DNS and LDAP persistent search

FreeIPA 3.0 and bind-dyndb-ldap 2.3

Petr Špaček <pspacek@redhat.com>

01-14-2013

FreeIPA DNS integration

- FreeIPA is able to store DNS zones in LDAP database as a tree of objects. DNS server BIND 9 uses **bind-dyndb-ldap** plugin for accessing the database.
- Configuration related to DNS is stored in LDAP and partially in `/etc/named.conf` on each FreeIPA server.
- Bind-dyndb-ldap supports two modes of operation:
  - Polling mode (legacy, used before FreeIPA 3.0)
  - Persistent search mode (used in FreeIPA 3.0)

Polling mode (before FreeIPA 3.0)

- LDAP database is periodically checked for new or modified DNS zones.
  - Default period was 30 or 60 seconds, see `zone_refresh` parameter in `/etc/named.conf` on specific server.
  - DNS server will notice added or deleted a zone with delay [0, zone_refresh] seconds.
- DNS records requested by clients are cached in DNS server's memory to lower a load of LDAP server.
  - Default cache TTL was 120 seconds, see `cache_ttl` parameter in `/etc/named.conf` on specific server.
  - DNS server will notice changed record with delay [0, cache_ttl] seconds.

Polling mode: configuration

- **FreeIPA before version 3.0 will configure poling mode automatically!**
- In `/etc/named.conf`:

```
dynamic-db "ipa" {
   library "ldap.so";
   <snip>
   arg "zone_refresh 30";
};
```

- Bind-dyndb-ldap (ldap.so) has built-in defaults:
  - `cache_ttl` = 120 seconds
  - `psearch` = no

Polling mode: zones

- New domain will be visible after [0, zone_refresh] seconds.
- Same rule applies to zone enabling/deleting/disabling.

```
[12:41:59]$ dig @127.0.0.1 example.test.
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 14361

[12:41:59]$ ipa dnszone-add example.test. \
            --name-server=ipa.corp.test. \
            --admin-email=hostmaster.corp.test.

[12:42:00]$ dig @127.0.0.1 example.test.
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 35154

[12:42:30]$ dig @127.0.0.1 example.test.
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31693
```

Polling mode: new records

- New records are visible immediately because there is **no negative caching in database back-end**.

```
[13:11:23]$ ipa dnsrecord-add example.test. \
            newrec '--txt-rec="""created on 13:11:23"""'

[13:11:24]$ dig @127.0.0.1 -t TXT newrec.example.test.
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59911
;; ANSWER SECTION:
newrec.example.test. 86400 IN TXT "created on 13:11:23"
```

Polling mode: modified/deleted records

- Record modification/deletion will be visible after [0, cache_ttl] seconds after the last query.

```
[13:11:24]$ ipa dnsrecord-mod example.test. \
           newrec '--txt-rec="""created on 13:11:23"""' \
           --txt-data="this is newer :-)"
  Record name: newrec
  TXT record: "this is newer :-)"

[13:11:25]$ dig @127.0.0.1 -t TXT newrec.example.test.
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24615
;; ANSWER SECTION:
newrec.example.test. 86400 IN TXT   "created on 13:11:23"

[13:13:25]$ dig @127.0.0.1 -t TXT newrec.example.test.
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43599
;; ANSWER SECTION:
newrec.example.test. 86400 IN TXT   "this is newer :-)"
```

Persistent search mode (from FreeIPA 3.0)

- DNS server watches LDAP DB with persistent search.
  (ancient RFC draft-ietf-ldapext-psearch-03)
- Any change in LDAP should be noticed by DNS server nearly immediately.
  - LDAP replication between FreeIPA servers can delay propagation of changes. It depends on replication topology, link latency etc.
- No periodic queries to LDAP server are done. Cache TTL is infinity: LDAP server will be asked for the same record only once.

Persistent search mode: configuration

- **FreeIPA version 3.0 will configure persistent search mode automatically!**
- In `/etc/named.conf`:

```
dynamic-db "ipa" {
  library "ldap.so";
  <snip>
  arg "zone_refresh 0";
  arg "psearch yes";
};
```

- Bind-dyndb-ldap (ldap.so) in persistent search mode ignores built-in `cache_ttl`.
- `zone_refresh` and `psearch` are mutually exclusive.

Persistent search mode: zones

- New domain will be visible nearly immediately.
- Same rule applies to zone enabling/deleting/disabling.

```
[15:31:49]$ dig @127.0.0.1 example.test.
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 58617

[15:31:49]$ ipa dnszone-add example.test. \
            --name-server=ipa.corp.test. \
            --admin-email=hostmaster.corp.test.

[15:31:50]$ dig @127.0.0.1 example.test.
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59435
```

Persistent search mode: zones – known BUG!

