URLfilterDB

Reference Manual

ufdbGuard version 1.33

the internet filter for the Squid web proxy

Table of Contents

1 Introduction

ufdbGuard is a URL filter software suite that can be used with the Squid web proxy to block unwanted web sites on the internet. ufdbGuard uses a URL database to determine which web sites are restricted and works with any text-based database and also the commercial database fro URLfilterDB. ufdbGuard also features the capability to enforce Google's SafeSearch, HTTPS tunnel detection and verification of TLS/SSL certificates used by HTTPS connections.

You may register as a trial user at www.urlfilterdb.com to receive a 60-day trial license for the URL database. The trial license is for an operational URL filter without restrictions.

Internet access restriction tools are usually implemented to increase the productivity and shield users from unwanted web content. ufdbGuard is also a network protection tool blocking access to proxies and tunnels.

To be able to use an internet access restriction tool, it is necessary to have a well-defined internet usage policy. It is common to actively share this policy with all employees and define clearly what is expected from employees.

1.1 What is URL Filtering?

URL filtering is a mechanism to block access to certain types of websites. Usually the reason to use URL filtering is to achieve higher productivity and to lower bandwidth usage. By only blocking advertisements and file sharing sites (p2p) the internet bandwidth usage can drop considerably. A URL filter can also block adult sites and prevent access to websites where an individual can copy an internal document to this public website with one or two mouse clicks. Finally, a URL filter can protect the IT infrastructure by blocking access to websites that promote themselves as "safe" but are not.

Users browse the internet and often without knowing it, they usually use a web proxy like Squid that is in between a PC and the internet. Squid and ufdbGuard can verify if a user has access to a particular (part of) a website that is visited. Approved websites can be visited without restriction. The browser displays a message that access is prohibited in case that a (part of) a website is restricted.

A URL filter can bock parts of a website. For example if news is allowed but entertainment is blocked, users will have access to www.cnn.com but not to www.cnn.com/showbiz.

1.2 Why is Tunnel Detection a Must-Have ?

Users who want to circumvent company internet and security policies, may use so-called *tunnels* to break through the firewall that protects your internal network from attacks from the outside. Also viruses which intention is to steal data also may try tunnels. Tunnels can be extremely harmful since they circumvent firewall rules, the anti-virus protection is circumvented and tunnels can be *bidirectional*, i.e. the firewall is open to attacks from the outside through the tunnel. ufdbGuard detects and blocks these tunnels.

SSH tunnels are one of the most dangerous types of tunnels where an entire corporate LAN can be exposed to anybody on the internet. Tunnels turns firewalls into open doors!

Very little knowledge is required to make a tunnel: everyone who knows how to type "how to punch holes in firewalls" at a search engine can do it. The reader is urged to get familiar with the risks of (SSH) tunnels to be able to implement a proper security fence. Security officers and senior management are advised to verify that a security solution is implemented to block tunnels.

1.3 About This Document

This document contains all information about installation and configuration of ufdbGuard. There are 2 options for installation: compilation from sources or by using RPM packages.

UfdbGuard can be installed from sources or by installing an RPM package. The installation of an RPM package is the recommended installation method which was introduced in version 1.31. The RPM obeys the guidelines from Redhat about installation directories which is different from the default directories used by the installation from sources. E.g. the binary `ufdbguardd` is installed in `/usr/sbin` by the RPM package installation and, unless configured otherwise, installed in `/usr/local/ufdbguard/bin` when compiled from sources.

This manual has references to directories and assumes that ufdbGuard is installed with the RPM package install. If ufdbGuard is installed from the sources, directories may be different than printed in this manual.

1.4 Latest Major Enhancements

Version 1.30 introduces easier integration with LDAP-based and other databases with usernames.

Version 1.31 introduces basic reports to be used by the network administrator. ufdbGuard now also has support for user-defined URL categories that change often and is also distributed as a binary package for Redhat and CentOS. The interface protocol of Squid 3.4 was changed and version 1.31 patch 13 is also compatible with Squid 3.4 (see also the `squid-version` keyword).

Version 1.32 introduces a new engine which has better performance with URLs that use HTTPS because of reduced server probing, has more information in the log file and has improved reporting. Version 1.32 supports the new peek+splice and peek+bump ssl-bump modes of Squid 3.5.x. Version 1.32 also has an improved performance for large lists of usernames and has a new feature for fast matching of URLs with a (set of) parameters and values without using regular expressions; e.g. one can make a category with:

```
popularvideos.com/watch?v=aaaabbbbcccc
popularvideos.com/watch?v=xxxx1111yyyy
```

ufdbGuard 1.32 supports a new database format (2.2) which supports the ZLIB compression algorithm. The URL database from URLfilterDB uses ZLIB compression since it is approximately 5 times faster decompressing URL tables although the space saving is a few percent less than with the BZIP compression algorithm. BZIP was used by ufdbGuard before version 1.32.

ufdbGuard 1.33 supports the new keywords `ipv6` and `ipv6list` which can be used in source definitions. ufdbgclient has a new `-m` option to enable multithreading which performs much better than the previous queueing mechanism that can be enabled with the `-C` option. ufdbGuard now fully supports URLs with %-encoded and native UTF8 characters.

ufdbGuard 1.33.2 has a fix for a bug that appears on overloaded or slow systems as well as inside virtual machines where ufdbguardd exits with the error “HUP signal received but could not acquire a lock on the configuration”.

A detailed list of all changes can be found at the top level of the source tree in the file CHANGELOG.

1.5 Copyright

The ufdbGuard software suite version 1.x is Open Source Software and free to use. To protect the ownership and the freedom of use of ufdbGuard, there is a copyright and you have a license to use and

modify the software freely, known as the GPL version 2 license. The full text of the license is here: www.gnu.org/licenses/old-licenses/gpl-2.0.html.

The software suite is divided in 3 packages known as *ufdbguardd,* the *ufdbguardd API* and *the ufdbguardd REST API*. Ufdbguardd is inspired by squidGuard and parts of it are written by third parties and have a GPL2 copyright by those authors. The two ufdbguardd APIs are copyright by URLfilterDB and are meant to be used by vendors who wish to integrate URL filtering in their products and does not contain any code of third parties.

The URL database is a commercial product and has a copyright by URLfilterDB. A license is required to use the URL database which is defined in The Terms of Contract document that can be downloaded at the website: www.urlfilterdb.com.

1.6 Support and Feedback

We welcome feedback from clients and those who test our software. Feel free to send your questions and feedback to the support desk: support@urlfilterdb.com.

2 Prerequisites

ufdbGuard runs on all flavors of UNIX and is usually installed on the same system where Squid is installed. The Squid proxy software can be downloaded at www.squid-cache.org.

2.1 System Resources

ufdbGuard needs 2 GB disk space and 450 MB memory for the 32-bit version and 700 MB memory for the 64-bit version. The required CPU power is compared to Squid relatively low, so it can run on one CPU for small to medium sized user groups. For large user groups, it is highly recommended to use an appropriate multicore system.

ufdbGuard uses probes to HTTPS sites to verify TLS/SSL certificates and to detect SSH tunnels and proxies. For these features ufdbGuard needs to be able to open direct communication channels with web servers. If any of these security enhancing features of ufdbGuard are required, firewall configurations may need to be modified to allow access to port 80 and 443 of external servers.

The URL database updates and upload of uncategorised URLs must have access to port 443 of `updates.urlfilterdb.com`.

2.2 Prerequisites for Installation from RPM Package

RPM packages are available for systems running Redhat or CentOS. Usually a base system already has all prerequisites installed. In case that a required package is not already installed, RPM installs all required packages together with the ufdbGuard package.

2.3 Prerequisites for Installation from Source

For all systems, ufdbGuard must be compiled with a C compiler. Most UNIX distributions come with the free GNU C compiler, `gcc` (see also gcc.gnu.org) or a native C compiler. In addition, the `wget`, `make` and `install` commands are required which are all included in most UNIX distributions.

ufdbGuard uses a compressed database and therefore requires the compression libraries `zlib` and `bz2`. These libraries are installed on most UNIX systems. In case that it is not on your system, it can be installed from your UNIX distribution CDs or downloaded from www.zlib.net and www.bzip.org. It also uses OpenSSL and needs the openssl header files which are included in your UNIX distribution (usually the package name is openssl-devel).

Starting with version 1.24 it is recommended to install the `gdb` debugging package to receive detailed information in the event that ufdbguardd terminates unexpectedly.

3 Architecture

Squid is a popular web proxy that is used as an internet cache and is usually part of the internet security solution where users are not given direct access to the internet. Squid is free and can be downloaded from www.squid-cache.org.

Squid uses child processes called *redirectors* or *URL rewriters* to verify URLs for blocking. The redirectors are lightweight processes (`ufdbgclient`) that communicate with a multithreaded daemon process (`ufdbguardd`). The multithreaded daemon has a single copy of the URL database in memory and does the actual URL verification. The daemon sends a reply indicating OK or BLOCKED to the redirector which sends the answer to Squid.

The `ufdbguardd` daemon uses a UNIX socket to communicate with `ufdbgclient`. Alternatively, a TCP socket can be used with port 3977 but ufdbGuard must be recompiled for this feature. `ufdbguardd` and `ufdbgclient` can be configured to use an alternative TCP port number.

To analyse SSL certificates of HTTPS sites. to detect proxies and SSL tunnels, and to detect various chat protocols, ufdbGuard connects directly to web servers on the internet and probes the web servers independently. Therefore ufdbGuard needs to be installed on a server that has direct internet access. Usually such server resides within a DMZ. A DMZ is not a requirement by ufdbGuard but a DMZ is highly recommended for all corporate environments for reasons of security.

In the next simplified diagram, all pieces are put together: the web browser on a PC connects to the Squid web proxy who uses ufdbGuard to block access to websites.

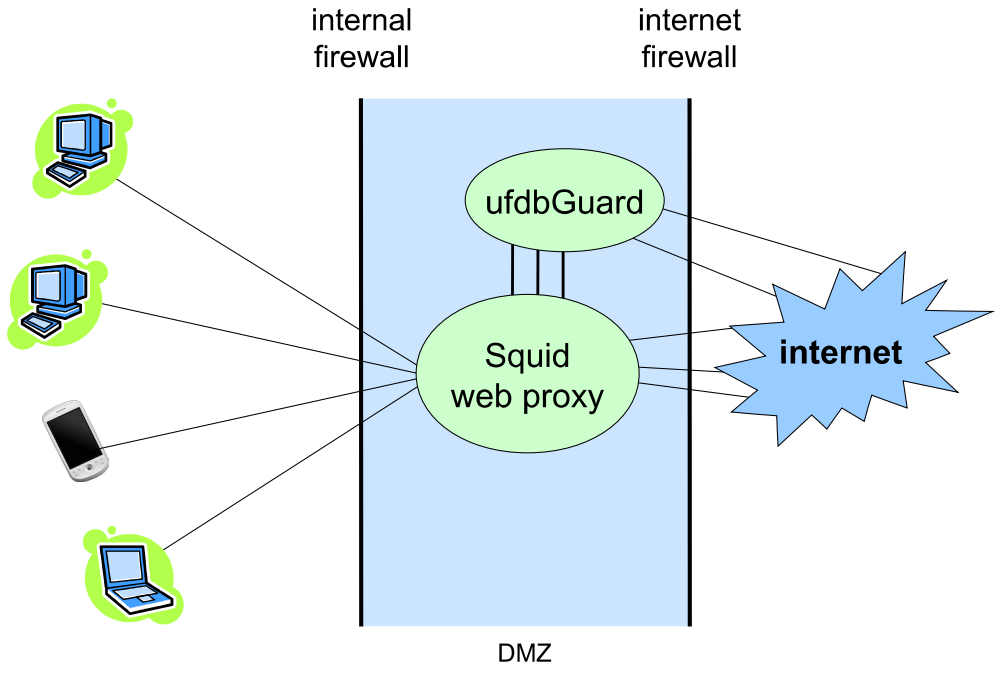


Figure 1: simplified architectural overview

Each request that Squid receives to retrieve a page from a website is first verified with ufdbGuard before Squid tries to fetch the content of the URL.

In case that the daemon is not running or when the daemon is loading a new version of the URL database, by default all URL verifications get an immediate OK and no web site is blocked. Configuration parameters exist to allow or block all access in case of fatal errors or a database reload.

ufdbGuard can be used with any URL database that comes in either a flat file format or the proprietary `.ufdb` format. The ufdbGuard software suite comes with a utility to generate an `.ufdb` database file from a flat file.

3.1 How URLs are blocked

3.1.1 Blocking HTTP URLs

The HTTP protocol supports URL redirection where a web server or a web proxy may respond with a `303 See Other` status code instructing the browser or application to retrieve the desired web object from an alternative location. With the help of the 303 status code, URLs can effectively blocked and the 'alternative location' is the URL of the web page that displays a message like “this site is blocked because ...”.

Squid must be configured to use ufdbGuard as the URL redirector. For each URL that Squid receives, it sends the URL to ufdbGuard which replies with an OK message to indicate that the URL is allowed or a 303 message to indicate that the URL is blocked.

3.1.2 Blocking HTTPS URLs

HTTPS is HTTP inside a layer of the secure SSL/TLS protocol and is designed to withstand any form of tampering. So there is not a straightforward way to block HTTPS by redirection of a URL request to an other URL. It must be stressed that HTTPS can be blocked in various ways and each blocking method has its own caveats.

To understand the various issues with HTTPS blocking, one needs to understand basic communication between the browser, proxy and server. A browser that wants to communicate to a server sends a CONNECT request to the proxy if it is configured to use a proxy. If nothing is blocked, the proxy then copies all bytes from the browser to the server and vice versa without any form of interference. Note that the proxy does not monitor the channel between the browser and server and that this property makes it a great instrument for applications to use non-HTTPS protocols between the application and the server. For example, Skype uses its own protocol using port 443. The web proxy and URL redirector need to be configured how to handle these non-HTTPS protocols.

If a browser or any application is not configured to use a proxy, instead of sending a CONNECT request, it simply starts the SSL/TLS handshake with the server as if there is no proxy and no firewall. In this scenario, Squid, the firewall and the routers can be configured to intercept such traffic and proxy it anyway. In interception mode, the SSL/TLS handshake is processed as a pseudo-CONNECT. But in interception mode, Squid only sees an IP address instead of a FQDN in the CONNECT request which cripples the URL filter since it needs at least the FQDN to be able to make correct decisions about blocking a connection or not. Note that receiving the FQDN in many cases is sufficient to make correct decisions for URL blocking but sometimes the full URL is required to make the correct decisions.

Squid has a feature called *ssl-bump* which allows the administrator to define how to behave for traffic on port 443 (note that this is HTTPS and other application-specific protocols). Squid can act as a man-in-the-middle, where to the browser it acts as if it is the server and to the server it acts as if it is the browser. This way, it can decrypt all HTTPS traffic and it mimics the server by generating a new SSL/TLS certificate that has all parameters of the original SSL/TLS certificate of the server, except that the generated certificate is signed by Squid itself. Squid uses a Squid CA certificate to perform the signing of generated certificates. Browsers have a list of all trusted CA certificates to detect tampering of SSL/TLS connections and by default complain about the certificates that Squid generates. To make the browsers trust the Squid CA certificate (and not complain about its certificates), one can add the CA certificate to the list of trusted certificates. It is relatively easy to administer the list of trusted

certificates for desktops and laptops with Windows or Linux, but not for smartphones. On some smartphones one can add the Squid CA certificate to the list of trusted certificates.

The blocking methods for HTTPS with Squid 3.5.x or later are in the following table:

redirect URL, *no ssl-bump*	ufdbguardd may redirect to an other HTTPS URL. caveat: the browser displays an error message to the end user like "the certificate does not match the domainname" or equivalent. second caveat: ufdbguardd only *sees the FQDN* and not the full URL to make blocking decisions.
redirect URL, with ssl-bump *peek+splice*	ufdbguardd may redirect to an other HTTPS URL. caveat: the browser displays an error message to the end user like "the certificate does not match the domainname" or equivalent. second caveat: ufdbguardd only *sees only the FQDN* and not the full URL to make blocking decisions.
redirect URL, with ssl-bump *peek+bump*	ufdbguardd sees the *full URL* and may redirect to an other HTTPS URL. The user sees an understandable message like "this site is blocked ...". caveat: the browser needs to trust the Squid bump by importing the Squid CA certificate. second caveat: there is a list of sites that malfunctions or refuses operation with Squid *peek+bump* mode and must be explicitly configured to use *peek+splice*.
terminate the connection	Squid can terminate a connection by acls. caveat: the browser displays an error message to the end user like "server terminated the connection" or equivalent. second caveat: ufdbguardd is not used.

Note that ssl-bump features client-first and server-first of older Squid versions are *not* supported.

peek+splice is an operating mode of Squid where it peeks at the SSL/TLS hello message of the browser, extracts the SNI, the Server Name Indication, and then splices the connection. Splicing in this context means that the connection between the browser and the webserver is untouched.

peek+bump is an operating mode of Squid where it peeks at the SSL/TLS hello message of the browser, extracts the SNI, the Server Name Indication, and then

The SNI that Squid peeked at, must be passed on to ufdbGuard for it to know to which webserver the browser connects to. See section 3.3.2 for more details.

A mix of *peek+splice* and *peek+bump* is also possible by means of acls of Squid. The mix can be used for example when one prefers to use peek+bump for the majority of HTTPS sites and use peek+splice for a small set of sites where bumping is not desired (e.g. sites of banks, sites with pinned certificates, etc.).

3.1.3 Recommended Configuration

In environments with desktops and laptops but without smartphones, the simplest configuration is not to use ssl-bump, and to configure all systems to explicitly use the Squid proxy, using a browser configuration or PAC.

