ufdbGuard Query Server v1.1

for clients with few resources

Reference Manual

Table of Contents

1 Introduction

Devices with limited resources like smartphones and small internet appliances cannot hold the URL database in memory like larger servers do. To support these limited devices, the *ufdbGuard Query Server* is installed on a server and responds to URL categorisation queries of the limited devices.

Value-added resellers may license the ufdbGuard Query Server and use it on their servers to serve clients like smartphones and small appliances. URLfilterDB provides the software for the server and the URL database while partners are expected to develop software for the devices.

To be released in Q3 2019: for devices that use the Squid proxy the product *ufdbGuard QS Client* can be used, which is a product derived from *ufdbGuard for Squid* and adapted to query the ufdbGuard Query Server and cache the query results.

2 Architecture

The Query Server holds the URL database in memory and serves requests from a large number of clients. Clients such as smartphones and small appliances can connect to the server, authenticate itself and send URL categorisation queries to the server.

The network latency between the clients and the server is expected to be over 10 ms. This implies that all clients must have a cache of URL lookups to prevent noticeable latencies by end users. Latency is important and servers should be located in such way that latencies are kept to a minimum.

The Query Server is a multithreaded process where the number of threads typically is equal to the number of available hardware threads and a is designed that a single daemon process can serve up to 10 million clients.

A load balancer must be used for redundancy and scalability. At this moment the load balancer must balance at the TCP level. A future version of the Query Server may support a load balancer that acts as the TLS endpoint.

Performance tests show that a server with 8 threads can process 100,000+ simultaneous queries per second and since clients have a cache and are mostly not sending queries, it is expected that a server with 8 threads can serve between 2 and 10 million clients. For systems with more than 16 threads, it is recommended to have multiple Query Server daemons on the same server or use multiple servers to increase redundancy. Since clients have a cache, they have an average number of queries that is estimated at one query per 100-500 seconds. The hard-coded maximum number of clients that a Query Server can serve is currently 10 million. Contact the support desk if you need a daemon capable of serving more clients.

The ufdbGuard Query Server is capable of loading a new URL database while serving clients uninterruptedly[1].

The communication channel between the app and the query server is a socket with TLS and TCP keepalive. The server must use a certificate that is signed by the CA certificate with Issuer CN=URLFILTERDB-QS-CA. Each vendor receives 3 certificate files, one certificate for the server, one private key for the server and the CA certificate of URLFILTERDB-QS-CA. The server certificate is signed by the CA certificate URLFILTERDB-QS-CA and the client must verify this. All certificates expire in 2035.

[1] in very rare cases a client may receive a TRYLATER response for a QUERY.

The sockets between the query server and clients are a resource on the server that cannot be kept open and unused for a long time. The server closes sockets that are unused for 45 minutes or more. In case of server overload the server may close sockets that are unused for 22 minutes.

The ufdbGuard Query Server is only supported for servers running RHEL/CentOS 7 or Ubuntu 18.04. Other Linux distros may be supported in the future if partners request it. The systems must be tuned for a large number of open files, and enough TCP resources for millions of clients. The network interfaces must be tuned for a large number of buffers.

2.1 Communication Protocol Overview

The communication protocol between the clients and the server is a line-based protocol that includes commands for

- exchange of software versions, app id, system id and shared secret
- query the list of URL categories
- query a domain, server answers with a list of categories (may have 0, 1 or more entries).

2.2 URL Database

2.2.1 The Full URL database

The full URL database contains database tables which have URLs that can be simple URLs which only hold a FQDN without any leading `www.`, e.g. `example.com`, and can be URLs consisting of a FQDN + path + optional parameters, e.g. `example.com/foo/cars.html` or `example.com/cgibin/query.cgi?type=suv`.

URLs may be in one or more categories. For example:

`www.cnn.com`	news
`us.cnn.com/entertainment`	news and entertain
`www.srbijadanas.com/zena/sex`	news and adult
`piratebay.tel`	adult, games and qmovies
`www.sex.com`	adult
`blogspot.com`	private
`zporn.blogspot.com`	adult and private
`actiongames37.blogspot.com`	games and private

Note that `blogspot.com` is only in the private category.

The full URL database can only be used if clients do all URL lookups with a server that has the full URL database and hence can only be used in environments with latencies less than 1 ms.

2.2.2 The Domain-only URL Database

The *domain-only* URL database is based on the full URL database where all URLs with a path or parameters have been removed. This simplified URL database is used by the ufdbGuard Query Server since the clients have insufficient resources and managing full URLs instead of domain names requires a relatively large amount of memory.

