# **Transducer controls**

## **Diagram**

Excalibur

Local

DB

Rethink

DB

ES

Virtue

Syslog-ng

message

?

Merlin

Unix

Domain

Socket

Filter

Virtue

Syslog-ng

message

?

Merlin

Unix

Domain

Socket

Filter

## **Components**

**Excalibur** – Excalibur will expose an API that will be available to the users in the form of a command line interface. It will implement all of the features that were specified in the Transducer part of the APL API:

* transducer list
* transducer get [transducerId]
* transducer enable [transducerId] [virtueId] [configuration]
* transducer disable [transducerId] [virtueId]
* transducer get enabled [transducerId] [virtueId]
* transducer get configuration [transducerId] [virtueId]
* transducer list enabled [virtueId]

In addition, users will be able to enable or disable a particular transducer for all Virtues. This will be useful in cases where a suspicious event causes administrators to want to increase logging for all Virtues, or if a particular transducer is spamming the system and must be disabled.

**Local DB** – This will be a database where Excalibur will store all of the objects specified by the APL API. Currently a relational SQL database would be ideal.

**RethinkDB** – RethinkDB is a database that can efficiently notify clients of changes to the data. In this design, RethinkDB will store a copy of the current ruleset (state of transducers) for each Virtue. Whenever a user requests a change to the ruleset, Excalibur will push a change to RethinkDB, which will trigger an alert to the Virtues that are impacted by this change. Specifically, the alert will trigger the Merlin part of the Virtue.

**Merlin** – When triggered by RethinkDB, Merlin will take the entire ruleset that is specified in RethinkDB for this Virtue, encode it as necessary, and push it to a Unix Domain Socket.

**Filter** – A custom Filter will live in Syslog-ng. Upon initialization, a thread will be launched, which will wait for content from the Unix Domain Socket. When it receives a new ruleset via this Unix Domain Socket (from Merlin), it will swap out the old ruleset with the new ruleset in its memory. Any new log messages that come in to syslog-ng will pass through this filter, and will either pass through to ElasticSearch (ES) or be dropped, depending on the ruleset.

### **Startup**

On Virtue creation, the Role’s list of default transducers is consulted. When the new Virtue first communicates with Excalibur, this initial ruleset is transmitted and set.

### **Excalibur**

The API has already been implemented into a CLI. We will add commands to broadcast enable or disable transducer messages to groups of Virtues:

transducer enable [tranducerId] [roleId] [configuration]

transducer disable [tranducerId] [roleId]

After verifying the user’s credentials, Excalibur will change a row in the RethinkDB table that corresponds to the user’s request. If the user has requested a broadcast message, Excalibur will use its local database to find all of the Virtues that have that Role, and will update all of the relevant rows in RethinkDB.

If the user is requesting the ruleset rather than changing it, the relevant rows can be directly read from RethinkDB.

### **RethinkDB**

Security features:

* One table for sending commands, one table for receiving acknowledgements – so that we can separate which commands have been sent and which commands have already been implemented.
* Each row in the tables is signed in order to make sure that Excalibur/the correct Virtue really sent that command/ACK.
* Include a message ID for each command, so that we can make sure we are getting an acknowledgement for the correct request, and to prevent replay attacks.
* RethinkDB allows for table-level user privileges. Galahad will have an account that allows for writing to the Commands table and reading from the ACKs table, while the Virtues will share an account that allows for reading from the Commands table and writing to the ACKs table. (Since each Virtue has its own key, the signatures will let Excalibur ascertain that the ACK came from the right Virtue.)
* TLS will be enabled for all connections to/from RethinkDB.

### **Merlin & Filter**

Merlin listens to a RethinkDB “changes” stream to get messages. It repackages the message into a format suitable for the Unix Domain Socket and sends it to the Filter. The Filter will send back an acknowledgement after it has implemented the change. If the update failed, Merlin can throw an error or otherwise signal to Excalibur that the update failed.

## **Data formats**

* The data format for Excalibur’s local storage is largely specified by the APL API
* A “transducer” will be an (application, log type) pair. This way, log statements can be turned off on a per-application basis.
* The RethinkDB table will have one row per Virtue/Transducer pair:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Virtue ID | Transducer ID | Configuration | Enabled | Timestamp | Signature |
| 5 | msword\_openfile | {} | True | 1521486930 | <signature> |

* + There are currently no plans for how to use a Transducer’s configuration. If during implementation we find data that should go there, it will be stored in a JSON format.
* The data format between Merlin and Filter can be JSON. If it turns out we need something more complex than a mapping of booleans (transducer name -> enabled), we can use Protobuf.

