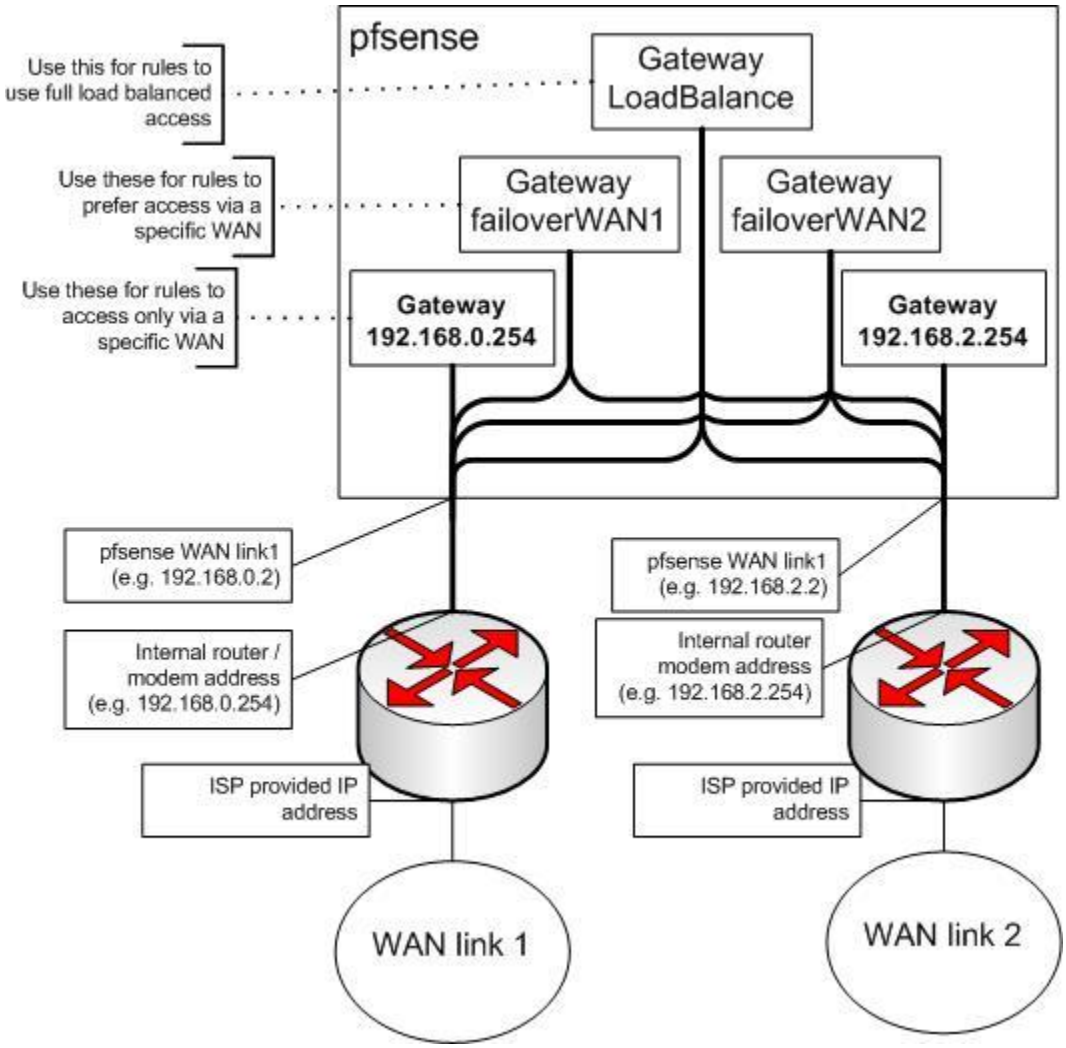
Three different VPN standards are supported: OpenVPN, IPSec, and PPTP. Under the current version of pfSense, both PPTP and IPSec have NAT limitations, making OpenVPN the most flexible solution. These limitations are well documented and a thumbnail of the issues is covered on the pfSense Capabilities Page.

pfSense Grade: B

Enterprise Capabilities

To paraphrase Doctor Strangelove, "What use is threat management if you don't have a network?" Safe network access has become indispensable. Any primary network gateway needs to provide for failover, at both the hardware and the provider level.

pfSense provides for hardware failover, network load balancing and failover, and a plethora of ways of monitoring its current and historical status. Hardware failover is handled through synchronized clustering of two separate pfSense boxes, utilizing the pfSense package CARP. Setting up CARP is outside the scope of this article (I don't have two pfSense boxes, but it appears to be straightforward).



pfSense has built -in Multi-Wan failover and load balancing, utilizing three tiers of cascading gateways: a single load balancer gateway and a gateway for each ISP fail-over point, each having a separate ping heartbeat (say the IPs for Google or Yahoo) that points to the gateway to the ISP. Here is the diagram from the pfSense tutorial.

Fail-over is pretty straightforward, active standby is dead simple. The tricky part comes with load balancing, which uses a connection-based simple round-robin algorithm.

Quite a few applications/protocols are stateful when it comes to your IP address, such as P2P, games, and IM applications. For each of these you'll need to set up routing rules that bypass the load balancer and direct the traffic through a particular ISP.

With HTTP connections, pfSense attempts to be sticky, that is, routing the same host through the same ISP, but this is hit and miss. You may see problems with web sites that count on your IP Address not changing, such as cloud based e-mail services and banks.

Regretfully, in the current stable version of pfSense, On-Demand connections, passive standby—like using USB Wi-Fi modems—is not currently supported. But this has been added in version 2.0. Without passive standby, failover is not very attractive to home networks, unless you are willing to incur two ISP bills a month. If you are, then load balancing becomes compelling, even with the routing hassles. Who wants to pay for bandwidth they don't use?
Enterprise capabilities would not be complete without talking about **monitoring**, pfSense offers out-of-the-box Syslog and SNMP logging, and several adaptor packages for other protocols, such as RADIUS, NetFlow, and Zabbix protocols. For bandwidth monitoring there is both RRD and a mostly integrated BandwidthHD web display, which breaks out traffic by host IP.

pfSense Grade: C

Closing Thoughts

One important factor that can't be ignored is that up-to-date content is needed for a UTM appliance to do its job. Without regular updates of IDS rules, host lists, and malware signatures, threat management is no better than a firewall.
For commercial vendors of these appliances, this is a major source of revenue. With pfSense, this content is largely free – making pfSense, with all of its patchwork flaws, very compelling. The value proposition of pfSense is significant. It is free, open, and no expensive subscriptions are needed to protect your network. Free something is better than nothing. So in Part 2, I'll step you through adding and configuring these UTM features to pfSense.

Introduction to Multi WAN Interfaces

Introduction

In Part One of this series, we established a working definition of our target, i.e. what has to be done, and in what order, to Cerberus the lowly IDS firewall to make it a UTM Appliance.

As we saw, there are six areas that need to be upgraded to grab the prize: IDS/IPS, Anti-Virus, Content Filtering, Traffic Control, Load Balancing and Failover, and finally Anti-Spam. We'll step through each of the six functional areas and show you how to install and configure the required packages.
Once we have everything set up, we'll look at performance and see if Cerberus with PFSense is able to be called a UTM appliance. But first, we need to attend to some prerequisites, which include setting up a second WAN interface for load balancing and fail-over and installing Squid, a critical piece needed for content filtering and anti-virus.

Multiple WAN Setup

For the purposes of this upgrade, we've ordered service from another ISP. You may remember we had previously set up a little-used guest wireless interface to use for our second ISP WAN connection for testing. Now we need the real thing.
The setup is straightforward—enable the interface using parameters provided by your ISP. In most cases this is just DHCP. Note, the FTP Proxy should be disabled on all WAN interfaces, including this one. Figure 1 shows the settings.

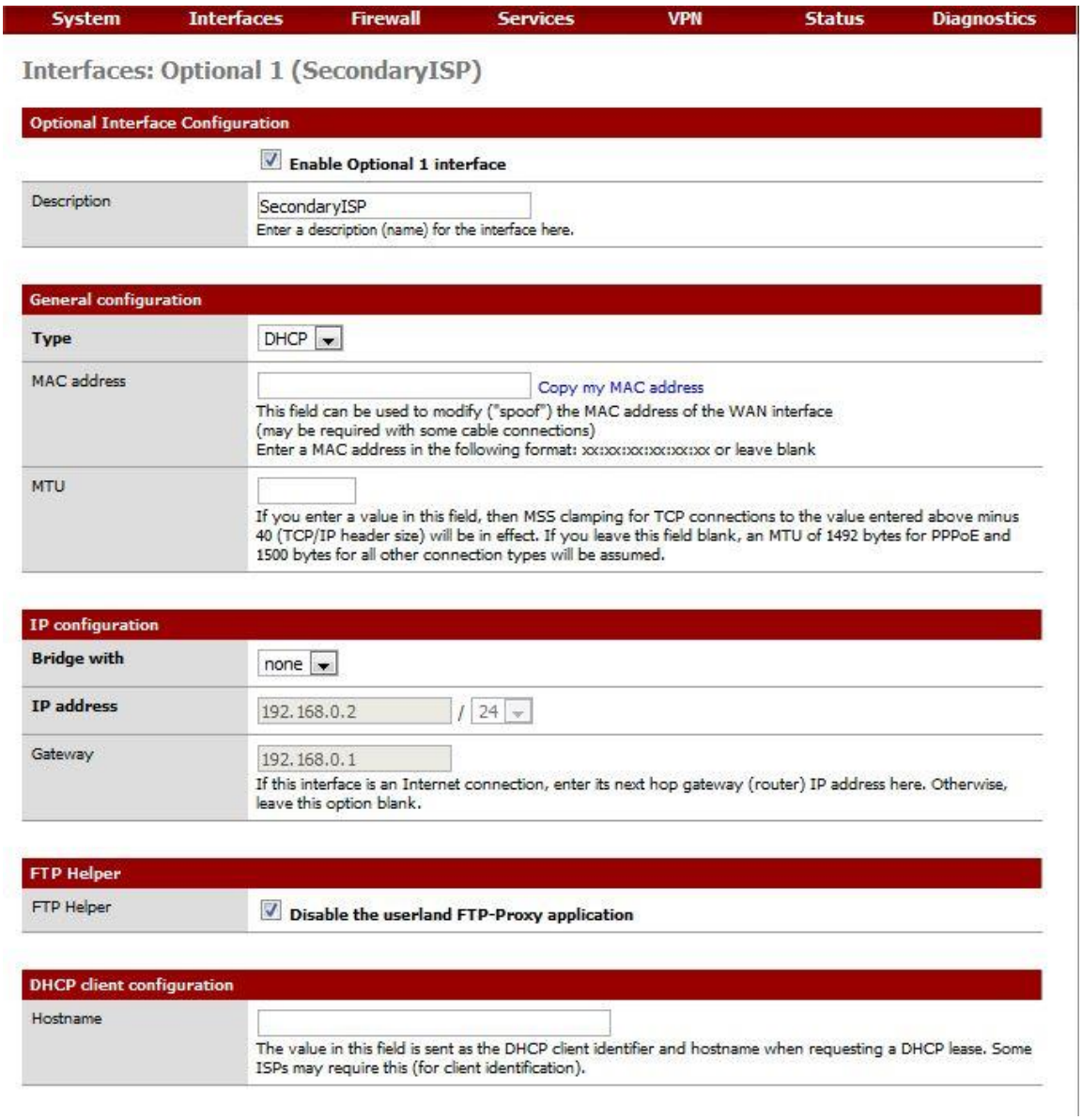


Figure 1: Enabling the second WAN interface

You can test your second WAN interface by changing the gateway on the already-established LAN routing rule, the one that directs LAN traffic through our current default gateway. Get the gateway for OPT1 from **Status Interfaces**, then under *Firewall->Rules*, edit the LAN rule, changing the gateway drop-down value to the OPT1 gateway IP as shown in Figure 2.

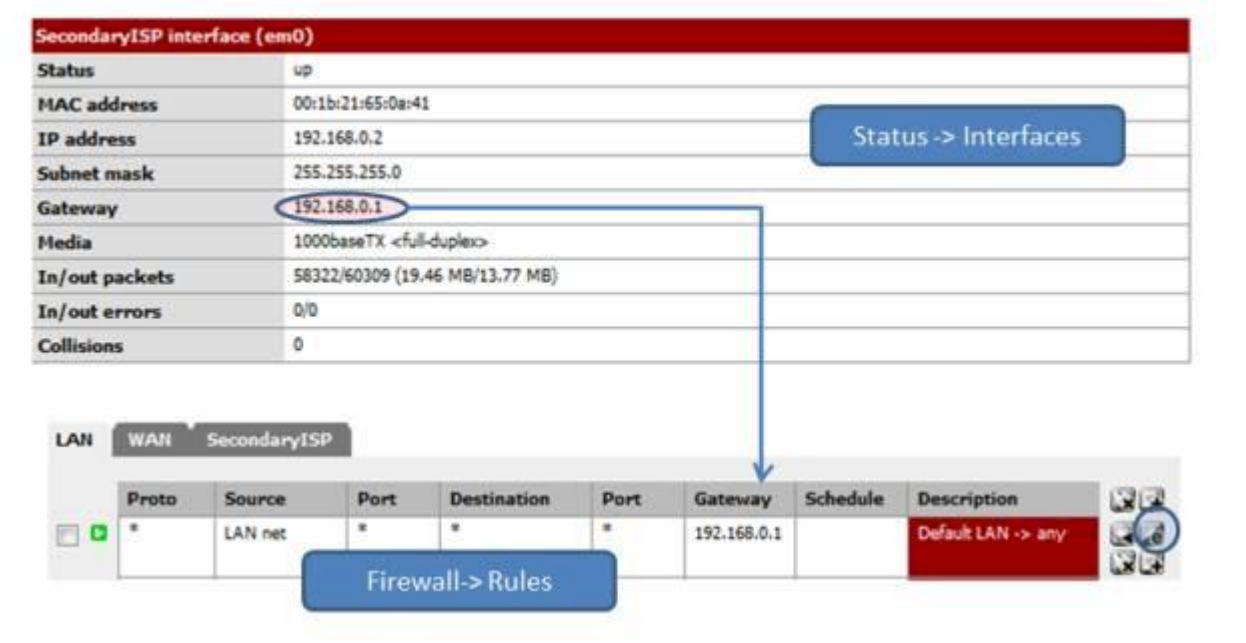


Figure 2: Testing the second WAN

Now from a web browser, visit the GRC Shields-Up Site. Your IP should correspond to your IP address from the secondary ISP. If you can't reach any web site, verify that the link is active by going to your modem/router diagnostics.

If the IP Address corresponds to your primary ISP, turn on logging for the routing rule, close your browser, and reboot your installation. Check the log once you are back up. If you still don't see the new IP address, verify your gateway settings. But hold off changing it back to the default gateway until after we've tested our IDS changes below.

That's it, done. We can now hang Snort on the Secondary WAN interface and set up the needed proxy servers. Load balancing and failover will come later.

Install Squid

Squid provides a tunable HTTP cache with traffic throttling. As with all cache servers, it trades disk I/O for network I/O. Your performance gain is largely dependent your bandwidth, the number of users, traffic volume, and the diversity of that traffic.

Significantly, there is a pretty cool chain here, and Squid is the heart of the whole thing. HAVP, the anti-virus proxy, runs as the parent of Squid, which in turn uses SquidGuard to filter content. All web requests travel through Squid's cache that contains (at least) twice-filtered content. This saves both bandwidth and scanning cycles for any subsequent reference to that content.

All packages are installed through the **Packages** menu on the **System** pull-down. Once installed, you need to configure Squid from *Services->Proxy Server*. We need to configure General settings and cache settings.

Most of the **General** settings are self-explanatory and PFSense has a tutorial to assist. The easy answer is that five fields have to be set as shown in

Table 1.

Setting	Explanation	Value
Proxy Interface	Interface Squid is bound to	LAN
Allow Users on Interface	Do not require separate subnet enumeration.	Checked
Transparent Proxy	Operate without separate network client configuration, everything through the proxy.	Checked
Log Store Directory	Where the logs live.	/var/squid/log
Proxy Port	Where other processes can find the proxy server, the default	3128

Table 1: Squid general settings

Figure 3 shows the settings for Cerberus.

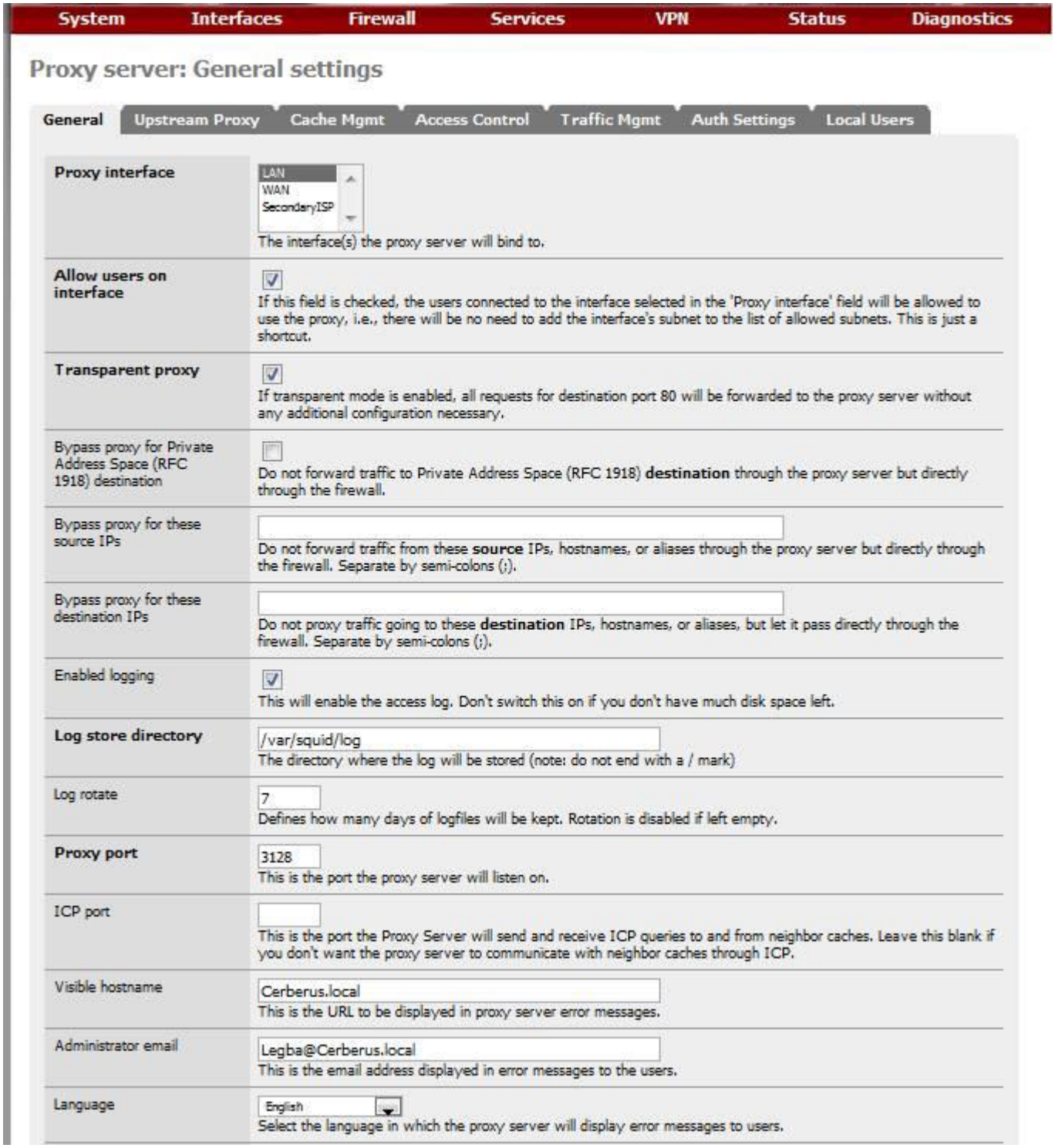


Figure 3: Squid proxy settings

And Figure 4 has a few more.

