Endpoint: dnsbl - DNSBL v5 with API v3

GDPR NOTICE

We used to snapshot the that rendered the blacklist reason.

However, as of 25th may 2018, when the data protection law changed the history of personal integrity, we no longer store this kind of content. It might sound strange that we do not store spam that works like a proof for why e-mail has been blacklisted. It also normally helps system administrators (especially those who administers email services) to trace the source of spam. But to protect the receivers part data, the mail spam storage project has been abandoned.

Table of contents

DNSBLv5 is a part of the deprecated APIv2 interface. However, we missed two things in the interface:

- Proper documentation
- API Configuration abilities (meaning, API keys was a nightmare if you wanted to create them, since you was forced doing this manually)

Both problems are solved: The documentation is being written as the DNSBL are developed and the API Configuration can be done by using TorneAUTHv4.
DNSBLv5 at the level of APIv3 is a bit different to the old API. Instead of doing weird requests, the APIv3 is properly utilizing the use of HTTP methods.

Take a look below at the big differences.

Permissions

Resolve first, use API last!

BEFORE USING THE API TO GET LIST OF BLACKLISTED HOSTS, READ THIS SECTION!

In the DNSBLv3-API a new set of permissions are assigned to the DNSBL. Normally, no permissions are required (to list blacklisted hosts).

Requesting permissions?

The special permissions need to be requested manually via support@tornevall.net

Permission flag	Description
allow_cidr	*The usage of CIDR-blocks are normally not permitted by the DNSBL API, in more functions than listing them. This permission also opens up for usage in DELETE/UPDATE cases (in our local e-mail requests where support sometimes cover CIDR-block removals, this would help a lot). However, adding data with CIDR and different flags might be a problem (this permission does not exist in the API yet)*
allow_cidr_update	*Setting that partially allows CIDR-block updates for the DNSBL (there migt me limitations linked to this permission - see the documentation for this information)* ***Current limitations: Not larger than a /24. This permission is also, currently, limited for internal use.***
can_purge	*Special ability to purge hosts instead of marking them deleted in the database*
dnsbl_update	*Standard DNSBL ability to update data in the DNSBL (dnsbl.tornevall.org and bl.fraudbl.org)*

fraudbl_update	*Extended ability to handle fraudbl-commerce (this is not the regular bl.fraudbl.org resolver)*
global_delist	*Global delisting permission (can use as delisting service for visitors)*
local_delist	*Local delisting permission (server can delist self)*
overwrite_flags	*When sending new or updated data to DNSBL, clients can only add more flags to the host. This feature makes it possible to overwrite old flags* ***Not yet implemented***

DNSBL Version

The DNSBL system itself is still running on 5.0, it is only the API that communicates with the system that is actually changed.

DNSBLv5APIv2

In DNSBLv5APIv2 a request of checking a listed ip looks like this:

Method	URL/data	The parameter	Expected response
HTTP POST	https://api.tornevall.net/2.0/dnsbl/ip/		
HTTP POST variables	?bulk[]=ipaddr1&bulk[]=ipaddr2	bulk[]=ipAddr	Arrayed (or not arrayed by only using bulk=ipAddr) request should return information about the current blacklisted ip (if it is blacklisted). To return fingerprints about the ip address, you could add ipAddr|e, where e stands for extended information. To add or delete a host in the blacklist, with specific permissions additional parameters was used: |a|<bitValue> for adding the address with a bit value (described here), or |d for deletion. Additional parameters (with permissions) could be used to purge (**p**) content. FIngerprints are used to make API requesters get more information about the

DNSBLv5APIv3

In DNSBLv5APIv3 a request is simplified like this:

Method	URL	Information	Data (POST parameters)	Expected response

HTTP POST	https://api.tornevall.net/3.0/dnsbl/
	
	or
	
	https://api.tornevall.com/3.0/dnsbl/request/ip/1.2.3.4
		HTTP POST has the same role as HTTP GET but with post parameters. Supports IPv6.	URL-encoded or JSON-formatted:
			
			`// simple json`
			`{"ip":"1.2.3.4"}`
			`// simple http post`
			`&ip=1.2.3.4`
			`// multiple json`
			`{"ip":["1.2.3.4", "5.6.7.8"]}`
			`// multiple http post`
			`ip[]=1.2.3.4&ip[]=5.6.7.8`
			
			This example is the same for the rest of examples in this table.
				Is ip listed?

<table>
<tr>
<td>HTTP PUT</td>
<td>https://api.tornevall.net/3.0/dnsbl/</td>
<td>Insert or update ip address</td>
<td>
<table>
<tr><td>ip</td><td>Array with ip address and bitmasked flag-per-ip</td></tr>
</table>
The bitmask flags mentions here

Example:

Do blacklist
<pre>{
        "ip": {
                "10.10.10.10": 64
        },
        "type": "dnsbl"
}</pre>
Response Look:

Response parameters (status) described
<table>
<tr><td>success</td><td>The insertion ID</td></tr>
<tr><td>address</td><td>The address that was update or inserted</td></tr>
<tr><td>state</td><td>Defines if the request already has the address blacklisted or if it was updated. Answers can be new or update.</td></tr>
<tr><td>arpaDelegations</td><td>A list of DNS-records that was registered or updated</td></tr>
<tr><td>flag</td><td>The flag of the blacklisted address</td></tr>
</table>
Do blacklist
<pre>{
        "dnsblResponse": {
                "status": [{
                        "success":
"1934699",
                        "address":
"10.10.10.10",
                        "state": "new",

"arpaDelegations": [

"10.10.10.10.dnsbl.tornevall.org"
                        ],
                        "flag": "64"
                }]
        }
}</pre>
</td>
<td></td>
</tr>
<tr>
<td>HTTP DELETE</td>
<td>https://api.tornevall.net/3.0/dnsbl/</td>
<td>Delist (delete/remove) ip address</td>
<td>
Example:

Do blacklist
<pre>{
        "ip":"1.2.3.4",
}</pre>
</td>
<td></td>
</tr>
</table>

Registration types

The default "type" used in some of the examples above is using the standard registration behaviour. It is simply just "dnsbl". As fraudbl.org is using the same flagset (see DNSBLv5: About and usage), normally we don't need to know anything else than this. Adding a host with the flagid 4 (phishing) will also automatically update the fraudbl tables with proper data.

In the new plugin for Wordpress, however, there's a new supported method that includes ecommerce. The refresh rate for this method are much higher than the normal hosts registry (and normally, you want to use the API for this as DNS requests might be a bit delayed). The ecommerce method is specifically used for fraud controls in ecommerce sites (with for example ecommerce plugins that supports fraud controls). As this method should be used with caution, since this directly might affect ecommerce sites, the hosts also have a shorter time to live and should - by a supported plugin - also handle ecommerce unfreezing (which means, if your site is flagging a host as fraudulent, it should also deflag hosts when they are confirmed non fraudulent).

We should also consider how to block website visitors as orders flagged as fraud, might also deny access to websites instead of order confirmations. However, the real purpose of this functionality is to block visitors to shop if they are really used for frauding (for example, fraud-customers that visits many shops at the same time to buy things before discovered). In this case, the checkout shoule be rejected.

The typeset for such scenarios is "ecommerce" and will be flagged differently to the regular lookups. Here's an example:

```
{
        "ip":"1.2.3.4",
        "type":"ecommerce"
}
```

allow_cidr_update