Rekall Forensic

We can remember it for you wholesale!

Michael Cohen Google Inc.
scudette@gmail.com

Rekall in a nutshell

- Rekall started life as Memory Forensic tool with a focus on
  - Speed
  - Reliability - supports more Operating systems reliably out of the box.
    - Automatic profile selection from a vast library.
    - Huge library of Windows kernel profiles.
    - Transparently support common Linux distributions through profile indexes.
  - Ease of integration into other tools (Simple JSON based API)

- Rekall has always had a focus on live analysis:
  - The Winpmem, MacPmem and LinPmem drivers provide direct access to the physical memory.

Rekall can now use the APIs

- Some things are faster and more reliable to do through the API
  - Memory analysis is still kind of fragile - have to have the right profiles.
  - Sometimes it is an overkill e.g.:
    - Detecting simple DLL injection is faster and more effective through the OS APIs. Kernel level subversion is not often used.
    - Yara scanning processes and user space is often more reliable (No need to worry about paging or smear).
- Rekall has live modes
  - --live Memory - Automatically inserts memory drivers, loads profiles and analyzes memory directly.
  - --live API - Switches Rekall into API mode where many plugins are available to use the API:
    - WMI - issue WMI queries.
    - Glob - file system based plugins
    - Yara scanners for files, process memory etc.

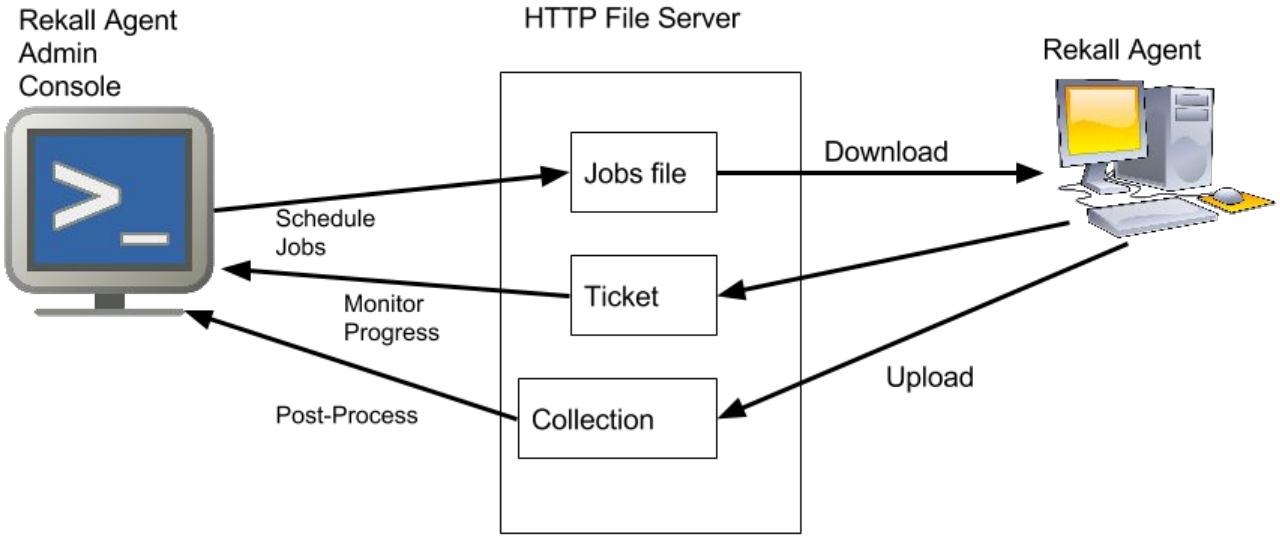
Rekall is focused on live analysis and collection.

- With the emergence of API based plugins we can implement some of the common collection tools:
  - **Forensic Artifacts** are community maintained tool agnostic recipes for collection of forensically critical information.
  - Timelining is a commonly used technique

The Rekall Agent

- GRR like agent focused on collection, performance and ease of deployment
- Was born from a thought experiment - Redesign GRR:
  - What if it was really simple?
  - What if it was easy and cheap to deploy?
- Everything is a file!
  - Just copy files around - turns out we are really good at doing this in scale!
- The Rekall Agent is simply:
  - Get a JSON message describing what to do.
  - Does it and prepares files (Result Collections, File Uploads)
  - Upload the files to the specified location.
- The Rekall Agent Controller:
  - Creates the JSON file instructing the client.
  - Collect and manage the result files.