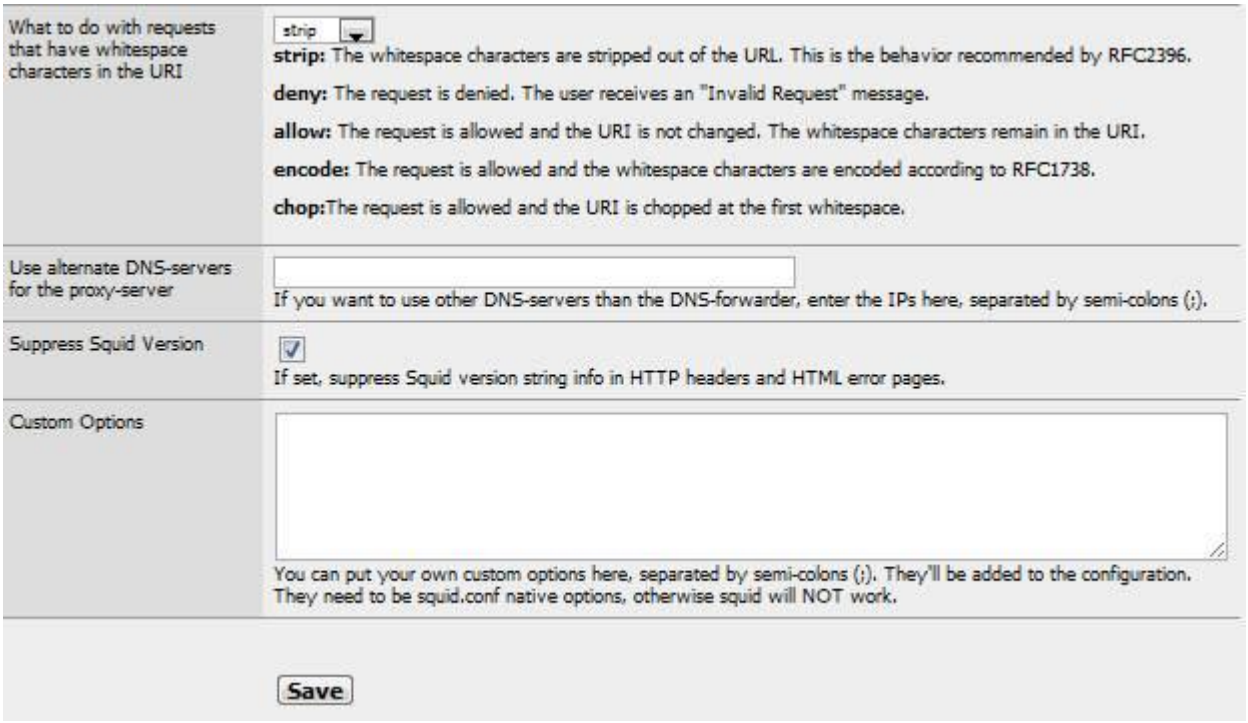


Figure 4: More Squid proxy settings

General Settings are now done. So save' em and move on to the **Cache Management** Tab.

We need to do some math before we determine cache size values. The temptation, since we have gobs of our 250 GB disk available, is to use a large chunk for web caching. The thing is that Squid uses an in-memory index to address the cache. So it is best to balance memory against disk cache size.

The Squid User Guide recommends 5 MB of memory for every Gigabyte of disk cache (you don't want to be thrashing, incurring a high swap rate). So determine how many megabytes of memory you have to spare for caching, divide that by 5, and you have the number of Gigabytes you should allocate to your cache.

With Cerberus under load and largely due to Snort, I run at 80% memory usage (according to *System->Status*), giving me about 600 MB free. I want some headroom for processing peaks, about half, so I have 300 MB available for my in-memory cache. Dividing that by the 5 to 1 guideline, I end up with a disk cache size of 60 GB.

Having calculated our sizes, we are ready to fill in the Cache Management configuration tab values, as summarized in Table 2.

Setting	Explanation	Value
Hard disk cache size	Disk size limit in megabytes	61400
Hard disk cache location	Where the cache is stored	/var/squid/log
Memory cache size	Megabytes of memory cache	300
Minimum Object Size	Smallest object to cache, in kilobytes.	0 (no limit)
Maximum Object Size	Largest object to cache, in kilobytes	256

Table 2: Squid Cache Management configuration tab values

I have also tweaked the optional tuning values: used threaded access to the UFS file system and since I have cycles to spare and a large cache, I've doubled the number of level 1 directories. I've also changed the memory replacement policy to Heap-LFUDA (Least Frequently Used with Dynamic Aging).
Figure 5 shows the settings for Cerberus.

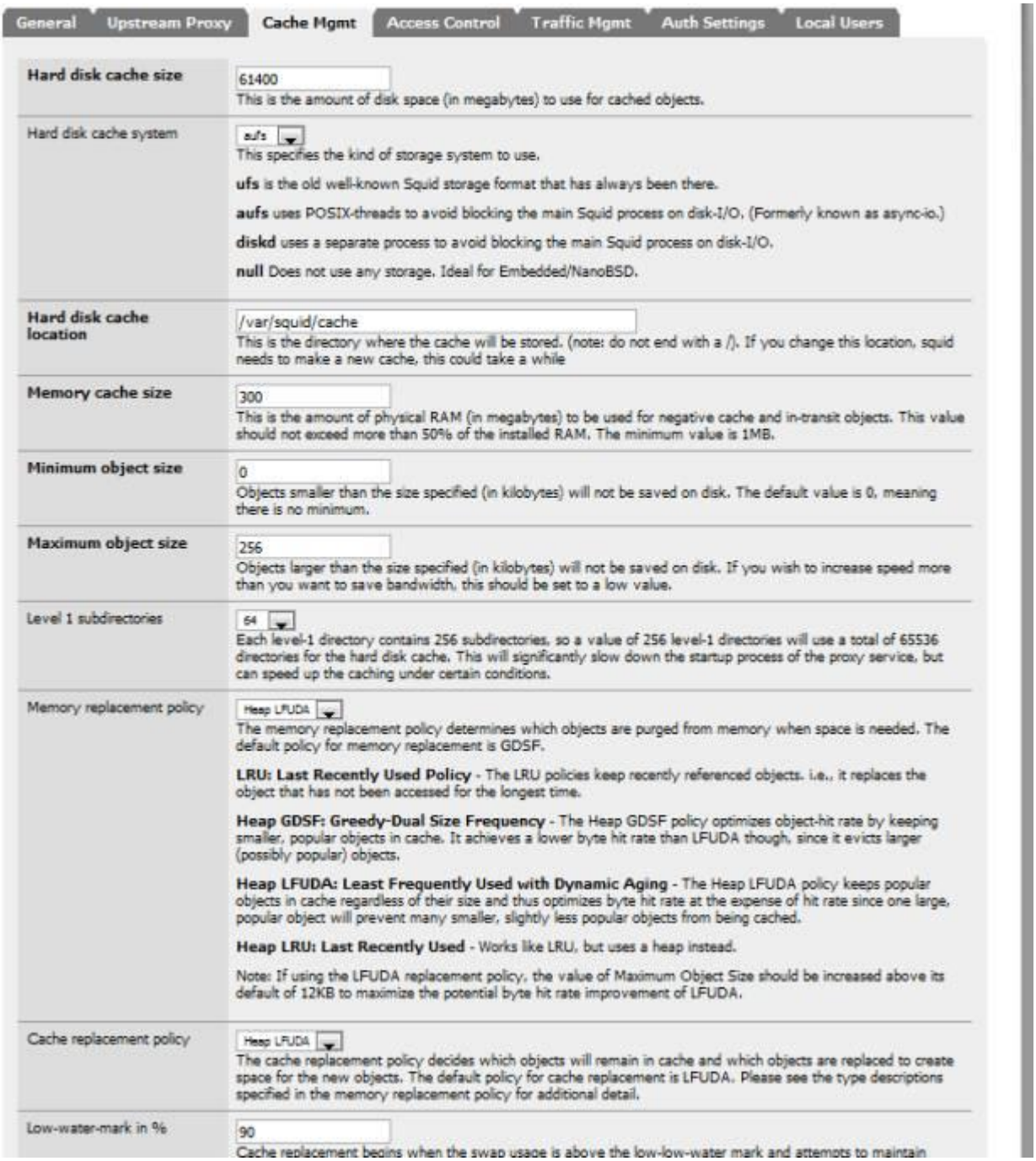


Figure 5: Squid Cache Management settings

To verify your Squid install, check the **System Log** (*Status->System Log*). If you need to track down any issues, there is a more detailed log you can use. Execute a BSD command (*Diagnostics->Command*) to access it; it is located here: */var/squid/log/cache* and should look like Figure 6.

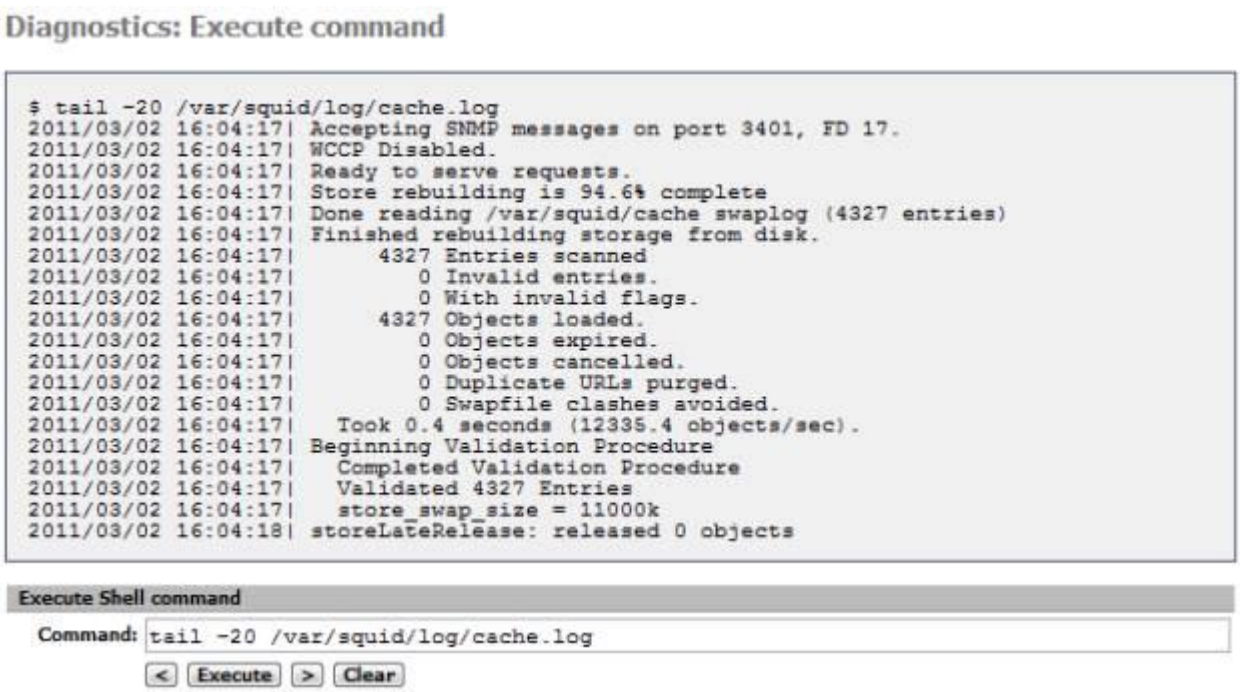


Figure 6: Squid cache.log

Additionally, to review web accesses, you can take a look at the *access.log* file in the same directory. Or install the partially-integrated Squid reporting tool, LightSquid, which gives you a view of cache hits, including Top Sites and hit percentages.

With Squid installed, we are done with the prerequisites. Let's start the main event, the functional upgrades needed to become a UTM.

Intrusion Detection and Prevention Configuration

Cerberus is already an IDS Firewall. In the previous article Build Your Own IDS Firewall With pfSense the installation and configuration of Snort was covered in detail. So there is little that needs to be done further for it. We do need to add our new **OPT1 WAN** connection, however and rearrange our rules.

We are going to want the same overall protection on both WAN interfaces. So under *Services->Snort*, add both the new **OPT1** interface and your **LAN** interface. The OPT1, Secondary ISP interface should be a clone of your Primary Interface, i.e. same pre-processor settings, same rules, as shown in Figure 7.

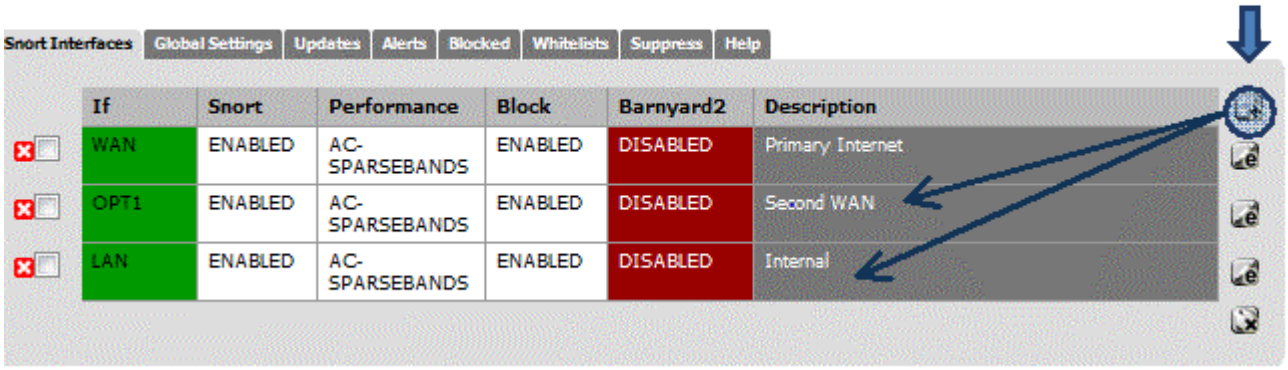


Figure 7: Squid interfaces

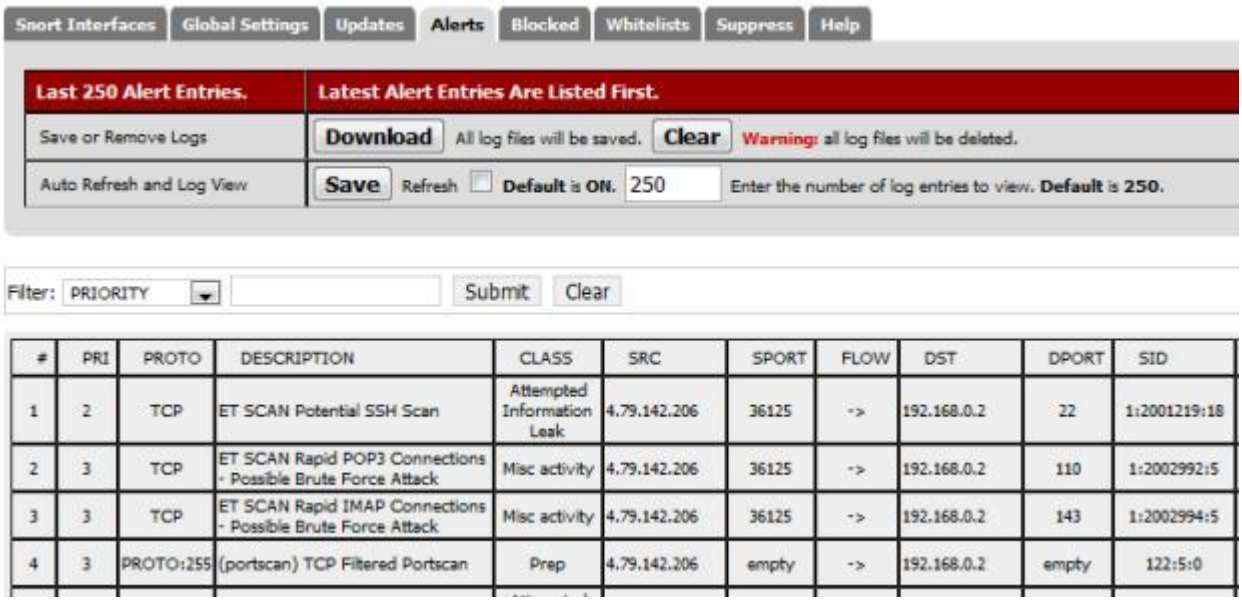
The **LAN** interface, on the other hand, is lightweight with just the pre-processor defaults and **HTTP Inspect** checked. It should handle just a few categories of rules. The idea here is to offload a few categories from your WAN interfaces to the LAN's where it would be good to know which LAN IP is being attacked and whether the attacks are coming from the inside. Examples categories would be **NetBios** and **ICMP**.
Your mileage may differ and you may want to expand the categories that generate alerts. Figure 8 shows the selected categories on Cerberus.