This implies that URLs which have a specific category based on the URL path or URL parameter do not have all categories returned by a query. The following table shows a comparison between querying the full URL database and the domain-only database.

URL	categories full DB	categories domain-only DB
`www.cnn.com`	news	news
`us.cnn.com/entertainment`	news ***and*** entertain	news
`www.srbijadanas.com/zena/sex`	news ***and*** adult	news
`piratebay.tel`	adult, games and qmovies	adult, games and qmovies
`www.sex.com`	adult	adult
`blogspot.com`	private	private
`zporn.blogspot.com`	adult and private	adult and private
`actiongames37.blogspot.com`	games and private	games and private

2.2.3 Maintenance

The URL databases are updated each hour. It is recommended to refresh a URL database at least once and at most 4 times per 24 hours. It is suggested to refresh it every day between 4 AM and 6 AM local time.

2.3 Uncategorised URLs

The ufdbGuard Query Server collects all URLs which are not yet in the full URL database. The server uploads the uncategorised URLs to the servers of URLfilterDB where they will be analyzed and categorised for inclusion in the URL database.

URLs are always stripped from username:password and parameters before uploading to URLfilterDB. See the privacy policy at https://www.urlfilterdb.com/privacystatement.html for more details.

2.4 Client Identification

The solution must of course have a protection mechanism against abuse and therefore all clients must be identified and authenticated. The only function of the authentication is to counteract abuse so there is not a unique password for each user but the client merely sends a system identification ID and a shared secret. The server will have a blacklist that blocks apps based on client IP, system identification ID and shared secret. The ufdbGuard Query Server has no automatic abuse detection and the blacklist is maintained in a configuration file.

After the TLS socket between the client and server is created, the client must authenticate itself and send a system identification ID and a shared secret. The system identification ID is 64 bytes long and a unique ID. It is suggested that the identification ID is generated when the client app is installed on the device and is the result of the SHA256 hash of the concatenation of random data, the application ID and an ID that identifies the system (a MAC address or other hardware ID). The SHA256 hash is an ID which cannot be traced back to a person or device.

The shared secret has 16 bytes. The Query Server has a configurable list of maximum 10 shared secrets. It is recommended that the shared secret is updated each 6 months and that apps receive new shared secrets with app updates and then to stop using the older shared secret.

NOTE: The blacklist feature is not yet implemented in version 1.1 but the ufdbGuard Query Server demands that the clients send a correct shared secret and send a system identification ID.

3 Client-Server Protocol Definition

The client opens a TLS-encrypted socket to the query server where both end use a certificate to identify themselves. The client must know the FQDN and IP of the ufdbGuard Query server. In most environments the FQDN will point to a load balancer. The client must verify the certificate of the server and the certificate chain must be signed by URLFILTERDB-QS-CA so it must have its public key.

The client can send the following commands:

command syntax	explanation
CLIENT <VERSION> <APP-ID> <SYSTEM-ID> <SECRET> “\n”	The app sends the CLIENT command to the server immediately after it opened the socket to the query server. If a client does not send the CLIENT command within 30 seconds, the connection will be closed by the server. The server answers with SERVER or ERROR: “SERVER” <VERSION> <IDENTIFIER> “\n” “ERROR” DOUBLEQUOTE <MESSAGE> DOUBLEQUOTE “\n” The version format is always major.minor. Clients and servers with the same major number are always compatible. Generally servers will be upwards compatible so when the server major is one greater than the client's major, they are compatible. If not, the server will reply with an error. The APP-ID has a maximum of 64 bytes and must contain a short form or abbreviation of the vendor that implements the client followed by a dash followed by a short identifier that identifier the client application and its version followed by the application version, i.e. ABCNET-smartfilter-3.2. The SYSTEM-ID is the 64-bytes system identification ID as described in section 2.4. The SECRET is a 16-byte shared secret as described in section 2.4.
LISTCATEGORIES “\n”	The app sends the LISTCATEGORIES command to retrieve a list of all the categories known to the server. The server replies with a single line with the names of all categories separated by spaces and terminated with a newline: <category-*1*> space ... <category-*n*> “\n”

command syntax	explanation
QUERY <DOMAIN> “\n”	Query the server for the set of categories that the <DOMAIN> belongs to. The server replies with a single line that contains one of the following replies: FOUND, NOTFOUND, TRYLATER, ERROR. When one or more categories were found in the URL database, the set of categories is sent with: “FOUND” space <category-*1*> space ... <category-*n*> “\n” The number of categories can be 1 or more. When no category was found, the server replies with: “NOTFOUND” “\n” When the server is in a state that it cannot answer the query, it may send “TRYLATER” “\n” When there is an error condition, the server replies with “ERROR \”<MESSAGE>\” “\n”
QUIT “\n”	Client terminates. The server replies with “BYE” “\n”. The server does not close the socket but waits until the client closes it.

