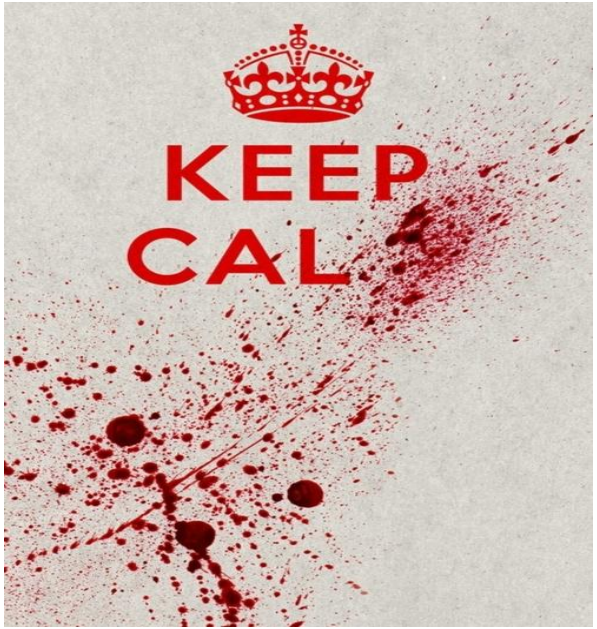
GRR Rapid Response

Practical IR with GRR
OSDF 2013



Darren Bilby, Joachim Metz - Google

Agenda

Presentation: GRR Architecture

Exercise 1: Installation and doing something useful

Presentation: How flows work

Presentation: Customization

Exercise 2: Client Customization

Break - 15 Minutes

Presentation: VFS and Pathspecs

Presentation: Audit controls

Exercise 3: Using the Console

Presentation: Hunting

Break 15 Minutes

Presentation: Getting data out of GRR

Exercise 4: Running a Hunt

Presentation: Artifacts

Exercise 5 Artifacts

Today

Extra instructions at:

https://code.google.com/p/grr/wiki/OSDFWorkshopInfo2013

Ask any question you want

Instructors:

Darren Bilby - Google

Joachim Metz - Google

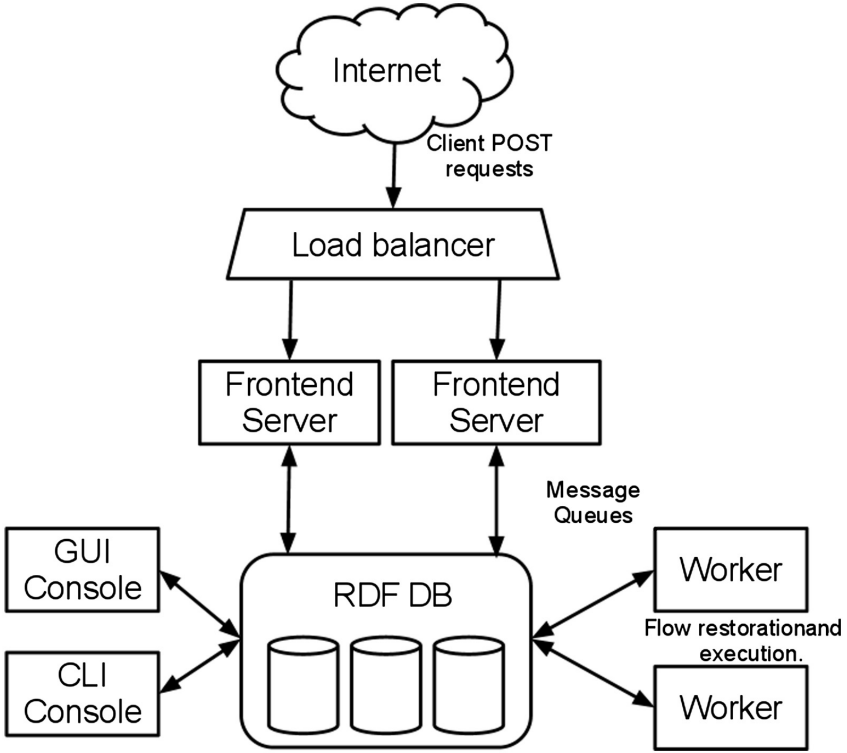
What is GRR?

