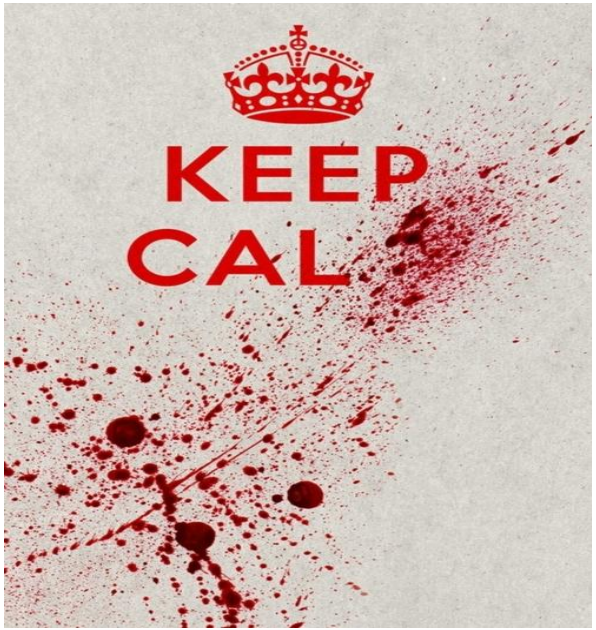
GRR Rapid Response

Practical IR with GRR
DFRWS EU 2015



Andreas Moser, Google

Agenda

- Introduction to GRR
- Demo: Setting up your own GRR server
- Hands on work
  - Easy stuff (Files, Registry, …)
  - More advanced stuff (Investigating live memory)
  - Super interesting stuff (Using Rekall on live memory)
  - Stuff at scale (Collect all the things everywhere at the same time)
- 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 100000 of them - i.e. "hunt" across the fleet)

What is GRR?

What is GRR?

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

What is GRR?

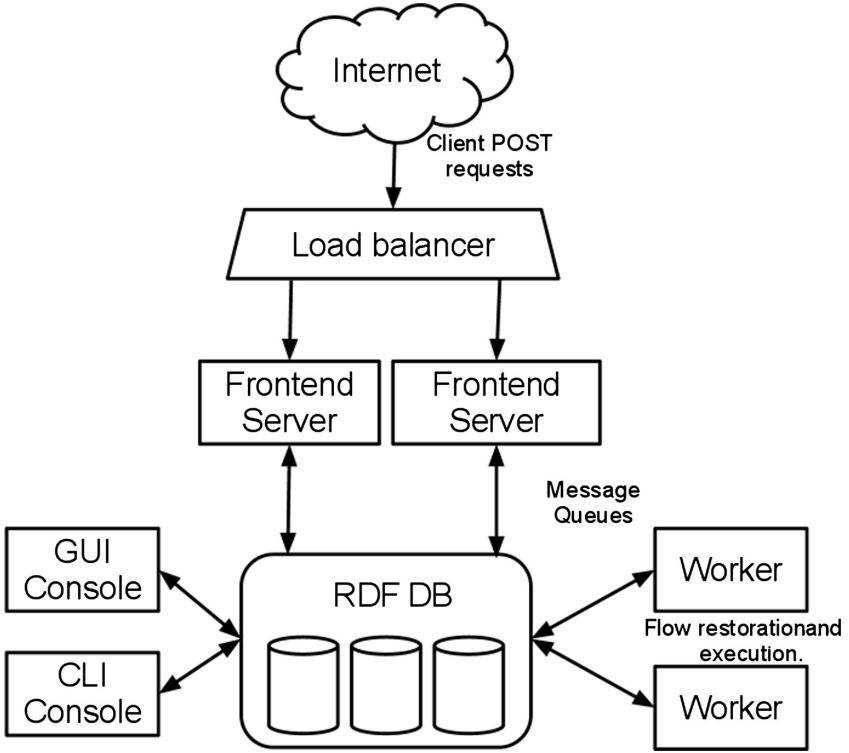
- Built, maintained, used by Google… and others
  - 6 full time developers
  - Lots of people helping out

- Built by engineers for engineers

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

Architecture

- Client
- Frontend Server
- Admin UI
- Worker
- Console



Datastore

- Default db is Mongodb (for now...)
- Can also run on Mysql, Filesystem (SQLite), RemoteDataStore (see presentation)
  - Abstraction makes replacing it easy
- Built on AFF4
  - Every object has a URN, and some attributes
    - Ex.: Client urn “aff4:/C.1c0162518681e509”
      - Attributes: architecture, mac_addresses, usernames, ...

Datastore

- 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
    - Memory, CPU limited
    - Watchdog process
  - Contains very little logic
    - encoded in “Flows” on the server

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”

Audit Controls

- GRR is remote root equivalent
- Audit controls
  - Multi-party authorization
  - Audit hooks
- Made possible by passing ACLToken objects
  - User, reason, expiry
  - Not enabled today, you can ignore them

Demo

- Setting up your own GRR server
  - System is fairly complex
    - but we have a script :)
  - Minimal hardware requirement: one box
  - Should be up and running in 10 - 15 mins
    - Including key generation, client customization and generation, ...