Architecture Overview



Installing the Rekall Agent - Create Bucket

Google Cloud Platform Google.com Rekall

Home
API Manager
Billing
Cloud Launcher
Support
IAM & Admin

COMPUTE

App Engine
Compute Engine

Storage

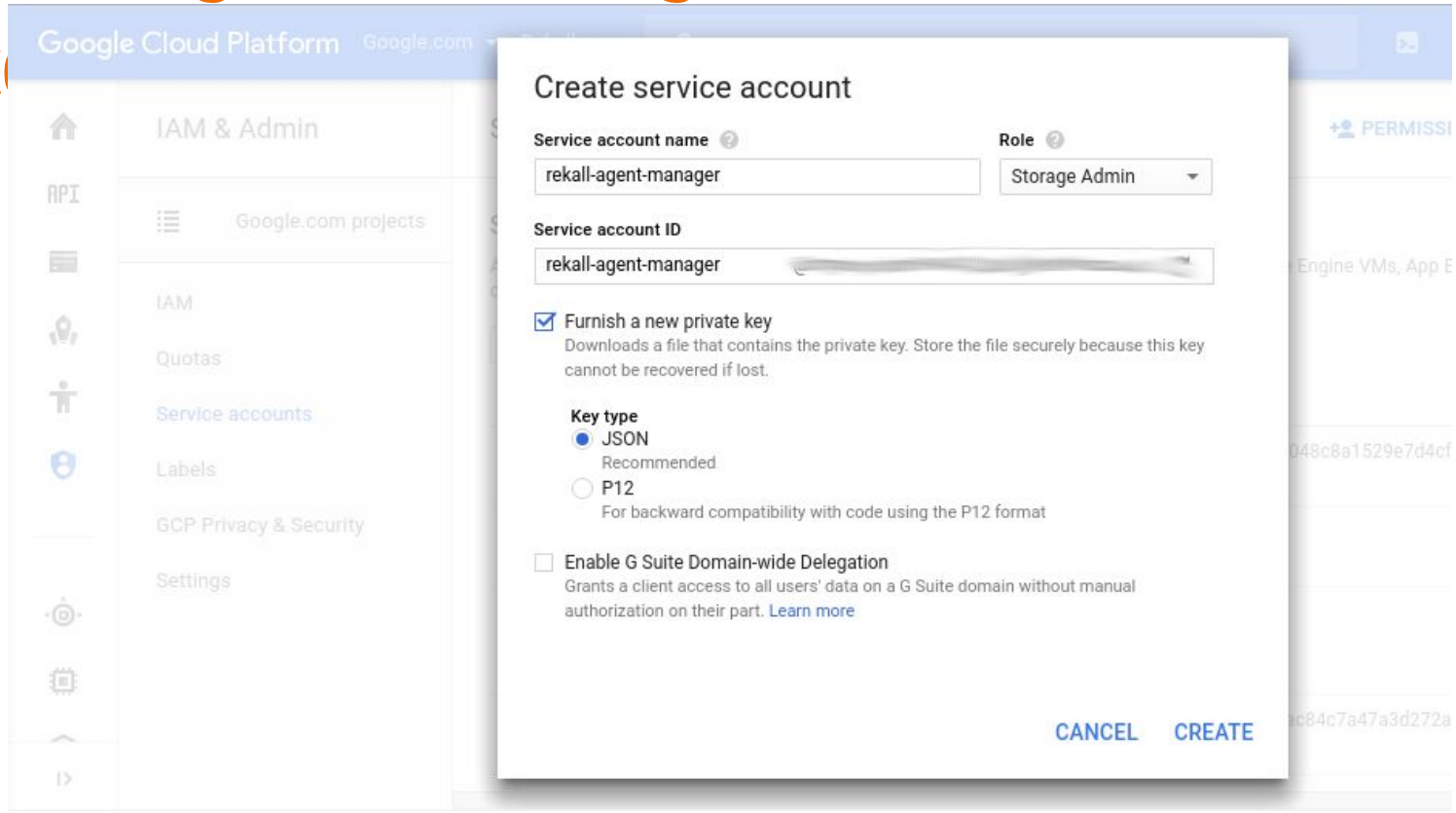
Browser
Transfer
Settings

Browser CREATE BUCKET REFRESH DELETE

Buckets Filter by prefix...