In environments where smartphones are used, often it is not possible to configure all applications on all phones to use the Squid proxy and hence Squid is often configured in interception mode. When Squid

uses interception, browsers perform TLS/SSL handshakes with servers directly and Squid does not see a URL nor a FQDN of the webserver. To collect the FQDN of a connection, Squid must peek at the TLS/SSL handshake. This peek is non-intrusive and one can choose to configure Squid in one of the following modes: peek+splice, peek+bump, or a mix of peek+splice and peek+bump. The easiest configuration is the non-intrusive peek+splice mode. However, peek+splice mode only sends the FQDN of a connection to ufdbGuard and not the full URL, which may result in not 100% accurate filtering results.

In environments where there is a need for 100% accurate filtering, the peek+bump mode is the most common mode of operation for Squid.

3.2 HTTP URL Redirection

Squid sends each URL that it receives from a browser to ufdbGuard for inspection. If ufdbGuard finds a URL that needs to be blocked according to the local internet usage policy, it sends a substitution URL to Squid. This substitution URL is then used by Squid and content of the substitution URL is sent back to the browser.

A substitution URL is by default similar to this:

```
http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=adult&username=John&url=www.sex.com
```

So for each blocked URL, an appropriate message is retrieved which was generated by `URLblocked.cgi` at `cgibin.urlfilterdb.com`.

For reasons of efficiency, it is recommended to use the lightweight http daemon, `ufdbhttpd`, which has a built-in `URLblocked.cgi`. `ufdbhttpd` is part of the ufdbGuard software suite, can easily serve 700 requests/sec, and is recommended for all environments except the very large ones. Users of ufdbGuard which generate a very large number of hits on `URLblocked.cgi` on the server `cgibin.urlfilterdb.com` will be asked to stop using it.

Very large environments should use a web server on their LAN and install `URLblocked.cgi` on a web server. The web server must be able to incorporate perl scripts.

It is recommended ***not*** to redirect to a simple HTML page because not all blocked web objects are HTML-based objects. `URLblocked.cgi` produces an image for blocked images and produces a valid javascript document for blocked javascript objects, etc. If a blocked image is redirected to an HTML page, the browser receives HTML code where it expects an image and gets confused, may generate errors, may become slow in response, or even terminate unexpectedly.

3.3 HTTPS URL Redirection

As discussed in section 3.1.2, there are various ways that HTTPS URLs can be blocked, each with its own caveats. For correct interaction between browsers, Squid and ufdbGuard it is required to have equal configurations for Squid and ufdbGuard.

Squid	ufdbGuard
no ssl-bump	Works with any version of Squid. Each URL category has its own redirection statement which are used for redirecting HTTP-based URLs. For HTTPS-based URLs, ufdbguard uses `redirect-https`. The default value of `redirect-https` is "`blockedhttps.urlfilterdb.com:443`". `squid-uses-active-bumping` must be "`off`".
with ssl-bump *peek+splice*	Squid 3.5.20 or higher is required. Each URL category has its own redirection statement which are used for redirecting HTTP-based URLs. For HTTPS-based URLs, ufdbguard uses `redirect-https`. The default value of `redirect-https` is "`blockedhttps.urlfilterdb.com:443`". `squid-uses-active-bumping must be "off".`
with ssl-bump *peek+bump*	Squid 3.5.20 or higher is required. Each URL category has its own redirection statement which are used for redirecting HTTP-based URLs. For HTTPS-based URLs, ufdbguard uses `redirect-`*`bumped`*`-https`. The default value of `redirect-bumped-https` is "`https://blockedhttps.urlfilterdb.com/cgi-bin/URLblocked.cgi?clientgroup=%s&category=%t&url=%u`". `squid-uses-active-bumping` must be "`on`". `squid-version` must be "`3.5`" or higher.

Note that ssl-bump features client-first and server-first of older Squid versions are *not* supported.

The default value of `redirect-https` is `blockedhttps.urlfilterdb.com:443` which may seem odd at a first glance. This is because Squid receives a CONNECT to *`FQDN`*`:443` and expects a similar URL if rewritten. So it is not possible to specify a full URL. To prevent browsers complaining about self-signed certificates, best practice it to redirect to a URL that has a valid certificate. So, if the value of `redirect-https` is overridden, make sure that it points to a site with a valid certificate.

Note that `redirect-bumped-https` is only used where there is active bumping, e.g. Squid uses peek+bump. To prevent browsers complains about certificates, best practice it to redirect to a URL that has a valid certificate.

3.3.1 Squid Example Configuration, No SSL-Bump

```
http_port 3128
```

3.3.2 Squid Example Configuration, SSL-Bump peek+splice

The following is an example configuration for Squid 3.5. See the Squid wiki and the FAQ about ssl-bump for more explanation.

```
http_port 3130 ssl-bump generate-host-certificates=on  \
               dynamic_cert_mem_cache_size=4MB  \
               cert=/local/squid35/etc/certs/SquidTLSbumpCA1.pem  \
               sslflags=VERIFY_CRL_ALL  \
               options=NO_SSLv2,NO_SSLv3,No_Compression
dhparams=/local/squid/etc/dhparam.pem
```

```
sslcrtd_program /local/squid/libexec/ssl_crtd -s  \
                /local/squid/var/lib/ssl_db -M 4MB
sslcrtd_children 5

sslproxy_options NO_SSLv2,NO_SSLv3,No_Compression
sslproxy_cipher  ALL:!SSLv2:!ADH:!DSS:!MD5:!EXP:!DES:!PSK:!SRP:!RC4:!IDEA:!
SEED:!aNULL:!eNULL

# TLS/SSL bumping definitions
acl tls_s1_connect      at_step SslBump1
acl tls_s2_client_hello at_step SslBump2
acl tls_s3_server_hello at_step SslBump3

# TLS/SSL bumping steps
ssl_bump peek   tls_s1_connect all   # peek at TLS/SSL connect data
ssl_bump splice all                  # splice: no active bumping

# ufdbGuard must receive the SNI and other parameters:
url_rewrite_extras "%>a/%>A %un %>rm bump_mode=%ssl::bump_mode
sni=\"%ssl::>sni\" referer=\"%{Referer}>h\""
```

3.3.3 Squid Example Configuration, SSL-Bump peek+bump

The following is an example configuration for Squid 3.5. See the Squid wiki and FAQ about ssl-bump for more explanation.

```
http_port 3130 ssl-bump generate-host-certificates=on  \
               dynamic_cert_mem_cache_size=4MB  \
               cert=/local/squid/etc/certs/SquidTLSbumpCA1.pem  \
               sslflags=VERIFY_CRL_ALL  \
               options=NO_SSLv2,NO_SSLv3,No_Compression
dhparams=/local/squid/etc/dhparam.pem

sslcrtd_program /local/squid/libexec/ssl_crtd -s  \
                /local/squid/var/lib/ssl_db -M 4MB
sslcrtd_children 5

sslproxy_options NO_SSLv2,NO_SSLv3,No_Compression
sslproxy_cipher  ALL:!SSLv2:!SSLv3:!ADH:!DSS:!MD5:!EXP:!DES:!PSK:!SRP:!RC4:\
!IDEA:!SEED:!aNULL:!eNULL

# TLS/SSL bumping definitions
acl tls_s1_connect      at_step SslBump1
acl tls_s2_client_hello at_step SslBump2
acl tls_s3_server_hello at_step SslBump3

# define acls for sites that must not be actively bumped
acl tls_allowed_hsts    ssl::server_name .akamaihd.net
acl tls_server_is_bank  ssl::server_name .abnamro.nl
acl tls_server_is_bank  ssl::server_name .abnamro.com
acl tls_to_splice any-of tls_allowed_hsts tls_server_is_bank
```

```
# TLS/SSL bumping steps
ssl_bump peek   tls_s1_connect   # peek at TLS/SSL connect data
ssl_bump splice tls_to_splice    # splice some: no active bump
ssl_bump stare  all              # stare(peek) at server
                                 # properties of the webserver
ssl_bump bump                    # bump if we can (if the stare succeeded)

# ufdbGuard must receive the SNI and other parameters:
url_rewrite_extras "%>a/%>A %un %>rm bump_mode=%ssl::bump_mode
sni=\"%ssl::>sni\" referer=\"%{Referer}>h\""
```

Squid 4 has a new configuration directive `on_unsupported_protocol` which must be set to an appropriate value. Squid 4 also uses a new `tls_outgoing_options` directive.

3.4 How ufdbGuard Deals with URLs

URLs can be a complex string of characters. Since characters like `%`, `:` and = have a special meaning so they must be %-encoded, hence a %3D is a literal '='.

ufdbGuard supports URLs with UTF8 characters and internally converts all %-encoded characters to their binary form before the URL lookup in the tables takes place. The ufdbGenTable utility that converts a list of URLs to a URL table, also converts %-encoded characters, hence when is URL lookup takes place, converted characters are compared against converted characters.

3.5 Dynamic Proxy Tunnel Detection

The HTTPS protocol is encrypted and designed for secure transactions like online banking and is considered a safe protocol since it cannot be decrypted. Unfortunately, the HTTPS protocol is at the same time a security risk since anybody can type "proxy tunnel" at Google and find out how to transfer data to and from any system on the internet *bypassing firewalls and internet access security measurements* using the HTTPS protocol. This is possible since HTTPS is encrypted and most security measurements like anti-virus software, malware detection and intrusion detection systems cannot scan encrypted data.

ufdbGuard has a unique protection mechanism that detects abuse of the HTTPS protocol and blocks all unauthorized transfers of data with these so-called proxy tunnels. When a URL with the HTTPS protocol is evaluated by ufdbGuard, ufdbGuard opens a separate connection to the web server and probes it for SSH tunnels, proxies, various chat protocols and valid SSL certificates. The results of the probes are cached to ensure a good performance.

Proxy tunnels can also be used with reverse port forwarding (using ssh) which means that from any system on the internet an unauthorized connection can be made into the protected network. It also does not matter how good the firewall is! As long as HTTPS is allowed and there is not a proper countermeasure against proxy tunnels, a high security risk exists.

The ufdbguardd daemon verifies all websites that are accessed with the HTTPS protocol and detects all popular proxy tunnels.

3.6 Enhanced HTTPS Security

As stated earlier, the HTTPS protocol is a useful protocol where secrecy is desired. Financial transactions are a good example where HTTPS is desired. Unfortunately, because HTTPS is encrypted, the HTTPS protocol is also used by applications that may introduce security issues like proxy tunnels, remote access software and viruses. To enhance the security of HTTPS connections on the internet, two configuration options control the use of HTTPS. See section 5.5 for more information.

4 Software Installation

Starting with v1.31 a RPM package can be used to install ufdbGuard on server running Redhat and CentOS. If ufdbGuard is installed on an other OS, the installation steps outlined in section 4.2 are to be taken. The 64-bit package is built on CentOS 6 and CentOS 7. By default, the ufdbguard daemon runs as user `ufdb` and therefore during the package installation this user account is created if it does not already exist.

When upgrading ufdbGuard from a previous version, follow the installation steps as described in the following sections and read section 4.6 to ensure that all new URL categories are defined.

4.1 Installation from RPM Package

4.1.1 File Layout on Redhat and CentOS

The ufdbGuard package for Redhat and CentOS follows the guidelines of Redhat for the location of files and uses the directories and files as presented in the following table.

`/etc/sysconfig/ufdbguard`	system configuration file
`/etc/ufdbguard/ufdbGuard.conf`	ufdbGuard configuration file
`/etc/init.d/ufdb`	system initialization file
`/usr/sbin`	binaries and scripts
`/var/ufdbguard`	home directory of user `ufdb`
`/var/ufdbguard/images`	image directory for `ufdbhttpd`
`/var/ufdbguard/blacklists`	URL database directory
`/var/ufdbguard/logs`	directory for log files
`/usr/share/man/man{1,8}`	man pages

The URL database and log files may need up to 2 GB disk space. This amount depends mostly on the configuration of the maximum size of the log files. So make sure that `/var/ufdbguard` has sufficient space or configure ufdbGuard to use an other location to store the URL database and log files.

4.1.2 RPM Package Installation

Download the package suitable for the system. There is a choice for 32-bit systems and 64-bit systems. The rpm command installs the ufdbGuard software suite in the locations specified in the previous section. The command to install the 64-bit package is

```
$ yum install ufdbGuard-1.33-3.x86_64.rpm
```

and the command to upgrade the ufdbGuard package, is

```
$ yum upgrade ufdbGuard-1.33-3.x86_64.rpm
```

After installation two configuration files must be edited before the ufdbguard daemon can be started.

4.2 Installation from Sources

It is assumed that the Squid web proxy is already installed, configured and operational. Please refer to the documentation of Squid to make it operational (see also www.squid-cache.org).

4.2.1 Upgrading from a Previous Version

A previous version can simply be overwritten by a new version.
CAVEAT: you must stop squid and ufdbGuard to overwrite executables, so you must stop them before a 'make install'.

When an upgrade is performed, it is recommended to perform all installation steps including the last step, retrieval of database update in section 4.4.

When an upgrade is performed, the default configuration file (ufdbGuard.conf) is not installed to prevent loss of previous settings. It is recommended to read the sections about configuration options and to add settings for the new options to the existing configuration file.

NOTE: when upgrading, the make install verifies the current configuration file and adds new URL categories if they do not exist. Always look carefully at the output that make install produces.

4.2.2 User Account

The ufdbGuard software suite should be installed with its own user account, ufdb. Formerly, it was recommended to use the username squid, but this is now considered inappropriate because two different applications should have two different application owners. There is no technical reason not to use squid as the owner of the ufdbGuard package, so if you do not agree you can overrule the default by using user squid when running configure, see section 13 for details.

The user account must be a regular user account with credentials to login with a shell and be allowed to use crontab. It is recommended to use a new group and make the user the only member of the new group. It is suggested to use the groupname ufdb.

NOTE: the following steps in this section should be done as the aforementioned user.

4.2.3 Installation Directory

ufdbGuard can be installed in any directory and by default is installed in /usr/local/ufdbguard. Note that the packaged version of ufdbGuard obeys the FHS standard and uses /usr as TOPDIR. In this manual, the word TOPDIR refers to the top level installation directory for ufdbGuard.

4.2.4 Unpack Software

The tar file that contains the ufdbGuard software suite must be unpacked in a source directory of your choice, e.g. /local/src.

Unpack the ufdbGuard tar file in the source directory:

```
$ cd /local/src
$ tar xzf ufdbGuard-latest.tar.gz
```

A directory with the name ufdbGuard-1.33.1 contains the software.

4.2.5 Configure the Software Build

Before ufdbGuard is compiled, it needs to be configured and told what the TOPDIR of your choice is:

```
$ cd ufdbGuard-1.33.1
$ ./configure --prefix=TOPDIR
```

So, assuming that TOPDIR is /usr/local/ufdbguard, the configure command becomes:

```
$ ./configure --prefix=/usr/local/ufdbguard
```

The command `./configure --help` displays a description on how to configure alternative locations and username.

UfdbGuard is installed with the assumption that the user `ufdb` is owner of the installed files. If you prefer that ufdbGuard is installed with a different user account, you may use the `--with-ufdb-user` option. The administrator should create such a user account and the configure script reminds the administrator of this. Since previous versions used a default user '`squid`', you may want to use `--with-ufdb-user=squid`.

The most common error is that the *development* packages for openssl, zlib and bzlib are not installed. The configure command checks for the existence of these packages and gives an appropriate message. For most operating systems, one can find the packages *openssl-devel*, *zlib-devel* and *bzip2-devel* on the installation media. Note that each Linux distribution uses different package names and you may find that openssl-devel has an other similar name like *libssl-dev*. Likewise, the bzip2-devel package may be called *libbz-dev*. Refer to the Operating System manual on how to install additional packages.

4.2.6 Configure Software for Multiple Systems

ufdbGuard will use UNIX sockets whenever they are available which implies that ufdbGuard and Squid must be executed on the *same system*. Alternatively, the less efficient TCP sockets can be used which have the benefit that Squid and ufdbGuard can be executed on multiple systems. The choice for the type of sockets is made when the software is build; it is not a runtime configuration option.

To force the usage of TCP sockets, use the configure command with the option `--without-unix-sockets`. This option must be used if ufdbGuard runs on a different system then where Squid runs.

4.2.7 Compile Software

Now compile the software suite:

```
$ make all
```

And ignore warnings about compiler options.

4.2.8 Install Software

In case that you perform an upgrade and squid is already running with ufdbgclient redirectors, squid must be stopped to be able to perform a proper installation. After the installation, squid is started after ufdbGuard.

Stop any running instances of squid:

```
$ su -
# /etc/init.d/squid stop
```

Stop any running instances of ufdbGuard. i.e.

```
$ su -
# /etc/init.d/ufdb stop
```

The programs can now be installed. Note that you must be `root`.

```
# cd .../ufdbGuard-1.33.1
# make install
```

Note: the installation will look for an old configuration file and will update it and give feedback. The automatic updates are adding new URL categories (religion, extappl etc.) and replacing references to www.urlfilterdb.com/cgi-bin/URLblocked.cgi with cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi to offload www.urlfilterdb.com. It is recommended to read the output of this installation step and to modify the ACLs where necessary.

It is recommended to start the ufdbguardd daemon at system boot and before Squid is started. The start script is in the `init.d` directory (this may vary and depends on the OS). Optionally modify `UFDB_OPTIONS` to use the `-T` option to use the test mode.

And then start ufdbGuard and Squid – in this order only.

```
# /etc/init.d/ufdb start
# /etc/init.d/squid start
```

4.3 Configure ufdbGuard

The system configuration file contains a few variables necessary to start the ufdbguard daemon. The variables in `/etc/sysconfig/ufdbguard` are in the table below. In case that the URL database of URLfilterDB is used, the `DOWNLOAD_USER` and `DOWNLOAD_PASSWORD` must be set here. In most cases, the other variables do not have to be changed.