https://code.google.com/p/grr/wiki/ProjectFAQ

- Built, maintained, used by Google…. and others
- GRR Rapid Response (no Google)
- Long term support
- Prioritized for IR capabilities but used for other things
- Built by engineers for engineers

Architecture

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



Exercise 0

- Install the server as per:
  - https://code.google.com/p/grr/wiki/GettingStarted
- Check the user manual for downloading clients
  - http://grr.googlecode.com/git/docs/user_manual.html#_downloading_agents

Communications

- Client polls the server for work
- Defaults to once every 10 minutes
- Client backs off
- Messages are protobufs
- Signed and encrypted end to end
- Unique key-pair generated at enrolment time

Client

- Python code compiled with pyinstaller
- Single directory
- Logging to syslog, event log, file
- Windows zip installer
- Linux deb/rpm
- OSX pkg file
- Resource constraints

Datastore

- Built on Mongodb
- Can also run on Mysql
- Abstraction makes replacing it easy

- Built on AFF4
  - Every object has a URN, and some attributes

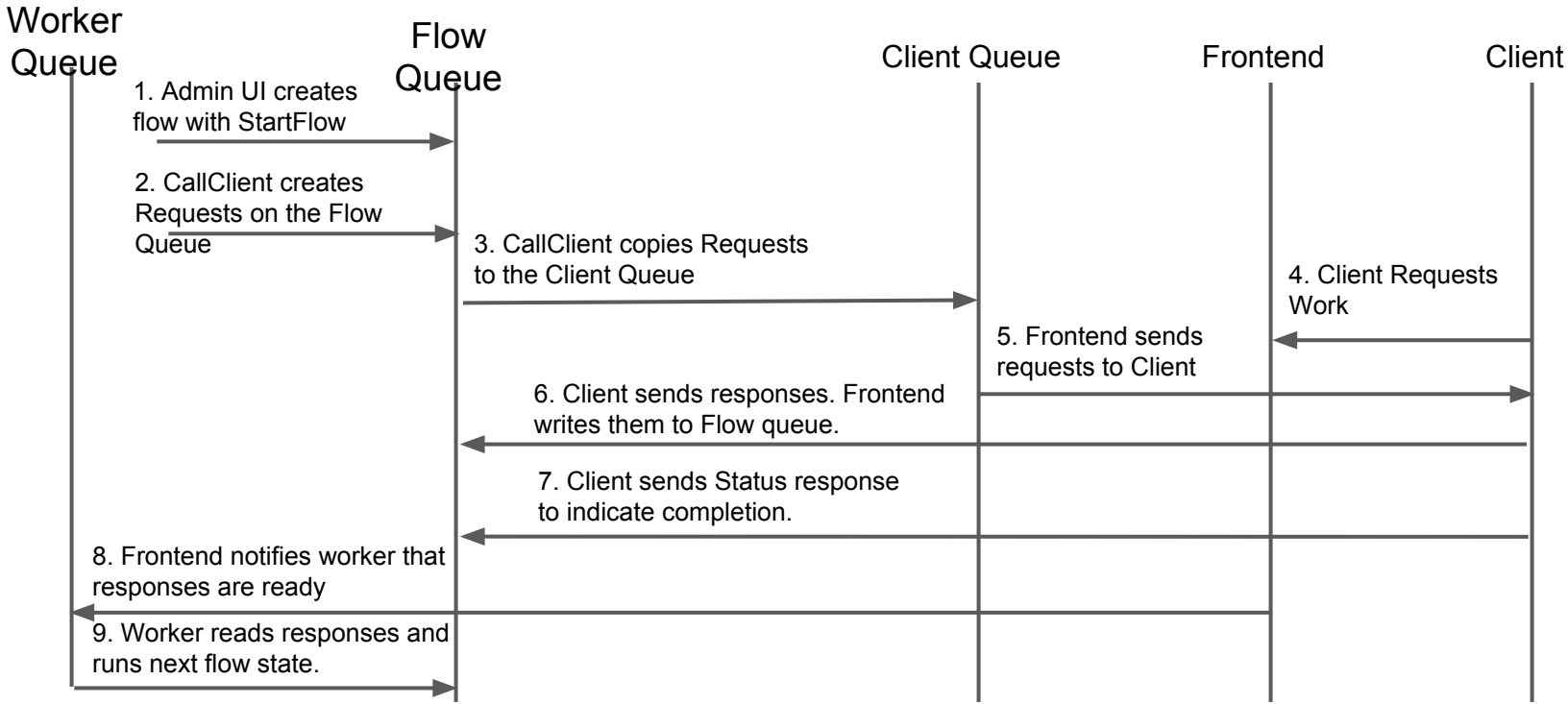
Exercise 1

Do something useful With GRR

Instructions at:

https://code.google.com/p/grr/wiki/OSDFWorkshopInfo2013

Life of a Flow



Life of a Flow

Step1: Flow created

1. StartFlow run, `Start` state is executed
2. Create the flow object `aff4:/C.0000000000001/flows/W:B1C77B76`
3. Create requests in the flow
4. Copy requests to the Client Queue `aff4:/C.0000000000001/tasks`

Grep Memory https://code.google.com/p/grr/source/browse/lib/flows/general/memory.py#462

WIN-KFDDWDYJ6CV
Status: 6 seconds ago.
ip-10-195-94-74.ec2.internal
Host Information
Start new flows
Browse Virtual Filesystem
Manage launched flows
Advanced
Client Performance Stats
Crashes
Debug Client Requests
MANAGEMENT
Automated flows
Cron Job Viewer
Hunt Manager
Show Statistics
Start Global Flows
Advanced
CONFIGURATION
Manage Binaries
Settings

State	Path	Flow Name	Creation Time	Last Active	Creator
	W:E6DF1C12	GrepMemory	2013-11-03 01:24:43	2013-11-03 01:24:43	admin

Flow Information | Requests

ID	Request		Last Response
flow:request:00000001	Id	1	
	Next state	Grep	
	Response count	0	
	Client id	aff4:/C.8ce383738bc72bde	
	Session id	aff4:/C.8ce383738bc72bde/flows/W:E6DF1C12	

WIN-KFDDWDYJ6CV
Status: 36 seconds ago.
ip-10-195-94-74.ec2.internal
Host Information
Start new flows
Browse Virtual Filesystem
Manage launched flows
Advanced
Client Performance Stats
Crashes
Debug Client Requests
MANAGEMENT
Automated flows
Cron Job Viewer
Hunt Manager
Show Statistics
Start Global Flows
Advanced
CONFIGURATION
Manage Binaries
Settings

Status	ID	Due	Flow	Client Action
	15821113764923517006	2013-11-03 01:24:43	aff4:/C.8ce383738bc72bde/flows/W:E6DF1C12/W:DF7C5538	GetMemoryInformation