Name	Storage class	Location
dfrws2016.rekall-forensic.com	Standard	US
images.rekall-forensic.com	Standard	US
pypi.rekall-forensic.com	Standard	US
rekall-aff4-test	Standard	US
rekall-images	Standard	EU
rekall-temp	Standard	US
releases.rekall-forensic.com	Standard	US

Installing the Rekall Agent - Create Service Ac



Installing the Rekall Agent - Configure Clients.

Service Account JSON file (from Cloud Console)

Deployment bucket name

Config files directory

```
(Dev) scudette@scudette-glaptop:~/rekall-agent/rekall/rekall-agent$ rekal agent_server_initialize_gcs /tmp/my_new_config/ -servi
ce_account_path ~/.rekall_agent_service_account --bucket rekall-temp --client_writeback_path /tmp/agent.local.json
Message
-------
Generating new CA private key into /tmp/my_new_config/ca.private_key.pem and /tmp/my_new_config/ca.cert.pem
Generating new Server private keys into /tmp/my_new_config/server.private_key.pem and /tmp/my_new_config/server.certificate.pem
Writing client config file /tmp/my_new_config/client.config.yaml
Writing server config file /tmp/my_new_config/server.config.yaml
Writing manifest file.
Writing manifest file to bucket rekall-temp path manifest
Done!
```

Rekall Management Console config file

Client's config file

Manifest file uploaded to bucket

Enrolling Client

- Rekall Agent is a zero configuration client.
  - Client is installed with a deployment config file.
  - The Client writes state into the client (Termed the **writeback** location)
    - Remembers its client ID, Client's private and public keys,
    - Last timestamp of executed server job (Termed **Flow**).
    - Currently executing action (to track crashes).
  - When the client is run for the first time it generates its own keys and client ID
- The Client's Startup sequence
  - Contact the manifest file of the deployment from its config file.
  - Verifies the manifest and executes any commands present there (termed the **Interrogate** flow).
  - Starts to query its message queues for commands.
- Note - no server side support required for clients to become operational.

Rekall Agent Result Collections

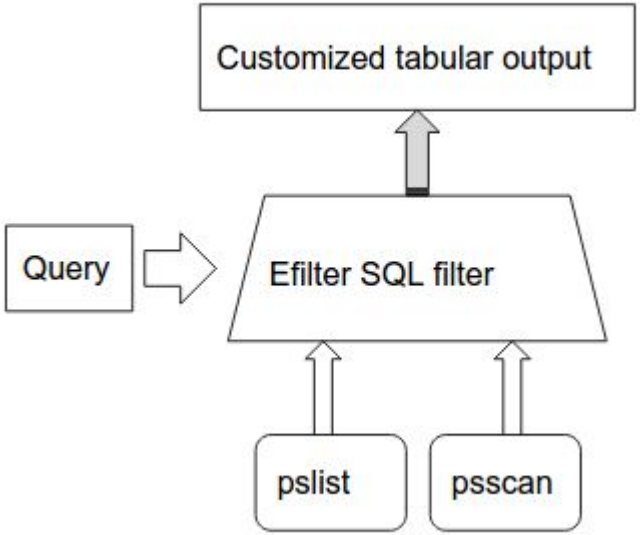
- At the end of the day the Rekall Agent produces Result Collections.
- A stand alone file with results of an EFilter Query over Rekall plugins
  - Currently SQLite files.
  - One Result Collection per Efilter query.
  - Note that EFilter queries can combine results from several Rekall plugins.
- An example query:

```
select proc.name, proc.pid, ppid, start_time from pslist()
```

What is efilter?

SQL like query language for Rekall plugins

Efilter queries can combine the output from several plugins in a flexible way.