EMERGING THREAT Rules	SNORT Rules	PFSense Rules
activex.rules	attack-responses.rules	voip-rules
attack_response.rules	backdoor.rules	
chat.rules	bad-traffic.rules	
compromised.rules	bad-traffic.so.rules	
current_events.rules	blacklist.rules	
deleted.rules	botnet-cnc.rules	
dns.rules	chat.rules	
dos.rules	chat.so.rules	
drop.rules	content-replace.rules	
dshield.rules	ddos.rules	
exploit.rules	deleted.rules	
ftp.rules	dns.rules	
games.rules	dos.rules	
icmp.rules	dos.so.rules	
icmp_info.rules	experimental.rules	
imap.rules	exploit.rules	
inappropriate.rules	exploit.so.rules	
malware.rules	finger.rules	
misc.rules	ftp.rules	
netbios.rules	icmp-info.rules	
p2p.rules	icmp.rules	
policy.rules	icmp.so.rules	
pop3.rules	imap.rules	
rbn.rules	imap.so.rules	
rpc.rules	info.rules	
scada.rules	local.rules	
scan.rules	misc.rules	
shellcode.rules	misc.so.rules	
smtp.rules	multimedia.rules	
snmp.rules	multimedia.so.rules	
sql.rules	mysql.rules	
telnet.rules	netbios.rules	
tftp.rules	netbios.so.rules	
tor.rules	nntp.rules	
trojan.rules	nntp.so.rules	
user_agents.rules	oracle.rules	
virus.rules	other-ids.rules	
voip.rules	p2p.rules	
web_client.rules	p2p.so.rules	
web_server.rules	phishing-spam.rules	
web_specific_apps.rules	policy.rules	
worm.rules	pop2.rules	
	pop3.rules	
	rpc.rules	
	rservices.rules	
	scada.rules	
	scan.rules	
	shellcode.rules	
	smtp.rules	
	smtp.so.rules	
	snmp.rules	
	specific-threats.rules	
	spyware-put.rules	
	sql.rules	
WAN Rules	sql.so.rules	
	telnet.rules	
LAN Rules	tftp.rules	
	virus.rules	
	voip.rules	
	web-activex.rules	
	web-activex.so.rules	
	web-attacks.rules	
	web-cgi.rules	
	web-client.rules	
	web-client.so.rules	
	web-coldfusion.rules	
	web-frontpage.rules	
	web-iis.rules	
	web-iis.so.rules	
	web-misc.rules	
	web-misc.so.rules	
	web-php.rules	
	x11.rules	

Figure 8: Alert categories

Remember, the more rules you select, the higher the probability of false positives, which can be an administration headache.

After adding the additional interfaces and configuring them, start Snort by clicking the green arrow next to the interface definition. We can test these additions to Snort by using the GRC Shields-Up Site to scan the added Secondary ISP WAN interface. Your Snort Alert log should look something like Figure 9.



#	PRI	PROTO	DESCRIPTION	CLASS	SRC	SPORT	FLOW	DST	DPORT	SID
1	2	TCP	ET SCAN Potential SSH Scan	Attempted Information Leak	4.79.142.206	36125	->	192.168.0.2	22	1:2001219:18
2	3	TCP	ET SCAN Rapid POP3 Connections - Possible Brute Force Attack	Misc activity	4.79.142.206	36125	->	192.168.0.2	110	1:2002992:5
3	3	TCP	ET SCAN Rapid IMAP Connections - Possible Brute Force Attack	Misc activity	4.79.142.206	36125	->	192.168.0.2	143	1:2002994:5
4	3	PROTO:255	(portscan) TCP Filtered Portscan	Prep	4.79.142.206	empty	->	192.168.0.2	empty	122:5:0

Figure 9: Snort Alert log

If you have an ISP-provided router instead of just a modem, you need to either put pfSense in the DMZ or configure your router to run as a transparent bridge.

Since ISP routers are a known attack vector, transparent bridging is recommended.

For example, out of the box, the Qwest branded Actiontec Q1000 has multiple ports open, including HTTPS for remote administration. For the purposes of obscuring my logged IP address in this article, Cerberus has just been put in the DMZ.

Once this is complete, you will want to reverse the changes made when testing your multi-WAN configuration and change your LAN traffic rule back to using the default gateway (our primary ISP).

Anti-Virus Install with pfsense

HAVP, our anti-virus solution, has pretty much a point and shoot setup. Once installed, there are only a few settings (*Services->Anti-Virus*) to change on the HTTP proxy tab:

Setting	Explanation	Value
Enable	Turn on scanning	Checked
Proxy mode	Define Run Mode	Parent of Squid
Proxy port	Connection Port. Must be different than Squid port	3125

Table 3: Anti-virus settings

There are several other discretionary settings including file types to scan, logging, etc. Figure 10 shows the settings for Cerberus.

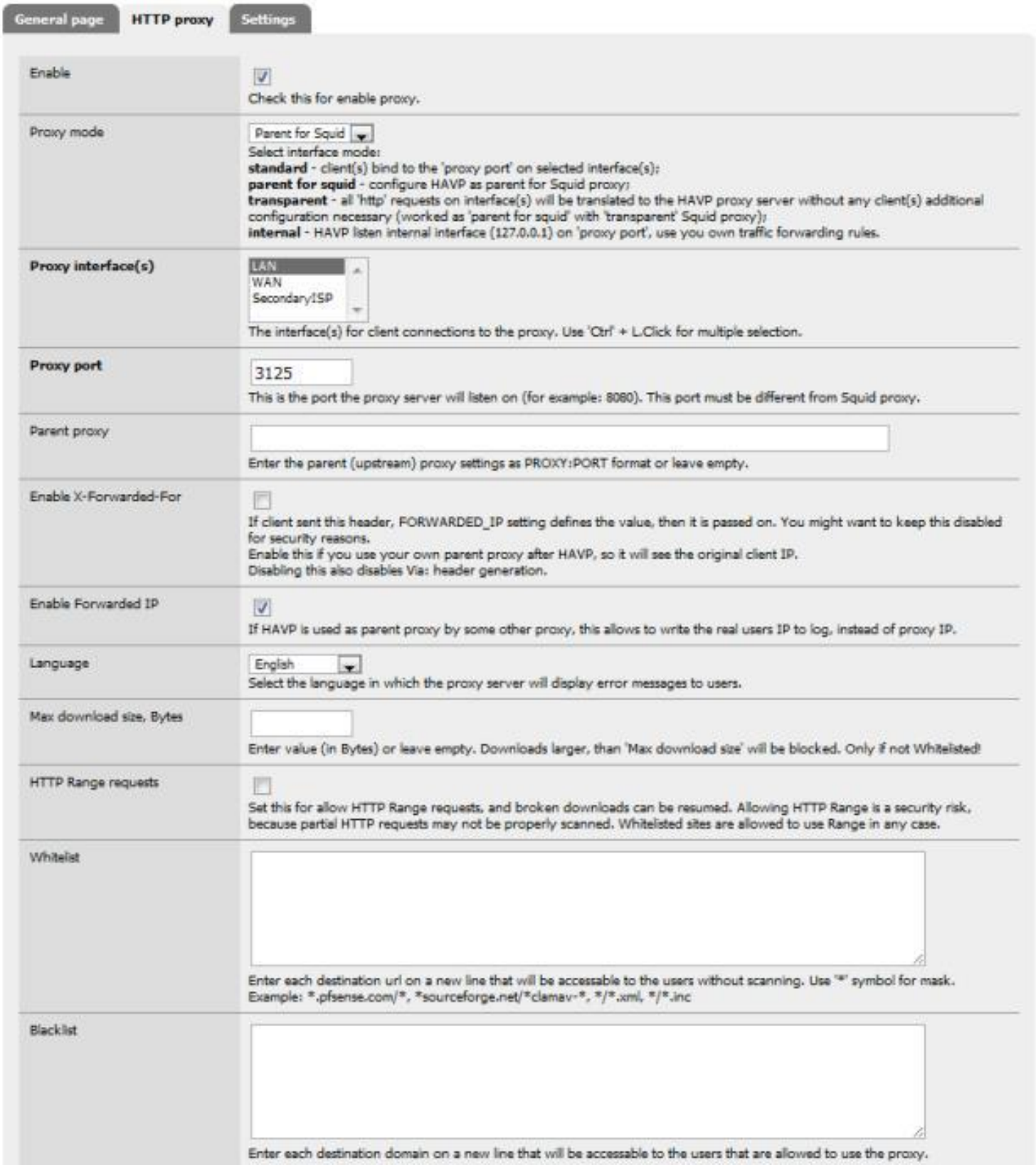


Figure 10: HTTP proxy settings for anti-virus

And a few more in Figure 11.

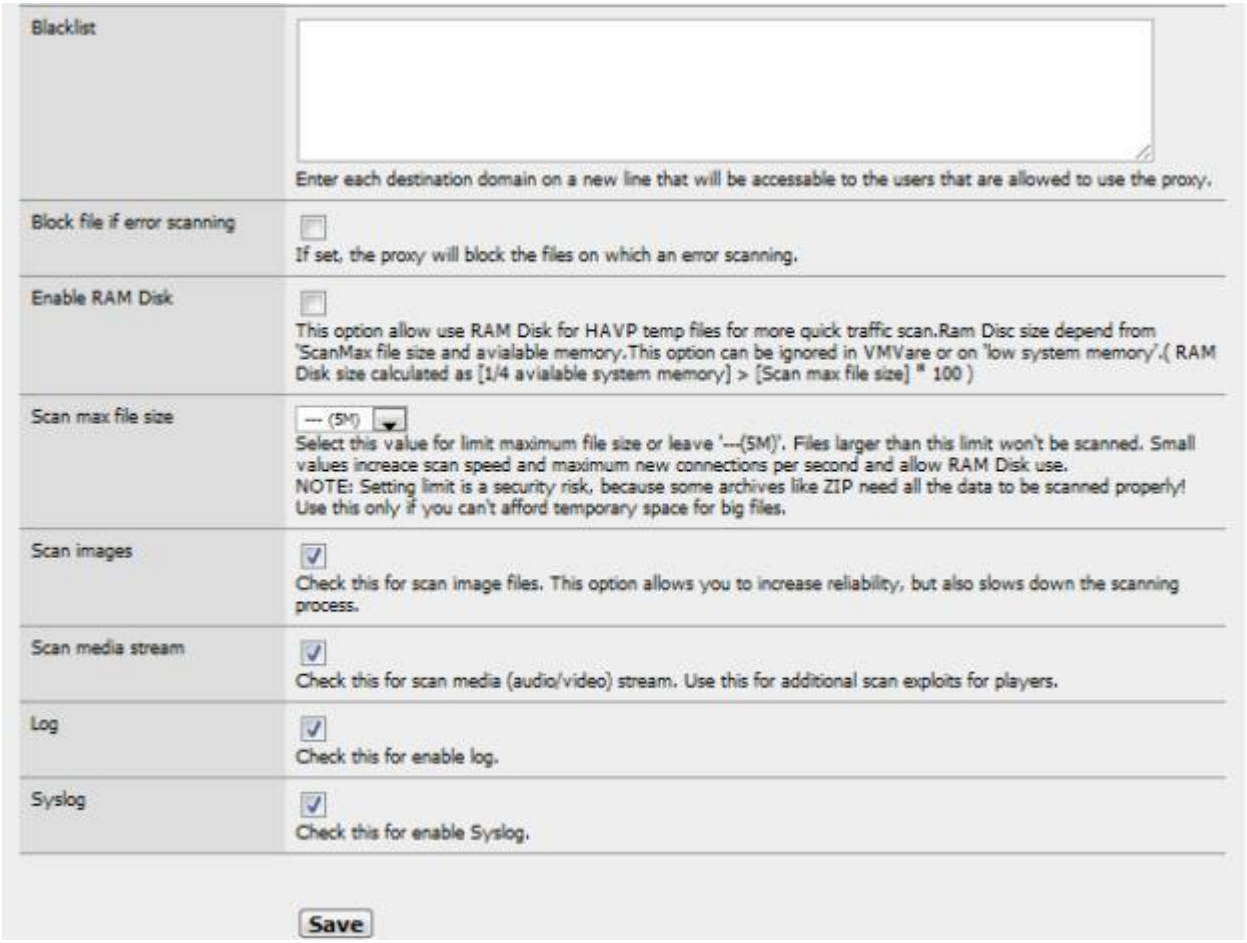


Figure 11: More HTTP proxy settings for anti-virus

There are also some minor settings under the **Settings** tab dealing with update frequency and logging. Figure 12 shows how Cerberus is configured.

Figure 12: Miscellaneous AV settings

Once you have saved your settings, you can verify that both the HAVP proxy and the ClamAV scanning engine are running under the **General** page tab:

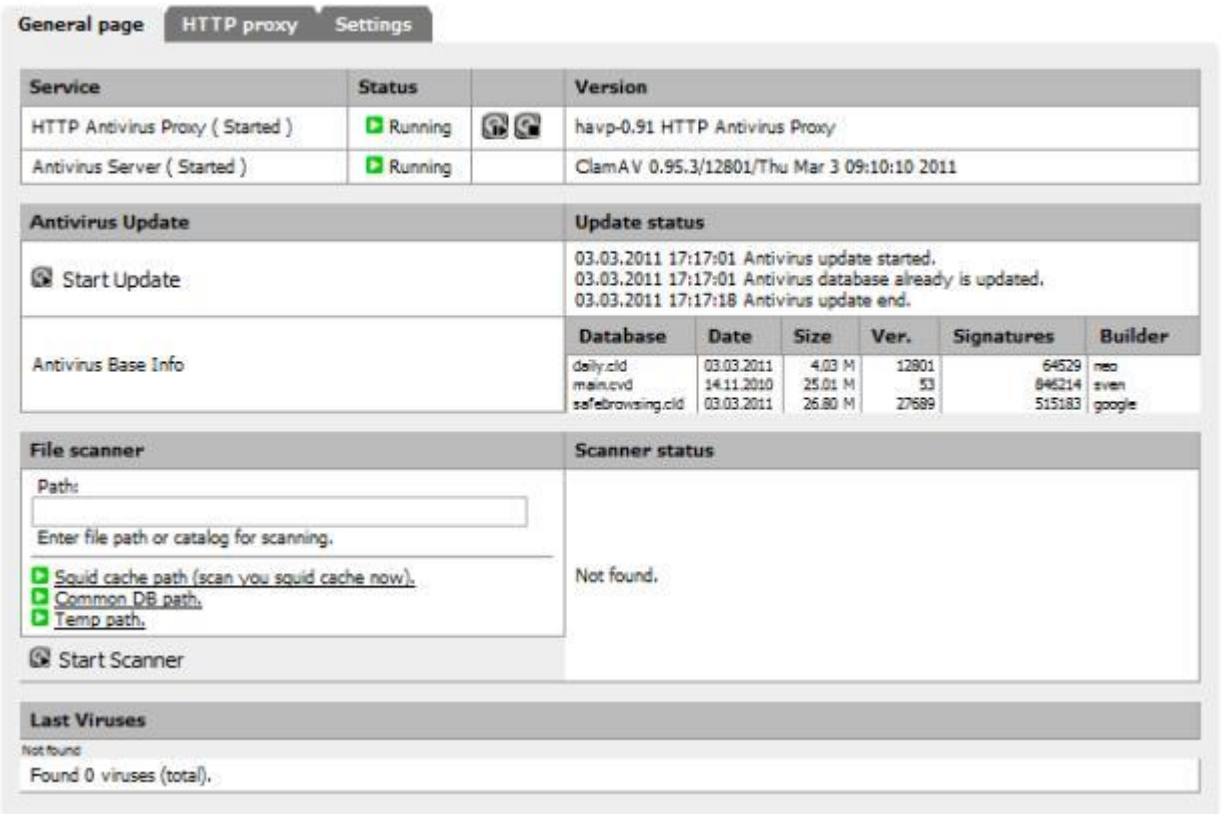


Figure 13: HAVP and ClamAV running

Once you are fully updated (should take about ten minutes), you can test your install using [safe virus simulation files provided by Eicar.org](http://www.eicar.org/anti_virus_test_file.htm).

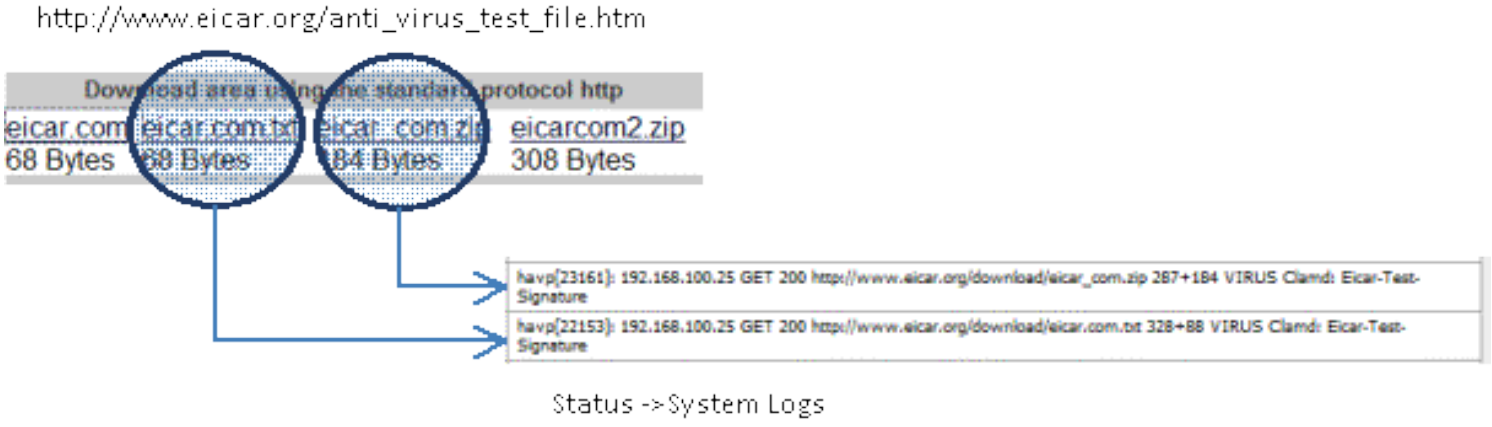


Figure 14: Eicar.org virus test file

Only two of the test files are recognized as threats. Files with the extension COM are not scanned, and embedded archives are not tested, underlining the need for separate anti-virus on each host machine.
Anti-Virus is now up and running.
That's it for this installment. Next time, we'll continue the conversion to UTM with Content Filtering setup and plenty more.