Workshop test environment

- GRR server at
  - <server taken down until the next workshop, sorry>
- Clients connected:
  - 2 Windows server 2008, 2 Windows server 2012
  - 2 Ubuntu 14.04, 1 RHEL 7.1
  - 1 Mac running Mavericks

Demo

- Workshop server

Exercise 1 - Introduction to GRR

- Server IP: <taken down>
- User accounts: User<n>/Password<n>
- Search the client database
  - “.” gives all clients
- Look at client info
  - Look at Mac, Linux clients as well
- 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)
  - Completely asynchronous
    - Triggered by incoming responses from a client or from a subflow
  - Flows schedule more tasks
    - Call one or more client actions
    - Call a subflow

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
    - Sent back to parent flows

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, send to socket, just report existence

FileFinder

- Demo, this will be next exercise

Exercise 2 - File downloading

- Client C.3718d5d27f51d6ea
- 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 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.a25e72587cd41c3e
- 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?

Memory Analysis

- GRR comes with memory acquisition drivers
  - Windows, Mac work out of the box
  - Linux is harder
    - needs driver compiled on target machine
    - or use /proc/kcore
- Memory Collector flow
  - Literal / Regex search
  - Download an image live!

Memory Collector

- Demo

Exercise 4 - Memory Inspection

- Client: any Windows
- Use the Memory Collector to find a short string (ex. “grr”) in memory and inspect the context.
  - Please use report only, don’t take memory images
  - Also, just get the FIRST_HIT, not all of them

Advanced Memory Analysis

- GRR has Rekall built in
  - https://github.com/google/rekall
- Memory analysis framework
  - Plugins to analyze kernel structures and extract forensics data
  - Usually works on images, we do it live :)

Advanced Memory Analysis

- Demo

Exercise 5 - Memory Analysis

- Use the AnalyzeClientMemory flow to run Rekall plugins directly on a client.
  - Candidates: pslist, dlllist, modules

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 Mutexes/RunOnce Keys/… from all machines
  - Which ones stand out?
  - Which ones are new compared to last week?

Hunts

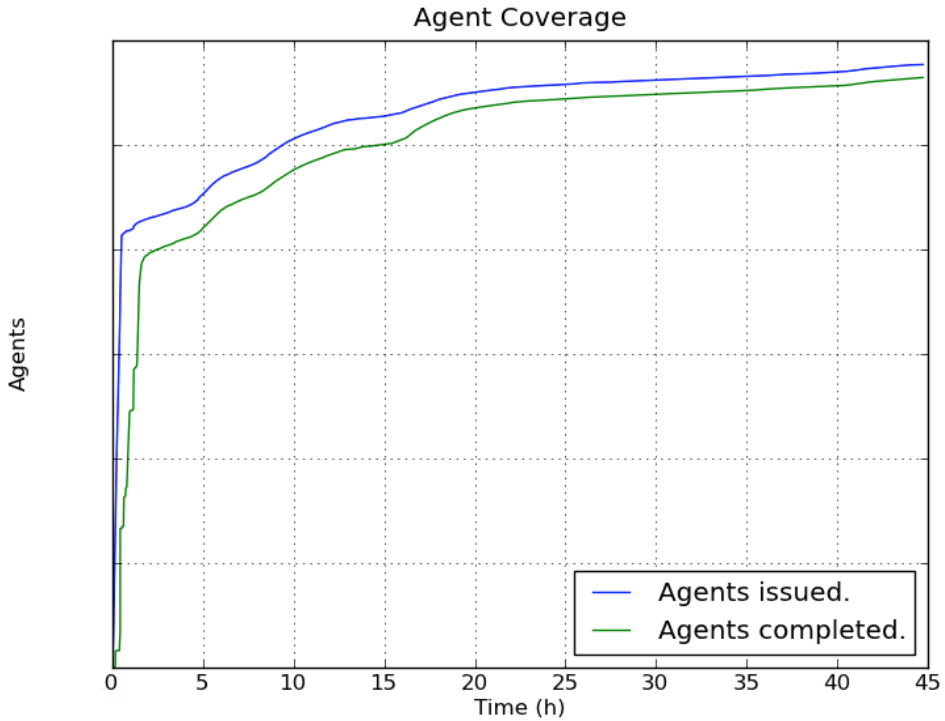
- Demo time - Collect Notepads and Export

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 6 - 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

Embedded Flash Malware

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

Exercise 7 - Hunt Embedded Flash

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

Artifacts

- Flows are too tricky for simple things
- We wish we could share information better
- Too much duplicate 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: SecurityEventLog
doc: Windows Security Event Log.
collectors:
- action: GetFile
  args: {path: '%%environ_systemroot%%\System32\winevt\Logs\SecEvent.evt'}
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 8 - Artifacts

- Check out the Artifact Collector flow
  - Collect an artifact
    - Event Log? …
- You suspect that the machine C.a25e72587cd41c3e was owned by a drive by download. Can you show one of the users went to pho8.com using Chrome?

The End...

grr-users@googlegroups.com

grr-dev@googlegroups.com