Variable	Meaning	Default value
`DOWNLOAD_USER`	Username to download URL database used by `ufdbUpdate`	*void*
`DOWNLOAD_PASSWORD`	Password to download URL database used by `ufdbUpdate`	*void*
`PROXY_USER`	Username for optional download proxy	*void*
`PROXY_PASSWORD`	Password for optional download proxy	*void*
`BLACKLIST_DIR`	Directory where URL database resides	`/var/ufdbguard/blacklists`
`CONFIGDIR`	Directory where `ufdbGuard.conf` resides	`/etc/ufdbguard`
`BINDIR`	Directory where binaries and scripts reside	`/usr/sbin`
`UFDB_OPTIONS`	Options for ufdbguard daemon	*void*
`RUNAS`	ufdbguard daemon runs as this user	`ufdb`
`MALLOC_ARENA_MAX`	max number of memory segments that the malloc (glibc 2.10 and newer) memory allocator uses.	`16`

The second configuration file is `/etc/ufdbguard/ufdbGuard.conf` which contains many parameters and rules for the ufdbguard daemon. Section 13 and onward contain all information about the configuration of ufdbguard.

4.4 Get Daily Updates

The URL database needs to be updated every day. If one has a trial license or a regular license from URLfilterDB, daily updates of the URL database can be downloaded with the provided ufdbUpdate script, see the next section for all details. In case that an other source for a URL database is used, the procedure outlined in section should be followed.

4.4.1 Daily Updates from URLfilterDB

The script `ufdbUpdate` takes care of downloading a new version of the URL database from URLfilterDB and signaling ufdbGuard that a new version is downloaded. Daily updates are available for everyone with a permanent or trial license of URLfilterDB.

The `ufdbUpdate` script should be run twice per day by the cron scheduler. Note that ufdbUpdate downloads a new version of the URL database *and* sends a signal to ufdbguardd to upload uncategorised URLs. A regular upload of uncategorised URLs is required to maintain the URL database up to date and therefor it is recommended to run ufdbUpdate twice per day.

There are 3 things to do:

- make sure that the `ufdb` user is allowed to run crontab jobs (`crontab -l` run as user `ufdb` should give no errors).
- enter the username and password in the system configuration file[1]. The name of the configuration file is system-dependent and may be `/etc/sysconfig/ufdbguard` or `/etc/conf.d/ufdb`. Edit the configuration file and set the variables `DOWNLOAD_USER` and `DOWNLOAD_PASSWORD`.
- to get the most recent updates, it is recommended to update the URL database by running `ufdbUpdate` in the early morning.

Edit the configuration file to include the username and password that you received when the (trial) license was received:

```
$ vi /etc/sysconfig/ufdbguard
...
DOWNLOAD_USER=lic99999
DOWNLOAD_PASSWORD=aa22bb
```

Users that evaluate the URL database may use the demoXX username and password. Just request a trial license on the website and you will receive a demoXX username. If a license is purchased, you will receive a unique username and password for daily downloads of the URL database.

Test the `ufdbUpdate` script with the verbose option:

```
# su - squid
$ /usr/local/ufdbguard/bin/ufdbUpdate -v
```

The output should be similar to:

```
http_proxy is not set: no proxy is used for downloads
Downloading the current database...
<retrieving URL database>
new database downloaded:
-rw-r--r-- 1 root root  17572147  May 16 05:04 /tmp/urlfilterdb-latest.tar.gz
Unpacking the database...
The downloaded database is installed in directory /local/squid/blacklists and
its subdirectories
```

[1] until version 1.30 the username and password were set inside the `ufdbUpdate` script. Later versions still support this but it is recommended to use of the system configuration file.

```
Sending HUP signal to the ufdbguardd daemon to load new configuration...
URL database creation date:  Mon May 16 06:15:47 CEST 2016
<retrieving license status>
URL database license status: OK
done.
```

To install the cron job for `ufdbUpdate`, edit the crontab table of the user `ufdb` and add the appropriate lines:

```
$ crontab -e
```

To run the URL database update each day at 6:15 AM, add the following line:

```
15 6  * * * /usr/local/ufdbguard/bin/ufdbUpdate
```

At this time, you may want to verify that the Squid housekeeping is also executed. Verify that the crontab for user `squid` has an entry to run "`squid -k rotate`" (see the squid manuals for more details). It is a common misconception that this command only rotates the logfile of Squid but it also does necessary housekeeping. Note that the Squid housekeeping must be done when there is almost no load (e.g. at 03:00 AM).

4.4.2 Exit Codes of ufdbUpdate

To monitor URL database updates, ufdbUpdate has a defined set of exit codes.

code	*explanation*
0	all OK
1	version mismatch warning; most likely there is a new version of ufdbguardd
2	license expiration warning: less than 2 months to renew license
3	license expired: a license renewal is required immediately
11	configuration error
12	temporary file error
13	download OK but cannot signal ufdbguardd to load the new URL database
21-40	exit code of ufdbUpdate is exit code of wget + 20. `wget` is the command that downloads the new URL database from the servers of URLfilterDB.
41-60	exit code of ufdbUpdate is exit code of gunzip + 40. `gunzip` uncompresses the downloaded URL database. There may be an issue with file system space.
61-80	exit code of ufdbUpdate is exit code of tar + 60. `tar` unpacks the downloaded URL database. There may be an issue with file system space.

In case of an error, it is advised to run `ufdbUpdate -v` from the command line to have more feedback about what is going wrong. License expiration warnings are also issued by ufdbguardd.

4.4.3 Database Updates from Other Sources

In case that a URL database is used that is not provided by URLfilterDB, the URL database is most likely a collection of flat files with domain names and URLs. In this case the user should implement itself a method for getting regular database updates. After downloading a new flat file URL database, the database must be converted to the proprietary database format of ufdbguardd using the `ufdbConvertDB` tool.

After a URL database conversion, `ufdbguardd` must be signaled to load the new URL database. There are various ways to do this:

```
/etc/init.d/ufdb reload       # systems without systemd
```

or

```
systemctl reload ufdbguard    # systems with systemd
```

or

```
/usr/sbin/ufdbsignal -C "sighup ufdbguardd"
```

4.4.4 Converting a URL Database

ufdbConvertDB converts a whole URL database without the need to call `ufdbGenTable` for each URL category. ufdbConvertDB requires one parameter which is the top level directory name where the URL database resides.

Example:

```
# ufdbConvertDB /usr/local/ufdbguard/blacklists
```

4.5 Firewall and Proxy Settings

4.5.1 ufdbGuard

ufdbGuard send probes to HTTPS sites to detect tunnels, detect proxies, detect Skype, detect Google Talk, other chat applications, and verifies SSL certificates. It also sends statistics[2] to the servers of URLfilterDB. Since tunnel and proxy detection are considered an important feature, it is strongly recommended to modify firewall rules to provide access.

ufdbGuard needs access to port 443 for all internet addresses and needs access to port 80 and port 443 of `updates.urlfilterdb.com`.

4.5.2 ufdbUpdate

ufdbUpdate downloads the URL database and obviously needs access to the servers of URLfilterDB. Firewall rules may need to be modified to provide access to `updates.urlfilterdb.com`.

A proxy can be used to download the URL database: edit the `/etc/sysconfig/ufdbguard` configuration file and assign the appropriate values to the variables `http_proxy`, `PROXY_USER` and `PROXY_PASSWORD`.

4.6 Extra Steps when Upgrading

New URL categories have been introduced since the release of ufdbGuard version 1.31. It is highly recommended to include the new URL categories in the configuration file since it prevents unnecessary processing of uncategorised URLs. So even if the new URL categories are not used for blocking, it is still recommended to define them.

[2] ufdbGuard and URLfilterDB B.V. respects the privacy of all persons and statistics do not include, directly or indirectly, any information about any person. See also the privacy policy of URLfilterDB B.V. at http://www.urlfilterdb.com/privacystatement.html.

The new URL categories are: *ms-data-collection*, *safe*, *dynaddress*, *religion*, *teamviewer*, *malware*, *dropbox*, *checked*. Below are the category definitions to be added to the configuration file.

```
# define the safe category
category safe {
   domainlist      "safe/domains"
   expressionlist  "safe/expressions"
   redirect        "http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=%t&url=%u"
}

# Microsoft collects vast amounts of user and system data from workstations,
# browsers and apps.  Define the Microsoft Data Collection category
category ms-data-collection {
   domainlist      “ms-data-collection/domains”
   redirect        “http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=%t&url=%u”
}

# define the dynaddress (dynamic DNS) category
category dynaddress {
   domainlist      "dynaddress/domains"
   expressionlist  "dynaddress/expressions"
   redirect        "http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=%t&url=%u"
}

# define the religion category
category religion {
   domainlist      "religion/domains"
   expressionlist  "religion/expressions"
   redirect        "http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=%t&url=%u"
}

# TeamViewer and VPNs are part of the proxies category.
# To distinguish Teamviewer it is a separate subcategory of proxies.
category teamviewer {
   domainlist      "proxies/teamviewer/domains"
   expressionlist  "proxies/teamviewer/expressions"
   redirect        "http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=%t&url=%u"
}

# define the malware category
category malware {
   domainlist      "malware/domains"
   expressionlist  "malware/expressions"
   redirect        "http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=%t&url=%u"
}

# define the checked category
category checked {
   domainlist      "checked/domains"
```

```
   redirect        "http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
category=%t&url=%u"
}
```

NOTE: modify the *redirect* lines to match the local policies.

NOTE: do not use the URL category *ufdbsafe* since it is a copy of *safe* and will be deleted from the URL database.

5 Configuration

This chapter and chapter 7 and 8 describe the features and configuration syntax for ufdbGuard in detail.

5.1 Infrastructure

Using a URL filter only contributes to a safer internet experience when the infrastructure is safe. The configuration of the infrastructure is out of scope of this document but it is highly recommended to use a DNS server that validates DNSSEC records, use a firewall that blocks incoming *and* outgoing traffic and to use antivirus solutions.

Google has invented the QUIC protocol that is used by some browsers and some apps on smartphones. Squid cannot proxy QUIC and it is recommended to configure the firewall to reject the QUIC protocol on UDP ports 80 and 443 with an ICMP icmp-port-unreachable packet to signal applications that try to use QUIC to abandon it immediately and revert to HTTP(S).

5.2 How Access is Controlled

Access is controlled by a access control list (ACL). Each ACL defines which *source* has access to which URL *categories*. URL categories are predefined sets of URLs and included in the URL database. The default configuration file already has all URL categories predefined. A *source* is a group of users and may be defined in various ways (see section 8.4 for more details).

5.3 Know your Internet Usage Policy

Before one can start with the configuration of ufdbGuard, an internet usage policy must already be defined and state which web site categories are considered unwanted and therefore must be blocked by ufdbGuard. The HR department might be a good starting point to find out which categories of the database must be used to block access to parts of the internet.

Note: the default configuration of ufdbGuard blocks only adult, security, p2p, proxies, gambling, malware, warez, violence, drugs and illegal. By default, ufdbGuard blocks all unsafe HTTPS connections with the options enforce-https-with-hostname, enforce-https-official-certificate and https-prohibit-insecure-sslv2. The SSLv3 protocol is also considered insecure but is at the time of this writing still used on some websites, so the https-prohibit-insecure-sslv3 options is by default off.

The internet usage policy may block certain categories which need exceptions. E.g. the internet usage policy may prohibit visiting shops but an authority may decide to to allow access to the local pizza delivery – which is usually included in the category shops – to provide a meal for those who do overtime. Or a policy may allow access to shops but block one or two particular shops. The URL categories *alwaysallow* and *alwaysdeny* are used to define these exceptions and contains URLs that should be allowed or denied according to the local policy. Section 7 explains on how to configure these categories.

ufdbGuard supports time-based ACLs which enable the implementation of internet usage policies that have different policies during different time of the day or week. With time-based ACLs it is, for example, possible to define that after working hours a different – usually more relaxed – policy is used. See section 8.6 for more information.

It is recommended to start with the URL filter in the test mode. In this mode, sites are not blocked but only logged in the log file. An analysis of the log file may help in defining the appropriate Internet Usage Policy and to find out which sites should never be blocked (see also section 7.1). At last a free tip: to prevent a storm at the help desk or system administrator, it is advised to inform users about (change in) the implementation of a URL filter *before* it is implemented and to allow users to request to define exceptions with the alwaysallow category.

The ufdbGuard software suite includes a tool to find out what type of sites are currently visited by analyzing the Squid log file. See section 10 for a detailed description of ufdbAnalyse. This tool can also be used in determining or adjusting the Internet Usage Policy.

5.3.1 Recommendations

URL filters are used for various and often good reasons. We like to make clear that a URL filter is not the solution for everything and that education of users of what is allowed and what is suspicious is always necessary.

The URL database cannot be used as a replacement for an antivirus solution. We recommend to educate users about viruses and phishing.

When your internet usage policy prohibits certain classes of websites, consider reasonable alternatives. For example, if the site of dropbox is prohibited, consider to have a server on your own network where users can have similar functionality. This can have unexpected advantages like higher productivity and better user acceptance of the URL filter.

5.4 Proxy Tunnel Detection

Proxy tunnels are a security risk and it is strongly recommended to use detection of proxy tunnels. Enable proxy tunnel detection in the configuration file `ufdbGuard.conf` with the following line:

```
check-proxy-tunnels queue-checks
```

This option queues checks for proxy tunnels to be detected a few seconds later. It means that a proxy *can* be used but only for 1-2 seconds. Alternatively, proxy tunnels can be detected in an aggressive mode, where all HTTPS traffic is tested for proxies *before* access is given. The aggressive mode introduces some delays for HTTPS traffic and is therefore only recommended in very high security environments. If proxy tunnels are allowed, the value for `check-proxy-tunnels` can be `off` or `log-only`. All valid options are:

```
check-proxy-tunnels queue-checks
check-proxy-tunnels aggressive
check-proxy-tunnels log-only
check-proxy-tunnels off
```

NOTE: proxy tunnel detection depends on the usage of the category *proxies*. Only when the URL category proxies is blocked, the proxy detection is performed.

NOTE: when the “aggressive” detection method is used, the minimum number of threads in the ufdbguardd daemon is increased to 132 and additional threads are created. See section 5.13.2 on how to configure the number of threads in ufdbguardd and to calculate how many threads ufdbgclient uses.

5.5 Control HTTPS Usage

Most websites that use HTTPS for legitimate business reasons use a TLS/SSL certificate that is signed by a well-known certificate authority and have a fully qualified domain name in the URL for maximum security and a clear identification of the website, while most websites that use HTTPS for other reasons, have self-signed certificates and IP addresses instead of domain names. HTTPS is usually secure enough to protect the connection to eavesdropping but has an old protocol option which is rarely used and insecure. The old and insecure SSLv2 protocol can be blocked by means of a configuration option.

ufdbGuard is able to detect a small subset of protocols used on port 443 on the internet: HTTPS (HTTP+TLS/SSL), Skype and other chat applications, SSH and proxies. When ufdbguardd probes port 443 it is quite possible that it stumbles upon a port of a website that uses an unknown protocol. There is a configuration option to allow or block unknown protocols. The default is to allow such protocol since there are many sites which use port 443 to deliver video content or other application-specific content.

When ufdbguardd probes a webserver it does a DNS lookup and ufdbguardd before version 1.32, only used the IPv4 addresses of the server. If the option `use-ipv6-on-wan` is set to `on` (default), ufdbguardd also probes the IPv6 addresses of web servers.

Access to HTTPS websites can be controlled with the following options.

The following options are default in ufdbGuard.conf:

```
enforce-https-with-hostname on
enforce-https-official-certificate on
https-prohibit-insecure-sslv2 on
https-prohibit-insecure-sslv3 off      # SSLv3 is insecure and still used
allow-unknown-protocol-over-https on
```

It is recommended to keep these options set to "`on`" to have a increased protection level against phishing sites, proxies and websites with untrusted SSL certificates. In case that a legitimate website uses an IP address in the URL or an SSL certificate that is not signed by a trusted authority, it is recommended to add this site to the locally trusted websites (see section 7.1).

If the option `allow-unknown-protocol-over-https` is `on`, ufdbguardd allows unknown protocols by URLs composed of an IP address and the HTTPS port (port 443). ufdbGuard detects regular SSL connections, SSH, Skype and other chat protocols. It is recommended to leave this option `on` since many sites use URLs of this type for video and other applications.

It is mandatory to use these options *inside* the definition of the category "security":

```
category security
{
   domainlist "security/domains"
   option     enforce-https-with-hostname on
   option     enforce-https-official-certificate on
   option     https-prohibit-insecure-sslv2 on
   option     https-prohibit-insecure-sslv2 off
   option     allow-unknown-protocol-over-https on
   redirect   ...
}
```

Finally, include the category security into the ACLs (see section 5.8) to block HTTPS abuse.

Note: the option to verify HTTPS certificates only works when the URL database of URLfilterDB is used since the URL database includes the necessary SSL certificates to verify HTTPS websites. The CA certificates are in the file `.../security/cacerts`. In case that an alternative file with certificates needs to be used, one can be defined with the keyword `cacerts` which must be used inside the security category definition. The configuration file may look like this:

```
category security
{
   domainlist "security/domains"
   cacerts    "/var/ufdbguardd/cacerts"
   ...
```

ufdbGuard can also be configured to use a directory with certificates. The keyword `cacerts-dir` followed by a quoted string indicates the name of this directory. ufdbGuard allows the following configurations:

- no `cacerts` and no `cacerts-dir` keywords: ufdbGuards uses the default builtin cacerts file,
- no `cacerts` but `cacerts-dir` keyword: ufdbGuard uses the configured directory,
- both `cacerts` and `cacerts-dir` keywords: ufdbGuard uses both (the file first and then the directory).

5.6 Which HTTP Server to use for Redirection Messages

Whenever a website is blocked, Squid receives a redirection URL from ufdbguardd. This URL must point to a web server that can display an appropriate error message. Version 1.15 introduced a new lightweight http daemon with a built-in URLblocked.cgi: `ufdbhttpd`.