Introduction to content Filtering squid guard

Here, we established a working definition of our target, i.e. what has to be done, and in what order, to Cerberus the lowly IDS firewall to make it a UTM Appliance. In Part Two, we started the conversion by installing and configuring multi-WAN support, Squid, IDS and anti-virus features. This time, we'll add and configure Content Filtering, Traffic Control, Load Balancing and Failover.

Content Filtering

As introduced in the first part of this article, pfSense has several packages for content filtering, from the simple to the sublime. When setting up Cerberus in the previous article, Build Your Own IDS Firewall With pfSense, we installed the first of these, **IP Blocklist**, which blocks IP addresses based on lists downloaded from a clearinghouse of list maintainers, i.e. iBlocklist.com. There you will find a large assortment of list flavors: Adult Sites, Compromised Sites, Torrent Sites, etc.

In addition to IP Blocklist, there are two very simple packages to install: Country Block and DNS Blacklist. **Country Block** is geared towards blocking the countries responsible for the highest volume of Spam, but can be used to block the Individual countries. It uses the national CIDR ranges from CountryIPBlocks.net.

Once installed, it is simple to configure. Select the countries you wish to block from a list of all countries. At the top, you'll find a list of countries responsible for the largest volume of spam. Enable the service, select the countries you want to block, commit your selections, and save. Done.

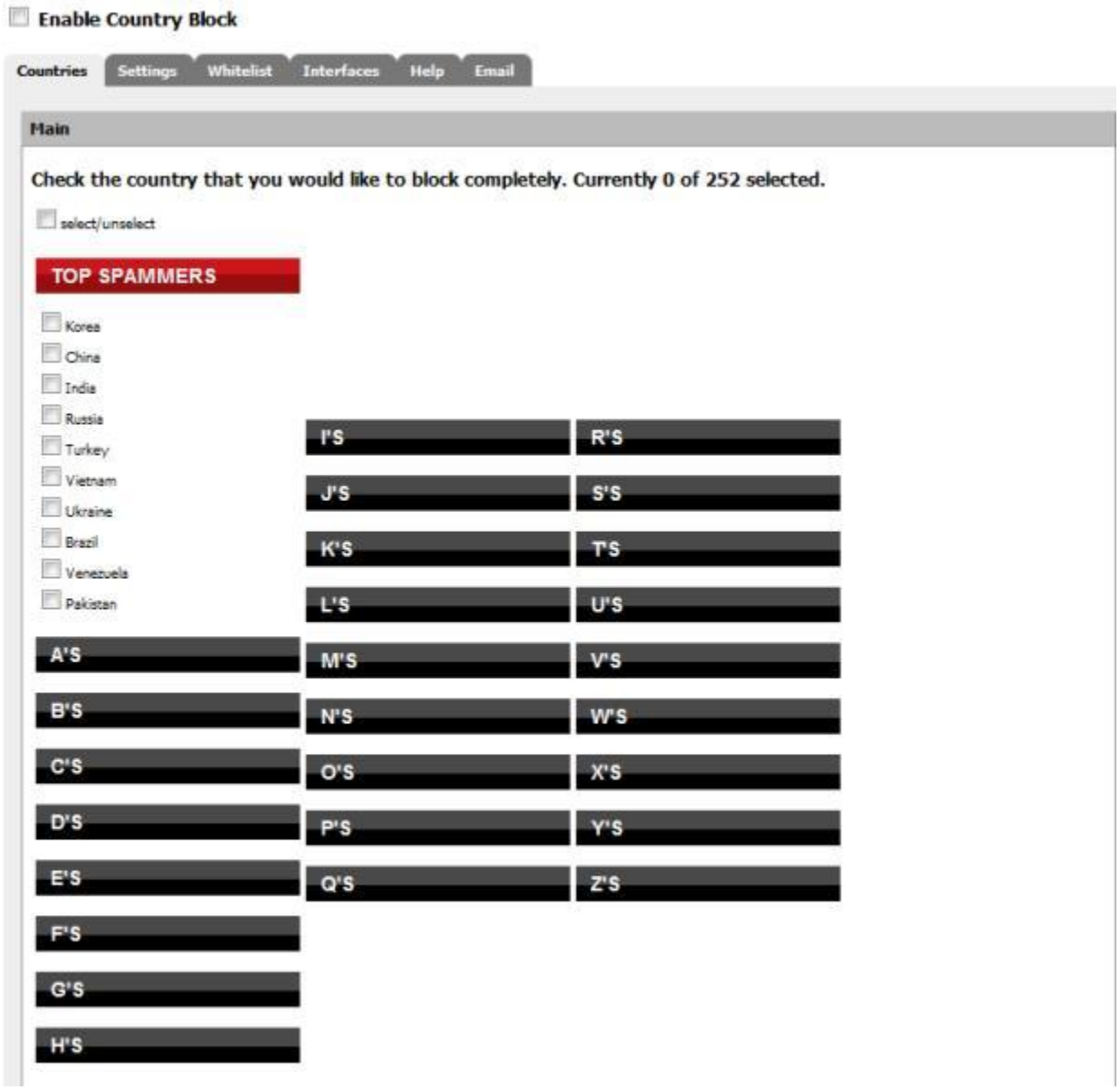


Figure 1: Country Block configuration

Incoming traffic is blocked by default, but this can be changed along with logging on the **Settings** tab. You can also limit blocking to a particular interface, but it defaults to all interfaces.

The other simple package, **DNS Blacklist**, allows you to block specific categories of domain names. The package forces DNS to resolve all domains listed in the selected categories to Google's IP address. The categorized domain list is originally from the Université Toulouse 1 Capitole, and has been wrapped into the release. This means the lists are static, and are not updated regularly, limiting overall usefulness, unless you choose to update them manually.

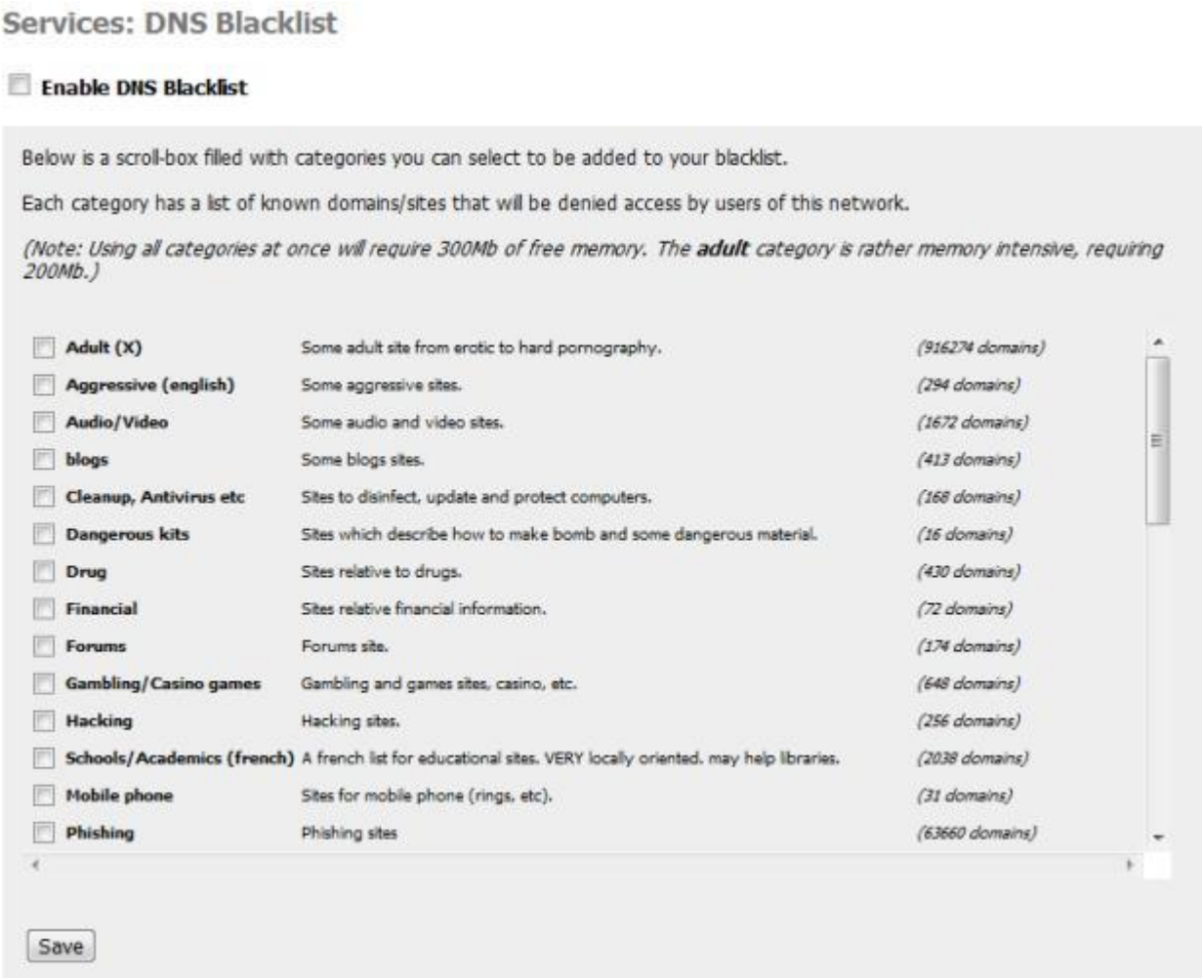


Figure 2: DNS Blacklist configuration

DNS Blacklist offers a very lightweight alternative to the content filtering heavyweight, **Squid Guard**. It uses DNSMasq as a DNS Forwarder, so requires no proxy server or complex indexing.

SquidGuard

The other alternative to content filtering is **SquidGuard**, a full bodied content filtering system that has more controls than a Gemini space capsule and is just as hard to get in and out of. To complicate this further, the SquidGuard tutorial on pfSense.org has gone 404.

Even with the difficulties of configuring SquidGuard, the functionality is compelling. You can choose what to block, for whom to block it, from what time to what time should the whole block thing happen, per entry.

The initial setup is a bit convoluted and requires a bit of dancing. First, you should select the blacklist provider you

want to use. A meta-list is available from SquidGuard.org. The recommended set of lists is Shalla's Blacklists (List Archive: http://www.shallalist.de/Downloads/shallalist.tar.gz).

Starting with the **General** tab (Services->Proxy Filter), enable the blacklist and paste the URL of your list archive, a tarball, into the value for Blacklist URL. Go ahead and save **without** enabling SquidGuard yet.

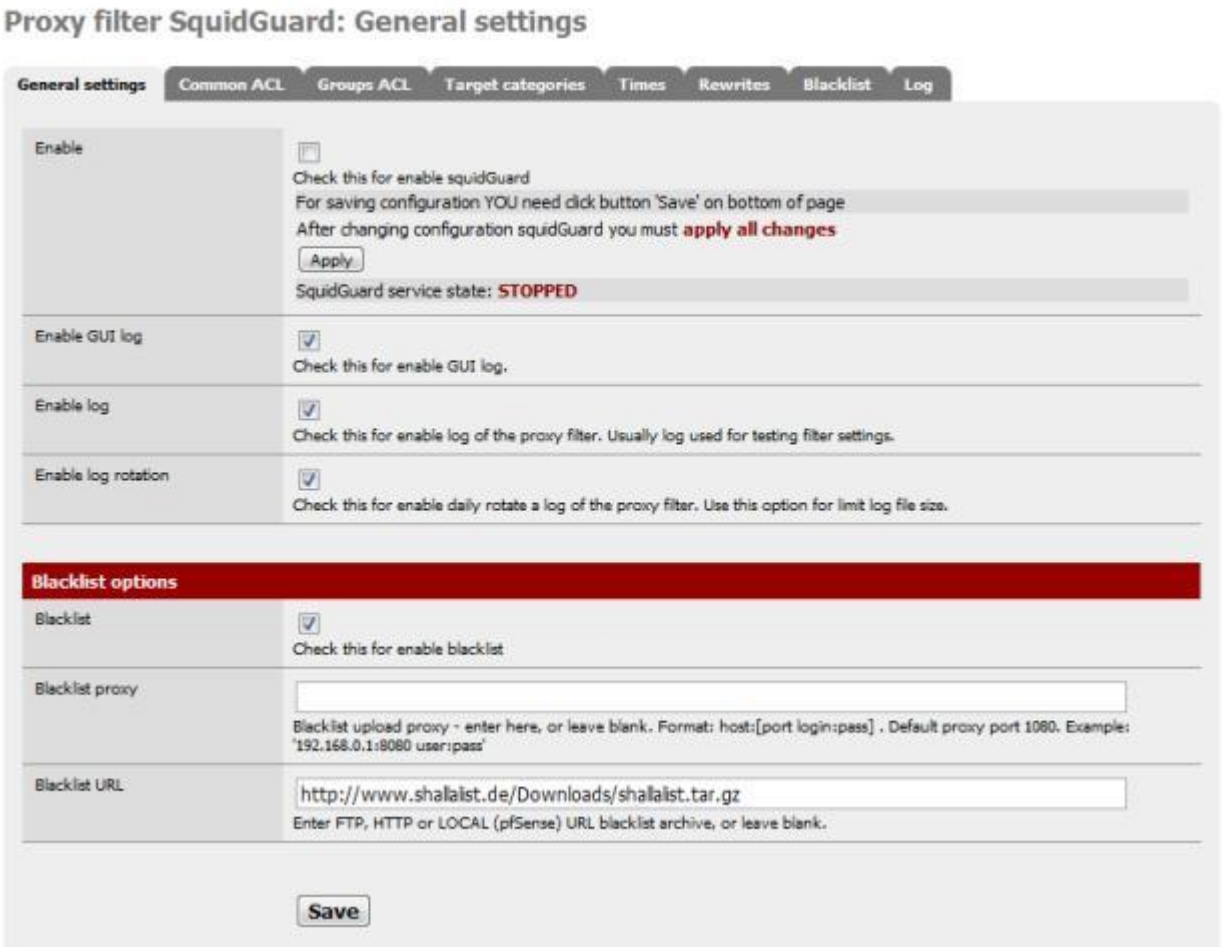


Figure 3: SquidGuard General Settings

Now move to the **Common ACL** tab. The common access control list handles filtering policy for everyone, and by default, web access is denied. We need to set it to *ALLOW* before enabling SquidGuard, otherwise we would lose all web access.

Expand the **Target Rules** List, there should be one entry, **Default Access**, set this to *ALLOW* and save.

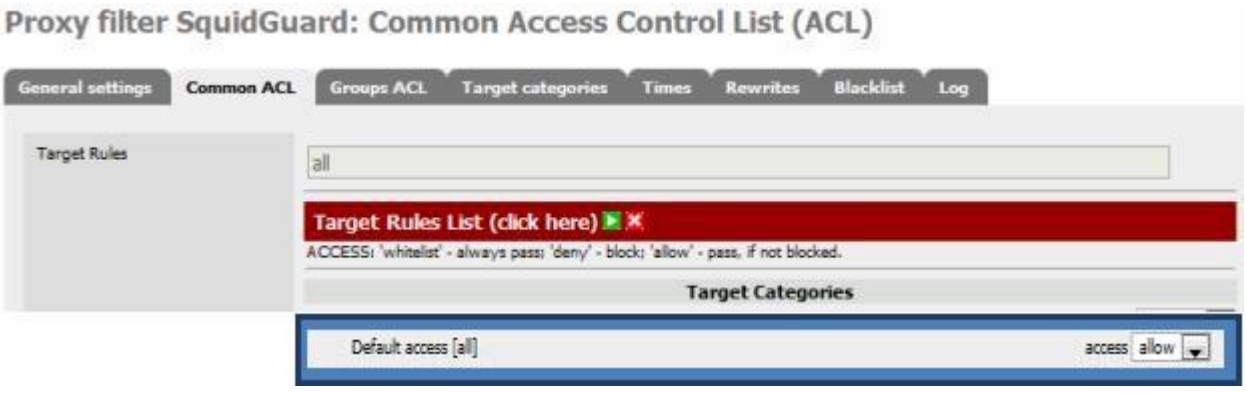


Figure 4: SquidGuard Common ACL setting

We are still not ready to turn the key yet. We need to go get our blacklists, so move to the **Blacklist** tab. If the URL field doesn't contain your selected list URL, copy it from the **General** tab and download the list. It will be downloaded and loaded into SquidGuard database. Wait for the download to complete; this may take up to ten minutes, depending on the list archive.