- bind-dyndb-ldap 2.3 + FreeIPA 3.0 contain a bug – a new zone with **relative name of name server** *could* return SERVFAIL until BIND reload. Red Hat bug 893571.
- The zone is visible immediately if you are lucky.
- Workaround - reload BIND with one of the commands:
  - `rndc reload` or `service named restart`

```
[15:38:36]$ dig @127.0.0.1 example.test.
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 39982

[15:38:36]$ ipa dnszone-add example.test. \
            --name-server=ns \
            --ip-address=192.0.2.1 \
            --admin-email=hostmaster.example.test.

[15:38:37]$ dig @127.0.0.1 example.test.
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 56265
```

Persistent search mode: records

- Any record operation will be visible nearly immediately.

```
[15:46:00]$ ipa dnsrecord-add example.test. \
            newrec '--txt-rec="""created on 15:01:00"""'

[15:46:01]$ dig @127.0.0.1 -t TXT newrec.example.test.
;; ANSWER SECTION:
newrec.example.test. 86400 IN TXT   "created on 15:01:00"

[15:46:01]$ ipa dnsrecord-mod example.test. newrec \
            '--txt-rec="""created on 15:01:00"""' \
            --txt-data="this is newer :-)"

[15:46:02]$ dig @127.0.0.1 -t TXT newrec.example.test.
;; ANSWER SECTION:
newrec.example.test. 86400 IN TXT   "this is newer :-)"
```

Bind-dyndb-ldap modes comparison

	Polling	**Persistent search**
Default in FreeIPA version	< 3.0	>= 3.0
Time to detect **record** change	up to 120 s	immediately
Time to detect **zone** change	up to 30 s	immediately*
SOA serial auto-increment	not supported	supported**

* Except the Red Hat bug 893571
** Be aware of Red Hat bug 895083

Debugging

- Intermediate servers can cache replies and cached records could hide real error sometimes.
    - Query FreeIPA servers directly with command `dig @IPA_server_IP_address`
- `/var/log/messages` or BIND logs should contain some hints if something doesn't work as expected.

Debugging: persistent search error messages

- Usually are in `/var/log/messages`
- Typical error message related to persistent search looks like (with added newlines):

```
named[2310]: update_record (psearch) failed,
dn 'idnsname=record,idnsname=zone.example.com,
cn=dns,dc=example,dc=com' change type 0x4.
Records can be outdated, run `rndc reload`:
not found
```

- Last line with **bold text** contains error code.
  - “Not found” – meaning depends on context.
  - Examine messages preceding and following this line to get some context.
  - Raise debug level when it is still not clear.

Debugging: not found errors

```
named[2310]: update_record (psearch) failed,
dn 'idnsname=record,idnsname=zone.example.com,
cn=dns,dc=example,dc=com' change type 0x4.
Records can be outdated, run `rndc reload`:
not found
```

- It usually means “object disappeared from the database before it was processed”.
  - Usually nothing to worry about.
  - E.g. some administration tool (FreeIPA CLI) changed some attribute (change type 0x4) and then deleted the whole object. BIND attempted to fetch changed object (after attribute change) but object disappeared in meanwhile.
  - E.g. records are added and deleted with speed BIND can't follow.
  - E.g. all records and then whole zone was deleted too fast for BIND to follow.

Debugging: zone not loaded

```
named[2310]: update_record (psearch) failed,
dn 'idnsname=record,idnsname=zone.example.com,
cn=dns,dc=example,dc=com' change type 0x1.
Records can be outdated, run `rndc reload`:
zone not loaded
```

(alternative error code is **`bad zone`**)

- It usually means “object inside invalid zone changed”.
  - Examine logs for the reason why the zone is invalid.
    - Missing glue records, missing NS records etc.
  - Fix errors in zone data and **reload BIND**.
  - Reload should be unnecessary after fixing Red Hat bug 893571. Manual intervention is inevitable in cases where zone data are invalid.

Debugging: more verbose logs

- Debugging messages are usually sent to `/var/named/data/named.run`
- Use `rndc debug trace <level>` to raise log level.
  - 0 is the default. Keyword “`notrace`” disables debugging.
  - 20 will print huge amount of data including each event received through persistent search mechanism and each LDAP query.
- Enable verbose checks in `/etc/named.conf:`

```
dynamic-db "ipa" {
        arg "verbose_checks yes";
};
```

  - `Verbose_checks` are **not supported** in bind-dyndb-ldap **<= 2.3**.

