First Steps Towards A (General) RPKI RTR C Lib & Its Integration into BIRD

Fabian Holler, Thomas C. Schmidt, and Matthias Wählisch

holler_f@informatik.haw-hamburg.de

{t.schmidt, waehlisch}@ieee.org

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Freie Universität Berlin

Project Outline

1. Implement general RPKI-RTR protocol in C
2. Integrate RPKI-RTR library into BIRD
   to allow for prefix origin validation
3. Performance tests

At the moment, we concentrate on 1. – 2.

General C RPKI-RTR library

- Objections of the implementation:
  - Fetch *validated* prefixes + origin ASes from RPKI cache
  - Keep the routers validation database in sync
  - Conform to *draft-ietf-sidr-rpki-rtr-11*
  - Provide an interface between local database and routing daemon to access validated objects
  - Allow also for validation of BGP updates:
    conform to *draft-ietf-sidr-pfx-validate*
- Open questions:
  - Which SSH implementation should be used?
  - Do RPKI validation records fit in the RAM? Do we need HDD storage?

Integration of RPKI-RTR Lib into BIRD

- Enable Origin Validation per BGP session
  - Validation operations on every BGP Update
- Required extensions: BGP routes include attribute PFX_origin_validity_state
  - Values: VALID, INVALID, NOT_FOUND
- Implement BIRD filters that consider origin validation and reflect route policies
- Open questions:
  - RPKI-RTR lib should be general and not depend on BIRD, but BIRD uses an own abstraction layer for OS operations. What is the best way to accomplish both?

Administration Data Types

-
  ```
  enum connectionState{
     CONNECTION_CONNECTING,
     CONNECTION_ERROR,
     CONNECTION_INITALSYNC,
     CONNECTION_SYNC,
     CONNECTION_ESTABLISHED
  };
  ```

-
  ```
  struct serverConfig{
    char* host;
    char* port;
    char* username;
    char* serverPubKey;
    char* clientPrivKey;
    u_int priority;
  } serverConfig;
  ```

Administration Functions

//**establish connection**, construct/update pfx_validate_table (draft-ietf-sidr-rpki-rtr-11)
int startClient(struct serverConfig config, u_int pollingPeriod, u_int cacheExpireTime);

//**update configuration**, client will connect to the new server with the highest priority
void updateConfig(struct serverConfig config, u_int pollingPeriod, u_int cacheExpireTime);

//**purge the pfx_validate_table** (and all associated validation callbacks?)
int resetCache();

//**sends immediately a serial Query** to the RTR-Server to force an update of the local pfx table
int updateCache();

//**returns the connected server** in struct config and the state of the connection
connectionState getConnectionState(serverConfig* config);

Validation Data Types & Functions

```
enum pfxvState{
	PFXV_VALID,
	PFXV_NOT_FOUND,
	PFXV_INVALID
      };
```

//definition of the callback function for validateOrigin:

```
typedef void (*verifyAnswer_cb)(u_int asNumber, char* prefix, pfxvState state);
```

//**returns immediately the validation state**

```
pfxvState validateOrigin(u_int asNumber, char* prefix, u_int maskLen, verifyAnswer_cb cb);
```

//unregister a validation callback

```
int unregister_validation_cb(u_int asNumber, char* prefix, u_int maskLen, verifyAnswer_cb cb);
```

//unregister all validation callback

```
int unregister_all_validation_cb();
```

Prefix Validate Cache Lookup

```
struct pfxRecord{
      char* prefix;
      u_int minLength;
      u_int maxLength;
      u_int quantity
    } pfxRecord;
```

//**return the pfxRecord** from the pfx_validate_table that matches
//prefix, maskLen, and origin AS; NULL parameters will be ignored
//e.g., return all pfxRecords for AS1234: cacheFind(cache, null, null, 1234)

```
pfxRecord[] cacheFind(pfx_validate_table* cache, char* prefix,
    u_int maskLen, u_int originAs)
```

Conclusion & Further Questions

- Developers of routing daemons need only to extend routes to reflect validation state
- We are in contact with BIRD developers …
- Beta version of lib scheduled for IETF 81@Quebec

However: **Do you find a general C RTR lib useful?**

Further detail questions:

- Are resetCache + updateCache necessary?
- Enable origin validation per BGP-Peering session or consider it as global parameter?