Greg Castle
@mrgcastle

Who am I

GRR Developer, Google IR team

OS X Security

Former lives: pentesting, IR, security audits etc.

Skillz++

Understand how GRR works
Setup test server/client
Collect from single machine
Memory analysis
Hunt multiple machines
Fleetcheck using artifacts

Live forensics

GET /beacon HTTP/1.1
Host: evil.com

from Joe’s machine


Joe


GET /beacon HTTP/1.1
Host: evil.com
Joe is on vacation with 3G internet

New APT Report

New malware report BEAR EAGLE SHARK LASER is out: check all the things

New malware report BEAR EAGLE SHARK LASER is out: check all the things

50+ IOCs for Win/Mac and “all the things” is the machines of a highly mobile global organisation with 50k+ employees

GRR: GRR Rapid Response

Open source live forensics

Agent -> Internet -> Server

Disk Forensics = Sleuthkit

Memory Forensics = Rekall

Scalable

Stable, low-impact client

Full-time devs

Why build?

Customize for our threats/detection/defense

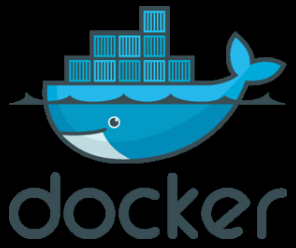
50 people analyzing 50 machines

Move as fast or faster than the attacker

Support Mac/Win/Linux

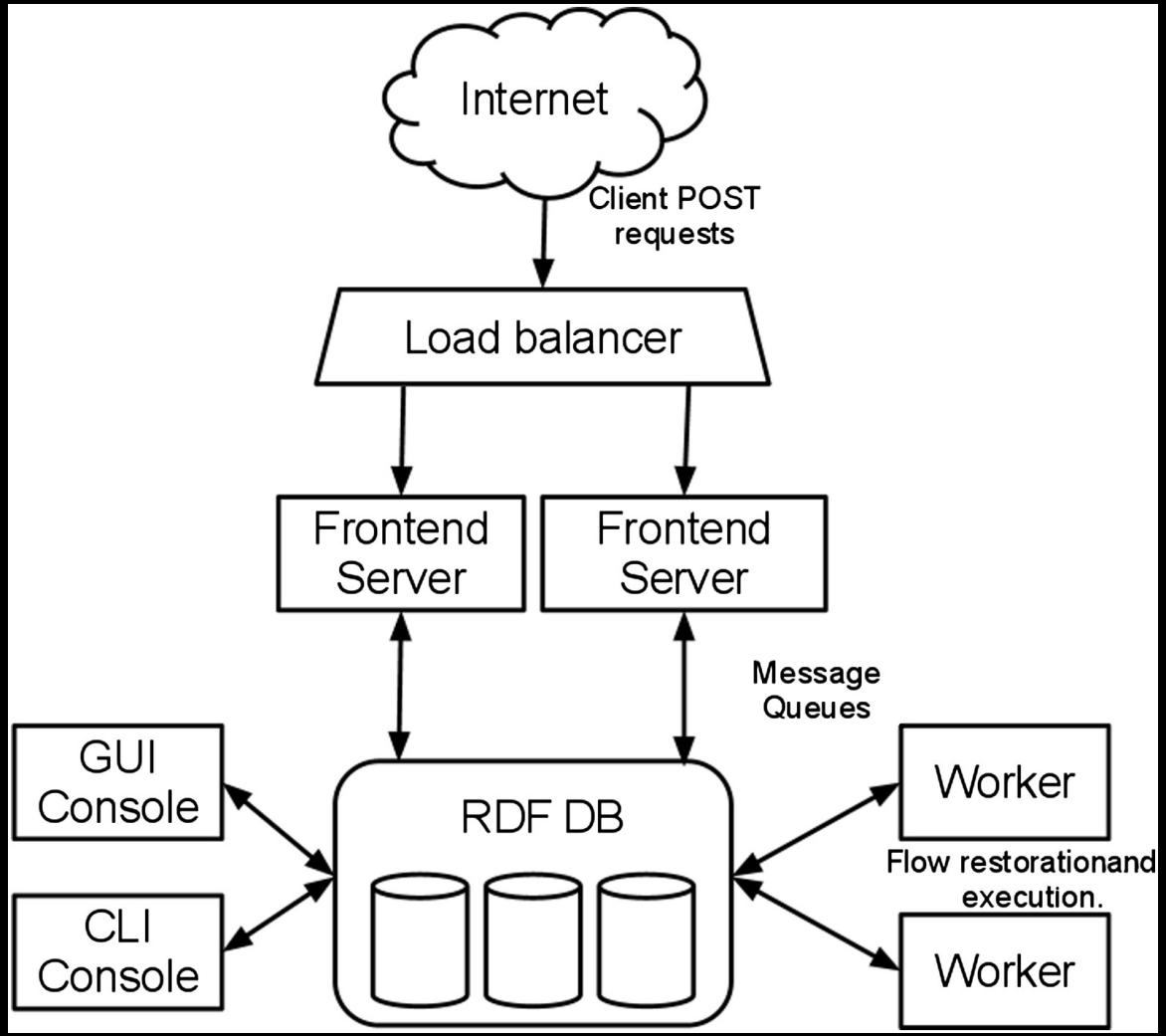

incident response
without remote live forensics

Demo - Server Installation



Install instructions

(pls don’t pull this image down now it will kill the WiFi...)


Internet
Client POST requests
Load balancer
Frontend Server
Frontend Server
Message Queues
GUI Console
CLI Console
RDF DB
Worker
Flow restorationand execution.
Worker

Clients

Stable, robust, low impact

Monitored

Limited

10min poll

Demo: Client searching

Search Box

Server Statistics

Exercise: Finding clients

Find all the windows clients

Find the client that has a user “gladstone”

- When was it installed?

Find client OS release breakdown stats

Solution: Finding clients

Top left search box:

- “windows”
- “gladstone” or “user:gladstone” (faster)

Install date: “First Seen” in client summary line (note all times are UTC)

Show statistics -> Clients -> All -> OS Release Breakdown

Smart Server, basic client

Time travel backwards

Faster build/fix/deploy

Less updating

Simpler backwards compatibility

Leak less intent

Server

Frontends pass messages

Workers do the real work

Everything is asynchronous

Queue work on the server

GRR ‘Cronjobs’ perform regular tasks

Datastore

Abstracted: easy to switch

MySQL Advanced | SQLite (sharded)

Versioned Data -> axis of time

Demo: Settings

Datastore.implementation

Client.control_urls

