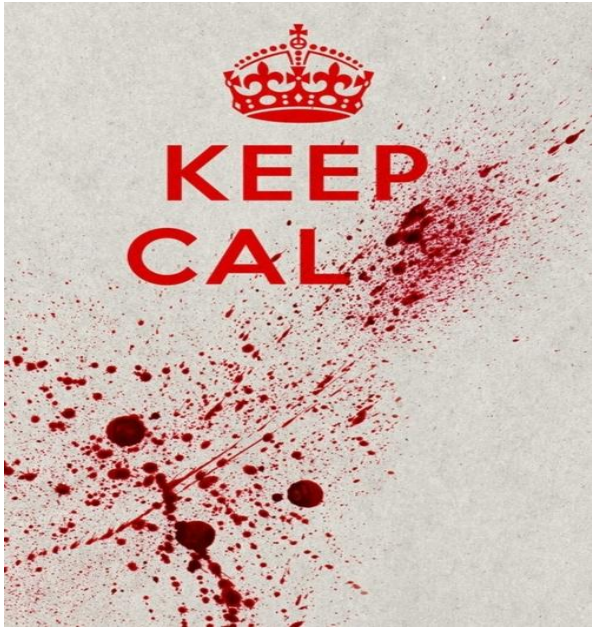
GRR Rapid Response



GRR Workshop @
ICDF2C Prague 2017

Mikhail Bushkov, Ben Galehouse

Agenda

- Introduction to GRR
- Hands on work - exercises
- Discussion

Remote Forensics at Google Scale

- Joe saw something weird, check his machine
  - (p.s. Joe is on holiday in Cambodia and on 3G)
- Forensically acquire 25 machines for analysis
  - (p.s. they're in 5 continents and none are Windows)
- Tell me if this machine is compromised
  - (while you're at it, check 100,000 of them - i.e. "hunt" across the fleet)

What is GRR?

- “GRR Rapid Response”
- Agent based forensics investigation tool
- Open source (Apache License 2.0)

What is GRR?

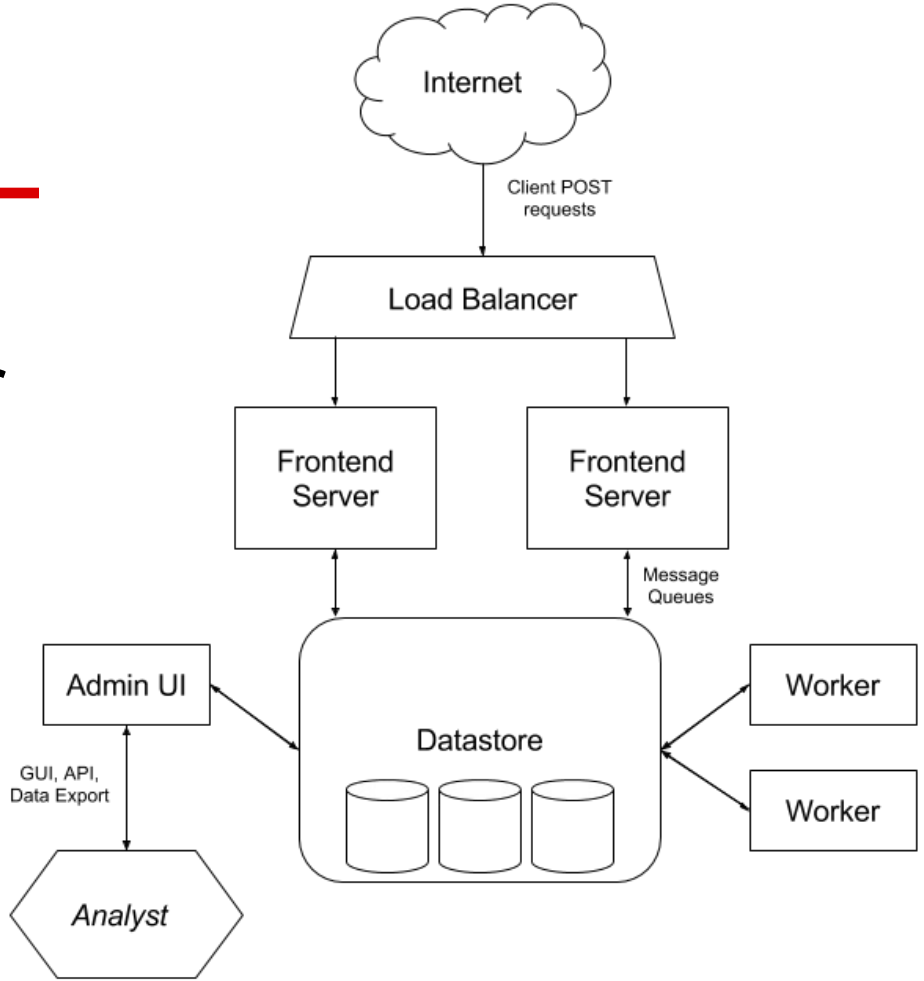
- Built, maintained, used by Google… and others
  - 5.5 full time developers (in ZRH)
  - Lots of people helping out