Request | Responses

Request 15821113764923517006

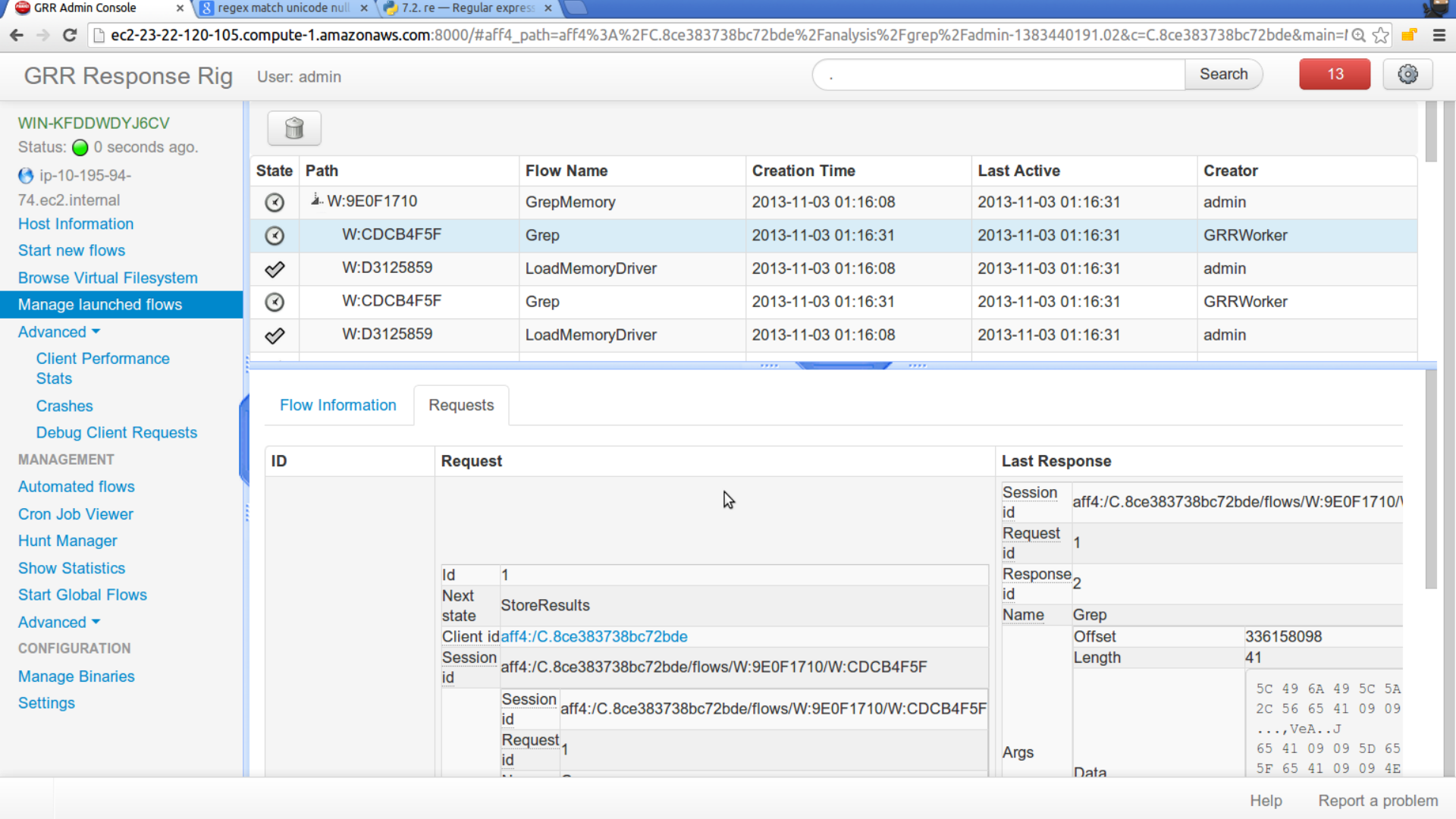
Task

Session id	aff4:/C.8ce383738bc72bde/flows/W:E6DF1C12/W:DF7C5538	
Request id	1	
Name	GetMemoryInformation	
Args	Pathtype	MEMORY
	Path	\\.\pmem
Priority	MEDIUM_PRIORITY	
Args rdf name	PathSpec	
Task id	15821113764923517006	
Queue	aff4:/C.8ce383738bc72bde/tasks	
Eta	1383441883968340	

Life of a Flow

Step 2: Client picks up requests

1. Client requests are marked leased for 10 minutes
2. Client sends multiple responses back using SendReply
3. Frontend writes responses to the flow state `aff4:/C.0000000000001/flows/W:B1C77B76/state`
4. Client sends a final Status reply
5. Frontend sees the Status and tells the worker to process the Flow by writing to queue `aff4:/W`
6. Cleans out requests


