GRR Meetup
Mar 2017

Agenda

- Fleetspeak
- GRR Golang client
- API client library
- An update on Components
- Memory Analysis in GRR
- Datastore Options
- Q & A

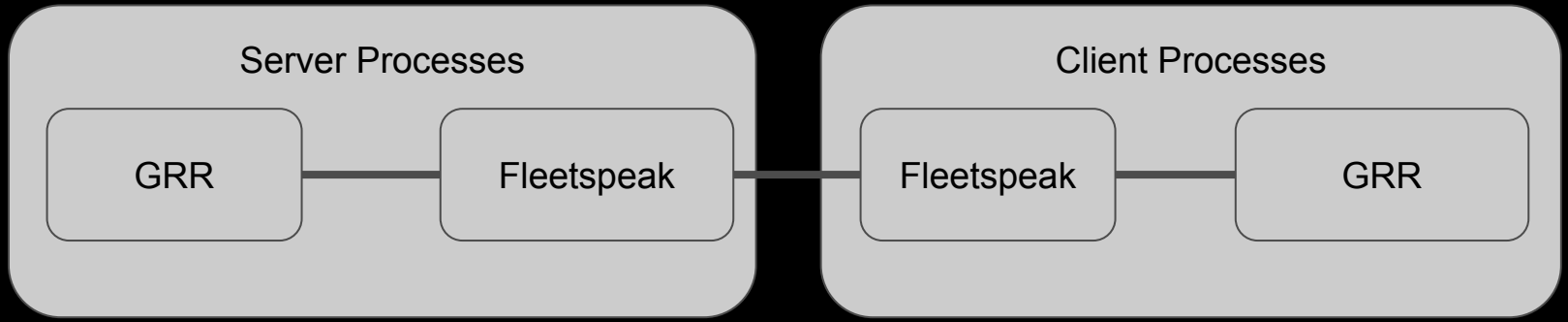
Fleetspeak

Separate the GRR communications layer into its own process.

- Allows the communications layer to be used by other agents.
- Reduces minimum resource footprint - Fleetspeak starts the rest of GRR when needed.

Fleetspeak - Big Picture

- Fits between the GRR client and server.
- Provides client identifiers and labels.
- Remembers communications history.



Fleetspeak - Technology Selection

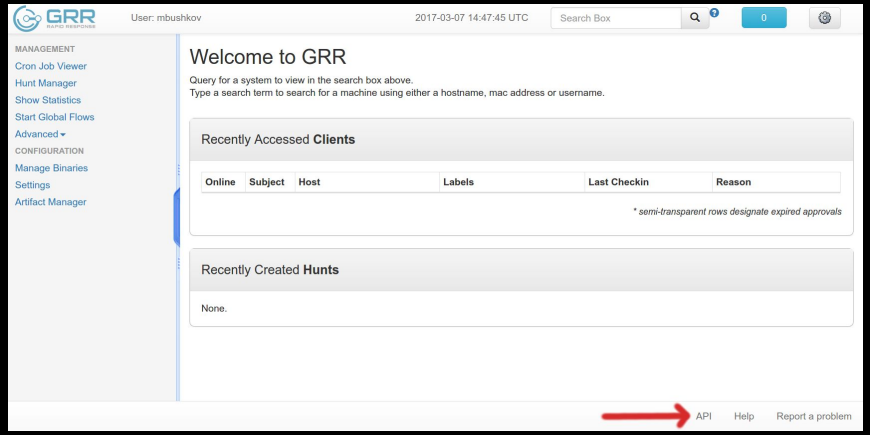
- Written in Go.
- Fleetspeak client and server communicate over HTTPS.
- Fleetspeak communicates with GRR using gRPC.

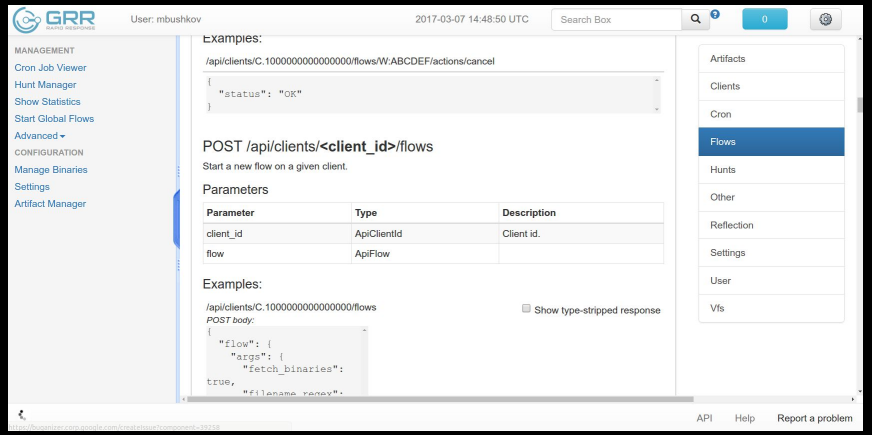
Golang Client

- Future of GRR might be Golang
  - Better performance, strong typing
  - Simplified packaging and deployment (no pyinstaller bundling)
  - Potential for cross-compilation
- Started porting the client to estimate effort as an experiment
  - Code coming soon

GRR API

- GRR has an HTTP API that can be used for scripting and automation
- Documentation is generated on the fly in GRR AdminUI





GRR Python API library

- Python client library to talk to GRR API:
  pip install grr-api-client

- Code and documentation on GitHub.

- Based on “requests” library: easy to support custom authentication methods.

- Has an interactive IPython-based API shell:
  grr_api_shell

Sample using Python API library

```
from grr_api_client import api
grrapi = api.InitHttp(api_endpoint="http://localhost:1234",
                      auth=("user", "pwd"))

search_result = grrapi.SearchClients("suspicious.corp.com")
result = {}
for client in search_result:
  client_id = client.client_id
  client_last_seen_at = client.data.last_seen_at
  result[client_id] = client_last_seen_at
print result
```

An Update on Components

- Components (pushing code to live GRR clients) are
  - stable but not stable enough
  - not truly client version independent
  - problematic for accountability
- We will deprecate them in the next client version
- Short term: build Rekall and Chipsec into the client
- Long term: standalone packages
  - installed by package managers or Fleetspeak
  - invoked by GRR client

Memory Analysis in GRR

- Regret that Michael Cohen is moving teams

- Current Memory Analysis Status and Roadmap:
  - Keep Windows supported
  - Linux is best effort
    - Works for now, might break in the future
  - Recent kernels on macOS not supported
    - Needs more time than we can allocate at this point
    - Patches to Rekall accepted :)

Datastore Options

- GRR data model is not optimal for SQL based data stores
  - limits scalability

- Working on a completely new model, will use Spanner
  - https://cloud.google.com/spanner/
  - SQL based
    - MySQL / SQLite support will be much better

Q & A