The server does not send commands. The server can close the communication socket when not in use for some time. Currently this timeout is set at 45 minutes but may change in a future version. The server may close the connection sooner if it wants to so do maintenance or has a high load. The client must be able to cope with this.

4 Clients

The client and server communicate over an encrypted socket that is encrypted with TLS v1.2 or higher and uses on the server a certificate chain (CA + server certificate) and on the client a certificate that is signed by the CA. The client must verify that the certificate chain of the server is correct and that the server certificate is signed by the CA with CN=URLFILTERDB-QS-CA.

The ufdbGuard Query Server may terminate and close the communication socket. This is done to free up resources and may also be used for server maintenance. The client software is *not allowed* to circumvent this mechanism with dummy queries and must accept that the socket may be closed by the server. It is more efficient for a client to send a QUIT command and close the socket itself before the server closes it to prevent that the client, after a period of not using the socket, receives an error and must open a new query socket.

For optimal efficiency and lowest latency clients must send each request with a single call to `SSL_write`, `gnutls_record_send` or equivalent function so that the TLS protocol sends a single record for each command. The Query Server does not support pipelining, so the client must send exactly one command and process the reply before sending another command to the server.

URLfilterDB does not provide software for the client other than `qstest0` that simulates a client. `qstest0` can be used to simulate one or more clients and can be used to test the Query Server and to perform volume tests.

The ufdbGuard Query Server may send an error reply to a client. It is suggested to show the full error to the end user.

4.1 Domain Queries

The client that uses the ufdbGuard Query Server is expected to extract the domain name from a URL and to simplify it. The simplification removes any optional protocol, username:password, port number, URL path and URL parameters from the full URL. So the URL
`http://john:secret@www.example.com:8088/cgi-bin/query.cgi?foo=bar`
is reduced to
`www.example.com`

A client uses the previously opened communication socket and sends a QUERY command with the domain name to the server. The server replies with a list of categories. The list may have 0 (NOTFOUND), 1 or more categories. In case that there are zero categories, the domain was not found in the URL database and is considered to be 'uncategorised'.

In rare cases the server may respond with TRYLATER which indicates that it temporarily cannot answer queries. The TRYLATER answer is only sent when the ufdbGuard Query Server is switching between an old and a new URL database which happens in a split-second. When a client receives TRYLATER it is suggested to treat it as NOTFOUND and to cache the result for at most 15 seconds.

Note that the result of a subdomain may be different than the result for a domain. For example, the server may return the following.

Query	Result
com	NOTFOUND
www.example1.com	FOUND news
games.example1.com	FOUND games news

The order of categories is not defined so `FOUND games news` is equal to `FOUND news games`.

4.2 Domain Query Cache

Since there is a latency between the client and the server, the client must have a query cache. The following table has suggested query cache sizes.

client type	minimum size	maximum size
smartphone	350	2,500
appliance	500	10,000

It is suggested that the client query cache is saved and restored when a client restarts to ensure that the the end user has a populated cache and low latencies whenever a device or the filter software is restarted.

The URL database of the ufdbGuard Query Server changes from time to time and the classification of cached domains may change. For example, a previously uncategorised domain may have a category, or a domain that was categorised in category X may have changed to category Y. This happens more often than one may anticipate and therefore it is required to expire cache entries at most in 24 hours.

To prevent a cache renewal storm it is suggested that each new cache entry has an expiration of 16+random(0,2) hours. The random(0,2) hours is added to prevent that a large number of domains must be queried at once when a user goes to his favorite complex website that uses 50+ domains.

It is suggested that when a cache entry expires, the client does not remove it from the cache but sends a new query to the server to have an up-to-date cache entry for a frequently used domain name.

The architect of the client software may even consider to preload caches when it knows it is a good time to do so. For example, if the software knows that the user of a smartphone becomes active every day at 6:45 AM, it may expire all domain names in the cache and re-query them at 6:30 AM. This may cause a peak load on the servers so try to spread the load and use different times to re-query on different clients.

5 Query Server Installation

5.1 Server Requirements

The minimal server must have at least 64 GB memory and a CPU with at least 12 threads. The server must have a minimum of 20 GB free space in the root file system. The server must have a high performance network interface. The server must be tuned for high volume TCP traffic and the limits for number of open files must be increased and set higher than the settings of the Query Server.