GRR Admin Console
regex match unicode null
7.2. re — Regular expres
ec2-23-22-120-105.compute-1.amazonaws.com:8000/#aff4_path=aff4%3A%2FC.8ce383738bc72bde%2Fanalysis%2Fgrep%2Fadmin-1383440191.02&c=C.8ce383738bc72bde&main=I
GRR Response Rig
User: admin
Search
13
WIN-KFDDWDYJ6CV
Status: 0 seconds ago.
ip-10-195-94-74.ec2.internal
Host Information
Start new flows
Browse Virtual Filesystem
Manage launched flows
Advanced
Client Performance Stats
Crashes
Debug Client Requests
MANAGEMENT
Automated flows
Cron Job Viewer
Hunt Manager
Show Statistics
Start Global Flows
Advanced
CONFIGURATION
Manage Binaries
Settings
State
Path
Flow Name
Creation Time
Last Active
Creator
W:9E0F1710
GrepMemory
2013-11-03 01:16:08
2013-11-03 01:16:31
admin
W:CDCB4F5F
Grep
2013-11-03 01:16:31
2013-11-03 01:16:31
GRRWorker
W:D3125859
LoadMemoryDriver
2013-11-03 01:16:08
2013-11-03 01:16:31
admin
W:CDCB4F5F
Grep
2013-11-03 01:16:31
2013-11-03 01:16:31
GRRWorker
W:D3125859
LoadMemoryDriver
2013-11-03 01:16:08
2013-11-03 01:16:31
admin
Flow Information
Requests
ID
Request
Last Response
Id 1
Next state StoreResults
Client id aff4:/C.8ce383738bc72bde
Session id aff4:/C.8ce383738bc72bde/flows/W:9E0F1710/W:CDCB4F5F
Session id aff4:/C.8ce383738bc72bde/flows/W:9E0F1710/W:CDCB4F5F
Request id 1
Session id aff4:/C.8ce383738bc72bde/flows/W:9E0F1710/
Request id 1
Response id 2
Name Grep
Offset 336158098
Length 41
Args
Data
5C 49 6A 49 5C 5A
2C 56 65 41 09 09
...,VeA..J
65 41 09 09 5D 65
5F 65 41 09 09 4E
Help
Report a problem

GRR Response Rig User: admin Search 13

WIN-KFDDWDYJ6CV
Status: 0 seconds ago.
ip-10-195-94-74.ec2.internal

- Host Information
- Start new flows
- Browse Virtual Filesystem
- Manage launched flows
- Advanced
  - Client Performance Stats
  - Crashes
  - Debug Client Requests

MANAGEMENT

- Automated flows
- Cron Job Viewer
- Hunt Manager
- Show Statistics
- Start Global Flows
- Advanced

CONFIGURATION

- Manage Binaries
- Settings

Status	ID	Due	Flow	Client Action
	1581899916783973865 2	2013-11-03 01:26:31	aff4:/C.8ce383738bc72bde/flows/W:9E0F1710/W:CDCB4F5F	Grep

Request | Responses

Task ID	Response		
flow:response:00000001:00000001	Session id	aff4:/C.8ce383738bc72bde/flows/W:9E0F1710/W:CDCB4F5F	
	Request id	1	
	Response id	1	
	Name	Grep	
	Args	Offset	312066850
		Length	41
		Data	C6 C6 C6 C6 39 39 39 39 1B 2C 56 65 41 09 09 4A ....9999.,VeA..J 65 41 09 09 5D 65 41 09 09 5F 65 41 09 09 4E 71 eA..]eA.._eA..Nq

Life of a Flow

Step 3: Worker processes responses

1. Once notified, a worker picks up requests
2. Checks for Status message and ensures all responses are complete
3. Worker loads the flow and executes the next state with the responses returned

```
@flow.StateHandler()
  def Done(self, responses):
```

Flow Summary

- Basic building blocks that can be chained together to do more complex tasks
- Completely asynchronous

Client Customization

Customization

- GRR is open source
- Attackers will end up knowing to look for it

Goal:

"The attacker has the same problem as us. They land on a machine, and have to find the response agent… just like we have to find the malware."

Customization

Building Clients

1. Use existing template and repack
    - registry keys, service names, logs
2. Install dependencies, build from source
    - Absolutely anything

GRR Configuration System

- 3 Levels
  - Default values in code
  - Master config file /etc/grr/grr_server.yaml
  - Overrides Config.writeback /etc/grr/server.local.yaml
  - Overrides in --secondary_config

- Handles multiple "Contexts"

GRR Client Configuration

How your config gets added to the client:

1. Apply the correct context to the server config
2. Extract the relevant variables for the client
3. Write the client config yaml file
4. Inject it into the template

Exercise 2

Exercise 2: Client Customization

- Edit the config to build a client with a different name
- Repack the new client and install it

VFS and Pathspecs

- Virtual filesystem
  - OS
  - TSK
  - Registry
  - Derived files
- AFF4 Namespace and System Namespace
- Mappings between are Pathspecs