## **Security Design**

* Excalibur to Merlin – RethinkDB
  + Implement heartbeats on Merlin, to make sure that the Virtue’s current Filter ruleset matches what is in its RethinkDB config.
* Merlin to Filter – Unix Domain Sockets
  + “In the Linux implementation, sockets which are visible in the filesystem honor the permissions of the directory they are in.” <https://stackoverflow.com/questions/28776502/unix-domain-socket-securing-receiver>
  + We will use filesystem permissions to make sure that only Merlin and the Filter can read and write to/from the socket.

## **Error conditions**

* Any part of the pipeline breaks down (or times out)
  + We will send ACKs back to confirm that the Filter actually put the changes into place. This won’t fix the problem entirely. There may be false negatives – the user will get a “failure” message, but only because the ACK took longer to return than the timeout value. Once the ACK comes back, the “get” command will show the state of the Filter properly.
  + Heartbeat messages should also help a bit with this problem.
* Partially successful broadcasts
  + Report how many / which ones failed, if possible.
* Contradictory simultaneous data
  + Timestamps should remove most of these issues, although we may still report false negatives if two users happen to make a change to the same Virtue and same transducer at the same second.
  + There’s nothing to prevent the user from a race condition between when they query for the ruleset and send a command to change it.

## **Implemented Security**

### **RethinkDB:**

* RethinkDB either lives on Excalibur or on a different host – doesn’t matter
* The web interface is turned off because it has no access control
* There are separate user accounts for Excalibur and for the Virtues (but only one account for all the Virtues) – run galahad/transducers/setup\_rethinkdb.py
  + The Excalibur account can write to the Commands table but only read from the ACKs table
  + The Virtue account can write to the ACKs table but only read from the Commands table
* There is a password on the admin account – run galahad/transducers/set\_admin\_pw\_rethinkdb.py
* Connections to RethinkDB must be over TLS; need the database’s CA cert to connect to the RethinkDB in addition to the account’s username and password
* RethinkDB needs to auto-restart if it ever goes down. The docker container is currently set to do that, but if we switch away from docker we need to remember to set that elsewhere.

### **Excalibur:**

* User logs into Excalibur CLI – needs to use a User Token – this is not implemented yet; not sure what the plan is (**TODO**)
* When a command is written to RethinkDB, each row is signed with Excalibur’s private key. On each Virtue, Merlin checks that each command it receives is properly signed.
* When an ACK (equivalent to the current state of a Virtue’s ruleset) is read from RethinkDB, each row is signed with a Virtue’s private key (each Virtue has an individual private key, there is not one key shared between all the Virtues). Excalibur checks the signature before reporting results to the user.
* There is a timestamp in each message (command and ACK) to prevent replay attacks.
  + To receive an ACK for a command, Excalibur checks that the ACK’s timestamp is the same or later than the command’s timestamp.
  + **TODO**: Not sure how else this helps against replay attacks. I guess we could check that the timestamps are within some tolerance? For example, the row must have been written in the past 5 minutes.

### **Virtues – Merlin and Syslog-ng:**

* Each command has a signature that is checked against Excalibur’s public key. Each row written to the ACKs table has a signature, generated from the current Virtue’s private key.
* Merlin and the syslog-ng filter communicate over a Unix domain socket. Access to the socket can be controlled via filesystem permissions. Syslog-ng must run as root, so there will be a group that contains only root and the user running Merlin, and the socket will only be accessible by users in that group.
* Heartbeats - Every x seconds (default is 30), Merlin prompts the syslog-ng filter to send its current ruleset over the Unix domain socket. Merlin then updates the ACKs table appropriately.
* If syslog-ng restarts, its ruleset will be repopulated by Merlin to be the latest settings, found in the ACKs table.
  + Each row retrieved from the ACKs table has its signature checked.
* The syslog-ng filter fails open – meaning that if there is no entry in the ruleset for a transducer, it will default to reporting the message, not blocking it.
* **TODO**: Need Merlin and syslog-ng to auto-restart if they ever get taken down.
* Merlin is constantly listening to the Commands table change-stream. This should ensure that it receives all of the messages from Excalibur. However, there is no guarantee because of broken connections and so on. For this reason, Merlin replies with an ACK to a successful command so that the user will receive feedback on the success/failure of their command.

## **Possible attacks:**

* Replay attack from an old ‘disable’ message for this virtue, for this transducer.
* One Virtue could mess with another Virtue’s entries in the ACKs table. The signatures wouldn’t match but the data would be overwritten.
  + The heartbeats should override the incorrect data.
  + If the attack is persistent and keeps overriding the correct data, we should at least notice that the heartbeats aren’t coming in properly (getting immediately overwritten).
  + If incorrect data is inserted into the ACKs table and then the filter goes down, we can’t reinstate the filter’s old ruleset when it starts up again. That Virtue will have to restart with its default ruleset.
* If the attacker got root access on Excalibur or the Virtue then they could do a lot of damage, but I hope that’s out of scope.
* Taking down Merlin or the Filter would stop heartbeats. However, we should be able to notice that there are no heartbeats and at least alert an administrator.