There are 3 options for serving redirection messages:

1. use `ufdbhttpd` on a non-privileged port on host *myhost.mylan*:
   `http://`*`myhost.mylan`*`:8080/cgi-bin/URLblocked.cgi`
2. use the perl script `URLblocked.cgi` on your own internal web server:
   `http://myhost.mylan/cgi-bin/URLblocked.cgi`
3. use `URLblocked.cgi` of URLfilterDB[3]:
   `http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi`

Option 1 is preferred for all but very large environments. Option 2 is preferred for very large environments. You need to have a web server like Apache with mod_perl or FastCGI enabled to implement option 2.

The default configuration file comes with option 3 preconfigured. If option 1 or 2 is used, the `redirect` statements must be changed accordingly. E.g., change the lines with

```
   redirect  http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?
color=orange&...
```

into

```
   redirect  http://myhost.mylan:8080/cgi-bin/URLblocked.cgi?color=orange&...
```

`ufdbhttpd` is lightweight and can be installed on the same server where `ufdbguardd` is installed.

For option 2: the perl script `URLblocked.cgi` is included in the distribution in the directory `samples`. The perl script uses some images which are also in the samples directory and which need to be installed on the webserver. The reader is referred to the documentation of the used web server on how to install this perl script.

Redirection of HTTPS-based URLs is more complex. See the next section for more information.

[3] URLfilterDB reserves the right to block access to `cgibin.urlfilterdb.com` to any site that uses it excessively.

5.7 Redirection of HTTPS-based URLs

Squid requires that HTTP-based URLS are redirected to other HTTP-based URLs and that HTTPS-based URLS are redirected to other HTTPS-based URLs. This causes a problem since most web browsers do not accept a redirection of a HTTPS-based URL. There is no solution for this issue: the standards of HTTP, ICAP and web proxies do not have support for such feature. Basically, this means that blocked HTTPS-based sites cause unexpected browser-generated error messages like "cannot connect to *www.example.com*" or "*www.example.com* does not have a valid SSL certificate".

5.8 Main Configuration Settings ufdbGuard

Use your favorite editor to edit the configuration file and define which categories must be blocked.

```
$ vi ufdbGuard.conf
```

There are 5 sections with a comment 'EDIT THE NEXT LINE'. Find each section and change the configuration where needed.

The first section defines the locations of the log directory and the blacklist database directory. This section does not have to be changed if the initial choice of these locations during the configuration of ufdbGuard (see section 4.2.5) has not changed. The section looks like this:

```
# EDIT THE NEXT LINE FOR LOCAL CONFIGURATION
dbhome "/var/ufdbguard/blacklists"
logdir "/var/ufdbguard/logs"
```

The process id of the ufdbguardd daemon is stored in the pid file which defaults to /var/run/ufdbguard/ufdbguardd.pid. The `pidfile` statement overrules the name pf the pid file.

```
pidfile "/var/ufdbguard/ufdbguardd.pid"
```

The second section defines the IP address range of your local network. The section looks like this:

```
source allSystems {
   ip 10.0.0.0/8
}
```

The appropriate network subnet must be entered in this section. `10.0.0.0/8` and `192.168.0.0/16` are the most common values for this. Consult your network administrator for assistance.

The third section defines the usage and settings of `ufdbhttpd`, the lightweight HTTP daemon:

```
http-server { port= 8080, interface= all, images="/usr/local/ufdbguard/images" }
```

If you don not use `ufdbhttpd`, then transform the line above into a comment (with a #).

The fourth and fifth sections are close to each other and define the list of categories to be blocked (one list for the systems with IP address defined in `allSystems` and one list for all other systems). Change the list of categories to be blocked. The default list of blocked categories contains the categories security, adult, p2p, proxies, gambling, violence and warez. The fourth section looks like this:

```
acl {
   allSystems {
      # EDIT THE NEXT LINE
      pass !adult !p2p !proxies !gambling !violence !warez !security … any
   }
```

To block a category, it needs to be present with an exclamation mark (!) that is used as a blocking indicator. So to block the *adult* category, `!adult` must be present in the line that starts with `pass`. If you prefer to allow gambling, the definition "`!gambling`" must be removed.

NOTE: because of the large set of URLs used by chat applications, configuration of the chat category is a little more complex. See section 8.8 for a detailed explanation.

At a site that only blocks security, adult, p2p and proxies, the section looks like this:

```
acl {
   allSystems {
      # EDIT THE NEXT LINE
      pass !security !adult !p2p !proxies any
   }
```

The fifth section is very similar to the fourth section and defines which categories to block for computer systems that are not part of `allSystems`.

The configuration for the use of Skype and other chat applications is more complex. Read section 8.8.3 on how to allow or block Skype.

5.9 Ensure Access to Internal and 3rd Party Websites

The domain name of your company may be included in the URL database of URLfilterDB. To ensure access to all own websites, the category *alwaysallow* should be configured (see section 7.1).

To prevent unhappy users, one should carefully examine which own sites and sites of 3rd parties are used for regular daily activities and make sure that these sites can be accessed without restriction. Therefore, it is recommended to run a few days in test mode (see section 5.13.3) and add sites of important 3rd parties to the category *alwaysallow* (see section 7.1).

5.10 Redirection Message Variables

A redirect statement may contain variables that are dynamically expanded to their values. E.g. `%u` is substituted by the URL string that is blocked. So when www.adult.com is blocked, the redirection URL

```
http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?url=%u
```

is dynamically expanded to

```
http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?url=www.adult.com
```

The variable `%u` is associated with the parameter *url*. The parameters are interpreted by the `URLblocked.cgi` script and used to produce an appropriate HTML message for the internet browser of the end user. The following table contains the list of all redirection message parameters.

parameter	*variable*	*explanation*
admin	%A	administrator
clientaddr	%a	source address (IP or FQDN)
clientuser	%i	username
clientname	%n	source domainname
clientgroup	%s	source identifier
category targetgroup	%t	URL category
categories	%C	all categories that a URL belongs to[4]
url	%u	URL
url	%U	parsed URL
httpcode	-	httpcode=204 defines that the redirected URL has an empty body and the HTTP code is 204 NO CONTENT
-	%%	the literal character '%'

redirection message parameters & variables

5.11 Redirection Message Styles

When a URL is prohibited to be visited, a message is displayed that access is forbidden. The style, size and background color can be set by the administrator.

The options for the background color are: orange (default), white, black, grey and red. The options for the font size of the message are: normal (default), small and large.

To change the style of a message, edit the configuration file and change the default settings of all `redirect` rules.

```
$ vi ufdbGuard.conf
...
redirect http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?admin=
%A&mode=default&color=orange&size=normal&clientaddr=%a&clientname=
%n&clientuser=%i&clientgroup=%s&targetgroup=%t&url=%u
...
```

Substitute the default value for a new value according to the options specified earlier in this section.

The default error message includes an explanation why the URL is blocked, the URL, the email address of the administrator and a link for more information. Alternatively, the output can be configured to be very simple and just consist of the word “Forbidden” (in different languages) and the category of the URL in red. To use this style one should set the *mode* parameter to `simple-red`.

The *mode* parameter can have additional values specific for advertisements only. Valid values for *mode* are: transparent (default), noads, square, cross and simple-red. The transparent mode displays nothing, the noads mode displays “no ads”, the square mode shows a grey square, the cross mode shows a transparent cross, and the simple-red mode shows a short message and for ads the text “*ads*”.

[4] expands to a comma-separated list of category identifiers. Each identifier may be preceded with an exclamation mark if it is blocked.

The *color* parameter defines the background color and determines automatically an appropriate foreground color. The *size* parameter defines the size of the text.

parameter	***value***
`mode`	default noads square cross simple-red
`color`	white black grey orange red
`size`	small normal large

redirection style parameters

5.12 Automatic Improvements of the URL Database

Although the URL database is updated daily, it may happen that some web sites are not included in the URL database and therefore users might be able to visit inappropriate websites. UfdbGuard has a configuration option to collect a sample of these uncategorized domains and to upload them to servers of URLfilterDB for inclusion in the database. This option is `on` by default. ufdbGuard does *not* track all websites that are visited. Instead, it only tracks websites which are not yet part of the current URL database.

Privacy is guaranteed since no identification of client, source or user is included, just the domain name is registered for categorization. URLfilterDB has a privacy policy which prohibits sharing of any information whatsoever with 3rd parties. URLfilterDB is a Dutch company and obeys Dutch and European privacy laws. See also http://www.urlfilterdb.com/en/privacystatement.html for further information.

All software tools of URLfilterDB strip all parts of (intermediate) data which may contain privacy-related items such as IP address of the user's PC, user name, password and query parameters. A secure HTTPS channel is used when data is uploaded.

To prevent analysis of uncategorized URLs, the `analyse-uncategorised-urls` configuration option must be set to `off`. It is, however, highly recommended to leave this option `on` to receive more updates in the URL database which results in better filtering. URLfilterDB categorizes URLs as soon as possible to include them in the URL database.

5.13 Configure Squid

5.13.1 Default configuration

To let Squid use the URL filter (in Squid terminology the URL filter is a *redirector* or a *URL rewriter*), 2 parameters must be added to the squid configuration file.

```
$ vi squid.conf
```

Add the following 2 lines for Squid **3.3** and newer:

```
url_rewrite_program /usr/sbin/ufdbgclient -m 4 -l /var/squid/logs
url_rewrite_children 16 startup=8 idle=2 concurrency=4
```

The `-m` parameter specifies the number of threads that each ufdbgclient (16) process uses, so the total number of threads used by squid/ufdbgclient is 16 * 4 = 64. The `-l` parameter is followed by the name of a directory where ufdbgclient stores its logfiles.

NOTE: the `-m` parameter and the `concurrency=` parameter must always have the same value.

NOTE: replace */var/squid/logs* by a suitable directory where the `squid` user has write permission.

NOTE: by default, the `ufdbguardd` cannot support more than 65 `ufdbgclient` threads so make sure that the total number of threads of all ufdbgclient processes does not exceed 65. See the next section for more information on the modify the parameters of ufdbgclient.

5.13.2 Large System Configuration

ufdbGuard 1.33 has a new feature for very large systems which is a multithreaded version of ufdbgclient. Squid can send a configurable number of requests to a URL rewriter and before the new multithreaded version those requests were queued. The multithreaded version, however, sends *NT* requests immediately to ufdbguardd, where *NT* is the number of threads per ufdbgclient process. Note that Squid can start multiple instances of ufdbgclient so the number of ufdbgclient processes and the number of threads for each process must be carefully chosen and must be less than the number of worker threads of ufdbguardd.

It is suggested to have the number of threads in each ufdbgclient process between 4 and 16 and to use between 4 and 64 ufdbgclient processes. ufdbguardd has by default a minimum of 65 threads when aggressive tunnel detection is off and a minimum of 132 threads when aggressive tunnel detection is on. For most systems the number of threads is sufficient and increasing this number can decrease performance. Ask help from the support desk if you have any doubts.

The number of threads per ufdbgclient process is specified with the `-m` option. The number of ufdbgclient processes and the number of request that Squid sends to an individual process are defined with `url_rewrite_children`.

The configuration file of squid 3.3 and newer may look like this:

```
url_rewrite_program /usr/sbin/ufdbgclient -m 8 -l /var/squid/logs
url_rewrite_children 32 startup=16 idle=4 concurrency=8
```

The above implies that a total number of 32 * 8 = 256 threads are used, so the number of worker threads in ufdbguardd ***must*** be increased to at least 256.

NOTE: the ufdbgclient process is a child process of Squid and runs as user `squid`. Therefore, the log directory of ufdbgclient must have write permission for the user `squid`.

On very large systems, the number of worker threads needs to be increased. The default number of worker threads is 65. Define the number of workers in ufdbGuard.conf:

```
num-worker-threads <NUMBER>
```

where the maximum number of worker threads is 1285. To not waste resources, it is recommended to have the number of worker threads slightly higher than the total number of threads used by ufdbgclient.

5.13.3 Test Mode

ufdbGuard has a test mode where internet access is not blocked and where the log file contains lines which web sites would have been blocked in normal mode.

In case that you want to use the test mode, add the `-T` option for `ufdbguardd`. The place to set the options is in `/etc/sysconfig/ufdbguard` on systems that have `/etc/sysconfig` or `/etc/init.d/ufdb` (might be elsewhere for your OS). The file should then have a line like this:

```
UFDB_OPTIONS="-T"
```

In test mode, the log file of ufdbguardd contains lines like this:

```
TEST BLOCK adult www.sex.com
```

5.13.4 Configuration for 2 servers

In case that ufdbguardd runs on a different system then where Squid runs, the server name and port number need to be configured for `ufdbgclient` with the following options:

```
-S servername -p 3977
```

The `squid.conf` file should then have a line like this:

```
url_rewrite_program /usr/sbin/ufdbgclient -S urlchecker01 -p 3977
```

Also at configuration time, it is required to define the use of 2 servers (see section 4.2.6).

The default port number is 3977 and can be configured to be a different number. To use an alternative port number, one uses the port of choice with the `-p` parameter for the `ufdbgclient` command. To configure an alternative port number for ufdbguardd, the keyword `port` followed by the port number is added to its configuration file. In addition, ufdbguardd can be configured to use all or one particular interface to listen on. To configure a particular interface, the keyword `interface` must be used in the configuration file. The keyword must be followed by the string "`all`" or an IP address. By default ufdbguardd listens on all interfaces.

Example configuration to listen on port 9999 of interface 10.2.3.4:

```
port 9999
interface 10.2.3.4
```

5.14 Configuration of Browsers

The browsers like Internet Explorer, Firefox, Opera and others need to be configured to

- use the Squid proxy
- display original error messages.

5.14.1 Configure Browser to use Squid

How to configure the browser depends much on the chosen technical infrastructure. If Active Directory Server is used, it can be used to configure browsers to use the Squid proxy as the HTTP and HTTPS proxy. Browsers can also be configured to automatically detect a proxy using the WPAD[5] discovery mechanism or can be configured manually.

5.15 Monitoring of ufdbGuard

ufdbGuard uses 2 log files and the system log to write messages to. The location of the log files depends on the choice that was made during the configuration phase (see section 4.2.5). The names of the log files are `ufdbguardd.log` and `ufdbgclient.log`. The location and name of the system log is OS-dependent and is usually `/var/log/messages` or `/var/adm/syslog.log`.

It is recommended that the log files are regularly inspected, either manually or automatically. Serious errors have a 5-star indicator (`*****`).

[5] See for example http://en.wikipedia.org/wiki/Web_Proxy_Autodiscovery_Protocol for more information on WPAD.

5.16 Compatibility between ufdbGuard and squidGuard

ufdbGuard 1.0 was based on squidGuard 1.2.0 and a lot has been changed since then. ufdbGuard has new features that squidGuard lacks: *SafeSearch, safer HTTPS options, logfile rotation, HTTP daemon, configurable error messages, analysis of uncategorised URLs, dynamic user lists, dynamic URL tables* and *CPU binding*. ufdbGuard does not support the squidGuard features *user quota, ldap, mysql* and *separate logfiles*. Despite the differences, a configuration file of squidGuard works with ufdbGuard with little or no modifications.

ufdbGuard uses the more logical keyword `category` where squidGuard uses `destination`. For reasons of compatibility, ufdbGuard recognizes the `destination` keyword.

ufdbGuard has a different database format than squidGuard and has the utility `ufdbConvertDB` to convert the flat files of the database of squidGuard to the database format of ufdbGuard.

5.17 Verification of the Configuration

The configuration file can be verified for errors without starting ufdbguardd. There are two ways to verify a configuration file. When the default configuration file is changed, the startup script can be executed with the parameter `configtest`, e.g.:

```
# /etc/init.d/ufdb configtest
```

Alternatively, the ufdbguardd program can be called directly to verify the configuration file. Use the normal options as usual and append the command line options "-C verify", e.g.:

```
# ufdbguardd –c new.conf –C verify
```

6 Start the URL Filter

6.1 On Linux-based systems

To start the URL filter daemon:

```
# /etc/init.d/ufdb start
```

Now you can restart squid to use the URL filter:

```
# /etc/init.d/squid reload
```

or

```
# /etc/init.d/squid stop
# /etc/init.d/squid start
```

7 Exception Rules

In cases where exceptions to the categories of URLfilterDB are desired, an administrator can define 2 extra categories that are managed by the administrator and never by URLfilterDB.

7.1 Allow Extra Sites

A common case is that you want to ensure access to your own websites and websites of 3rd parties that are used for normal activities. To grant users access to the company websites, the URL `yourcompany.com` needs to be added to the category *alwaysallow*.

For universities that want to allow access to all other universities in the United States, a simple `edu` in the alwaysallow list. In the UK, only `ac.uk` needs to be included.

Edit the file that contains the extra sites that should always be allowed:

```
$ cd /usr/local/ufdbguard
$ vi blacklists/alwaysallow/domains
```

Add the appropriate URLs and always remove a leading `www.`:

```
yourcompany.com
news.google.com
google.com/news
```

In case that you have a file with many URLs having a leading `www.`, you may use the -W option to remove the `www.` prefix automatically.

Additional domains can be added according to the local internet usage policy. For example, if news should be blocked but access to CNN allowed, then `cnn.com` should be added also. Alternatively, when news should be blocked but Google news allowed, `news.google.com` and `google.com/news` should be added.

ufdbGuard only uses proprietary database files, so generate an `.ufdb` database file from the ASCII file with `ufdbGenTable`:

```
$ cd /var/ufdbguard
$ ufdbGenTable -W -n -t alwaysallow -d blacklists/alwaysallow/domains
```

The above command generates the file `blacklists/alwaysallow/domains.ufdb` and should be invoked each time that the domains file is changed. `ufdbGenTable` does a sanity check which performs a few checks on the validity of the URLs. The `-W` option removes the initial "`www[0-9]{0,2}.`" from all URLs. The `-n` option prevents encryption of the URL table. The `-h` option shows all options.