5.2 Files and Directories

The init script is `/etc/init.d/ufdbqs` and other files are installed in `/etc/cron.d`, `/etc/ufdbqs.d`, `/usr/sbin`, `/etc/defaults/ufdbqs` (Ubuntu and Debian) or `/etc/sysconfig/ufdbqs` (RHEL and CentOS) and `/var/ufdbqs` and subdirectories. The configuration file of the Query Server daemon is `/etc/ufdbqs.d/ufdbqsd.conf` and contains references to `dbhome` (the top directory for the URL database), `logdir` (the directory for log files), certificate and DH parameter files.

5.3 Installation

Packages for RHEL/CentOS 7 and Ubuntu 18.04 LTS can be downloaded from a link provided by the support desk.

Install the package in the usual way with `yum` or `dpkg`. On the first install Diffie-Hellman parameters are generated and stored in the file `/etc/ufdbqs.d/dh2048.pem` which can take a minute, so be patient.

The Query Server needs certificates which are created by URLfilterDB and signed by Issuer URLFILTERDB-QS-CA. The support desk will send partners a tar file with the server certificate, server private key file and the CA certificate. The CA certificate is only needed by the client application to enable it to verify the server certificate.

The tar file with the certificates and key must be unpacked in `/etc/ufdbqs.d` and contains the files pa_*partner*.crt, pa_*partner*.key and `ufdbqs_ca.crt`.

The last step of the installation is the download of the URL database but this can only be done after the configuration step in section 5.4. Use the `ufdbqs-update` script with the `-v` (verbose) flag to download the URL database. The `-v` flag is only required on the first download or when troubleshooting.

Verify the time of the crontab job to refresh the URL database (see section 6.2).

After the database download the ufdbGuard Query Server daemon can be started:

```
systemctl start ufdbqsd
```

5.4 Configuration

5.4.1 System configuration file

The support desk will send you a username and password for the downloads. Edit /etc/defaults/ufdbqs (Ubuntu and Debian) or /etc/sysconfig/ufdbqs (RHEL and CentOS) and assign the username and password to the variables DOWNLOAD_USER and DOWNLOAD_PASSWORD, e.g.

```
DOWNLOAD_USER="qs00000"
DOWNLOAD_PASSWORD="xhH2ss2dhJkL"
```

5.4.2 Query Server configuration

The Query Server configuration file is /ufdbqs.dufdbqsd.conf which is installed when the package is installed.

The server-ip may be set to 127.0.0.1 for testing purposes but usually must be changed to an appropriate IP address where clients or a load balancer can connect to.

The default port where the Query Server listens to is 1567 and may be changed with port followed by a number, i.e.

```
port 1567
```

The server-cert and private-key are preset to pa_test.crt and pa_test.key. After installation of the tar file with the partner-specific certificates, the server-cert and private-key must be updated to the file names of the received certificate and key files. The server needs for TLS sessions that use Diffie-Hellman key exchange a file with DH parameters. The 2048-bit DH parameters are generated when the package is installed and are therefore unique for the server where the ufdbqs package is installed. The parameter dh-params-file is preset to /etc/ufdbqs.d/dh2048.pem and usually does not need to be changed.

Each partner manages its own shared secrets that are used to authenticate clients apps. The secrets are 16 bytes and shared by all installed apps so there is no individual password for a user or app. The Query Server supports up to 10 shared secrets so that shared secrets can be rotated where newer versions of an app use new secrets and old versions of apps can be phased out or forced to upgrade to get a more recent app with a more recent and valid shared secret. Set at least one 16-byte shared secret, e.g.

```
secrets { “0123456789abcdef” “abcdef0123456789” }
```

On a modern x86-64 CPU with 12 threads the Query Server can process 100,000+ queries per second (see also section 2). The maximum number of clients that the server will accept is configured with max-connections. The minimum value is 2048 and the maximum value is 10,000,000, e.g.

```
max-connections 250000
```

The number of threads that is used can be configured with server-threads. The minimum value is 8 and the maximum value is 256, e.g.

```
server-threads 12
```

The directory where log files are stored is preset to /var/ufdbqs/log and may be changed with logdir followed by a directory name, e.g.

```
logdir “/local/ufdbqs/log”
```

Log files have a maximum size and are rotated when they reach the maximum size. A maximum of 8 log files exist. The maximum log file size it set with max-logfile-size, e.g.

```
max-logfile-size 1000 mb
```

The top directory where the URL database is stored, is preset to /var/ufdbqs/blacklists and may be changed with dbhome followed by a directory name, e.g.

```
dbhome “/local/ufdbqs/blacklists”
```

<u>Do not forget</u> to update the `BLACKLIST_DIR` variable in the system configuration file (see section 5.4.1) to match the `dbhome`. This ensures that `ufdbqs-update` puts a new URL database in the right place.