Note: lines highlighted in blue are modified from defaults.

Demo: VFS browse and download

Refresh, recursive refresh

Multiple versions of /etc/lsb-release

Download new version

Text/Hex views

Exercise: VFS time travel

On client-ubuntu-trusty-m a malicious modification has been made to /home/gcastle/.bashrc

What was it?

Solution: VFS time travel

Browse Virtual Filesystem -> fs -> os -> home -> gcastle -> .bashrc

Click Age window and download latest and oldest. Diff.

Find LD_PRELOAD line.

GRR…
It’s a botnet essentially

Authorization, Auditing

2-party authorization for machine access

DB logging

Audit events

Approval emails with justifications

Demo: Flows/hunts run recently

Show Statistics -> Server -> Flows|Hunts

Fast, reliable, remote.

Advanced live forensics at scale.

Be really really good at collecting

Filesystem/Registry artifacts (Sleuthkit)

Memory artifacts (Rekall)

From difficult-to-specify locations

Demo: Running FileFinder

Search by:

path, name, contents (literal / regex), time

For matches:

download, hash, send to socket, just report existence

Exercise: FileFinder

Pick a windows machine and:

- Get a list of all DLLs (*.dll) in C:\Windows\System32
- Get the partition boot sector C:\$BOOT

  Windows API will hide this! Requires TSK
- There is a file containing the string "malware" in C:\Temp. Try to find it.

Solution: FileFinder

Filesystem->File Finder:

- path: C:\Windows\System32*.dll
- pathtype: OS
- action: STAT

Filesystem->File Finder:

- path: C:\$BOOT
- pathtype: TSK
- action: DOWNLOAD

Solution: FileFinder cont.

Filesystem->File Finder:

- path: C:\Temp*
- pathtype: OS
- condition: contents literal match = malware, FIRST_HIT
- action: DOWNLOAD

Windows Registry

Keys = Directories, Values = Files

Same operations supported!

Globbing

Content match on values

Exercise: RegistryFinder

Get the values for these run keys:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run

(copy from http://pastebin.com/eijGRcFu)

Browse the registry VFS

Solution: RegistryFinder

Registry->Registry Finder:

keys path:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run

Memory Acquisition

Drivers for Win and OS X

Linux is trickier:

- /proc/kcore
- or driver per kernel

Demo: Memory Collector

Download a small chunk of memory

Exercise: Grep raw memory

On a windows client, use the Memory Collector to find a short string (eg. “svchost”) in memory and inspect the context.