- Built by engineers for engineers

FAQ: https://github.com/google/grr-doc/blob/master/faq.adoc

Architecture

- Client
- Frontend Server
- Datastore
- Admin UI
- Worker
- API



Datastore

- GRR runs on MySQL (and SQLite for demo purposes)
  - Abstraction makes replacing possible
- Forensic data is stored by client id
  - GRR datastore schema is organized around AFF4 paths: it’s generic but not user-friendly
  - Datastore changes in progress
  - GRR UI/API is used to export data

Datastore

- Forensic data is versioned
  - Usually nothing is deleted ever
    - Just new version are added
    - -> The complete history is kept in the GRR DB

Clients

- Clients for Windows, Mac, Linux
  - Stable, robust, low-impact
    - Python + PyInstaller
    - Memory, CPU limited
    - Watchdog process
  - Contains very little logic
    - encoded in “Flows” on the server
  - We’re experimenting with making the client smarter

Communications

- Client polls the server for work
- Defaults to once every 10 minutes
- Messages are protobufs
- Signed and encrypted end to end
- Default connection via “HTTP”
- Fleetspeak subproject will modularize and modernize comms
  - Separate process (only running GRR when needed)
  - TLS

Audit Controls

- GRR is remote root equivalent
- Audit controls
  - Multi-party authorization
  - Audit hooks
  - Audit log
- Approval-based system built in
  - User, reason, expiry
  - Disabled for the workshop for simplicity

GRR Quick Install

- Setting up your own GRR server
  - Many moving parts but DEB package is available
    - Should be up and running in a few mins
  - DEBs releases are done periodically
  - HEAD DEB is built on every commit

- Can also be installed via:
  - Docker
  - Pip (easiest for development)

Workshop test environment

- GRR server at
  - http://35.203.141.126:8000/
- Clients connected:
  - 1 Windows Server 2008 machine
  - 1 Windows Server 2016 machine
  - 1 Debian Linux machine
  - 1 Ubuntu Linux machine

Demo

- Workshop server

Exercise 1 - Introduction to GRR

- Server: http://35.203.141.126:8000/
- User accounts: user<n>/password<n>
- Search the client database
  - “.” gives all clients
- Look at client info
  - Look at different OSs
- Check out /fs/os in the VFS
  - Also /fs/tsk, /registry for Win clients

Flows

- Flows encapsulate logic
  - Clients are “dumb”
    - Client actions are basic building blocks
      - “Get me this file”, “List this directory”
    - -> Clients don’t need to be updated frequently
  - Flows interpret the data received
    - Ex.: Get browser plugins
      - Downloads file(s) with known paths
      - Parses received data to find plugin directories
      - Downloads those directories

Flow Processing

- Flows are processed on the Worker(s)
  - Flows are then suspended and stored in the datastore
    - If client goes away, flow just resumes at a later time
  - In the end, results are produced
    - Shown in the UI
    - Can be exported from the UI

Launching Flows

- Launching flows demo

FileFinder

- Flow to search for files by multiple criteria
  - path, name, contents (literal / regex), time
- When a file matches, an action is run
  - Download, hash, stat (report existence)

FileFinder

- Demo, this will be next exercise

Exercise 2 - File downloading

- Client C.510b34c59daf8393
- Get a list of all DLLs (*.dll) in C:\Windows\System32
- Get the partition boot sector C:\$BOOT
  - Windows API will hide this! Use pathtype TSK
- There is a file (or two) containing the string "malware" in <Desktop>\Browsercache. Try to find it.

Registry Analysis

- Registry analysis works like file analysis
  - Keys / Directories, Values / Files
  - Same operations supported!
    - Globbing
    - Content match on values

Exercise 3 - Registry

- Client C.510b34c59daf8393
- Poke around using the Registry finder
  - Should be straightforward - similar to FileFinder
  - Please don’t schedule huge recursive listings.
- One of the values in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer contains the string "malware". Which one?

YARA process memory scanning

- YARA process memory scanning built in for all platforms
- Process memory collection is in the works
- GRR has Rekall built in but:
  - It’s deprecated/unsupported
  - Turned off by default

YARA process memory scanning

- Demo

Exercise 4 - YARA memory scanning

- Use the “Yara Process Scan” flow to run YARA scanning directly on a client.
  - Enable “Advanced” mode to be able to use the flow
  - Set per-process timeout to 60
  - Use sample YARA signature:

```
rule Example
{
    strings:
        $text_string = "grr"
    condition:
       $text_string
}
```

Hunts

- Hunting is running a flow on all the clients in the fleet

- Fleet checks
  - I found this suspicious file on one machine, which other boxes have it too?