Then activate the category by editing the `ufdbGuard.conf` file and uncomment the category definition for *alwaysallow*. The configuration file should have the following lines:

```
category alwaysallow
{
   domainlist alwaysallow/domains
   redirect ...
}
```

Also add the category alwaysallow to the ACL `allSystems`. The ACL should then start with

```
   pass alwaysallow !adult ...
```

Finally signal the ufdbguardd daemon about the new configuration:

```
# /etc/init.d/ufdb reconfig
```

7.2 Block Extra Sites

In case that you want to block access to a site that is not in any category, you can add this site to the category *alwaysdeny*. For example, `google.com` is not in any category but if you like to block access to this popular search engine, `google.com` can be included in the alwaysdeny category. Analogous to the alwaysallow category, the domain (without leading `www.`) must be added to the category domains file, and the ACL `allSystems` must be extended.

Edit the file that contains the extra sites that should always be blocked:

```
$ cd /var/ufdbguard
$ vi blacklists/alwaysdeny/domains
```

Add the appropriate URLs (always remove a leading `www.`):

```
google.com
```

ufdbGuard only uses proprietary database files, so generate an `.ufdb` file from the ASCII file with `ufdbGenTable`:

```
$ cd /var/ufdbguard
$ ufdbGenTable -W -n -t alwaysdeny -d blacklists/alwaysdeny/domains
```

The `-W` option removes the initial `www.` from all URLs. The above command generates the file `blacklists/alwaysdeny/domains.ufdb` and should be invoked each time when the domains file is changed. Then activate the category by editing the `ufdbGuard.conf` file and uncomment the category definition for alwaysdeny. The configuration file should have the following lines:

```
category alwaysdeny
{
   domainlist alwaysdeny/domains
   redirect ...
}
```

Also, add the category alwaysdeny to the ACL `allSystems`. The ACL should then start with

```
   pass alwaysallow !alwaysdeny !adult ...
```

Finally signal the ufdbguardd daemon about the new configuration:

```
# /etc/init.d/ufdb reconfig
```

7.3 Block Extra Sites – a more Advanced Example

If *parts* of a website need to be blocked, the method described in the previous section does not provide a solution. Suppose we allow users to watch CNN news but do not want them to look at the CNN news videos.

It was observed that the CNN videos are made available through the following URLs:

```
http://www.cnn.com/video
http://edition.cnn.com/video
http://vid.cnn.com
```

Since access to `www.cnn.com` is allowed and access to `www.cnn.com/video` is not, a *domain block* is not an option, but a so-called *url block* is. Therefore, in this example the above URLs need to be divided into 2 categories: those which can be blocked with domain blocks and those which can be blocked with url blocks.

The block referring to a domain goes into the file …`/alwaysdeny/domains`:

```
vid.cnn.com
```

and the blocks referring to a full URL go into the file …`/alwaysdeny/urls`:

```
cnn.com/video [6]
edition.cnn.com/video
```

The `ufdbGenTable` command has a `-u` option to include the url blocks and the command to generate a new `.ufdb` file including all options becomes this:

```
$ cd /var/ufdbguard/blacklists
$ ufdbGenTable -W -n -t alwaysdeny -d alwaysdeny/domains -u alwaysdeny/urls
```

Do not forget to signal the ufdbguardd daemon about the new configuration:

```
# /etc/init.d/ufdb reconfig
```

[6] Note that there is no leading `www.` since the URL filter works internally with stripped URLs that do not contain a `www.` part. Use the `-W` option for files with URLs that have a `www.` prefix.

7.4 Block Extra Sites – Using URL Parameters

URL tables usually contain only simplified URLs without parameters like `?foo=abc&t=4&bar=23`. Starting with version 1.32 of ufdbGuard one can define URL tables with URLs that contain parameters so that URLs with parameters can be matched. The URL table can have a subset of parameters that are used in a particular URL and if all parameters specified in the URL table are matched, the URL is matched. See below for a few examples:

Browser URL: `www.popularvideos.com/watch?v=aaaabbbb`

URL table contains:	matched?
`popularvideos.com/watch?v=aaaabbbb`	yes
`popularvideos.com/watch?foo=x&v=aaaabbbb&bar=y`	yes
`popularvideos.com/watch?v=yyyyzzzz`	no

Browser URL: `www.popularvideos.com/watch?v=aaaabbbb&n=2`

URL table contains:	matched?
`popularvideos.com/watch?v=aaaabbbb&n=2`	yes
`popularvideos.com/watch?n=2&v=aaaabbbb`	yes
`popularvideos.com/watch?n=2`	no
`m.popularvideos.com/watch?v=aaaabbbb&n=2`	no

WARNING: although the performance of parameter matching is very good, performance can degrade when 2 or more parameters are used inside URL tables and URLs have many parameters.

Due to technical limitations, individual parameter names and values cannot exceed 244 characters. If exceeded, the parameter match fails for the particular parameter and ufdbguardd continues processing.

8 Advanced Options

8.1 Filtering HTTPS Traffic

HTTPS is encrypted and designed not to be tampered with and filtering HTTPS traffic is a challenge (see section 3.1.2 for more information). Filtering HTTPS traffic is sometimes required often based on a requirement to block sites that use HTTPS *partially*. When parts of a site must be blocked, the URL filter can no longer rely on a filter policy based on domainname but must have the full URL to decide if a URL must be blocked or not.

To filter HTTPS traffic, the administrator needs to

- configure Squid ssl-bump in peek+bump mode

- configure ufdbguard for active ssl-bump.

8.1.1 Active HTTPS Bumping with ufdbGuard

To configure ufdbGuard for ssl-bump peek+bump mode, add the following to the configuration file:

```
squid-uses-active-bumping on
```

The above feature modifies the behavior of ufdbGuard and no longer blocks a `CONNECT example.com` from browsers, but allows the CONNECT request and waits for the GET https://www.example.com/somethingnaughty/index.html.

However, for sites that use the HTTPS port 443 for non-HTTPS protocols the above does not work since ufdbguardd only receives the CONNECT and never a GET or POST that it may block. So for categories where it is known that the protocol is not HTTPS, the CONNECT must be blocked even when `squid-uses-active-bumping` is on. Use `block-bumped-connect on` inside a category definition to achieve this. For example:

```
squid-uses-active-bumping  on
...
category skype {
   domainlist                     chat/skype/domains
   option block-bumped-connect    on
   option allow-skype-over-https  off
}
```

When Squid uses active bumping, ssl-bump in peek+bump mode, ufdbguardd receives a CONNECT with an IP address instead of a CONNECT with a full qualified domainname, FQDN. This implies that the option `enforce-https-with-hostname` must be automatically disabled by ufdbguardd when `squid-uses-active-bumping` is on.

8.2 Blocking Adult Images produced by Search Engines

Search engines like Google, Yahoo and MSN have a capability to search for images and allow users to view adult images that can not be blocked in a simple way since the images come from Google, Yahoo and MSN themselves and in general one would not like to block all images from the search engines.

Google[7], Yahoo and other search engines offer a S*afeSearch*™ feature which blocks most adult images. UfdbGuard has the configuration parameter `safe-search` that enforces the safesearch policies of the search engines. The default value for the parameter is ON.

The safesearch feature enforces safe searches for the following search engines: A9, AOL search, Alltheweb, Ask, Bing, Blinkx, Buscamundo, Dogpile, Excite, Foxnews, Google, Infospace, Live, Lycos, Metacrawler, Metaspy, MSN, Terra, Webcrawler, webpile, Yahoo and other less popular search engines.

Note that many search engines use HTTPS exclusively and that HTTPS filtering is required for the safe-search feature to work.

8.2.1 SafeSearch of Google

Google allows users to search with HTTPS. Searches with HTTPS can be modified by a URL filter to force a SafeSearch but but only when the ssl-bump peek+bump feature of Squid 3.5+ is used.

Google offers an alternative to enforce SafeSearch which is explained on their website: https://support.google.com/websearch/answer/186669?hl=en option 3 and configure DNS to have a CNAME record entry for www.google.com pointing to forcesafesearch.google.com. Also make CNAME entries for Google on popular TLDs in your region, e.g. www.google.de, www.google.es or www.google.com.br.

8.2.2 Content Restriction on Youtube

Youtube content also can be restricted using DNS. Youtube uses the same mechanism as Google and is explained here: https://support.google.com/youtube/answer/6214622. To implement it one needs to add

[7] Google SafeSearch filtering is a trademark of Google.

a CNAME restrict.youtube.com for the following domains: www.youtube.com, m.youtube.com, youtubei.googleapis.com, youtube.googleapis.com and www.youtube-nocookie.com.

8.2.3 SafeSearch of Bing

Bing offers an alternative to enforce SafeSearch which is explained on their website: http://help.bing.microsoft.com/#apex/18/en-US/10003/0 and configure DNS to have a CNAME record entry for www.bing.com pointing to strict.bing.com.

8.2.4 Global SafeSearch Option

If Squid is configured to use ssl-bump peek+bump, one can enforce SafeSearch for all users. Use the following:

```
safe-search on
```

8.2.5 Per-ACL SafeSearch Option

If Squid is configured to use ssl-bump peek+bump, one can have fine-grained control over the SafeSearch feature. The global safe-search feature must be turned off and a category must be defined that includes the safe-search option. The newly created category can be used in the ACLs as demonstrated in the following example.

```
safe-search off
...
category safesearch {
   option safe-search on
}
...
acl {
   ...
   allSystems {
      pass !alwaysdeny !proxies ... safesearch any
   }
```

8.3 YouTube Education Filter

YouTube has a filter for schools. Details about the filter are on the help system of YouTube: http://support.google.com/youtube/bin/answer.py?hl=en&answer=1686318. To be able to use the educational filter of YouTube one must setup a school account and request an edufilter ID.

The use of the YouTube educational filter is controlled with two parameters:

```
youtube-edufilter    on
youtube-edufilter-id "ABCD1234567890abcdef"
```

8.4 Different Policies for Different Users

Suppose that ufdbGuard is used in a bank. The internet usage policy could be defined as: block sex, chat, dating, entertainment, finance, news, webmail. This policy can be appropriate for most users but not for staff working in a dealing room where access to news and finance-related sites is required. This section explains how to achieve this.

Always be careful with the order of rules and make sure that the more privileged user (groups) is defined first in the ACL.

8.4.1 Policy based on IPv4 Address

If the dealing room has a separate IPv4 subnet, the *dealing room* policy can be defined in the following way in the ufdbGuard configuration file.

```
source dealingroom {
   ipv4 10.4.0.0/16      # e.g. the dealing room uses this subnet
}

acl {
   # more privileged users first
   dealingroom {
      pass !security !adult !warez any
   }
   allSystems {
      pass !security !adult !p2p !proxies !dating !entertain !warez any
   }
   default {
      pass none
      # the following redirect is for the pseudo category 'none'
      redirect http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?...
   }
}
```

Alternatively, one can also use a single IP address, a range of IP addresses or a list of IP addresses.

Single IPv4 address:

```
   ipv4 10.4.2.240
```

Range of IPv4 addresses:

```
   ipv4 10.4.2.0 - 10.5.9.255
```

List of IPv4 addresses:

```
   ipv4list "<filename>"
```

where the file contains one IPv4 address per line. For backward compatibility the quotes are optional. Each source can have multiple occurrences of the `ipv4` and `ipv4list` keywords.

Keyword Compatibility

The keywords `ipv4` and `ipv4list` were introduced in ufdbGuard 1.33 and have identical behavior to the keywords `ip` and `iplist`.

8.4.2 Policy based on IPv6 Address

If the dealing room has a separate IPv6 subnet, the *dealing room* policy can be defined in the following way in the ufdbGuard configuration file.

```
source dealingroom {
   ipv6 2001:db8:2::/24      # e.g. the dealing room uses this subnet
}

acl {
   # more privileged users first
   dealingroom {
      pass !security !adult !warez any
   }
   allSystems {
      pass !security !adult !p2p !proxies !dating !entertain !warez any
   }
```

```
   default {
      pass none
      # the following redirect is for the pseudo category 'none'
      redirect http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?...
   }
}
```

Alternatively, one can also use a single IPv6 address or a list of IPv6 addresses.

Single IPv6 address:

```
   ipv6 2001:db8:2::12:1001
```

IPv6 subnet:

```
   ipv6 2001:db8::/32
```

List of IPv6 addresses:

```
   ipv6list "<filename>"
```

where the file contains one IPv6 address or one IPv6 subnet per line. For backward compatibility the quotes are optional.

Each source can have multiple occurrences of the `ipv6` and `ipv6list` keywords. The keywords `ipv6` and `ipv6list` were introduced in ufdbGuard 1.33.

8.4.3 Policy based on Username

Internet access policies can also be based on an individual user or a list of usernames. In this example, a list of usernames is used to use the dealingroom policy. The `source` definition defines which users are members of the group *dealingroom*.

```
source dealingroom {
   user      "admin"
   userlist  "/usr/local/ufdb/etc/dealers"
}
```

The file `/usr/local/ufdb/etc/dealers` should contain the usernames of all dealers. The final policy definition is the same as a policy based on IP address:

```
acl {
   # more privileged users first
   dealingroom {
      pass !security !adult !proxies !dating any
   }
   allSystems {
      pass !security !adult !proxies !chat !dating !entertain !webmail any
   }
   default {
      pass none
      redirect http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?...
   }
}
```

CAVEAT: additional configuration is required to make Squid able to find out which users are using Squid. You may configure Squid to use user authentication *or* you have to install `identd` on all PCs to support this feature. Please read the Squid documentation for more information and do not forget to use `acl` *`foo`* `ident REQUIRED`.

8.4.4 Usernames with Domainnames

In case that users are authenticated by Squid, squid sends the authenticated username to ufdbGuard. The authenticated username can include a domainname, e.g. "johndoe@MYDOMAIN". To be able to match usernames with an included domainname, ufdbGuard has a new[8] option `strip-domain-from-username`. This option is `off` by default, and should be turned `on` when applicable.

Example of LDAP authentication

Let's assume that users are authenticated by Squid using LDAP. The PAM LDAP module (see also `man pam_ldap`) should be configured and ufdbGuard should have an appropriate source definition. E.g.

```
source managers {
   unix group "managers"
}
source staff {
   unix group "staff"
}
```

where `managers` and `staff` are LDAP groups.

Note that starting with v1.30 there is an alternative way to use LDAP-based groups. See section 8.4.6 for more information.

8.4.5 Policy based on UNIX Group name

Internet access policies can also be based on a UNIX group. In this example, a groupname is used to use the dealingroom policy. In this case, the `source` definition defines that the group `dealer` represents the dealingroom.

```
source dealingroom {
   unix group dealer
}
```

The group `dealer` must be a valid UNIX group name. The final policy definition is the same as a policy based on IP address:

```
acl {
   # more privileged users first
   dealingroom {
      pass !security !adult !proxies !dating any
   }
   allSystems {
      pass !security !adult !proxies !chat !dating !entertain !webmail any
   }
   default {
      pass none
      redirect http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?...
   }
}
```

CAVEAT: additional configuration is required to make Squid able to find out which users are using Squid. You may configure squid to use user authentication or you have to install `identd` on all PCs to support this feature. Please read the Squid documentation for more information and do not forget to use `acl` *`foo`* `ident REQUIRED`.

[8] the option `strip-domain-from-username` was introduced in version 1.29.

8.4.6 Policy based on Dynamically Generated List of Usernames

ufdbguard can import lists of users from any user database, whether it is LDAP, Kerberos-based, Active Directory, MySQL or any other type of database with usernames. The database is queried using an external script that produces a list of usernames, one username per line. ufdbguardd regularly executes the external script to always be up-to-date.

Internet access policies can be based on dynamically generated list of usernames. In this example, a list of usernames is used to use the finance policy. The `source` definition defines which users are members of the group *finance*.

```
source finance {
   execuserlist  "/usr/local/ufdb/bin/execuserlist.sh finance"
}
```

The user-defined shell script `/usr/local/ufdb/bin/execuserlist.sh` must produce a list of usernames, one per line. An example shell script `execuserlist.sh` is located in the `.../samples` directory. The example script uses ldapsearch to produce a list of usernames based on a OU in an example LDAP database and can be adapted to produce a list of users based on any type of user directory that can be queried from the command line.

`execuserlist` must be followed by a quoted command which may include parameters.

The command defined by each `execuserlist` is executed when ufdbguardd starts and also every 15 minutes to automatically pick up changes in user groups. Only one execuserlist definition may exist in any source. ufdbGuard does not support a mix of user sources.

The final policy definition is the same as a policy based on IP address:

```
acl {
   # more privileged users first
   finance {
      pass !security !adult !proxies any
   }
   allSystems {
      pass !security !adult !proxies !chat !dating !entertain !webmail any
   }
   default {
      pass none
      redirect http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?...
   }
}
```

CAVEAT: additional configuration is required to make Squid able to find out which users are using Squid. You may configure Squid to use user authentication *or* you have to install `identd` on all PCs to support this feature. Please read the Squid documentation for more information and do not forget to use `acl` *`foo`* `ident REQUIRED`.

The list of usernames retrieved by execuserlist is retrieved again every 15 minutes to automatically pick up changes in user groups. The refresh time (default 15 minutes) can be set using the `refreshuserlist` keyword followed by a number representing the number of minutes for the refresh interval. The interval must be between 5 and 1440 minutes. For example:

```
refreshuserlist 5
```

Debugging scripts which generates a list of usernames may be difficult and therefore ufdbguardd has an option to debug and log the output generated by shell scripts. For example:

```
ufdb-debug-external-scripts on
```

The logfile of ufdbguardd will contain lines with `execuserlist: received "foo"` when the debug option is `on`.

8.4.7 Policy based on Domain Name

Internet access policies can also be based on the domain name of a PC. In this example, the PCs in the dealingroom have a unique domain name that can be used to define a policy.