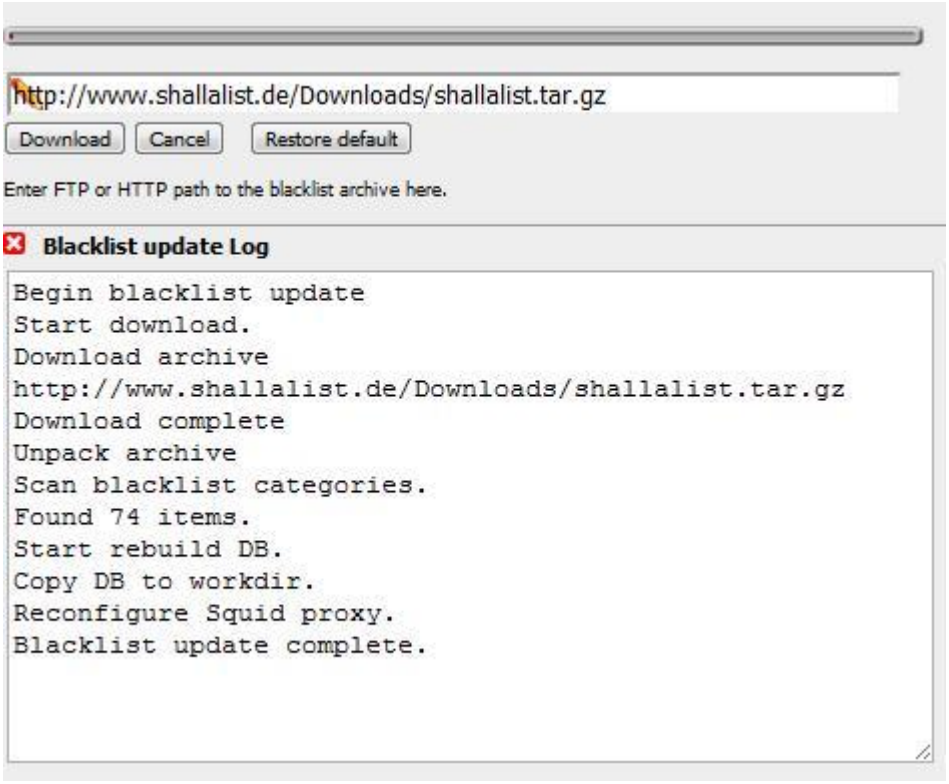


Figure 5: Blacklist download

Once we verify we have a blacklist, we will be ready to kick-start this beast. Return to the **Common ACL Tab** and expand the **Target Rules** List. It should look like this now:

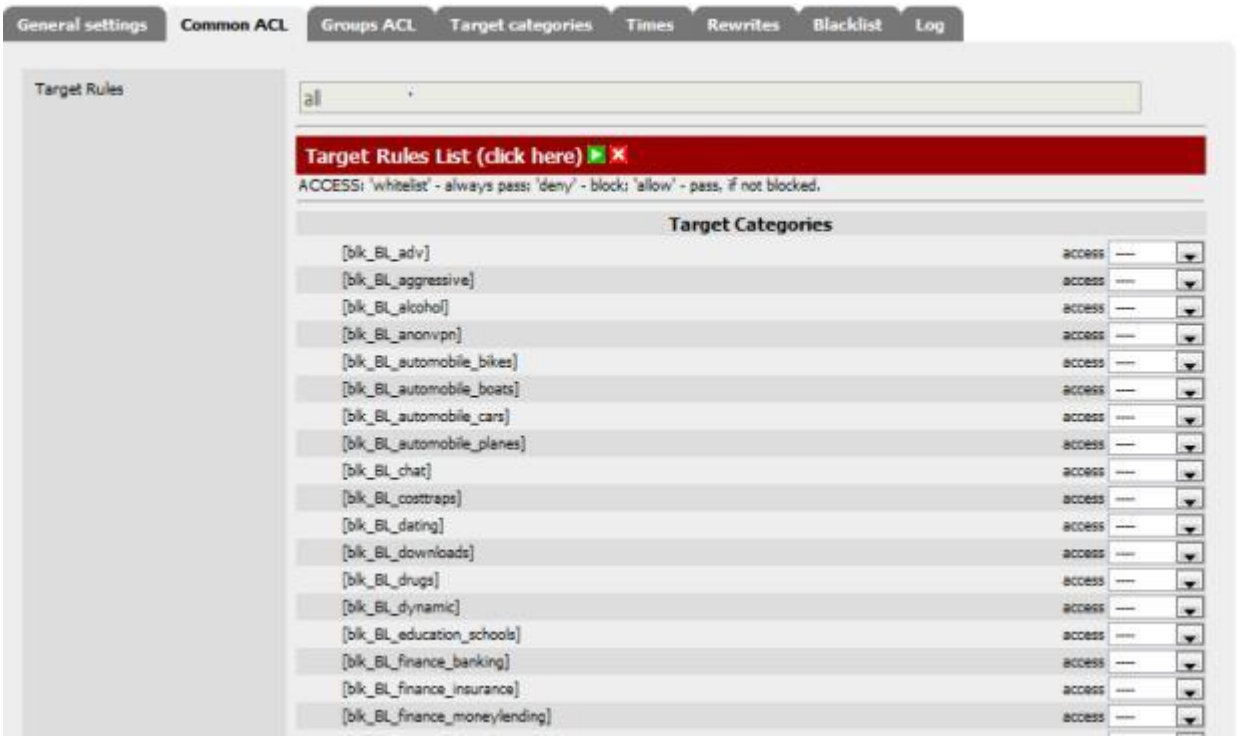


Figure 6: SquidGuard Common ACL Target rules

Now return to the **General Settings** tab, check all the logging you can, and check *Enable*. Save these changes and wait for the SquidGuard Service State to change to **Started**.

To verify that it is up and running, check the **Filter Log** under the **Logging** tab. If all looks good, go to the **Common ACL** tab and set the blacklist *blk_BL_hobby_pets* to DENY and Save. Return to the **General Settings** tab and click Apply. Now, try to go to the French Bulldog Club.
You should see:

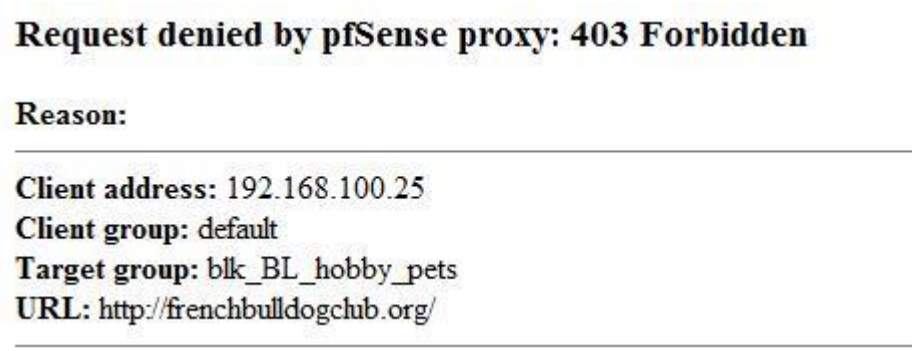


Figure 7: URL denied

This is just the tip of the iceberg for SquidGuard. For example, it would be possible to redirect any references to the Fox News site to that of the NY Times, from 9 AM to 9:10 AM ...on only Karl the programmer's machine. Or more importantly, ensure that your kids are actually using the Internet to do their homework after school, instead of Facebook.

Traffic Control

Though Traffic control is central to pfSense, there are some serious limitations in the current version. Traffic shaping in Version 1.2.3 doesn't handle either Squid HTTP traffic or failover. (Squid uses your loopback interface, which is not shaped, but there is a workaround). Version 2, to be released soon, supposedly does.

Traffic shaping can be effective on a single WAN system or multi-WAN, but just on a single WAN interface with static routing. For example, you can direct all file transfer protocols (P2P, FTP, etc) through your secondary WAN interface, and leaving HTTP on the primary interface.

I will introduce traffic shaping. But full traffic shaping is complex, requiring specific details of not only your traffic, but of use patterns. This kind of traffic shaping is outside the scope of this article; more details can be found in the pfSense forums.

The Wizard sets up initial traffic queues and rules that can then be tuned; it uses your actual bandwidth figures to allocate traffic across the defined queues. So, before you start, you will need to gather your bandwidth figures, both up and down, using any number of sources (DSLReports, for example).

The first time you go to the **Traffic Shaper** (Firewall->Traffic Shaper) you will be presented with the wizard interface, which will step you through setting up traffic queues for the traffic you want to shape.

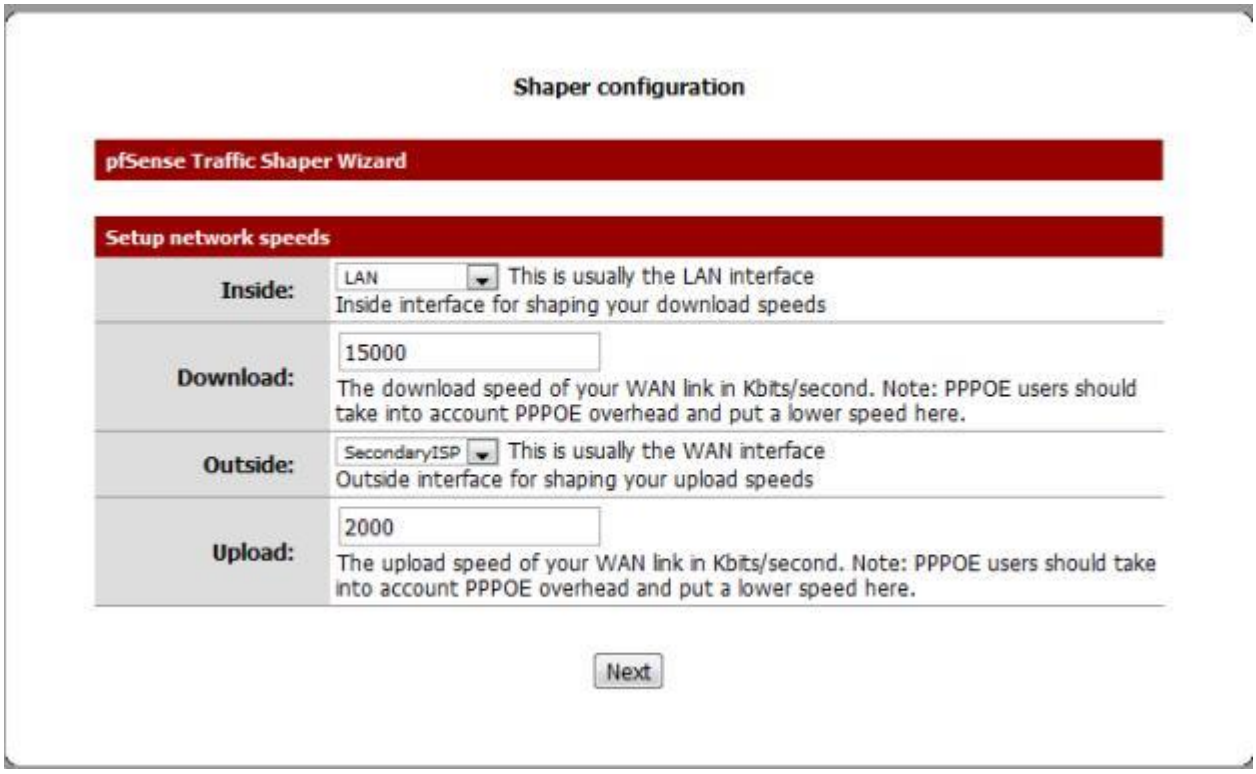


Figure 8: Traffic Shaper Wizard

Here are the options for types of traffic that can be prioritized:

Traffic Type	Description
VoIP	Higher priority for VOIP traffic, generic or Vonage, Voice Plus, Asterisk
Peer To Peer	Allocate Bandwidth to generic P2P traffic, or Disable and Lower priority for about 20 protocols of P2P traffic
Gaming	Increase priority for about 20 Games, including BattleNet, WOW, Xbox360
Other	Set priority for about eight categories including VPN, IM, HTTP, and Multimedia

Table 1: Traffic Shaper options

You can also define a **Penalty Box**, a specific IP or alias to limit if traffic levels are high.

Once you finish the wizard, it will generate traffic queues, which are essentially separate sets of routing rules. When you return to the Traffic Shaper, you will now have three tabs: **Rules**;**Queues**; and a tab for rerunning the wizard.

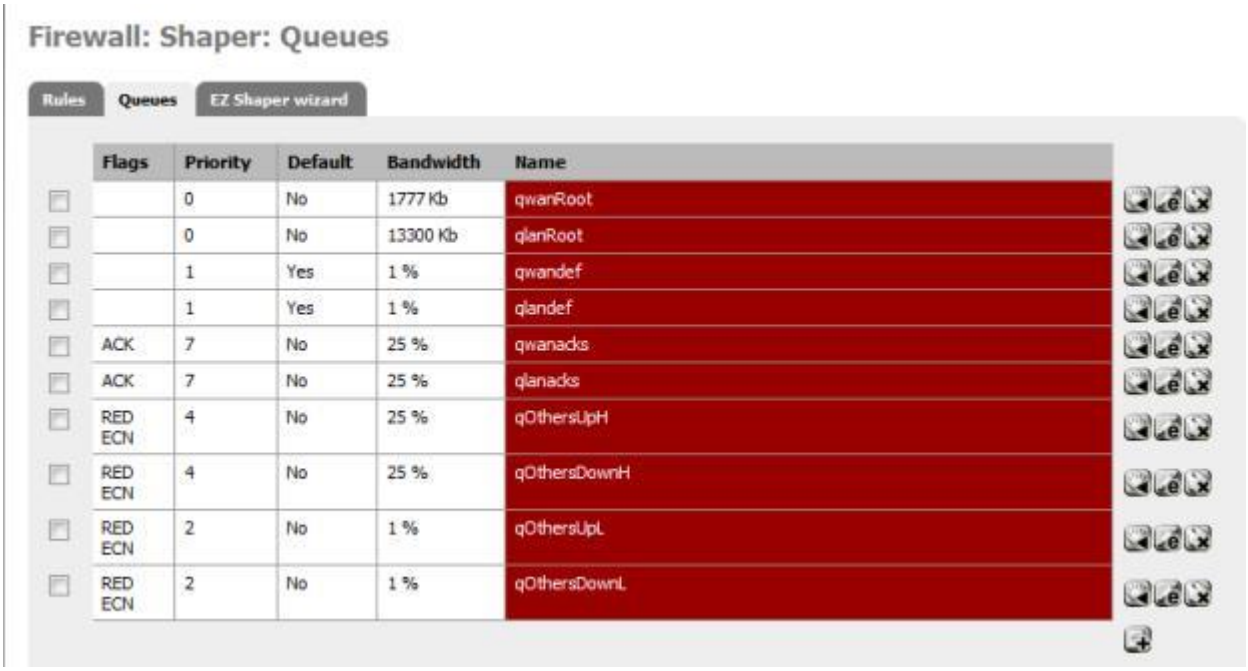


Figure 9: Traffic Shaper queues

The values and order of the rules can all be tuned to prioritize traffic. By editing a queue, you can change the traffic percentage, and the corresponding priority of the traffic.

To verify that traffic is moving through your queues, go to the **Queue Status** page (Status->Queues). The various bar graphs should dynamically show changes in traffic patterns after a short delay. Attention should be paid to any drops, which indicate traffic problems.

Figure 10: Traffic Shaper queue status

Both Squid and Snort offer traffic control facilities. Squid offers both transfer caps and throttling under the **Traffic Management** tab of the Squid page (Services->Proxy Server). These settings are straightforward, and allow for throttling of particular categories of downloads.

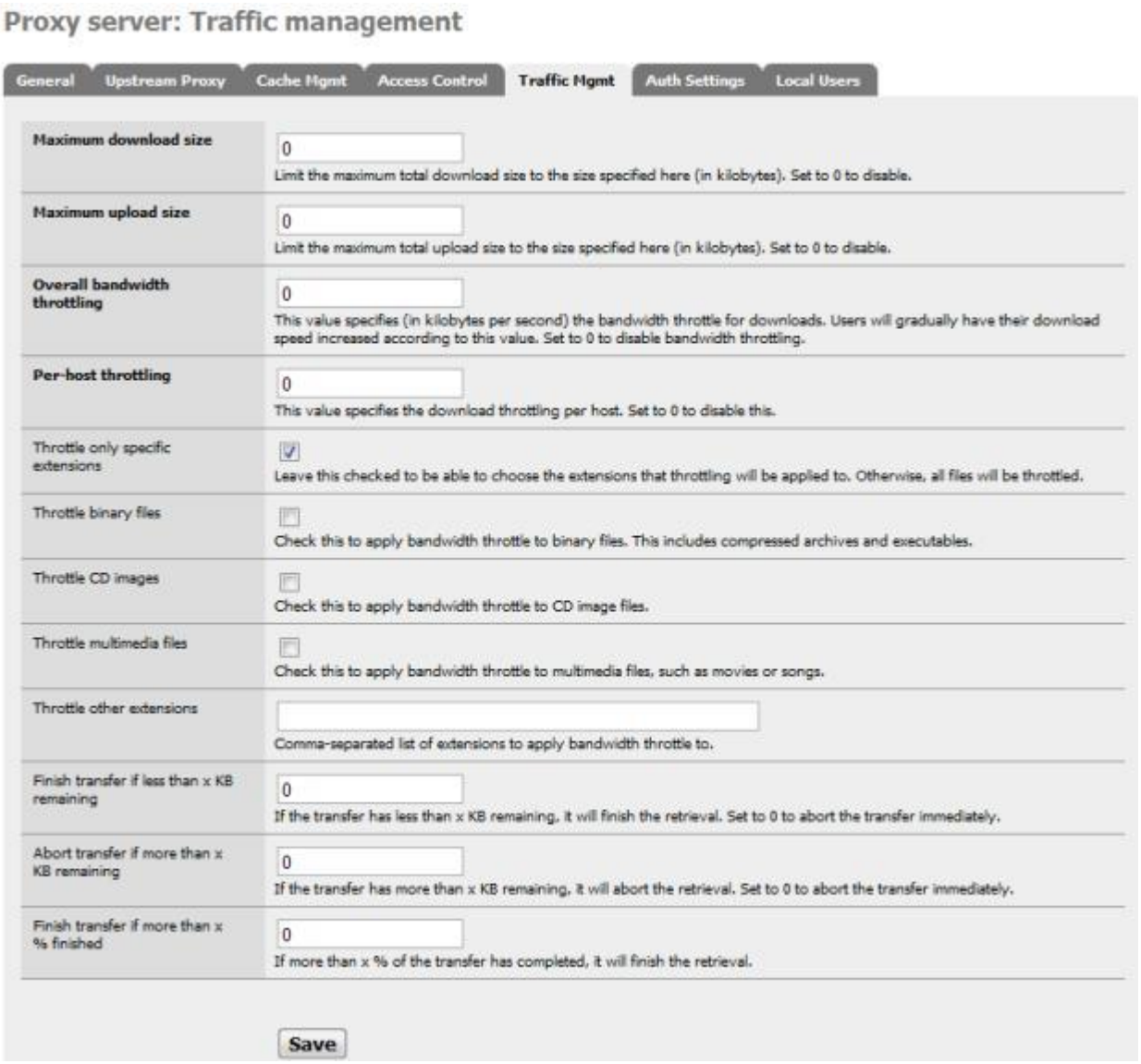


Figure 11: Squid traffic management

Snort, on the other hand, offers rules for blocking certain protocol traffic , such as IM Traffic (emerging and snort chat.rules) and P2p traffic (snort and emerging p2p.rules).