Pathspecs

- aff4:/C12345/fs/os/C:
- Literal vs Case Insensitive
- Handles recursion

VFSFile			
PATHSPEC	Pathtype	OS	
	Path	/\\?\Volume{853d6ac8-41c5-11e3-9b94-806e6f6e6963}	
	Nested path	Pathtype	TSK
		Path	/$MFT
		Path options	CASE_LITERAL
		Inode	0
	Path options	CASE_LITERAL	
	Aff4path	aff4:/C.8ce383738bc72bde/fs/tsk/\\?\Volume{853d6ac8-41c5-11e3-9b94-806e6f6e6963}/$MFT	
	St mode	r-xr-xr-x	
	St ino	0	
	St nlink	1	

The Console

- IPython
  - Explore and execute code
  - <tab><tab> ? ?? cpaste
  - x = !ls /etc/

- grr_console
  - Raw interface to everything GRR

Audit Controls

- GRR is remote root equivalent
- Audit controls
  - Multi-party authorization
  - Audit hooks
  - Gateway mechanism to allow console with audit
- Made possible by passing ACLToken objects
  - User, reason, expiry
  - You can mostly ignore them or set None

Exercise 3: Using the Console

- Searching for clients
- Starting flows
- Reading data from console

Hunts

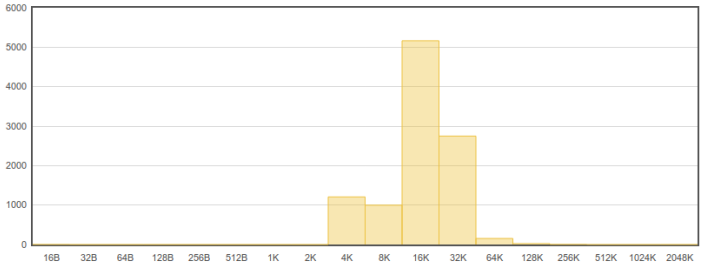
- Specialized Flows that run on multiple clients
- Scheduled using rules
- Maintain detailed statistics and outliers

- SendReply data from all flows go to one place

Hunt Stats

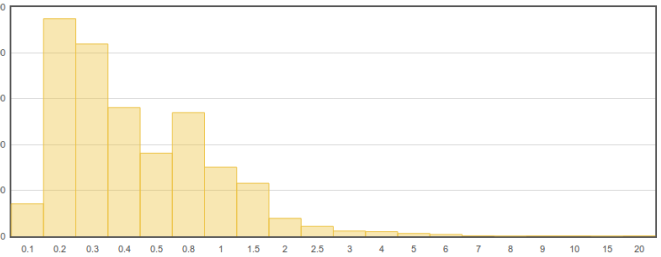
Network bytes sent

Network bytes sent mean 14521.8
Network bytes sent stdev 8035.0
Clients Hisogram



User CPU

User CPU mean 0.5
User CPU stdev 1.0
Clients Histogram



Worst performers

Client Id	User CPU	System CPU	Network bytes sent
aff4:/C.26289f627ee6445d	44.0	3.8	239853
aff4:/C.ab45e712f59b2d2b	43.9	3.5	239437
aff4:/C.153af81376c4b36b	35.2	3.8	93218
aff4:/C.0bddc1bc95f9c95b	15.4	2.8	15695
aff4:/C.f6c6d3af360b21f4	14.1	1.1	80153
aff4:/C.5125d4db87d89f3d	10.4	0.7	73760
aff4:/C.29ee79aff6e06683	10.2	0.7	114687

Exercise 4: Running a Hunt

1. Run a hunt on all Windows machines to retrieve the Event Logs
2. Figure out how much CPU it used on the machine to do that
3. Run the file exporter to dump the results to the disk using the command line

Artifacts

- Flows are too tricky for simple things
- We wish we could share information better
- Too much duplicate code

----------> Let's generalize to Artifacts

- Define what to collect
- Define how to parse it
- Define what they produce

Artifacts

Knowledge Base Interpolation

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

%%systemroot%% → c:\Windows\System32

%%users.name%% → c:\Documents and Settings\foo\AppData\Roaming

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

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

https://code.google.com/p/grr/source/browse/proto/knowledge_base.proto

Artifacts: Path Syntax

All Chrome History Files can be written as:
%%users.localappdata%%\Google\Chrome\User Data*\History