Let's assume that the dealingroom PCs have a name with the same subdomain like pc31.dealingroom.bank.com, and then the `source` definition for the `dealingroom` group is as follows:

```
source dealingroom {
   domain "dealingroom.bank.com"
}
```

To use this feature in older versions of Squid, the reverse name lookup must be enabled in `squid.conf`:

```
log_fqdn on
```

8.4.8 Policy based on Multiple Source Types

A source may have multiple source types: usernames, domainnames and IP addresses. Depending on the requirements, one may wish to define that *either* source type or *all* source types must be matched to match a source. By default, ufdbguard matches a source when either source type is matched. Consider the following definition of source "teacher":

```
source teacher {
   user  "MrB"
   ip    10.2.1.1
}
```

ufdbGuard matches the source teacher when the username is "MrB" or the IP address is 10.2.1.1 and implies `evaluate-or`. To define that both source types must be matched one can add to the source definition `evaluate-and`:

```
source teacher {
   evaluate-and
   user  "MrB"
   ip    10.2.1.1
}
```

8.5 Multiple ACLs

In case that there are exceptions for (groups) of users, multiple ACLs will be used (see also the examples for a dealing room in the previous sections).

The order of the definition of sources is important since the *first* source that matches a URL/user/IP is used to determine a block or a pass and no further sources are taken into account. The following example demonstrates this.

Suppose there is an administration PC that should have access to all websites and that the PC has a fixed IP address: 10.2.3.4. The configuration file should have the source definition for the administration PC *in front of* the source allSystems. Otherwise, the administration PC is considered part of allSystems and the ACL rule for adminpc is never used to decide a block or pass. So the order of the definitions is always important.

```
source adminpc {
   ip 10.2.3.4
}
source allSystems {
   ip 10.0.0.0/8
}
acl {
   adminpc {
      pass any
   }
   allSystems {
      pass !security !adult !proxies !chat !dating !entertain !webmail any
   }
   default {
      pass none
      redirect http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?...
   }
}
```

Note that the `redirect` statement is required because the pseudo-category `none` is used.

8.6 Time-Based ACLs

ufdbGuard supports time-based ACLs which enable the implementation of internet usage policies that have different policies during different time of the day or week. As the first step, a definition of one or more time intervals is given with the `time` statement. A time interval definition can contain two types of time intervals: the `weekly` directive is used to define reoccurring time intervals and the `date` directive is used to define special dates. The syntax is explained with the following example.

```
time "working-hours" {
   weekly mon,tue,wed,thu,fri  08:00 - 19:00
   weekly sat,sun              08:00 - 12:30
   date   *-*-01   # every first of the month
   date   2010-12-31
}
```

Weekly reoccurring hours have one or more days separated by commas or a wild character, followed by a time interval: *HH:MM - HH:MM*. The names of the days may be replaced by the wild card "*" to denote all days. Dates have the format *YYYY-MM-DD* and may contain a wild character for the year, month and/or day.

Once the time intervals are defined, they can be applied to the ACLs with the directives `within`, `outside` and `else`. The following example shows how these directives can be used.

```
acl {
   allSystems within "working-hours" {
      pass !security !adult !proxies !p2p !dating !entertain !webmail any
   } else {
      pass !security !adult !proxies !p2p any
   }
   ...
```

In the above example users are not allowed access to sites in the categories dating, entertainment and webmail during office hours. The `else` part defines an alternative ACL for all other hours that are not inside the *working-hours* time interval. The `else` part is optional.

As an alternative to the `within` directive, the `outside` directive can be used which has opposite semantics.

8.7 Whitelisting

Whitelisting is used in case that users are only allowed to visit a predefined set of websites. In this case, the ACL for allSystems contains the categories *alwaysallow* and *none*. The ACL in the configuration file looks like this:

```
acl {
   allSystems {
      pass alwaysallow none
      # the following redirect is for the pseudo category 'none'
      redirect "http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?..."
   }
   default {
      pass none
      redirect "http://cgibin.urlfiterdb.com/cgi-bin/URLblocked.pl?..."
   }
}
```

Note that the extra `redirect` statement is required because the pseudo-category `none` is used.

8.8 Chat Applications

Many chat applications have more functionality than a simple chat. They can share files or even do remote control of another workstation. These additional features bring security risks and may violate the local Internet Usage Policy. It is suggested to evaluate the use of ebuddy (see http://www.ebuddy.com) since this chat application can connect to many major chat sites but does not have the additional features.

In case that one or more chat applications are allowed, the ACL to configure this requirement is a little more complex because the chat applications use a large and complex set of URLs and most chat applications use the HTTPS port for a proprietary protocol which may violate HTTPS security options. The next sections describe in detail on how to block or allow one or more chat applications.

It is possible to block or allow a subset of chat applications, but since the chat applications use a large set of URLs and use login server, chat servers and direct peer to peer communication, both over HTTP and HTTPS, the ACLs for correct and efficient implementation of the requirements are different than other URL categories. This section explains the details and it is recommended to read the whole section to understand all aspects.

8.8.1 Chat and HTTPS

Many chat protocols use the HTTPS port. Many chat protocols use a proprietary protocol and hence not the normal SSL encrypted data stream that the normal HTTPS websites use. ufdbGuard, however, has important security features to detect SSH tunnels and other proxies and VPNs over HTTPS and probes HTTPS ports to detect them.

To enable chat applications *and* to detect tunnels and proxies correctly, there are rules for the ACLs as explained in the following sections.

8.8.2 Ebuddy

Ebuddy is a web-based chat application that provides chat, voice and video calls. Ebuddy does not have additional features like file sharing that may violate a local Internet Usage Policy and is therefore considered safer than other chat and VOIP applications.

Blocking Ebuddy is easy because one only needs to block a small set of URLs to block the whole application. To block Ebuddy, the subcategory `chat-ebuddy` must be defined and the relevant ACL must contain `!chat-ebuddy` to block Ebuddy.

The URL database has a subcategory for ebuddy called `chat-ebuddy`. The default definition is as follows:

```
category chat-ebuddy {
   domainlist      "chat/ebuddy/domains"
   expressionlist  "chat/ebuddy/expressions"
   redirect        ...
}
```

To allow ebuddy chat, the ACL must include the `chat-ebuddy` category in front of the `security` and `proxies` category. The ACL must be similar to the following:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat-ebuddy
           !security !proxies ...
      ...
```

8.8.3 Skype

Skype is a VOIP application that also can be used to transfer files and give remote access using screen sharing. These additional features *may* violate the local Internet Usage Policy.

Blocking Skype is easy because one only needs to block a small set of URLs to block the whole application. To block Skype, the subcategory `chat-skype` must be defined and the relevant ACL must contain `!chat-skype` to block Skype.

Skype is a popular VOIP application that requires a more complex configuration because it uses the HTTPS port 443 for its proprietary protocol, which would be blocked if the options for safer HTTPS are used. Skype also uses IP addresses instead of hostnames, so Skype conflicts with the security options `enforce-https-with-hostname` and `enforce-https-official-certificate`. The URL database of URLfilterDB contains a subcategory for Skype with many URLs that Skype uses. Since Skype connects to IP addresses of Skype users, it is not feasible to have all these IP addresses in the URL database and therefore ufdbGuard dynamically detects the use of Skype.

An extra subcategory needs to be configured:

```
category chat-skype {
   domainlist      "chat/skype/domains"
   expressionlist  "chat/skype/expressions"
   redirect        ...
}
```

For those who want to use Skype and have a safer use of HTTPS, the option `allow-skype-over-https` must be used. The category `security` must have this option set to ON:

```
category security {
   option          allow-skype-over-https on
   ...
}
```

Note: URLs are matched against categories in the order that they appear in an ACL and therefore the category `chat-skype` must be placed *before* the categories `security` and `proxies` in the ACLs to prevent that the security category blocks access to Skype URLs, e.g.:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat-skype
           !security !proxies ...
      ...
```

Skype must be configured to use Squid as its proxy: menu Options, tab Advanced, tab Connection, HTTPS proxy.

8.8.4 Yahoo Messenger

Yahoo instant messenger, or Yahoo IM, is a chat and VOIP application that also can be used to transfer files. These additional features *may* violate the local Internet Usage Policy.

Blocking Yahoo IM is easy because one only needs to block a small set of URLs to block the whole application. To block Yahoo IM, the subcategory `chat-yahoo` must be defined and the relevant ACL must contain `!chat-yahoo` to block Yahoo IM.

Yahoo IM is a popular chat application that requires a more complex configuration because it uses the HTTPS port 443 for its proprietary protocol, which would be blocked if the options for safer HTTPS are used. Yahoo IM also uses IP addresses instead of hostnames, so Yahoo IM conflicts with the security options `enforce-https-with-hostname` and `enforce-https-official-certificate`. The URL database of URLfilterDB contains a subcategory for Yahoo IM with many URLs that Yahoo IM uses. Since Yahoo IM connects to a large set of IP addresses, it is impossible to have all these IP addresses in the URL database and therefore ufdbGuard dynamically detects the use of Yahoo IM.

Yahoo IM also uses URLs which are used by other services of Yahoo. These other services may be blocked by the configuration which is a potential problem. Therefore there is a second subcategory called `chat-allowed` which contains these URLs. This subcategory contains only a few URLs and should be placed in an ACL immediately following `chat-yahoo`.

The two subcategories are in the default configuration file ufdbGuard.conf of version 1.25 or later but inside comments. The default definition of the subcategories is:

```
category chat-yahoo {
   domainlist      “chat/yahoo/domains”
   expressionlist  “chat/yahoo/expressions”
   redirect        ...
}
category chat-allowed {
   domainlist      “chat/allowed/domains”
   expressionlist  “chat/allowed/expressions”
   redirect        ...
}
```

And the `security` category must have the following option settings:

```
category security {
   option          allow-yahoomsg-over-https on
   ...
}
```

Note: URLs are matched against categories in the order that they appear in an ACL and therefore the category `chat-yahoo` must be placed *before* the categories `security` and `proxies` in the ACLs to prevent that the security category blocks access to Yahoo IM URLs, e.g.:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat-yahoo chat-allowed
           !security !proxies ...
      ...
```

Yahoo IM must be configured to use Squid as its proxy: menu Messenger, item Preferences, category Connection, option connect via a proxy server, option HTTP proxy.

8.8.5 Facebook Chat

Facebook Chat is a chat and VOIP application. Blocking Facebook Chat is easy because one only needs to block a small set of URLs to block the whole application. To block Facebook Chat, the subcategory `chat-facebook` must be defined and the relevant ACL must contain `!chat-facebook` to block Facebook Chat.

Facebook Chat is a popular chat application that requires a more complex configuration because it uses the HTTPS port 443 for its VOIP protocol, which would be blocked if the options for safer HTTPS are used. Facebook Chat also uses IP addresses instead of hostnames, so Facebook Chat conflicts with the security options `enforce-https-with-hostname` and `enforce-https-official-certificate`. The URL database of URLfilterDB contains a subcategory for Facebook Chat with many URLs that Facebook Chat uses. Since Facebook Chat connects to a set of IP addresses, it is impossible to have all these IP addresses in the URL database and therefore ufdbGuard dynamically detects the use of Facebook Chat.

Facebook Chat also uses URLs which are used by other services of Facebook. These other services may be blocked by the configuration which is a potential problem. Therefore there is a second subcategory called `chat-allowed` which contains these URLs. This subcategory contains only a few URLs and should be placed in an ACL immediately following `chat-facebook`.

The two subcategories are in the default configuration file ufdbGuard.conf of version 1.27 or later but inside comments. The default definition of the subcategories is:

```
category chat-facebook {
   domainlist      “chat/facebook/domains”
   expressionlist  “chat/facebook/expressions”
   redirect        ...
}
category chat-allowed {
   domainlist      “chat/allowed/domains”
   expressionlist  “chat/allowed/expressions”
   redirect        ...
}
```

And the `security` category must have the following option settings:

```
category security {
   option          allow-fb-chat-over-https on
   ...
}
```

Note: URLs are matched against categories in the order that they appear in an ACL and therefore the category `chat-facebook` must be placed *before* the categories `security` and `proxies` in the ACLs to prevent that the security category blocks access to Facebook Chat URLs, e.g.:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat-facebook chat-allowed
           !security !proxies ...
      ...
```

8.8.6 AOL Messenger

AOL instant messenger, or AIM, is a chat and VOIP application that also can be used to transfer files. These additional features *may* violate the local Internet Usage Policy.

Blocking AIM is easy because one only needs to block a small set of URLs to block the whole application. To block AIM, the subcategory `chat-aim` must be defined and the relevant ACL must contain `!chat-aim` to block AIM.

AIM is a popular chat application that requires a more complex configuration because it uses the HTTPS port 443 for its proprietary protocol, which would be blocked if the options for safer HTTPS are used. AIM sometimes also uses IP addresses instead of hostnames, so AIM conflicts with the security options `enforce-https-with-hostname` and `enforce-https-official-certificate`. The URL database of URLfilterDB contains a subcategory for AIM with many URLs that AIM uses. Since AIM connects to a large set of IP addresses, it is impossible to have all these IP addresses in the URL database and therefore ufdbGuard dynamically detects the use of AIM.

AIM also uses URLs which are used by other services of AOL. These other services may be blocked by the configuration which is a potential problem. Therefore there is a second subcategory called `chat-allowed` which contains these URLs. This subcategory contains only a few URLs and should be placed in an ACL immediately following `chat-aim`.

The two subcategories are in the default configuration file ufdbGuard.conf of version 1.25 or later but inside comments. The default definition of the subcategories is:

```
category chat-aim {
   domainlist      "chat/aim/domains"
   expressionlist  "chat/aim/expressions"
   redirect        ...
}
category chat-allowed {
   domainlist      "chat/allowed/domains"
   expressionlist  "chat/allowed/expressions"
   redirect        ...
}
```

And the `security` category must have the following option settings:

```
category security {
   option          allow-aim-over-https on
   ...
}
```

Note: URLs are matched against categories in the order that they appear in an ACL and therefore the category `chat-aim` must be placed *before* the categories `security` and `proxies` in the ACLs to prevent that the security category blocks access to AIM URLs, e.g.:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat-aim chat-allowed
           !security !proxies ...
      ...
```

AIM must be configured to use Squid as its proxy: menu Menu, item Settings, category Connection, option connect using proxy and protocol HTTP/HTTPS.

8.8.7 Google Talk

Google Talk is a chat and VOIP application that also can be used to transfer files. This additional feature *may* violate the local Internet Usage Policy.

Blocking Google Talk is easy because one only needs to block a small set of URLs to block the whole application. To block Google Talk, the subcategory `chat-google` must be defined and the relevant ACL must contain `!chat-google` to block Google Talk.

Google Talk is a popular VOIP application that requires a more complex configuration because it uses the HTTPS port 443 for its proprietary protocol, which would be blocked if the options for safer HTTPS are used. Google Talk also uses IP addresses instead of hostnames, so Google Talk conflicts with the security options `enforce-https-with-hostname` and `enforce-https-official-certificate`. The URL database of URLfilterDB contains a subcategory for Google Talk with many URLs that Google Talk uses. Since Google Talk connects to a large set of IP addresses, it is impossible to have all these IP addresses in the URL database and therefore ufdbGuard dynamically detects the use of Google Talk.

To allow Google Talk the following subcategories are used:

```
category chat-google {
   domainlist      “chat/google/domains”
   expressionlist  “chat/google/expressions”
   redirect        ...
}
category chat-allowed {
   domainlist      “chat/allowed/domains”
   expressionlist  “chat/allowed/expressions”
   redirect        ...
}
```

And the `security` category must have the following option settings:

```
category security {
   option          allow-gtalk-over-https on
   ...
}
```

Note: URLs are matched against categories in the order that they appear in an ACL and therefore the category `chat-google` must be placed *before* the categories `security` and `proxies` in the ACLs to prevent that the security category blocks access to Google Talk URLs, e.g.:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat-google chat-allowed
           !security !proxies ...
```

```
      ...
```

Google Talk must be configured to use Squid as its proxy: menu Settings, category Connection, option Use the following proxy.

8.8.8 Live Messenger - MSN

Live Messenger, a.k.a. MSN, is a chat and VOIP application that also can be used to transfer files. This additional feature *may* violate the local Internet Usage Policy.

Blocking MSN is easy because one only needs to block a small set of URLs to block the whole application. To block MSN, the subcategory `chat-msn` must be defined and the relevant ACL must contain `!chat-msn` to block MSN.

MSN is a popular VOIP application that requires a more complex configuration because it uses a large set of URLs and some URLs are used for other purposes than only chat.

Note: MSN clients also communicate directly with `login.live.com` on port 443 and therefore firewalls rules need to be set. This behavior is at least *odd* and MSN messenger should use the proxy for all traffic but this is not the case.

To allow MSN the following subcategories are used:

```
category chat-msn {
   domainlist      "chat/msn/domains"
   expressionlist  "chat/msn/expressions"
   redirect        ...
}
category chat-allowed {
   domainlist      "chat/allowed/domains"
   expressionlist  "chat/allowed/expressions"
   redirect        ...
}
```

Note: URLs are matched against categories in the order that they appear in an ACL and therefore the category `chat-msn` must be placed *before* the categories `security` and `proxies` in the ACLs to prevent that the security category blocks access to MSN URLs, e.g.:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat-msn chat-allowed
           !security !proxies ...
      ...
```

MSN must be configured to use Squid as its proxy: menu Tools, item Options, category Connection, Advanced Settings, HTTP proxy server.

8.8.9 All chat applications

Skype and other chat application can also be used to transfer files and give remote access using screen sharing. These additional features may violate the local Internet Usage Policy.

Blocking all chat is relatively easy because one only needs to block a small set of URLs to block the chat applications. To block all chat, the URL category `chat` must be defined and the relevant ACL must contain `!chat` to block all chat and set the relevant options in the `security` category OFF.