5.4.3 User-defined Categories

The most common user-defined categories are a whitelist and a blacklist. The addition of a user-defined category is always with two steps: first convert a the text file 'domains' into a database table file 'domains.ufdb' and then add a category definition to the configuration file.

The conversion of the text file is done with the tool `ufdbqsGenTable` which must have the mandatory `-d` flag to specify the text file with domains (URLs composed of a domainname only) and an optional `-u` flag to specify the urls file which contains URLs that have a path. To create and convert the new table 'mylist' one does the following:

```
cd /var/ufdbqs/blacklists
mkdir mylist
cd mylist
echo mydomain.com > domains
echo mydomain.net/foobar.html > urls
ufdbqsGenTable -d domains -u urls
```

To configure a category 'mylist' one adds a category definition to the configuration file like this

```
category mylist {
   domainlist mylist/domains
}
```

Note that the Query Server will load the URL table from `mylist/domains.`*`ufdb`*.

Note that the clients of the Query Server only send domain names so the use of the `-u` flag has no effect. This may change if clients will be capable to manage URLs in their query cache.

5.5 Firewall

The query server daemon and the database update script must have access to port 443 of `updates.urlfilterdb.com`. Ensure that firewalls allow these connections for IPv4 and IPv6.

5.6 Verification

`qstest0` is a utility to test the server. `qstest0` produces a log file `qstest0.log` which contains a lot of debug information when the `-d` debug flag is used. `qstest0` supports the following command line options:

`-d` enable debug output to logfile

`-v` print version of qstest0 and exit

`-i` *`file`* specify input file with client commands, usually QUERY www.example.com

`-p` *`port`* server port number. default is 1567

`-s` *`server`* name of server. default is localhost.

`-P` *`secret`* mandatory flag. *secret* refers to the shared secret that client uses for authentication

`-m` *`nthreads`* create *nthreads* threads (max. 5000) Each thread processes the input file or default queries

`-S` *`nsock`* each thread creates *nsock* sockets (max. 10) to the server and uses them alternating

`-r` perform small random sleeps (0-1 seconds) between queries

`-rr` perform larger random sleeps (0-8 seconds) between queries

`-l` *`logid`* to distinguish multiple running copies of `qstest0`, the `-l` flag substitutes `qstest0` in the log filename and error messages by *`logid`*.

If no input file is specified, qstest0 sends a CLIENT command, six QUERY commands and a QUIT command to the server.

Verify the correct installation and configuration with `qstest0`, e.g.

```
qstest0 -d -P 0123456789abcdef
qstest0 -d -P 0123456789abcdef -s server-IP -m 100 -S 5 -i queries.qs
```

Note that the OS does not have more that 65,535 source ports per IP address so `qstest0` can only be safely used to test up to ~50,000 connections.

6 Query Server Management

6.1 Stop, Start, Status

The query server daemon is managed like most software with `systemctl`. See the `man systemctl` for more information.

The Query Server has logfile `ufdbqsd.log` in the directory `/var/ufdbqs/log`. Logfiles have a maximum size and are rotated. `ufdbqsd.log.8` is the oldest logfile. In case that there is an issue, it is recommended to always look in the logfile to see if the daemon has something to report. Sometimes it may be necessary to set the debug level to get more information in the logfile. Contact the support desk of URLfilterDB in case of doubts.

6.2 Database Refresh and Uncategorised URLs

The `ufdbqs-update` script downloads a new URL database and sends a HUP signal to the Query Server daemon to refresh its in-memory copy of the database. The URL database must be refreshed at least once per 24 hours. When the ufdbqs package is installed, it creates a default crontab job for `ufdbqs-update` to run at 5 AM in `/etc/cron.d/ufdbqs`. Edit this file and adjust the time to start the job according to local scheduling policies.

The Query Server Daemon maintains a set of URLs which are not yet in the database, the so-called *uncategorised URLs*, and uploads the URLs together with statistics to the server `updates.urlfilterdb.com`. Unless the partner has an unlimited use license, the statistics must be uploaded at least once per 24 hours. Make sure that all instances of the daemon are refreshed every 24 hours.

6.3 Monitoring

The Query Server daemon has verbose messages in its log file. The logfile should be regularly monitored for lines with

```
FATAL ERROR:
ERROR:
*****
```

Fatal error messages and important informative messages have five stars.

The Query Server also sends all fatal errors to the system log with the parameters `facility=local0` and `priority=alert`. The script `ufdbqs-update` sends errors to its stdout and the system log with `facility=local0` and `priority=error`.