Load Balancing & Failover

Now we are going to set up load balancing and failover. Let's look at the diagram from the pfSense tutorial again, and gather our required parameters before we begin.

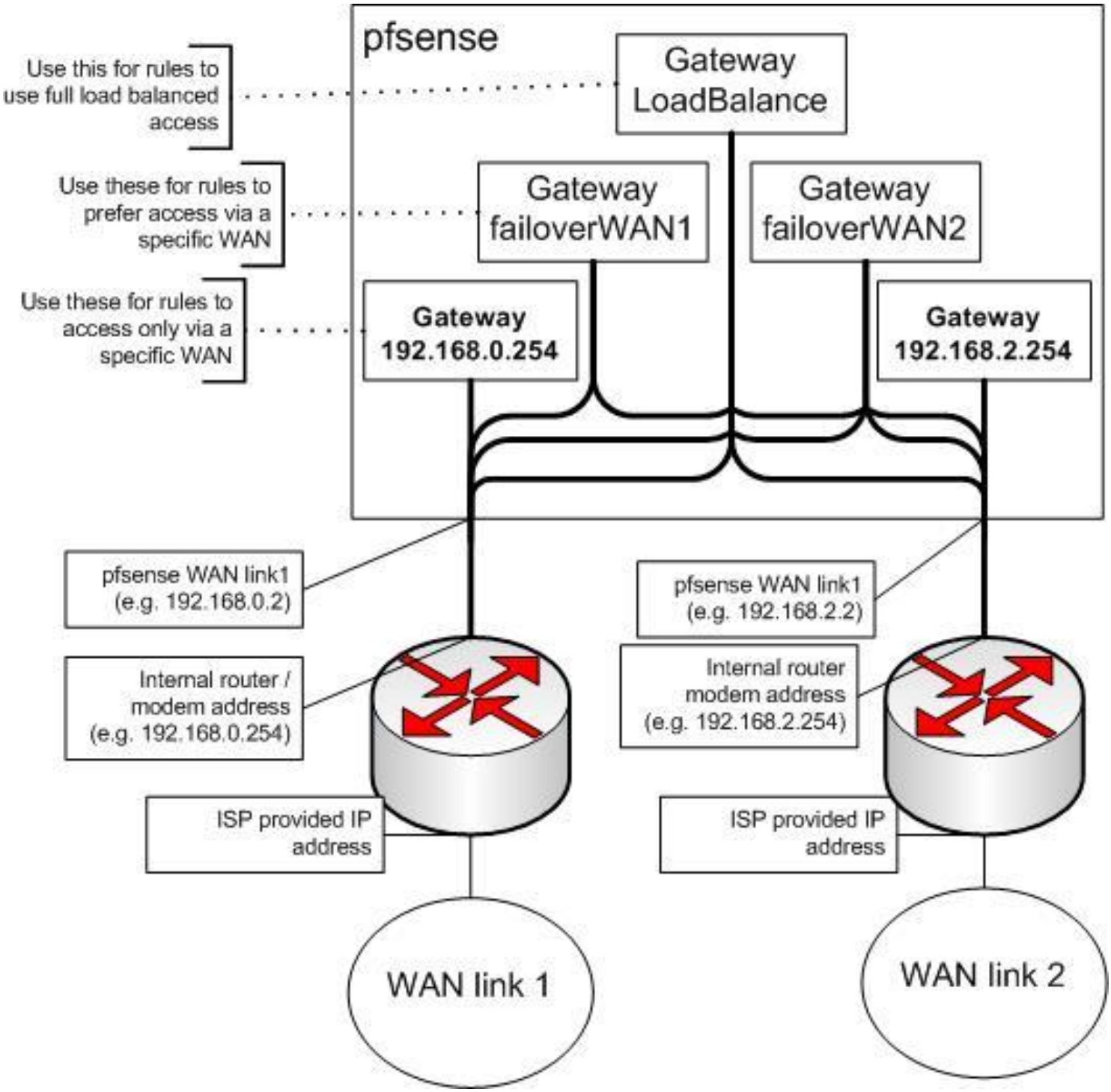


Figure 12: pfSense block diagram

We need our interface IP gateway addresses and the address for a ISP DNS server used on the corresponding interface. We will be using the DNS address as the monitor address, to verify the interface is up and running via a simple ping to that address. The values in Table 2 are actual addresses I used for Cerberus. Your values may be different.

Interface	IP address	DNS address
Gateway Primary ISP(WAN)	192.168.100.100	68.105.28.12
Gateway Secondary ISP (OPT1)	192.168.0.2	205.171.3.25

Table 2: IP address assignment

There are five steps to setting up failover and load balancing, one of which we have already accomplished.

1. **Set up Multi-WAN Configuration** – done in Part 2.
2. **Set up Required Values** – List DNS Servers, Turn on Sticky Sessions
3. **Define Failover Gateways** – One for each WAN connection
4. **Set Up Load Balancing Gateway** – Handles Round Robin Traffic Assignment
5. **Define Rules for LAN Traffic** – Direct LAN Traffic to Load Balancer

We will also need to test load balancing and failover and write a rule for outbound HTTPS traffic. This rule will serve as an example of traffic that needs to bypass the load balancer and travel directly out a single selected ISP interface.

Since we have already set up Cerberus for multi-WAN, we'll jump to step two, setting values. We need to do two things here; the first is make sure the two DNS addresses we are going to be using (*68.105.28.12, 205.171.3.25*) are listed under **General Setup**.

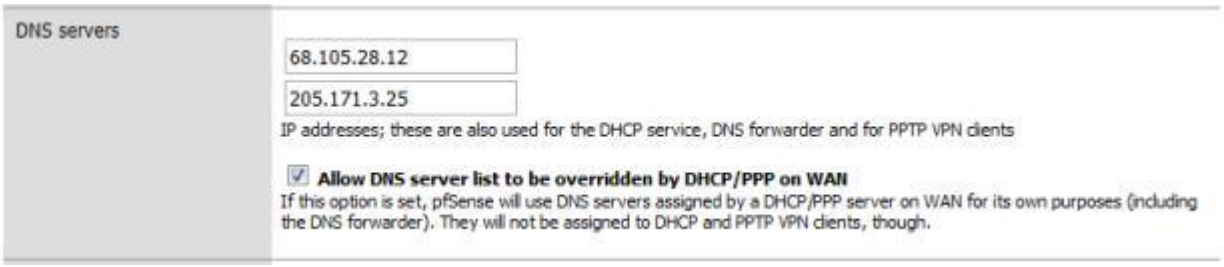


Figure 13: DNS address assignment

In **Advanced** Setup, we want to turn on **sticky connections**, so traffic started on a particular ISP WAN interface stays there, preventing sites that use your IP Address, such as your bank, from getting confused.

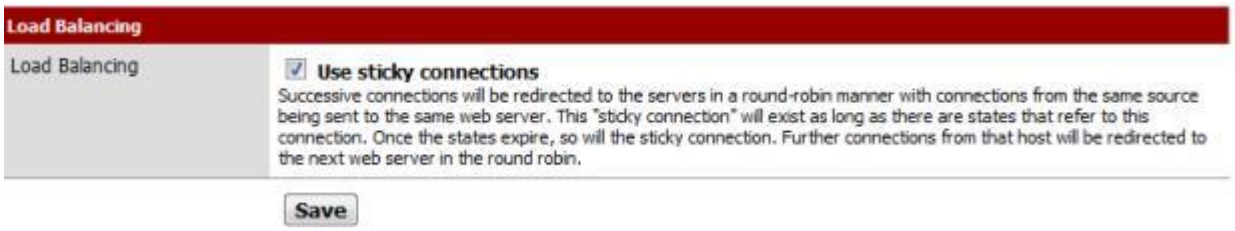


Figure 14: Enable Sticky Connections

I also recommend editing your **Snort Whitelist** (Services->Snort), ensuring DNS servers are automatically added. Depending on your ISP, DNS irregularities may cause Snort to block them, giving you a false failure.

Figure 15: Snort Whitelist auto-add DNS servers

The next step is setting up the **failover gateways** in the **Load Balancer** (Services->Load Balancer). Each failover gateway has a pool of interfaces, each with a monitoring IP. We have two pairs of Interface and Monitor IPs that need to be added to each pool. The only difference between the two gateways is the order of these pairs.

Pair One is the Primary ISP, and the WAN DNS Server: *[WAN, 68.105.28.12]*
Pair Two is the Secondary ISP, and the OPT1 DNS Server: *[OPT1, 205.171.3.25]*

The first pair in each gateway is the opposing interface, the one that it fails over to. The second is its own Interface. So the pools look like:

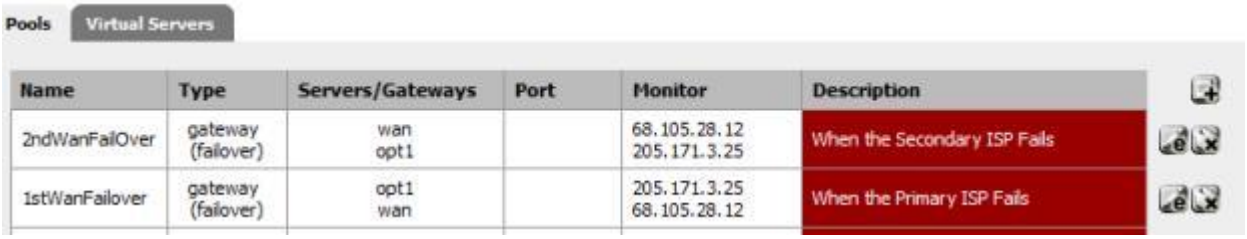


Figure 16: Failover gateway address pool

Here is the pool setup for the Primary ISP, note the the Secondary ISP Failover gateway only differs in pair order:

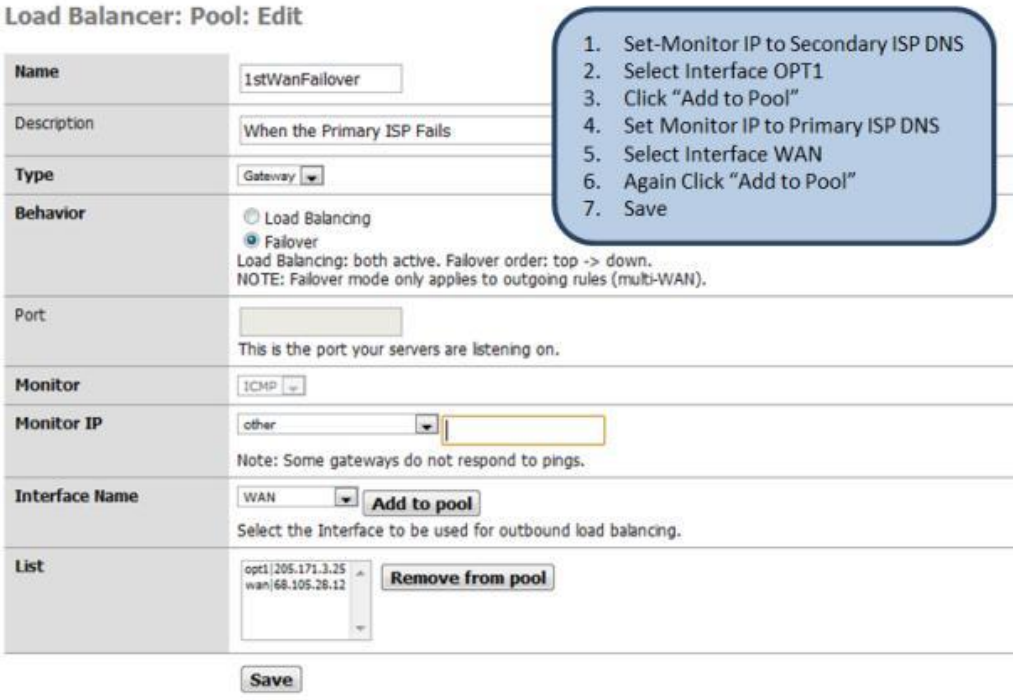


Figure 17: Primary Failover pool IP setup

With the failover gateways up, we can define the load balancer gateway – this looks just like our 2ndWanFailover gateway, except the behavior is *Load Balancing* instead of *Failover*.

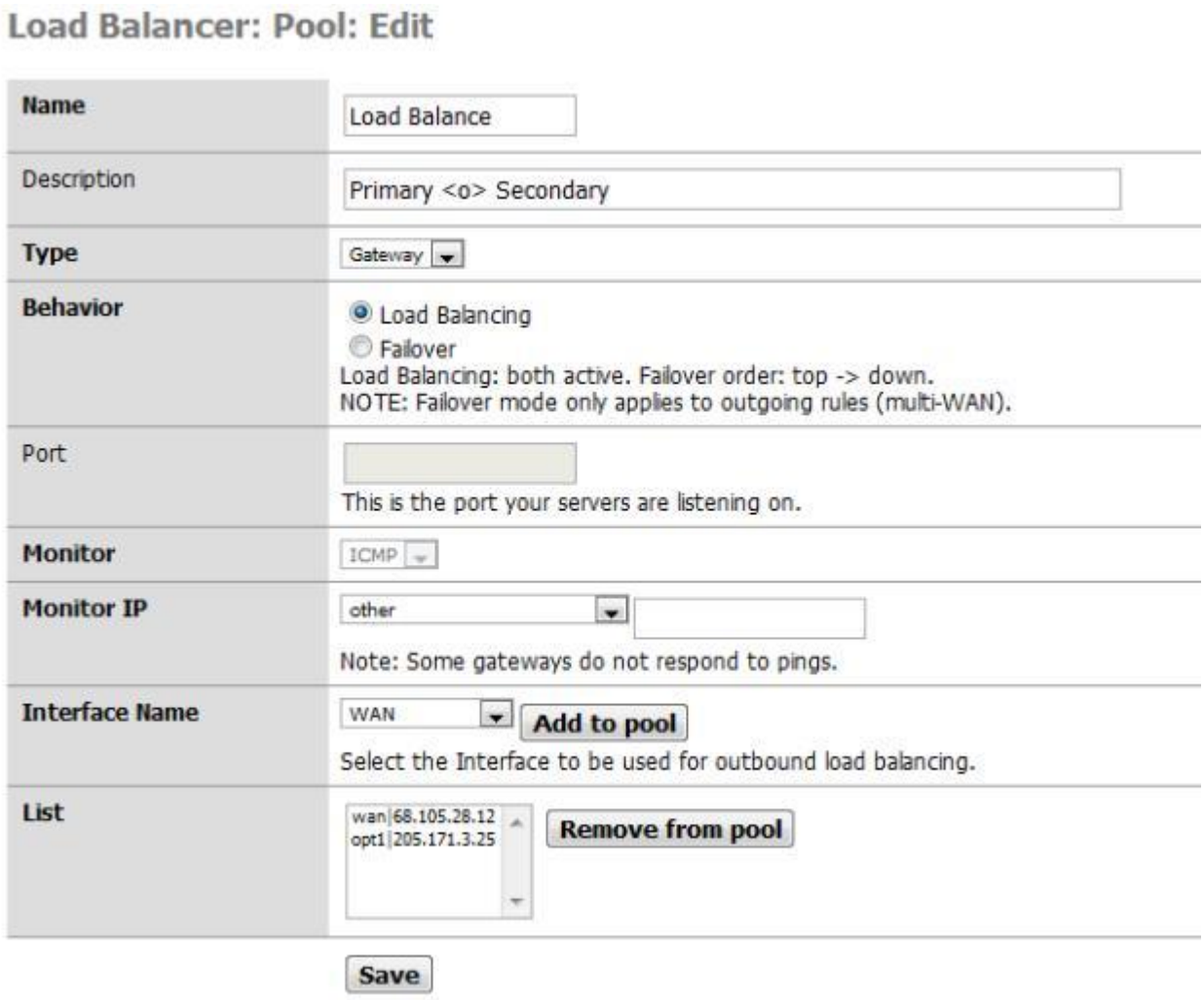


Figure 18: Load Balancer gateway setup

With the failover gateways up, we can define the load balancer gateway – this looks just like our 2ndWanFailover gateway, except the behavior is *Load Balancing* instead of *Failover*.