```
category security {
   option          allow-aim-over-https off
   option          allow-gtalk-over-https off
   option          allow-skype-over-https off
   option          allow-yahoomsg-over-https off
   ...
}
```

To allow all chat applications, the ACL must have the `chat` and `chat-allowed` categories in front of the `security` and `proxies` categories and the `security` category must have the correct settings for options that control the behaviour of ufdbguardd for HTTPS ports (port 443).

The default configuration file of ufdbguardd version 1.25 and higher have the following URL categories defined.

```
category chat {
   domainlist      “chat/domains”
   expressionlist  “chat/expressions”
   redirect        ...
}
category chat-allowed {
   domainlist      “chat/allowed/domains”
   expressionlist  “chat/allowed/expressions”
   redirect        ...
}
```

The ACL to allow all chat applications is as follows:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           chat chat-allowed
           !security !proxies ...
      ...
```

Various chat applications use the HTTPS port (port 443) but do not use URLs with a FQDN and do not use SSL. Therefore the security category must have the following options set.

```
category security {
   option          allow-aim-over-https on
   option          allow-gtalk-over-https on
   option          allow-skype-over-https on
   option          allow-yahoomsg-over-https on
   option          allow-unknown-protocol-over-https on
   ...
}
```

8.9 Anti-phishing from PhishTank

PhishTank (www.phishtank.com) provides a URL list with URLs used for phishing. ufdbGuard downloads an up-to-date version of the URL list from PhishTank. Users of ufdbGuard can choose to include this URL list in the access control list.

URLfilterDB and PhishTank are not associated in any way and the URL list of PhishTank is not part of the URL database of URLfilterDB. The URL list of PhishTank is always downloaded independently on the system where `ufdbUpdate` is executed.

To use the URL list of PhishTank, the configuration file should include the definition of the URL category, e.g.

```
category phishtank {
   domainlist      “phishtank/domains”
   expressionlist  “phishtank/expressions”
   redirect        ...
}
```

The ACL to block URLs of the phishtank category is as follows:

```
acl {
   allSystems {
      pass alwaysallow !alwaysblock
           !phishtank
           !security !proxies ...
      ...
```

In case that the URL list of PhishTank is not desired and must be prevented to be downloaded, the configuration file `ufdbGuard.conf` must contain the following line.

```
# do-not-download-phishtank
```

Alternatively, to prevent downloads of the URL list of PhishTank, an empty file with the name `.nodownloads` in the directory `.../phishtank` prevents downloads.

8.10 Default blocking behavior

8.10.1 ufdbguardd

The URL filtering system has two conditions where it cannot perform the regular filtering function:

a) whenever there is a fatal error

b) when the URL database is reloaded.

By default, whenever one of the 2 conditions occur, ufdbguardd sends a message to Squid telling Squid that the URL is not filtered, i.e. allowed to see. The only possible alternative to allowing all URLs, is to block all URLs. For those environments where it is necessary to block all URLs under the above conditions, the following 2 configuration settings can be set in the configuration file:

```
url-lookup-result-during-database-reload  deny
url-lookup-result-when-fatal-error        deny
```

When any of the above two parameters is set to “deny”, browsers are redirected and display an appropriate message. To overrule the default messages one can configure URLs that point to web pages to display own messages for fatal errors and when it is loading a database with the following two parameters.

```
redirect-fatal-error       "http://www.example.com/fatalerror.html"
redirect-loading-database  "http://www.example.com/loadingdb.html"
```

8.10.2 ufdbgclient

ufdbgclient is the glue between Squid and ufdbguardd. By default, when ufdbgclient cannot communicate with ufdbguardd, it allows all URLs, i.e. all URL filter requests of Squid are answered with a message indicating that the URL is allowed. The only possible alternative to allowing all URLs

is to block all URLs. To block all URLs when ufdbguardd is not running or there is a fatal communication error, ufdbgclient can be started with the "`-e deny`" option.

In case of a fatal error, ufdbgclient returns by default the URL http://cgibin.urlfilterdb.com/cgi-bin/URLblocked.cgi?category=fatal-error which displays an appropriate message. To overrule the default redirection URL for fatal errors, the `-E` option can be used which takes a valid URL as its argument, e.g. `-E http://example.com/error.html`

The `-e` and `-E` options must be specified in the configuration file of Squid, `squid.conf`. E.g.:

```
url_rewrite_program /var/sbin/ufdbgclient -e deny
```

8.11 Dynamic User-defined URL Categories

ufdbguardd support any number of user-defined categories of which *always-block* and *always-allow* are examples. Sites may use URL categories that change often and for the changes to become effective one would have to reload the whole configuration.

The Dynamic User-defined URL categories prevent the need for ufdbguard reloading the whole configuration by periodically checking if the URL category has been refreshed, and if so, only reloads the particular URL category. A user-defined URL category is made dynamic by adding the execdomainlist keyword followed by a command. For example:

```
category hotallow {
   domainlist     "hotallow/domains"
   execdomainlist "/local/scripts/updatehotallow.sh"
}
```

The `samples` directory of the sources has an example script called `execdomainlist.sh` which includes comments about what ufdbguard expects to happen.

The configuration parameter `refreshdomainlist` defines the interval for ufdbguard to execute the configured scripts. The default refresh period for execdomainlist is 15 minutes.

8.12 Extended Logging

ufdbGuard has 3 options for extended logging. The keyword `logblock` followed by `on` or `off` tells ufdbGuard whether to register blocked URLs in its logfile. The keyword `logall` followed by `on` or `off` tells ufdbGuard whether to register *all* URL verifications in its logfile. The keyword `logpass` followed by `on` or `off` tells ufdbGuard to log explicitly allowed URL categories.

`logpass` was introduced in v1.29 to log explicitly allowed URL categories. The most obvious example of such URL category is *always-allowed*. One can also define more explicitly allowed URL categories. For example, if a site allows webmail but likes to monitor the use of webmail, it can explicitly allow webmail in the ACL and use logpass to log its use. An ACL for this example looks like this:

```
acl {
   allSystems {
      pass always-allow !always-block !proxies !adult webmail any
   ...
```

Note: `logall on` requires more resources and has a slight performance impact on ufdbGuard.

8.13 Displaying URLs

URLs are displayed in the log file and in the message that says that a site is blocked. The administrator can control if the parameters of a URL are showed or not with the options `ufdb-show-url-details` and `ufdb-log-url-details` followed by `on` or `off`. The default value for both options is *off*. E.g.

```
ufdb-log-url-details on
```

8.14 Logfile Rotation

ufdbGuard rotates its logfile automatically whenever it grows beyond 200 MB. When the logfile is rotated, the file `ufdbguardd.log` to `ufdbguardd.log.1` and recreates `ufdbguardd.log`. A maximum of 8 log files are kept. Note that with the default size of 200 MB, hence the file system where ufdbGuard is installed needs 1.6 GB free space for the log files.

On receipt of the USR1 signal, ufdbguardd also rotates the logfile. The most convenient way to do this is to use `/etc/init.d/ufdb rotatelog`, which takes care of sending the USR1 signal to the right process.

The maximum logfile size is configurable with a parameter in the configuration file. The following line sets the maximum size to 50 MB.

```
max-logfile-size 50000000
```

8.15 Using Quotes

ufdbGuard has many reserved words that cannot be used as labels without using double quotes. In case that you have the need to use a source or category name that is identical to a reserved word, you may use it surrounded by double quotes. E.g.

```
category “aggressive” {

source “unix” {

pass ... !”aggressive” ...
```

Also file names, URL redirection strings, ACLs and parameters `dbhome` and `logdir` accept quoted parameters. E.g.

```
logdir “/usr/local/ufdbguard”
time “working-hours” {

category “aggressive” {
   domainlist     “aggressive/domains”
   expressionlist “aggressive/expressions”
   redirect       “...”

source “unix” {

pass ... !”aggressive” ...
```

8.16 Monitoring

ufdbGuard has two monitoring options: monitoring by email and monitoring by execution of an external command. It is recommended to use these features since an error always occurs unexpectedly and often needs a quick response.

ufdbGuard maintains a status of itself and whenever the status changes, it may send an email message to a configurable email address and/or execute a configurable external command. The external command can be any program or script. Usually it is a script that can send an appropriate command for a monitoring tool. By default, ufdbguardd does not send emails and does not execute external commands.

The status values used in the emails and scripts are *started*, *reloading*, *reloaded*, *fatal error, crash report uploaded, crash report NOT uploaded* and *terminated*.

In the rare event that ufdbGuard crashes, it calls the debugger gdb and produces a crash report which is very useful in finding and resolving bugs. By default ufdbGuard uploads any crash report that it finds when it is restarted. Crash reports reside in `/tmp` and the filename starts with "urlfilterdb.crashreport". Uploaded crash reports are renamed and the filename starts with "uploaded.urlfilterdb.crashreport". To prohibit the automatic upload of a crash report, use the following in the configuration file:

```
upload-crash-reports off
```

8.16.1 Monitoring by Email

The configuration parameters to enable monitoring by email are:

```
mail-server "hostname"
admin-email "email-address"
sender-email "email-address"
```

The mail server must accept SMTP connections on port 25. ufdbguardd uses the email address specified with `admin-email` as the recipient address. The `sender-email` is optional, if it is not specified, ufdbguardd uses the value of `admin-email` as the address of the sender.

The content of the email messages are:

```
ufdbGuard with pid <PID> on hostname has a new status: status
database status: <dbstat>
license status: <licinfo>
configuration file: <filename>
version: 1.33.1
```

The *status* can have one of the following values: virgin, started, terminated, reloading, reloaded or error. The *dbstat* is a text about the URL database and looks like this:

up to date
one or more tables are more than 4 days old. Check cron job for ufdbUpdate.
one or more tables are EXPIRED. Check licenses and cron job for ufdbUpdate.

The *licinfo* is a self-explanatory text about the license. It may contain a text with the words OK, warning or expired. Warnings are given for licenses that will expire in 2 months or less.

8.16.2 Monitoring by Command Execution

The configuration parameter to enable monitoring by command execution is:

```
external-status-command "path-to-executable"
```

The command must be specified including the full path, e.g. `/usr/local/bin/ufdbmon`. When executed, the program will receive the parameters `-s status -d dbstat -l licinfo`.

See the previous section for an explanation about *status*, *dbstat* and *licinfo*.

9 Performance Tuning

9.1 Web Proxy Infrastructure

For those organisations that use internal websites, it is recommended to configure and use PAC files that instruct browsers to connect directly to internal servers bypassing Squid and a URL filter. This offloads Squid and usually makes browsing more responsive.

Browsers itself have a cache. The default value for the size of the cache can be very large and it is recommended to set it to a reasonable value, e.g. 100 MB. Note that smaller browser caches usually make the browser faster due to less cache management. A very small cache may make the browser inefficient.

9.2 Upgrade C libraries

UNIX distributions that have a glibc 2.3.4[9] or older are recommended to be upgraded. This is due to a bug in the regular expression matching subroutines in the standard C library which is fixed in glibc 2.3.5. On the older systems, ufdbguardd will work flawlessly but at the expense of a performance penalty since only one single thread is allowed to perform lookups based on regular expressions at any point in time. Since ufdbGuard cannot distinguish version 2.3.4 from 2.3.5 but can distinguish version 2.3.x from 2.4.x, an upgrade to glibc version 2.4 or higher is recommended.

9.3 Squid performance

The following is recommended to improve the performance of Squid and ufdbGuard:

- use Linux 2.6 or later instead of earlier versions since the multithreading code in the libraries and kernel is much more efficient.
- use the `noatime` mount option for the file system with the cache. If reiserfs is used, also use the `notail` mount option. All options reduce the number of I/Os to the disk.
- use squid 3.5.20 or newer since older versions have too many known issues.
- make sure that Squid can use a proper number of open files. You may need to use the `ulimit` command and configure the kernel. There is no official guideline for a proper value for the number of open files, but you may find that 100+2.5*NUSERS with a minimum of 1024, is appropriate. Squid has a configuration parameter `max_filedescriptors` which can be added to `squid.conf`.
- use a moderately sized disk cache; a very large cache might have a slightly larger cache hit ratio, but the housekeeping of the cache requires more memory and CPU resources. Note that a larger number of cached objects also requires more physical memory for the index.
- use more than one disk for the disk cache, use only one cache directory per disk and do not stripe unless you are using an advanced disk array with internal striping.
- For optimal I/O performance, use the `aufs` cache directory type, but use `diskd` on FreeBSD. The latest versions of Squid support multithreading and rock store. You mileage may vary with these features.
- Squid does a lot of translations of hostnames to IP addresses and the translations are done by first trying to translate to an IPv6 address and if that does not work to translate to an IPv4 address. If the infrastructure does not have IPv6 routers, disabling IPv6 on the host where Squid

[9] consult your system documentation on how to retrieve version information about the glibc library

runs will give a performance gain because the IPv6 lookups and nameserver delays are prevented.

- Use a moderately sized memory cache. Allow enough memory for the kernel's file system cache and other processes. Remember that the configuration parameter `cache_mem` specifies only the amount of memory to be used for objects and the total process size is usually 2-3 times the amount of `cache_mem`. So, for a machine with 8 GB memory which only runs Squid, a good starting value for `cache_mem` is 1600 MB (Squid will occupy about 4.8 GB).
- Experiment with a memory-cache only (disable the disk cache) on large memory systems to get rid of the delays of I/O to disk. The benefit of not having disk I/O is usually bigger than the increased bandwidth usage caused by a reduced cache hit ratio. Note that this only works well with a suitable sized internet connection.
- Experiment with a cache for small objects only. Since up to 90% of all objects are smaller than 32 KB, Squid may be configured to use all memory and disk resources to cache small objects and not cache larger objects. Note that this only works well with a suitable sized internet connection.
- Hyperthreading is only beneficial for a limited set of applications and the performance gain with hyperthreading is different for each type of CPU. Intel Haswell and newer tend to perform well with hyperthreading since the cache bandwidth was doubled. But on older Intel CPUs, it is not recommended to use hyperthreading since it trashes CPU caches faster. If you are really keen on using hyperthreading then we suggest to measure its performance.
- do not forget to run "`squid -k rotate`" from `cron` as user `squid` because it also does important housekeeping of the cache. Perform the housekeeping on a moment when Squid is not used much, preferably in the middle of the night.
- visit http://wiki.squid-cache.org for more tips.

9.4 Linux performance

There are various sources for Linux system tuning and this section is not a replacement. This section contains recommendation for tuning Linux 2.6 and newer systems that runs Squid and ufdbguardd only.

9.4.1 Optimize System Memory Usage

Linux has various settings that control allocation of memory and how the system behaves when real memory gets scarce. Below are given the setting that gave satisfactory system behavior on our test systems.

Add to `/etc/sysctl.conf` or `/etc/sysctl.d/60-local.conf` the following lines:

```
# swappiness can have a value of between 0 and 100
# swappiness=0 tells the kernel to avoid swapping processes out of physical
# memory for as long as possible
# swappiness=100 tells the kernel to aggressively swap processes out of
# physical memory and move them to swap cache
# default: 60
# For an application server we use a less aggressive setting of 10-20.
# Use 15% for systems with 64 GB memory or more.
vm.swappiness=15

# For large memory systems, performance increases by tuning these:
# vm.dirty_background_ratio is the maximum percentage of ((Cache + Free) -
# Mapped) memory that can be dirty before it is written to disk by pdflush
```

```
# vm.dirty_ratio is the value that represents the percentage of MemTotal that
# can consume dirty pages before all processes must write dirty buffers back
# to disk
# vm.dirty_ratio=20               # default is 40
# vm.dirty_background_ratio=5     # default is 10

# VFS cache pressure:
# default: 100
# With values lower than 100, the data cache is reduced more and the
# inode cache preserved
vm.vfs_cache_pressure=50

# Overcommit or not memory allocations:
# This manual previously recommended vm.overcommit_memory=2 but this value
# works only well with old Linux kernels.
# With Linux 2.6.x and higher it is recommended to use vm.overcommit_memory=0
vm.overcommit_memory=0
```

NOTE: the memory allocator in the standard library on Linux has been reimplemented and has an impact on the above recommendations. Starting with glibc 2.10, the memory allocator (malloc) uses on 64bit systems many memory segments of 64 MB. There is nothing wrong with the implementation except that for multithreaded applications the virtual process size is much larger than the real memory that the process uses. In version 1.32 the effect of the new memory allocator is reduced by setting the environment variable `MALLOC_ARENA_MAX=20` in `/etc/sysconfig/ufdbguard`. This effectively reduces the virtual size of ufdbguardd from 5 GB to 1,25 GB. The real memory that ufdbguardd needs is still around 400 MB on 64bit systems. The change of implementation of the memory allocator implies that the previous recommendation for `vm.overcommit_memory` is no longer valid and for glibc 2.10 and higher we recommend the default setting `vm.overcommit_memory=0`.

Run the following command to make the new configuration active:

```
# sysctl -p /etc/sysctl.conf
```

9.4.2 Optimize TCP Connections

There is a relatively new parameter to tune TCP supported by Linux 3.7 and newer: *TCP fastopen.* Add to `/etc/sysctl.conf` or `/etc/sysctl.d/60-local.conf` the following lines:

```
# we want TCP fastopen
net.ipv4.tcp_fastopen = 0x3
net.ipv4.tcp_fastopen_key = <unique-string-of-hex-digits>
```

The unique hexdigit string has 4 substrings of 8 hexadecimal digits, separated by a minus sign, e.g. `f7792304-bbf0895f-46f4a2e2-dbd23e58`.

There are many other tuning parameters for TCP connections and many websites contain advise on how to change them, so they are not described here.

9.4.3 Bind ufdbguardd to a Fixed Set of Processors

For large sites that have both Squid and ufdbGuard running on a system with Linux 2.6 and with 2 or more real CPU cores, additional performance can be gained by optimizing the CPU cache efficiency. By separating the squid and ufdbguardd processes over different CPUs with their own memory caches, the individual CPU caches are used in an optimal way.