- Baselining
  - Download the RunOnce Keys from all machines
  - Which ones stand out?
  - Which ones are new compared to last week?

Hunts

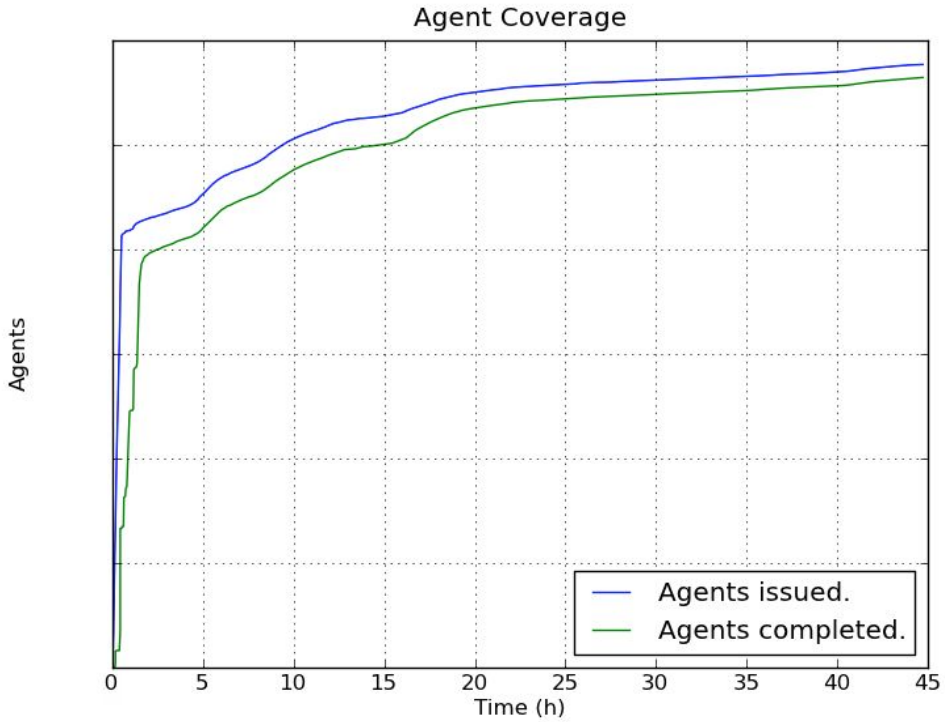
- Demo time - Collect Notepads and Export

Hunt Performance

- Longish lead time
  - Foreman delay
  - Client poll delay

- Once started, checks the whole fleet in hours
  - Mostly depending on client availability

Hunt Performance



Exercise 5 - Fleetwide Process List

- Get a list of all processes running on Windows machines in the test setup
  - Bonus task, do it also for Linux
- Look at hunt stats
  - Cpu used, network used, worst performers
- Export hunt results in CSV, YAML or SQLite format and find all processes with the process name containing “net”

Embedded Flash Malware

- Inspired by Hacking Team attack
  - Flash based attack inside Office document
- How would we go around finding this using GRR?

Exercise 6 - Hunt Embedded Flash

- There are .doc files in C:\Windows\Temp on the Windows machines
- Run a hunt to find the documents that contain embedded Flash
  - That is, they contain the literal “ShockwaveFlash.ShockwaveFlash”
- Download ZIP archive with found files

Artifacts

- Flows are too tricky for simple things
- We wish we could share information better
- Too much duplicated code
  - -> Let's generalize to Artifacts

Artifacts

- Define what to collect
- Define how to parse it
- Define the result they produce
- Data only, no code
- Yaml based format

https://github.com/ForensicArtifacts/artifacts

Artifacts

- Example Artifact:

```
name: WindowsEventLogSecurity
doc: Security Windows Event Log.
sources:
- type: FILE
 attributes:
   paths: ['%%environ_systemroot%%\System32\winevt\Logs\SecEvent.evt']
   separator: '\'
conditions: [os_major_version < 6]
labels: [Logs]
supported_os: [Windows]
urls: ['http://www.forensicswiki.org/wiki/Windows_Event_Log_(EVT)']
```

Artifacts

Knowledge Base Interpolation

%%environ_allusersprofile%% → c:\Documents and Settings\All Users

%%systemroot%% → c:\Windows\System32

%%users.appdata%%

→ c:\Documents and Settings\foo\AppData\Roaming

→ c:\Documents and Settings\bar\AppData\Roaming

→ c:\Documents and Settings\baz\AppData\Roaming

https://github.com/google/grr/blob/master/proto/knowledge_base.proto

Artifacts

- Demo - Artifact Collector flow

Exercise 7 - Artifacts

- Check out the Artifact Collector flow
  - Collect an artifact
    - Event Log? …
- You suspect that the machine C.510b34c59daf8393 was owned by a drive by download. Can you show one of the users went to pho8.com using Chrome?

The End (for this part)...

grr-users@googlegroups.com

https://github.com/google/grr