Use action NONE

Also, just get the FIRST_HIT, not all of them

Solution: Grep memory

Memory->Memory Collector

Condition: Literal match, FIRST_HIT

Action: NONE (reports the literal match and some context)

Memory Forensics

Memory analysis framework

Built into GRR client

Live memory analysis

Demo: lsmod on ubuntu

Exercise: Rekall lsof

Get a list of file handles from raw memory on a ubuntu machine

Use lsof plugin

Solution: Rekall lsof

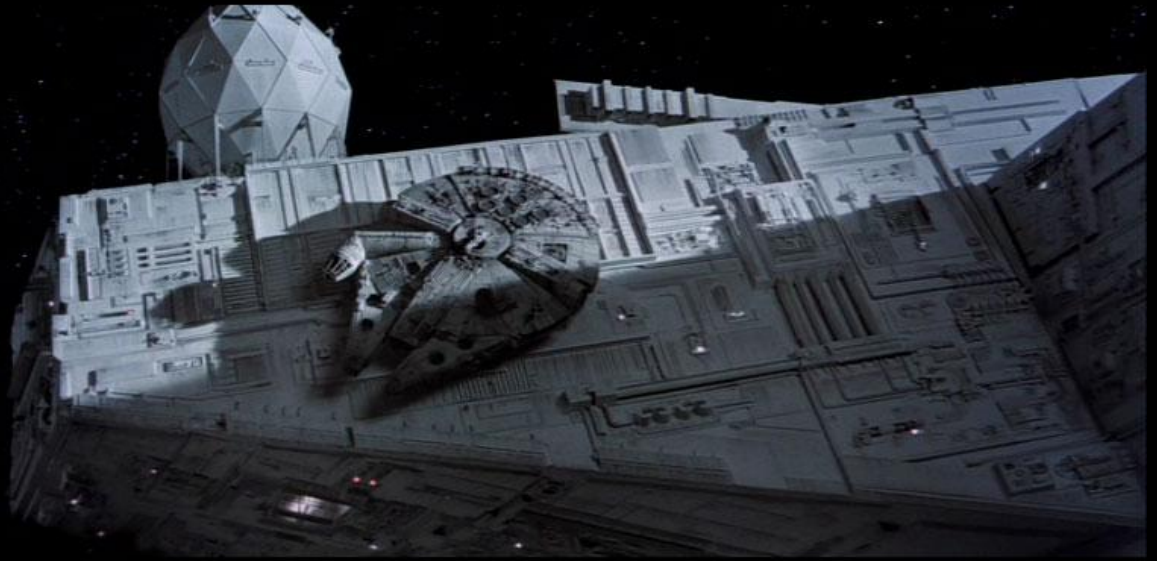
Memory -> AnalyzeClientMemory

Plugins: lsof

Hunting: flows on many machines

Hunting: Outlier analysis

Hunting: fleetcheck and pivot



Demo: Hunt to collect notepad.exe

Download with FileFinder

Export results as .zip

Smart download: only unique files

Exporting data for analysis

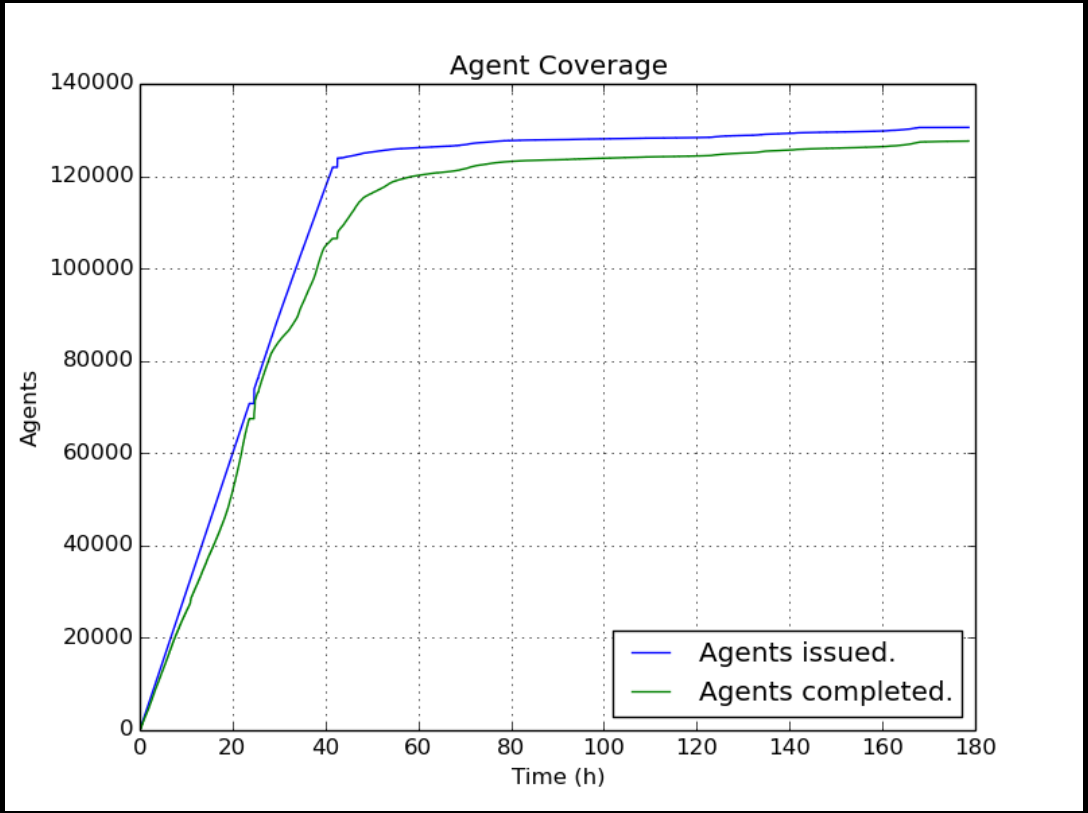
Heavy data analysis outside GRR

HTTP RPC APIs

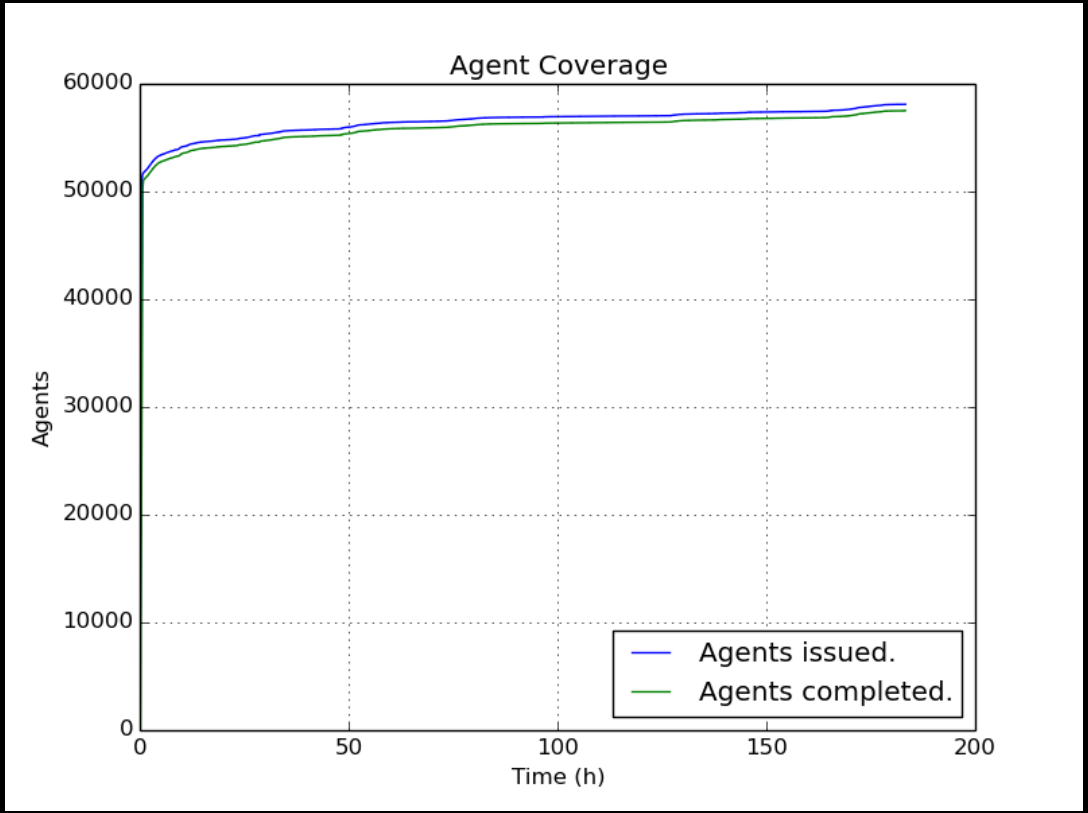
Export plugin system:

- CSV
- <elasticsearch or your plugin of choice here>

Hunts: Optional rate limiting



Hunts: No limit, go fast



Exercise: ListProcesses hunt

Get a list of Processes from all machines using ListProcesses flow

Look at hunt stats:

- Cpu used
- Network used
- Worst performers

Solution: ListProcesses hunt

Hunt Manager -> + -> Processes -> ListProcesses

Remove windows rule to run on all OSes

Press play on the paused hunt

Hunting: Malware inside .doc

Flash exploits embedded in office docs

How could we find these?

Exercise: Hunt for flash inside docs

Find doc with embedded flash in ~\Downloads\

Use %%users.homedir%% for user's homedir
Contains "ShockwaveFlash.ShockwaveFlash"

Solution: Hunt for flash inside docs

Hunt Manager -> + -> Filesystem -> FileFinder

Paths: %%users.homedir%%\Downloads*.doc

Condition: literal match “ShockwaveFlash.ShockwaveFlash” FIRST_HIT

Action: Download

Collection Problems

We mostly want to collect the same things, but:

- Too many details to remember
- No good way to share
- Too much duplicate code

As seen in the wild

HardDrive\Documents and Settings\USERNAME\Local Settings\Application Data\Google\Chrome\User Data\Default\History

HKU\S-1-5-21-xxxxxxxxx-xxxxxxxxx-xxxxxxxxx-xxxx\Software\Microsoft\Windows\CurrentVersion\Uninstall\Dropbox\InstallLocation

/Users/<user>/Library/Mail Downloads/

/home/user/.local/share/Trash/

What do I do with these?

HardDrive\Documents and Settings\USERNAME\Local Settings\Application Data\Google\Chrome\User Data\Default\History

HKU\S-1-5-21-xxxxxxxxx-xxxxxxxxx-xxxxxxxxx-xxxx\Software\Microsoft\Windows\CurrentVersion\Uninstall\Dropbox\InstallLocation

/Users/<user>/Library/Mail Downloads/

/home/user/.local/share/Trash/

Common language for interpolation

%%users.localappdata%%\Google\Chrome\User Data*\History

HKEY_USERS**%%users.sid%%**
\Software\Microsoft\Windows\CurrentVersion\Uninstall\Dropbox\InstallLocation

%%users.homedir%%/Library/Mail Downloads/

%%users.homedir%%/.local/share/Trash/

Artifact

```
name: ApplicationEventLog
doc: Windows Application Event log.
collectors:
- collector_type: FILE
  args: {path_list: ['%%environ_systemroot%%\System32\winevt\Logs\AppEvent.evt']}
conditions: [os_major_version >= 6]
labels: [Logs]
supported_os: [Windows]
urls: ['http://www.forensicswiki.org/wiki/Windows_Event_Log_(EVT)']
```

Artifact repository: get it here

~200 artifacts:

github.com/ForensicArtifacts/artifacts

Independent and reusable by any tool

Used and maintained by us

Review, bug reports, patches very welcome

Demo: Collect Run Keys

Exercise: Artifact Collector

Linux machines are beaconing to sysupdate81.appspot.com

Suspect malicious cronjob

Use AllLinuxScheduleFiles artifact to download cron files

Download results, find malicious one

Which machines was it on?

Solution: Artifact Collector

Hunt Manager -> + -> Collectors -> ArtifaceCollectorFlow

AllLinuxScheduleFiles

GenerateZip

Download, unzip:

grep -r “sysupdate” *

find -type l -ls | grep [hash match from grep]

What’s coming

Event triggered collection, powerful API

Usability improvements

Simple cloud server deployment

More data export options

Great, how do I try it?

Run the server docker image

Open a browser

Download and install the client on a machine

GRR (and friends) links

github.com/google/grr

github.com/ForensicArtifacts/artifacts

rekall-forensic.com

plaso.kiddaland.net/

github.com/google/timesketch

github.com/libyal/libyal/wiki/Overview

These slides

These slides and everything you need to run your own workshop will be published here:

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

Short link: https://goo.gl/GzsleU