DNS caching: authoritative server

- Each DNS record has own TTL value.
- TTL value is in the second column of answer section, in following examples written in **bold font**.
- FreeIPA uses 86400 seconds (1 day) as default value.
- TTL was artificially lowered to 60 seconds for following examples.

```
[11:18:54]$ dig @IPA ipa.corp.test.
;; ANSWER SECTION:
ipa.corp.test.    60 IN A  192.0.2.1

[11:19:05]$ dig @IPA ipa.corp.test.
;; ANSWER SECTION:
ipa.corp.test.    60 IN A  192.0.2.1
```

DNS caching: authoritative server

- Model situation: “`ipa`” is authoritative DNS server, “`cache`” is some random recursive caching DNS server in the world.
- Answers from authoritative servers are always “fresh”, i.e. answers have TTL as specified in zone.
- Changes are visible immediately if persistent search is enabled.

```
[11:18:54]$ dig @IPA ipa.corp.test.
;; ANSWER SECTION:
ipa.corp.test.     60 IN A  192.0.2.1

[11:19:05]$ dig @IPA ipa.corp.test.
;; ANSWER SECTION:
ipa.corp.test.     60 IN A  192.0.2.1
```

DNS caching: caching server: positive cache

- Caching servers obtain answers from some other servers, authoritative or also caching.
- TTL contained in each cached record is used as counter for count down. TTL is decremented by 1 each second and the cached record is flushed when TTL reaches 0.
- Remaining TTL value is sent to the client as part of the answer.
- **Remaining TTL is propagated downstream**, from authoritative servers through the whole tree of caching servers to the clients.

DNS caching: caching server: positive cache

- A record for `ipa.corp.test.` is retrieved at the first request, timestamp 11:19:05.
- Next query for the same record is answered from the cache. TTL in the second answer is lower than TTL in the first (“fresh”) reply.
- The record is served from the cache until TTL drops to zero. Any change immediately visible on authoritative server will be hidden until cache expires!

```
[11:19:05]$ dig @CACHE ipa.corp.test.
;; ANSWER SECTION:
ipa.corp.test.    60 IN A  192.0.2.1

[11:19:15]$ dig @CACHE ipa.corp.test.
;; ANSWER SECTION:
ipa.corp.test.    50 IN A  192.0.2.1
```

DNS caching: caching server: negative cache

- Answers for non-existing names obtained from caching non-authoritative servers behaves in the same way.
- Only difference is how initial TTL is determined.

```
[11:24:12]$ dig +multiline @CACHE blah.corp.test.
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 1523
;; AUTHORITY SECTION:
ipa.corp.test. 60 IN SOA   ipa.corp.test. (
<snip>
            3600       ; minimum (1 hour)
)

[11:24:21]$ dig +multiline @CACHE blah.corp.test.
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 22846
;; AUTHORITY SECTION:
ipa.corp.test. 51 IN SOA   ipa.corp.test. ( <snip>
            13 86400 3600 604800 3600 )
```

DNS caching: caching server: negative cache

- Effective TTL for negative cache is minimum from pair (SOA record minimum, SOA record TTL), i.e. 60 seconds in the following example:

```
$ dig @127.0.0.1 +multiline blah.corp.test.
;; AUTHORITY SECTION:
ipa.corp.test. 60 IN SOA   ipa.corp.test. hm.corp.test. (
            13         ; serial
            86400      ; refresh (1 day)
            3600       ; retry (1 hour)
            604800     ; expire (1 week)
            3600       ; minimum (1 hour)
            )
```

DNS caching: caching server: manipulating cache

- BIND command `rndc flush` will clean **server's local** cache.
- `rndc dumpdb -cache` will save current cache to file `/var/named/data/cache_dump.db`
  - Dump has classical DNS zone format.

DNS caching: recommendation

1. Don't use extremely small values for TTL (seconds) for normal operation. It will add a lot of network round trips and raise latency visible by end users.
2. Plan changes in DNS carefully :-)
3. Lower TTL for affected records to some small value (e.g. 60 seconds).
4. Wait until original (longer) TTL expires.
5. Do the changes.
6. Test changes thoroughly.
7. Raise TTL to original value if everything works.

Limitations

- Persistent search is expected to consume a lot of resources in 389DS.
- Persistent search can't detect object renaming (LDAP `moddn` operation) if the new DN is outside of `cn=dns` subtree. It normally should not happen.
- Some people use bind-dyndb-ldap with OpenLDAP but OpenLDAP doesn't support persistent search (ancient RFC draft-ietf-ldapext-psearch-03), only the newer SyncRepl (RFC 4533) mechanism. Patches are welcome :-)