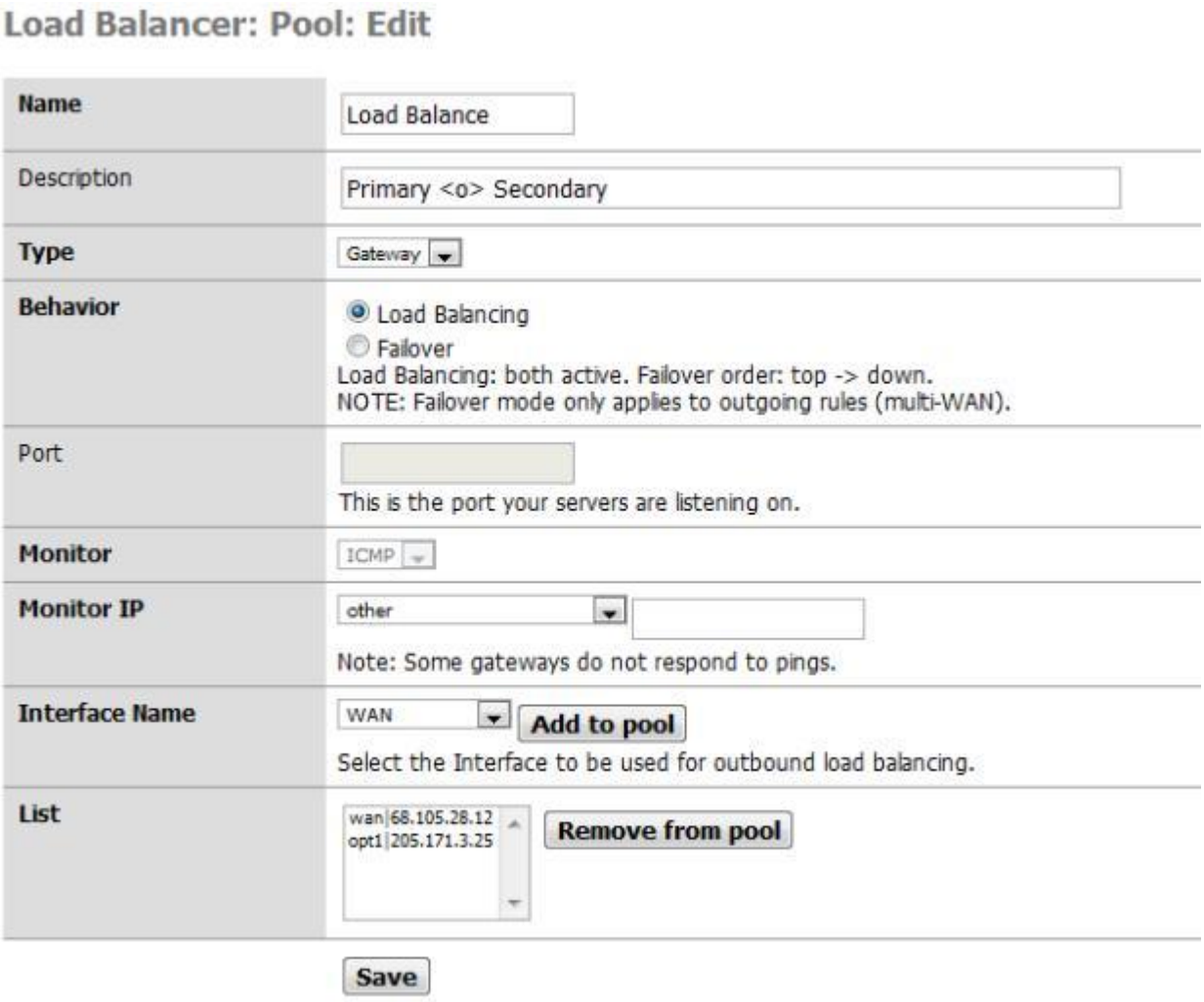


Figure 18: Load Balancer gateway setup

With that, we have completed our Gateway setup:

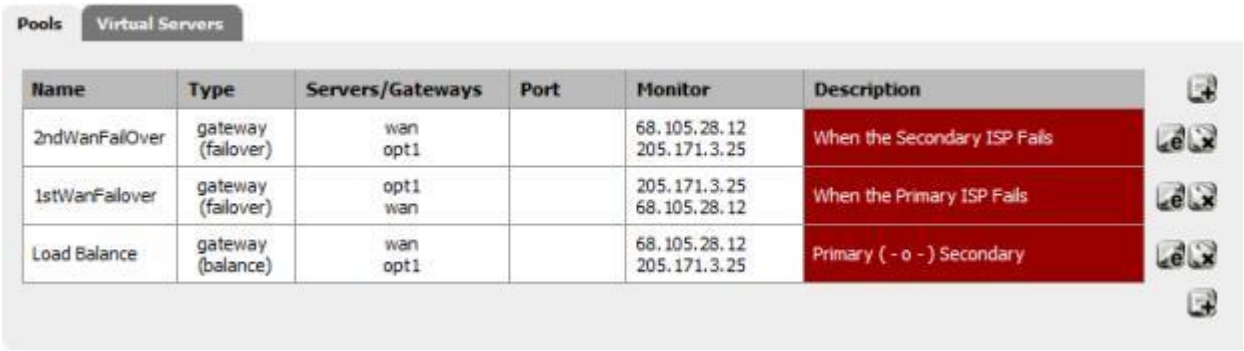


Figure 19: Gateway setup complete

Load Balancing & Failover - more

The final step is to start routing traffic through the load balancer. For that ,we need to define three firewall rules:

Rule	Explanation	Order
Primary ISP Traffic	Drive traffic to your Primary ISP	First
Secondary ISP Traffic	Traffic Destined for Second ISP	Second
Load Balancer Traffic	Direct Traffic across ISPs	Last

Table 3: Load balancer rules

To define the rules, go to the **Rules** page (Firewall->Rules). The rules handle outbound LAN traffic, so go to the **LAN** tab. Let's first add the new rules, then delete existing rules.

Action	PASS
Interface	LAN
Protocol	ANY
Source	LAN subnet
Destination	Network, 192.168.100.0/24
Log	Yes (For Testing)
Gateway	Default

Table 4: Primary ISP Traffic rules

For the secondary rules, we just change the destination:

Action	PASS
Interface	LAN
Protocol	ANY
Source	LAN subnet
Destination	Secondary ISP Subnet
Log	Yes (For Testing)
Gateway	Default

Table 5: Secondary ISP Traffic rules

For the Load Balancing Rule, we want any traffic that doesn't have a determined destination to go through the load balance gateway:

Action	PASS
Interface	LAN
Protocol	ANY
Source	LAN subnet
Destination	ANY
Log	No
Gateway	LoadBalance

Table 6: Load balancer Traffic rules

This is what your finished rules will look like:

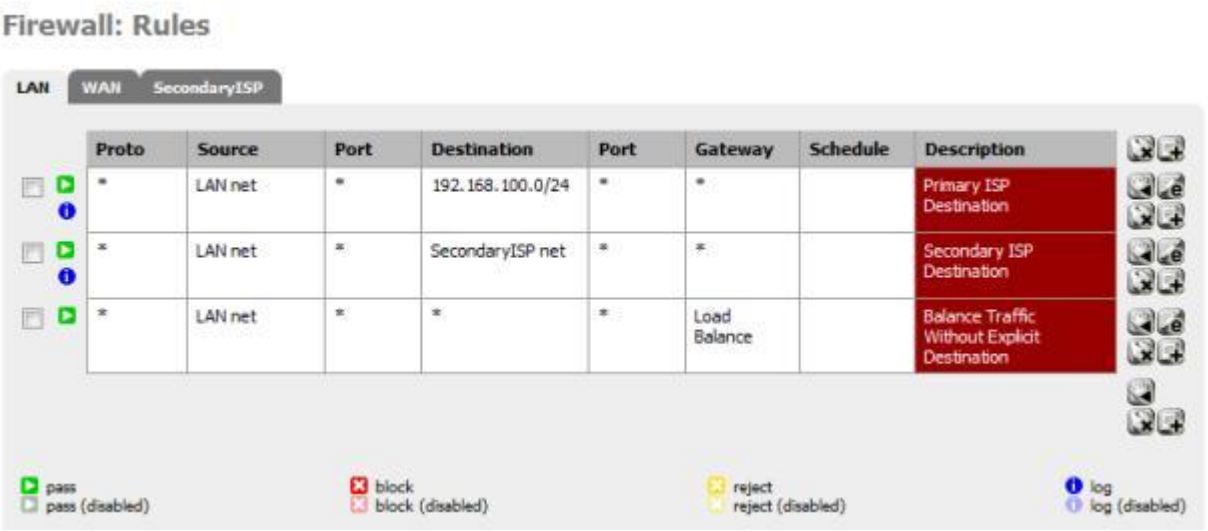


Figure 20: Firewall rules complete

To verify that everything started properly, go to the **Load Balancer** status page (Status->Load Balancer). It should be all green:

Figure 21: Load Balancer ready

Before we go any further, we should test load balancing and failover. Remember,

Squid and HAVP are not multi-WAN enabled. These packages use a single interface and bypass the load balancer to push traffic out the interface you configured it to use, in our case the WAN PrimaryISP interface. So to test failover we'll take down the SecondaryISP by simply disconnecting the cable. The system log should record the failure:

Last 500 system log entries	
Mar 8 21:22:47	check_reload_status: reloading filter
Mar 8 21:22:35	apinger: ALARM: 205.171.3.25(205.171.3.25) *** down ***
Mar 8 21:22:25	kernel: em0: link state changed to DOWN

Figure 22: Log showing failover event

If the failure is not logged, or shows the wrong interface, most likely you've confused your pairs, using the wrong DNS address.

To test load-balancing, use a protocol other than HTTP, say FTP, POP, IM, etc. that doesn't go through Squid. You should see the rule trigger on the balancer gateway in the firewall log:

Figure 23: Load Balancer rule trigger

You can also check your **States Table** (Diagnostics->States). It should list some states associated with your Secondary ISP if load balancing is working.

Your network should now be ready for the next unplanned outage by your ISP.
Sticky Connections solves most requirements for persistent sessions, but you may want to do your own pre-emptive load balancing, especially if you will be running a proxy server such as HAVP and Squid. The template for these rules is:

Action	PASS
Interface	LAN
Protocol	TCP
Source	LAN subnet
Destination	ANY
Destination Port	HTTPS
Log	No
Gateway	2ndWANFail

Table 7: HTTPS Rule for Balancer Bypass

The *HTTPS* Destination Port in Table 7 can be FTP, SMTP, etc. This rule needs to be at the top of the list of rules—the load balancer rule should always be last. Using a failure gateway, traffic will, of course, fail over. If that isn't what you want, change the gateway address to use the direct Gateway instead. In our example, that is *Opt1/192.168.0.2* or *WAN/192.168.100.100*.

P2P traffic is much the same. You will have to use a static port and the destination port will need to agree with the configuration of your BitTorrent client (uTorrent uses 2000-3000). For incoming connections, you'll need to define a port forwarding rule on your NAT, instead of using UPnP. More details are available in this pfSense tutorial.

That's all for this time. We'll try to wrap this up next time and run some performance tests to see if our hardware platform can handle all the extra duties we have piled onto it.

Monitoring Logging

Above of this post, we established a working definition of our target, i.e. what has to be done, and in what order, to Cerberus the lowly IDS firewall to make it a UTM Appliance. In above, we started the conversion by installing and configuring multi-WAN support,

Squid, IDS and anti-virus features. above , we added and configured Content Filtering, Traffic Control, Load Balancing and Failover.

In this last part, we'll wrap things up with Monitoring and Logging configuration, performance testing, final grading and reflection on the whole process.

Monitoring and Logging

There are numerous packages for logging and interfaces to external monitoring packages, summarized in Table 1.

Capability	Explanation	Features
Built-in Logging	Protocols for logging system events	SNMP, Syslogd, WebGui
RRD Graphs	System Resources Graphic Monitoring Tool	CPU Load, Traffic Throughput, Quality Handling, and Shaping Queues
Snort	Alert Tracking and Status	Barnyard2 package interface, Dashboard Widget
Squid	Web and Cache statistic	LightSquid
System Status	Hardware and Package Status	Dashboard, PHPsysinfo, WebGui, BandwidthD
External Interfaces	Monitoring and Management Agents	Zabbix, Radius, ntop

Table 1: Logging and monitoring packages

Several of these are built in, RRD Graphs are available is available from the Status menu, SyslogD can be configured there too, under *Status->System Logs->Settings*. SNMP is a built-in, find it under *Services->SNMP*.

Installing the others is straightforward, and can be found in the packages menu, these include **LightSquid**, **BandwidthD**, **PHPsysinfo**, and the **Dashboard**, including several dashboard widgets (Snort, Havp status). The interface to **Barnyard2** is included with Snort.

The only issue with a couple of these packages, LightSquid, ntop and BandwidthD, is that they are not fully

integrated into the pfSense webGui - the pfSense banner and menus disappear, but backing out of the reports will lead you back to the web GUI.

Here are some screenshots of some of the logging and reporting options:

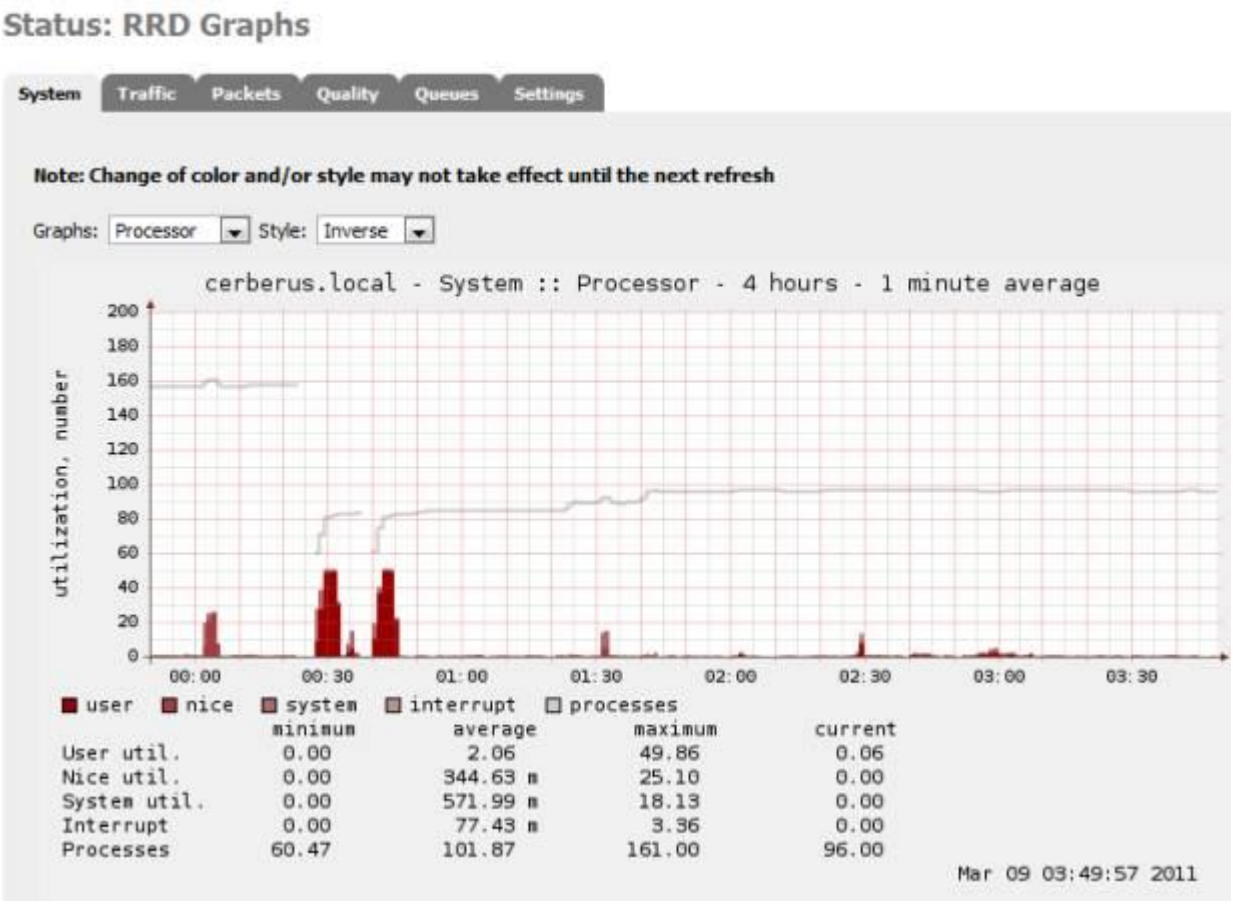


Figure 1: RRD Graphs

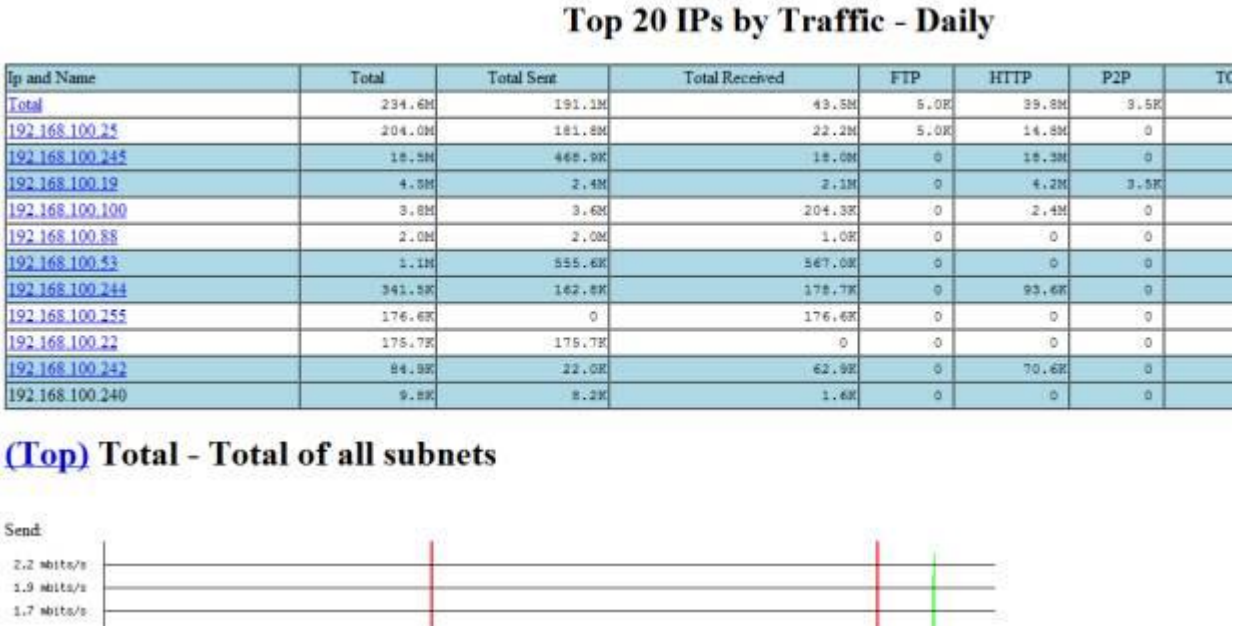


Figure 2: BandwidthD add-on Package

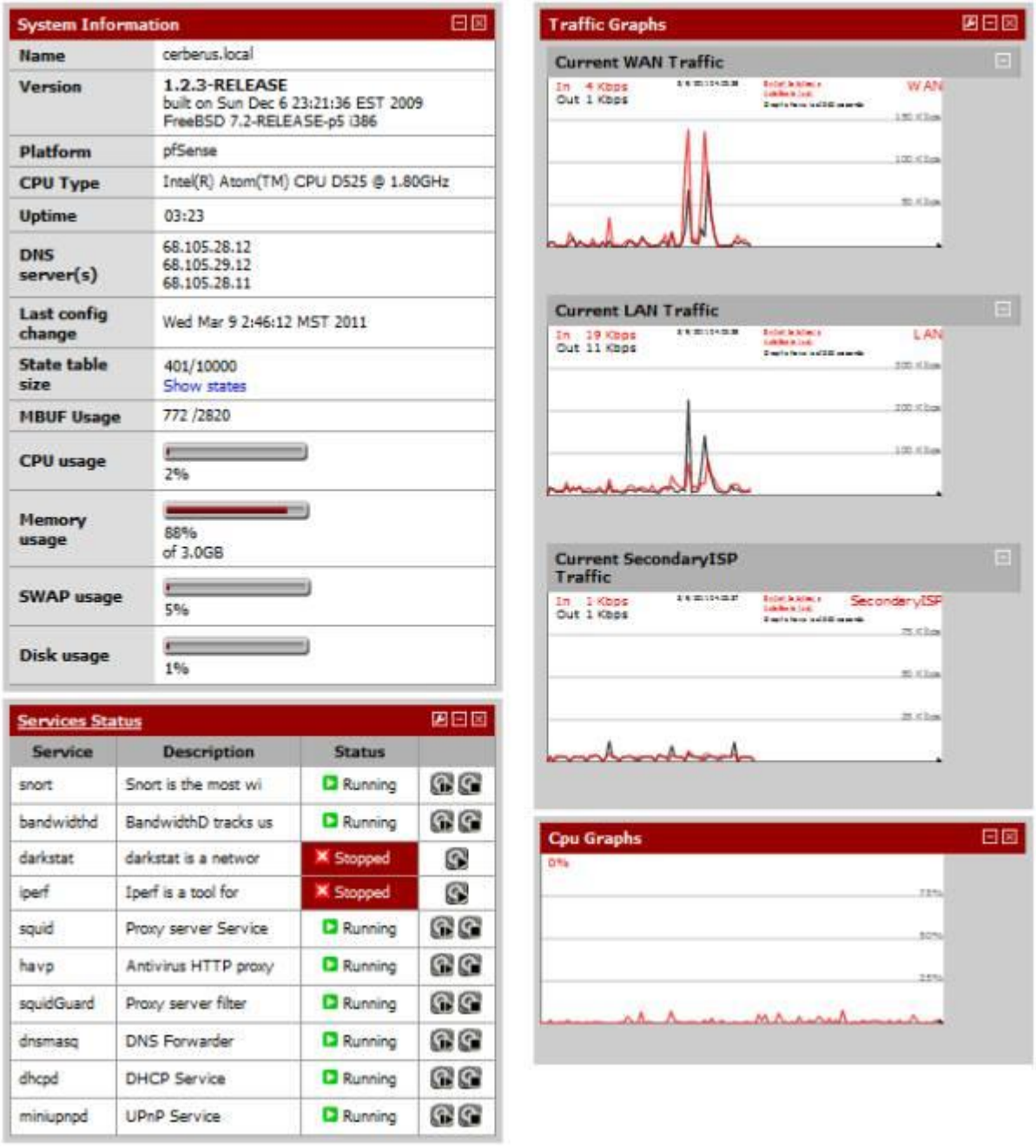


Figure 3: Dashboard

Squid user access report
Top Sites
Work Period: **Whole YEAR - 2011**

	Accessed site	Connect	Bytes	%
1	who graphics8.nytimes.com	56 193	73.6 M	1.5%
2	who talkgadget.google.com	47 822	32.5 M	0.6%
3	who twitter.com	34 919	69.1 M	1.4%
4	who mail.google.com	19 803	18.4 M	0.3%
5	who www.thetvdb.com	16 502	49.5 M	1.0%
6	who 192.168.0.1	14 107	100.1 M	2.0%
7	who js.nyt.com	10 034	6.1 M	0.1%
8	who api.twitter.com	9 591	15.4 M	0.3%
9	who stork138.dropbox.com	9 093	2.4 M	0.0%
10	who www.google.com	8 277	51.4 M	1.0%
11	who api.echoenabled.com	7 490	2.6 M	0.0%
12	who bullmarketfrogs.com	6 775	19.2 M	0.3%
13	who css.nyt.com	6 642	2.2 M	0.0%
14	who www.facebook.com	6 461	9.7 M	0.2%
15	who i1.nyt.com	6 059	10.7 M	0.2%
16	who toolbarqueries.clients.google.com	5 899	2.5 M	0.0%

Figure 4: Light Squid

Performance

First, a bit of review. Cerberus was introduced in Build Your Own IDS Firewall With pfSense as an inexpensive build (around $350) for an IDS Firewall. The build list is in Table 2.

CPU	Intel Atom D525 (Pineview-D) Dual Core, 1.8GHz (13W) processor	Incl in mobo
Motherboard	Supermicro X7SPA-H-D525 Mini-ITX Server	$180
RAM	2 x non-ECC DDR3 1066MHz SO-DIMM (running @800MHz)	$50
Storage	WD Scorpio Blue 2.5” 250Gig drive	$40
Ethernet	Intel 10/100/1000 PCIe NIC	$30*
Case	Antec Mini-Skeleton-90	$90
DVD	Sony DVD-ROM	*

Table 2: Cerberus component list

That previous article explained the whole decision process, the components and why. On top of that hardware we installed pfSense, Snort, and IP Blocklist – all to provide an extraordinary level of protection for a home network. As an IDS Firewall, Cerberus made a good showing, not a speed demon, but in the top third of SNB’s router performance charts. Running iPerf as the server on Cerberus, directly over gigabit LAN to jPerf, Figure 5 shows an average throughput of **236 Mbps**, with a peak of **253 Mbps** with a fair amount of CPU headroom left over.

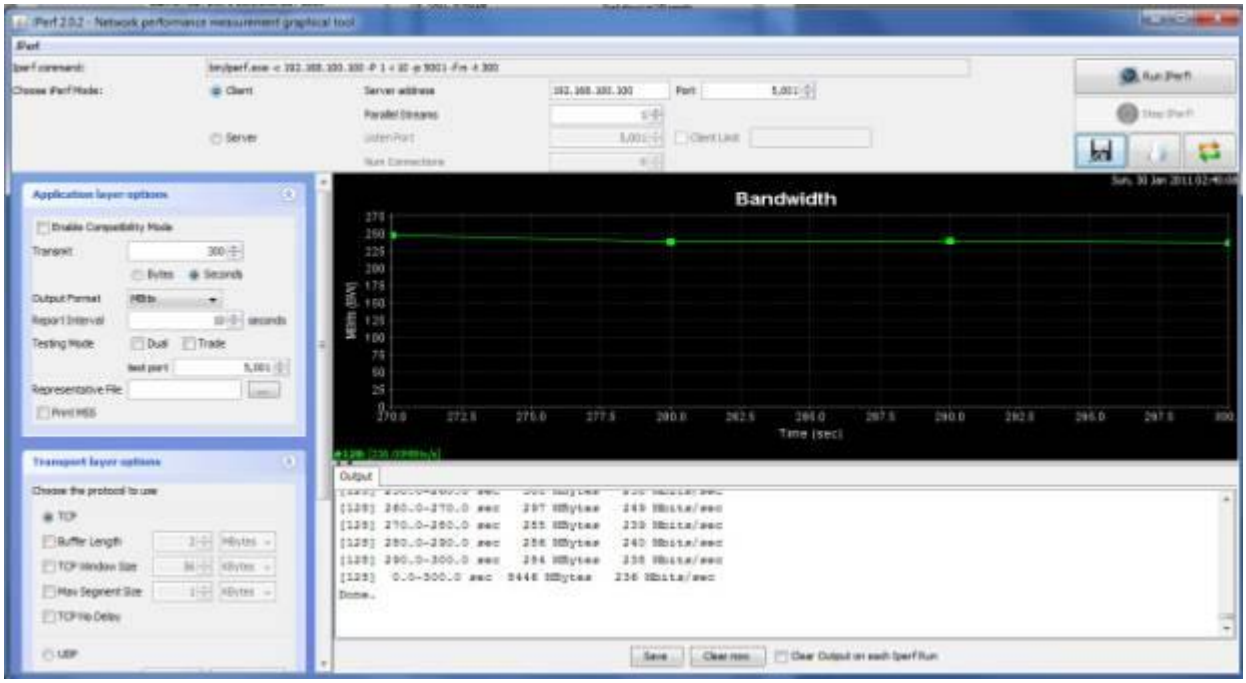


Figure 5: Running iperf on Cerberus as IDS

In our goal to convert Cerberus to a UTM, we poured on a whole lot of additional functionality. We added Squid and Squid Guard for caching and content filtering, we expanded Snort to cover three interfaces instead of just the single WAN interface, added HAVP and its scanning engine ClamAV for anti-virus, and instituted QOS and set-up multiple WAN load balancing and fail-over.

And finally we added some minor packages, SpamD for anti-spam, and DNS Blacklist and Country Block for targeted content filtering, BandwidthD, Lightsquid and Darkstat for reporting. In all, a complete package, our UTM.

So how did Cerberus the UTM fare performance-wise? Let's look at Figure 6, running the same iperf test, under the same conditions that we used for our IDS Firewall.

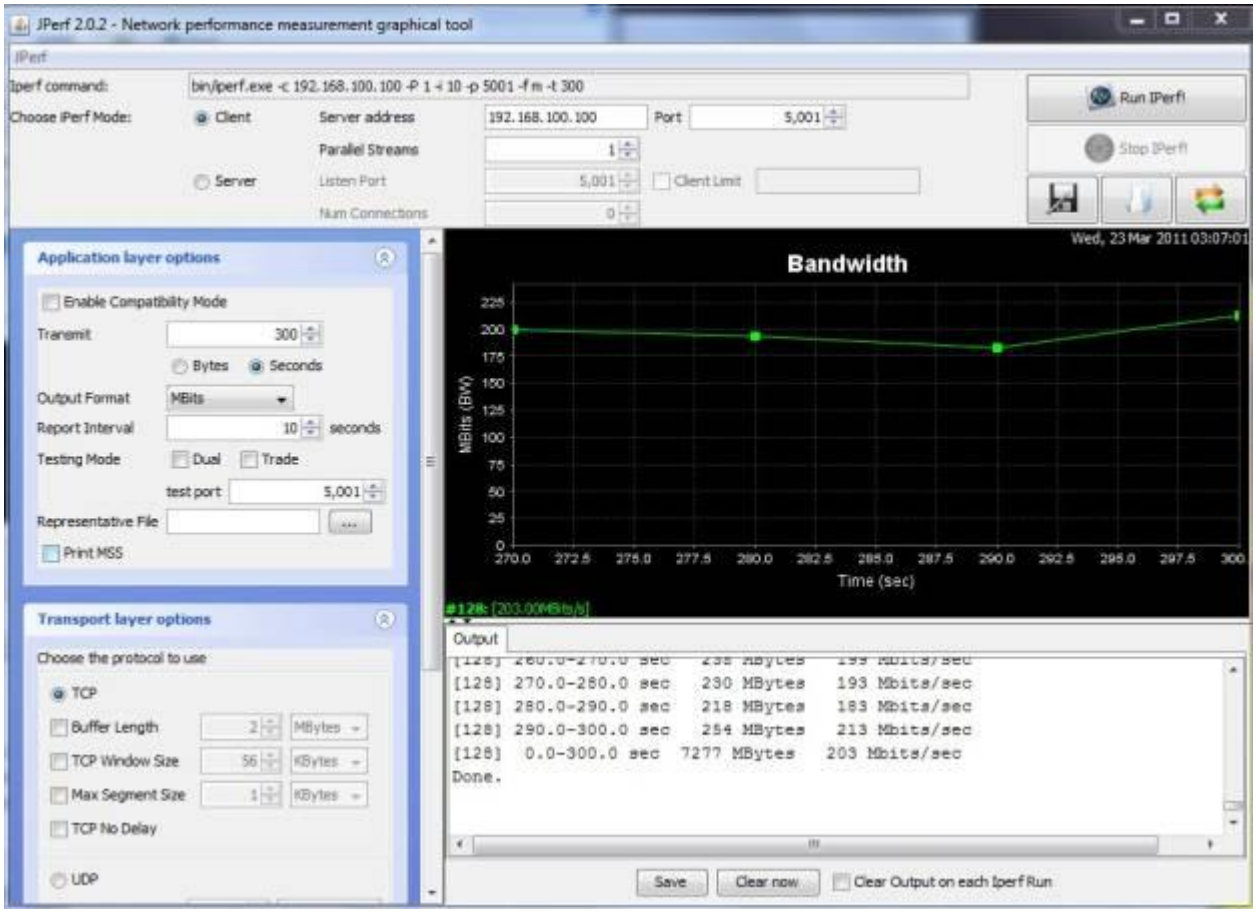


Figure 6: Running iperf on Cerberus as UTM

This time, I measured an average throughput of **203 Mbps**, with a peak of **231 Mbps**; CPU hit a utilization of just over 80% with using about 93% of available memory. Not too shabby, only a **14% drop** in performance, but without CPU headroom. This shows how much we overestimated the processing requirements of pfSense; a dual core Atom 510 would probably been sufficient vs. the D525.

Conclusion

Without a doubt, Cerberus has been transformed. Take a look at the packages and features we have enabled in the summary Table 3.

Package/Feature	Pros	Cons
Snort IPS/IDS	Comprehensive, Quick Rules engine supporting dynamic rules	High Memory Demands, Requires both thoughtful configuration and administration
Squid Proxy Server	Fast capable proxy server, allows for traffic throttling	Not just point and shoot, doesn't work with QOS
HAVP/ClamAV Anti-Virus	Non-Blocking, Easy to set-up	Not comprehensive, non-commercial AV scanning
pfSense QOS	Wizard-based setup, queue based administration	Limited Level-7 Support
pfSense Multi-Wan Load Balancing and Failover	Provides for resilient failover	Not integrated with QOS or packages, uses simple load balancing algorithm,

		complex non-intuitive set-up
Squid Guard Content Filtering	Full featured content filtering down to who and when, ability to use external well maintained lists	Difficult install, no stock blacklist, poor documentation
IP Blocklist	Dynamic list based blocking	Slow, manually updated list administration has bugs, lists can be a mixed bag
DNS Blacklist	Quick and simple category-based host blocking	Static list requires manual updating
Country Block	Easy and quick blocking of country CIDRs	Geared more towards anti-spam
SpamD Anti-Spam	Simple, clever spam protection	Not integrated into pfSense, set-up requires hacking
Reporting: RRD, BandwidthD, LightSquid	Comprehensive and easy to set up, dynamically updated	Not fully integrated into webGui

Table 3: Cerberus UTM packages

So can Cerberus take home the UTM Crown? Have we hit our target? Let's take a look at the big picture. The first step is reviewing the summary of grades from

Function	Grade
Intrusion Prevention & Detection	A
Anti-Virus	C-
Content Filtering	B
Anti-Spam	D
Traffic Control	B
Enterprise Capabilities	C
Overall Grade	**C+**

Table 4: Cerberus UTM grading

I do feel this is an accurate grade, based on functional capabilities. But the overall grade does not reflect what you personally might need from a UTM - in that case the grade drops to that of your most urgent requirement. If you are being pummeled with spam, or run an environment with a lot of unknown users, where anti-virus is significant, the grade you give pfSense drops dramatically. If home network protection is most important, the grade gets much better.

We could stop now, and say Cerberus is a UTM, sort-of. But that would be disingenuous, because of what we learned in the upgrade process. There are three other important aspects of our system in grading whether we hit our goal. These are: our installation experience; how well the system performs; and finally, the degree of integration, i.e. how well do the pieces work together.

The installation experience varied greatly, spanning the spectrum from seamlessly simple, with the installation of HAVP, our anti-virus solution, to the convolutions of origami we saw with installing SquidGuard, the cornerstone of content filtering. None of the more significant packages was what would be called turnkey.

It is understood that difference between an amateur and a professional is consistency - a professional chef makes the same dish over and over and it tastes the same, we cook at home, the meal can vary dramatically. PfSense's install processes are not consistent.
pfSense **Installation Process** Grade: **C-**

Performance is the bright spot, even with several layers on top of our TCP/IP stack, a multitude of processes poking and prodding packet after packet, Snort, QOS, load balancing, and a couple proxy servers, Cerberus still rendered excellent performance.

pfSense **Performance** Grade: **B**

Now the big one, the degree of integration: the pieces just don't meld together to form one appliance. Squid doesn't work with QoS, HTTP traffic will remain unmetered. The reporting tools, LightSquid and BandwidthD, are only partially integrated into the webGUI. And most significantly, virtually none of the packages are compatible with the critical enterprise aspect of running multiple WAN connections, not the built-in QoS, not any of the various proxy servers.
pfSense **Integration** Grade: **F**

If a UTM is defined by the six functional groups we identified in Part 1 of this article, then yes, pfSense and Cerberus is a UTM, all the boxes are checked. But if a UTM is an appliance where all the pieces work together, are really unified, then no, we can't say that Cerberus is a UTM. The whole must be bigger than the sum of the parts, or a checklist of functionality.

What we learned in this upgrade is that pfSense is a patchwork of packages, some excellent, others not so much. But overall, the pieces don't gel. The updated scorecard in Table 5 calculates out to a C. But it feels more like a **Fail**, or if you are charitable, an **Incomplete**.

Function	Grade
Intrusion Prevention & Detection	A
Anti-Virus	C-
Content Filtering	B
Anti-Spam	D
Traffic Control	B
Enterprise Capabilities	C
Installation Process	C-
Performance	B
Integration	F
Total Grade	**C**

Table 5: Cerberus final UTM grading

This judgment, our final grade, only applies to our well-formed definition of what a UTM is, and does not imply that pfSense is not suitable for solving your problem, especially if you don't need Multi-Wan. If all you want to do is protect your home network, Cerberus is an all-star.

However, there is hope on the horizon. While writing this article, pfSense moved the long awaited **Version 2.0** out of beta. 2.0 is reported to sport fully integrated multi-wan support, and expanded support for packages like SpamD. So we may get to do this all over again!