Squid is a very CPU intensive application and in the following examples we reserve CPU 0 for Squid. By configuring ufdbGuard to use the other CPUs, the kernel will move squid automagically to CPU 0. The remaining CPUs can be used by ufdbguardd and the process is bound to these remaining CPUs by including the `cpus` keyword in the ufdbGuard.conf file.

system	*for optimal performance, use*
2 simple CPUs	`cpus 1`
2 simple CPUs without hyperthreading	`cpus 1`
2 simple CPUs with hyperthreading	`cpus 2,3`
4 simple CPUs	`cpus 2,3`
1 dual core CPU	`cpus 1`
2 dual core CPUs	`cpus 2,3`
1 quad core CPU	`cpus 2,3`
2 quad core CPUs	`cpus 4,5,6,7`

UfdbGuard and Squid use large amounts of memory and therefore the CPU caches are flushed many times. Always try to separate Squid from ufdbGuard and do not let ufdbGuard use more than 50% of the CPU cores. It is also very rare that ufdbGuard needs more than 2 cores. When in doubt, ask the support desk of URLfilterDB for advice.

ufdbGuard uses a large amount of memory and performance tests showed that the performance of ufdbGuard degrades if cores on different NUMA nodes are used, so always try to run ufdbGuard on the cores of a single CPU.

9.5 Solaris Performance

Use a multi-core system and use a processor set of 2 or more CPUs for Squid and ufdbGuard, and disable interrupts for the CPUs of this processor set. On systems with many cores, use 2 or more cores for Squid and 2 or more cores for ufdbGuard.

10 Analysis of User Behavior

To learn more about which types of websites are visited by the users and how much bandwidth they use, the tool `ufdbAnalyse` can assist. See section 10.3 for more information.

10.1 Basic Analysis Tools

The ufdbguard software suite has a number of utilities which scan the log files of ufdbguardd. The analysis tools are command-line driven and meant to be used by the administrator of ufdbguard and provide a simple means to analyse the most common questions about top usage and user investigation.

NOTE: Analysis of log files is slow since (very) large log files are processed.

It may take 50 seconds to process 400 MB on a common server.

10.1.1 Most Frequently Used URLs

The `ufdb_top_urls` utility produces a list of most frequently used URLs. See `man ufdb_top_urls` for details.

10.1.2 Most Frequent Users

The `ufdb_top_users` utility produces a list of the most frequent users. See `man ufdb_top_users` for details.

10.1.3 Analyse URLs

The `ufdb_analyse_urls` utility produces a list of all visits to a set of URLs. See `man ufdb_analyse_urls` for details.

10.1.4 Analyse Users

The `ufdb_analyse_users` utility produces a list of all visits of a set of users. See `man ufdb_analyse_users` for details.

10.2 How to get More Detailed Reports

To get more detailed reports from the basic analysis tools, ufdbguard may need a more advanced configuration. To find out which sites a user visits, the option `logall` must be `on` which logs blocked *and* passed URLs.

Since most ACLs only include the blocked categories, ufdbguardd logs a passed URL with the category *any*. So, for example if the ACL contains

```
pass !proxies !security !adult any
```

and a user browses to the site `www.cnn.com`, the log file contains a line with

```
... PASS johndoe 10.1.1.1 regusers any http://www.cnn.com
```

Note that the URL is logged as if it belongs to the pseudo-category *any* while in fact it is part of the category *news*. Ufdbguardd never compared the URL against the URL category *news* and therefor is not able to report that the actual category is *news*. To resolve this, the ACL must also have the URL categories which are not blocked. Hence, if the ACL is as follows:

```
pass !proxies !security !adult ads entertain sports news audiovideo ...
any
```

the logfile contains the URL category as one would expect for the URLs:

```
... PASS johndoe 10.1.1.1 regusers news http://www.cnn.com
... PASS johndoe 10.1.1.1 regusers entertain http://www.cnn.com/showbiz
```

10.3 Offline Analysis

`ufdbAnalyse` reads one or more Squid log files, strips all non-relevant and personal information, and uploads the URLs to the servers of URLfilterDB for analysis. The stripped result file has no details about persons, passwords, session parameters or times, and the processing is done in compliance with our privacy policy (see section 13).

The support desk will send an email with the outcome of the analysis. The output contains for each URL category the #URLs, percentage of URLs, Kbytes and percentage of Kbytes. An example of the output is below.

```
The log files contain 51479 lines of which 2250 contain HTTP error codes.
49229 lines were processed for URLfilterDB categories.

Category                    #URLs     %         KB     %
Adult                        1935   4.0      78204  15.7
Audio & video                 128   0.3      58207  11.7
Advertisements               2449   5.0       2296   0.5
Chat                           18   0.1         56   0.1
Dating & Personals            190   0.4       4425   0.9
Drugs                           0   0.0          0   0.0
Entertainment                6379  13.0      59174  11.9
External Applications           0   0.0          0   0.0
Finance & Investment         1623   3.3      12780   2.6
Forums                        152   0.3        817   0.2
Gambling                      119   0.3        482   0.1
Games                         413   0.9       2565   0.6
Illegal                         0   0.0          0   0.0
Jobs                          357   0.8       3042   0.6
News                         2055   4.2      22224   4.5
Weblogs & private sites       882   1.8       5902   1.2
Peer-to-peer                    0   0.0          0   0.0
Web Proxies                   148   0.3        812   0.2
Religion                      119   0.3        816   0.2
Security violations             0   0.0          0   0.0
Shops                        3733   7.6      32542   6.5
Social Networks                 0   0.0          0   0.0
Sports                        813   1.7       6938   1.4
Toolbars                       23   0.1        290   0.1
Travel                       2586   5.3      21847   4.4
Hacking & warez                 9   0.1         89   0.1
Violence & hate                 0   0.0          0   0.0
Web-based email               105   0.3        809   0.2
Own domain                   1556   3.2       3249   0.7
Allowed sites               23556  47.6     183274  36.5
```

The `ufdbAnalyse` command uses options to specify the logfile, the own domain, your email address and your full name. The `-l` option may occur up to 64 times to produce a single aggregate report:

```
$ cd /usr/local/ufdbguard/bin
$ ufdbAnalyse -l .../access.log -d example.com -e "me@example.com" -n "John Doe"
```

10.4 Statistics

UfdbGuard prints in its log file total numbers of URL lookups, number of blocked URLs, numbers of tunnels detected, number of enforced SafeSearches, number of times Youtube edufilter was applied, and the total number of clients, and prints basic statistics about its rules. The statistics about the rules contain for each category and for each source the number of times it was matched. The statistics are printed every time that ufdbGuard receives a signal to reload the configuration (`ufdbUpdate` sends such signal) or every 48 hours, whichever event comes first.

Example of statistics:

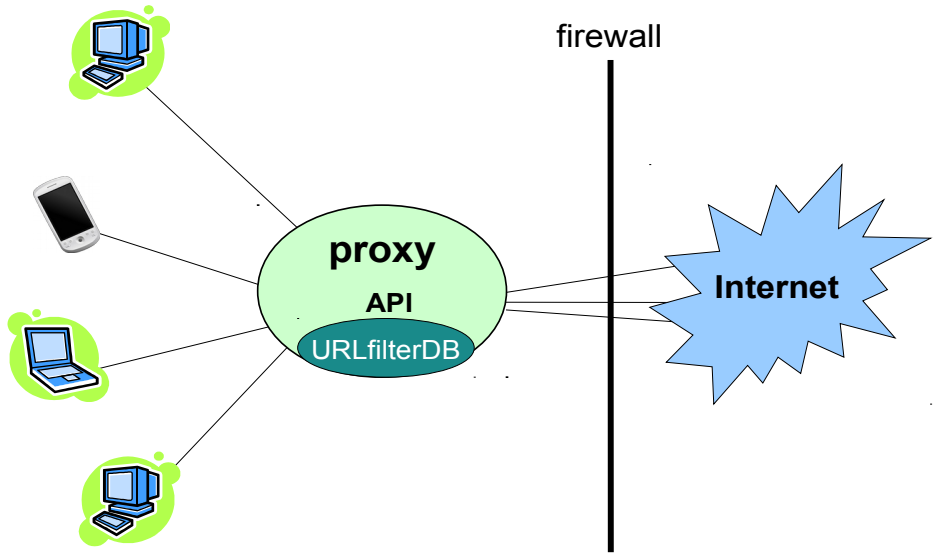
```
$ grep " statistics: " ufdbguardd.log
2016-06-20 22:01:09 [11845] statistics: 2407 URL lookups (0 https). 58 URLs blocked.
   0 tunnels detected. 0 safe searches. 0 Youtube edufilter. 0 uncategorised URLs.
   6 clients.
2016-06-20 22:01:09 [11845] statistics: category security was blocked 0 times
2016-06-20 22:01:09 [11845] statistics: category entertainment was blocked 31 times
2016-06-20 22:01:09 [11845] statistics: category proxies was blocked 2 times
2016-06-20 22:01:09 [11845] statistics: category adult was blocked 5 times
```

```
2016-06-20 22:01:09 [11845] statistics: category games was blocked 18 times
2016-06-20 22:01:09 [11845] statistics: category p2p was blocked 2 times
2016-06-20 22:01:09 [11845] statistics: source privileged_users was blocked 0 times
2016-06-20 22:01:09 [11845] statistics: source allSystems was blocked 58 times
2016-06-20 22:01:09 [11845] statistics: acl privileged_users-within-one-hour:
   !security:0 !entertainment:0 !adult:0 !games:0 all:0
2016-06-20 22:01:09 [11845] statistics: acl privileged_users-else:  !security:0
   !proxies:0 all:0
2016-06-20 22:01:09 [11845] statistics: acl allSystems-within-working-hours:
   !security:0 !proxies:2 !adult:5 games:18 p2p:2 !entertainment:31 !safesearch:0
all:0
2016-06-20 22:01:09 [11845] statistics: acl allSystems-else:  !security:0  all:0
2016-06-20 22:01:09 [11845] statistics: acl default:  none:0
```

11 Integration with 3rd Party Products

ufdbGuard also has an API that allows it to be easily integrated with 3rd party products. With the use of the API, any application has access to the core functions of ufdbGuard and can perform URL verifications from any program written in C with a high performance: 75,000 URL verifications per second on a single core of an Intel Xeon CPU E5-1620 v2 (introduced by Intel in 2013). The benchmark was performed with 100,000 URLs matching against all URL categories. Using 4 threads on the same CPU, the performance is 275,000 URL verifications per second.

The following diagram gives a simplified overview of how a 3rd party product could operate. In this example, a proxy is linked with the URLfilterDB library for extremely fast URL verifications due to the lack of any inter-process communication overhead.



the URLfilterDB library is part of the web proxy

Contact the support desk for additional information and licensing for 3rd parties using the URL database.

12 Frequently Asked Questions

ufdbGuard does not block

The most common reasons are either a configuration error (check `ufdbguardd.log`) or that the daemon is not active (verify with `ps -ef | grep ufdbguardd`).

Another common reason is that your trial license has expired.

Verify the log file for warning messages. The most serious errors have a 5-star indicator (`*****`).

ufdbGuard filters HTTP but not HTTPS

ufdbGuard filters everything that Squid asks to filter. The most common mistake is to *not use Squid for HTTPS*. Perform a test: browse to https://www.google.com and verify that the `access.log` file has a line with "`CONNECT www.google.com:443`".

A less common mistake is an error with the Squid configuration parameter `url_rewrite_access`.

I want to allow access to site *myfavoritesite.com*

See section 7.1. Do not forget to run `ufdbGenTable` after each change.

Can I define or add new categories ?

Yes. The ufdbGuard software suite includes a utility to convert a plain text file into a URLfilterDB database file.

Can I use ufdbGuard with a free URL database?

The ufdbGuard software is free and can be used with free URL databases if the database is in .ufdb format or in an ASCII format. The tools `ufdbGenTable` and `ufdbConvertDB` convert ASCII formatted (flat file) databases to the `.ufdb` format.

Which URLs are in the database?

URLfilterDB does not disclose the content of their database. You are encouraged to test the URL database yourself to find out its effectiveness for your user base. Please refer to section 12.1 for a more detailed description of what you may find in each URL category.

What does it cost?

The URL filter software is free. Prices for a subscription to the URL database are on the website and you can request a quote online. The prices are in EURO and we use a fair exchange rate for countries that prefer to be billed in US dollars. Non-profit and educational institutions receive a 40% discount. Some countries may have lower prices.

Any other question

You may contact the support staff at support@urlfilterdb.com to ask any other question. To get the best answers, always provide as much details as possible and supply a relevant fragment of the log files.

12.1 URL Categories

The URL database of URLfilterDB uses the following URL categories.

Ads

Websites with advertisements, user behavior monitors, traffic trackers and web page counters.

P2P

P2P stands for point-to-point file sharing. The P2P category contains websites that can be used directly or indirectly to upload, download and share files. Most P2P sites have copies of movies, adult content, warez and entertainment, and many sites have content that violates copyright.

Proxies

Sites that can be used to download content of other sites, URL rewriting sites and VPNs. Proxies are commonly used in an attempt to circumvent a URL filter and should always be blocked. A subcategory exists for `teamviewer`. Teamviewer is a popular remote access tool that gives full control to any other PC (including a home PC) and therefore is a proxy. Another subcategory is `translators` which translate web pages and may be used to circumvent the content filter.

Adult

Websites suitable for adults only, including but not limited to pornographic content and obscene content.

Warez

Websites with illegal software, illegal software codes, hacker's sites, warez and cracks.

Malware

Websites with malicious software.

Toolbars

Websites for toolbars of browsers. A toolbar is an extension to a web browser that may violate your privacy or make private files public.

Illegal

Websites explaining how to perform Illegal activities.

Violence

Websites about violent behavior and aggressive sales of arms.

Gambling

Websites offering gambling opportunities.

Drugs

Websites about hard drugs and soft drugs.

Webmail

Email accessible with a web browser. Webmail of business sites is not included. Webmail of ISPs is included in this category.

Dating

Websites about love, dating, romantic poetry, and friendship.

Chat

Websites to use IRC and chat. Subcategories exist for AIM, Ebuddy, ICQ, Facebook Chat, Google Talk, MSN Messenger, Oovoo, Skype, Telegram, Whatsapp and Yahoo Chat.

Forum

Websites where people exchange non-business information in a forum.

Private

Blogs and sites of private persons.

Web Radio

Radio and collections of music.

Web TV

TV and collections of video including Youtube.

Dailymotion

Dailymotion videos.

Vimeo

Vimeo videos.

Youtube

Youtube videos.

Audio-Video

Audio and video streams. Many entertainment sites are both in the category entertainment and in audio-video (e.g. www.youtube.com).

Sports

Websites related to sports including sports sections of news sites, fans of sports, sites about actively doing a sport.

Finance

Websites of banks, insurance companies, stock markets and stock brokers.

Jobs

Websites about and for job applications.

Games

Websites to play games and information about gaming.

Entertainment

Entertainment, lifestyle, hobby, arts, museums, food, fashion, electronic cards, magazines, horoscopes, desktop wallpapers, clip art, photos, portals, events, fan sites, baby-related, child sites, picture sharing and other sites for interest of private persons that are not related to business.

Religion

Websites with religious content.

Shops

Websites with shops, price comparisons, and auctions aimed at consumers (b2b is excluded).

Travel

Websites about travel agencies, airliners, tourism sites, hotels, holiday resorts.

News

Websites providing news and opinions.

External Applications

Free web-based document editors, spreadsheet applications, desktops, groupware, etc. where "internal" documents can be stored on external servers.
Subcategories exist for CitrixOnline, Dropbox, iCloud and Telegram.

Search Engines

Search Engines.

Security

This is an administrative category to be able to enforce strict HTTPS usage options (see section 5.5).

Safe

Websites with content that is considered safe to access, including but not limited to antivirus sites, SSL/TLS certificate verification, and some popular webfonts.

Social Networks

Sites that focuses on building and reflecting of social networks or social relations among people. Subcategories exist for Badoo, Facebook, Google+ and Twitter.

Dynamic domains

Computer systems without a static address use dynamic addresses which are usually managed by dynamic DNS servers (DDNS servers). DDNS is often used to gain remote access to computer systems at home and can also be used as proxies.

Mozilla

Mozilla Firefox uses a range of IP addresses to connect to using the HTTPS protocol and may need to be whitelisted with this URL category.

Chrome

The Chrome browser does at startup 3 queries to random non-existent webservers. This URL category matches those queries.

Checked

URLs that are verified by URLfilterDB not to be part of any other category and hence always are allowed by the URL filter. This category contains business sites, governmental sites and useful sites for the general public. This administrative category is used by the URL filter to track uncategorized URLs and users do not have to configure this category.

The URL database is used to block web content and not to classify web content. So, a website that has a business use, is always part of the category "checked" and will never be blocked. For example, access

to a website to sell medical equipment for hospitals is always allowed because it is not part of the category “shops”. Also governmental sites and all sites for basic human needs like electricity, water and housing are never blocked.

The nature of the content is more important than the strict definition, so an advertisement with a nude person is classified as adult rather than advertisement, and a forum about games is classified as games rather than a forum.

The general impression is also taken into account when a site is categorized. For example, most buyers at `ebay.com` are consumers rather than business users and therefore `ebay.com` is considered a shop for consumers and part of the shops category.

URLs may be part of two or more categories, e.g. `www.usatoday.com` is news while `www.usatoday.com/sport` is both news and sports. If a configuration blocks sports and allows news, it may occur that parts of websites are accessible while other parts are not.

13 Privacy Policy

The privacy policy of URLfilterDB is stated on the website: www.urlfilterdb.com/privacystatement.html.

14 More Information

More information can be found on the internet at the following addresses.

URLfilterDB	www.urlfilterdb.com
Squid	www.squid-cache.org