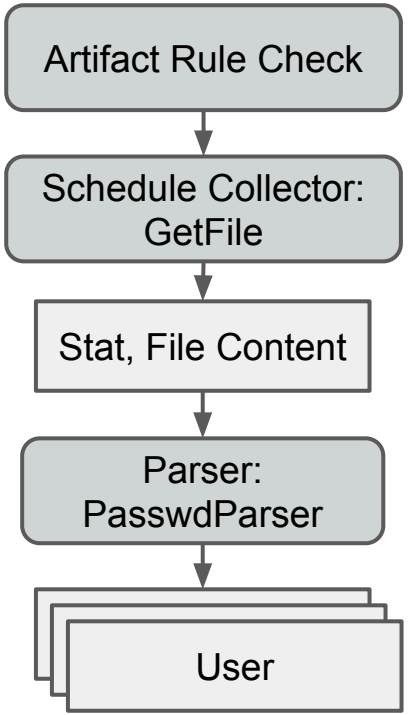
```
COLLECTORS = [
    Collector(action="GetFiles",
              args={"path_list": ["%%users.localappdata%%\\Google\\Chrome\\User"
                                  "Data\\*\\History"]},
             )
 ]
```

- Works across all Windows versions
- No need to remember paths!

Artifacts: Simple File Artifact

```
class LinuxPasswd(Artifact):
  """Linux passwd file."""
  SUPPORTED_OS = ["Linux"]
  LABELS = ["Authentication"]
  COLLECTORS = [
     Collector(action="GetFile",
               args={"path": "/etc/passwd"},
              )
  ]
```

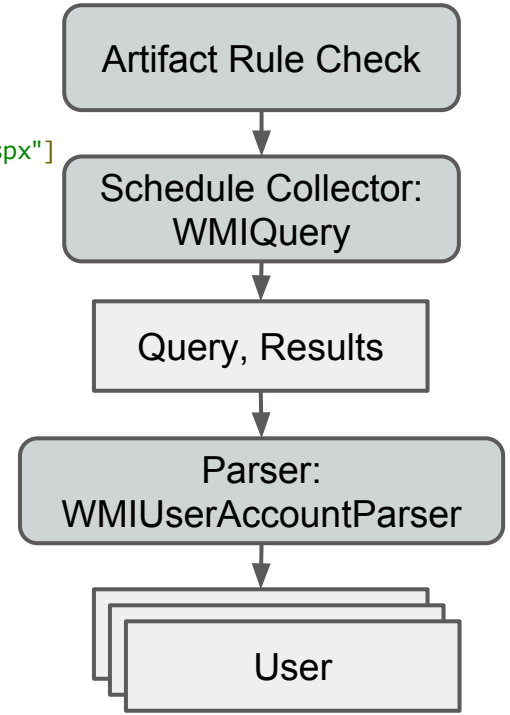
```
class PasswdParser(parsers.CommandParser):
  """Parser for passwd files. Yields User semantic values."""
  output_types = ["User"]
  supported_artifacts = ["LinuxPasswd"]
  def Parse(self, stat, file_object, knowledge_base):
    """Parse the passwd file."""
    ...
    yield user
```



Artifacts: Simple Artifact

```
class WindowsWMIProfileUsers(Artifact):
  """Get user information based on known user's SID.
  ....
  """
  URLS = ["http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507(v=vs.85).aspx"]
  SUPPORTED_OS = ["Windows"]
  LABELS = ["Users", "KnowledgeBase"]
  COLLECTORS = [
     Collector(action="WMIQuery",
               args={"query": "SELECT * FROM Win32_UserAccount "
                              "WHERE name='%%users.username%%'"}
              )
  ]
  PROVIDES = ["users.username", "users.userdomain", "users.sid"]

class WMIUserAccountParser(parsers.WMIQueryParser):
  """Parser for WMI Win32_UserAccount output."""
  output_types = ["KnowledgeBaseUser"]
  supported_artifacts = ["WindowsWMIProfileUsers"]
  def Parse(self, query, result, knowledge_base):
    """Parse the wmi Win32_UserAccount output."""
   ...
   yield kb_user
```



Exercise 5: Create an Artifact

- Make our own simple Artifact
- /usr/share/pyshared/grr/artifacts/
- Start with WinHostsFile

- Bonus time: Create a parser

The End...

grr-users@googlegroups.com

grr-dev@googlegroups.com