Glob for all *.exe files in the windows directory, and yara scan them for "Microsoft" encoded in UTF16.

```
glob "c:\windows\*.exe"
```

Now use the search plugin to insert query parameters:

```
plugins.search('select * from
file_yara(paths: (select
path.filename from
glob("c:\windows\*.exe")).filename,
binary_string: {str})',
query_parameters=dict(str="Microsoft"
.encode("utf-16-le").encode("hex")))
```

Rekall Agent Flows

- A Flow is a logically related sequence of requests directed at the client.
  - Each action will have its own Result Collection
  - All actions run at approximately the same time.
- For example, Collect the following artifacts: Running Processes, Open sockets, Registry artifacts.
- Flows can have a pre condition:
  - Precondition is an efilter expression so it can use the output of any plugin.
  - For example, collect all running processes if a registry key is present.
- Flows are directed at each client:
  - They are written into the client's private jobs queue
  - Result collections are collected under the client's namespace in the file store.

Rekall Agent Hunts

- Sometimes we want to run the same flow on multiple clients at once.
  - Rather than write the flow object to 10,000 individual queues we simply post the job message on a shared queue.
  - All clients query the **All** queue (i.e. they constantly read that file).
- Client participation in the hunt is based on self selection:
  - Hunts typically have a pre-condition, for example:
    - Run this hunt on all windows machines

```
Select * from agent_info() where key=='system' and
value=='Windows'
```

    - Run this hunt on all machines with this process running

```
Select * from pslist() where regex_search(Name,
```

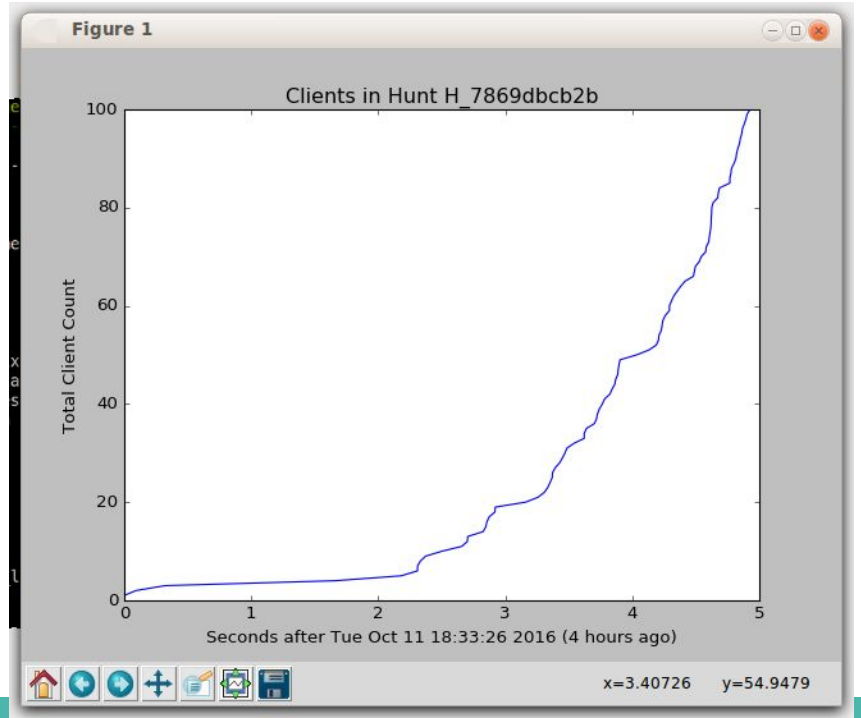
How do I check 100,000 hosts for badness?

- Step 1: Drop a file on a cloud bucket

```
[1] Rekall Agent 19:05:34> launch_hunt "ListProcessesFlow"
-------------------------> launch_hunt("ListProcessesFlow")
      Field                                   Value                                                   Description
------------------- ------------------------------------------------------------------------------- -----------
  client_id         C.4dd70be22bc56fc3                                                              A client id to target this flow on.
  queue             All                                                                             A queue to launch this one. When specified this flow is run as a hunt.
  flow_id           F_224c8bed27                                                                    Unique ID of this flow, will be populated when launched.
  created_time      2016-10-07T19:05:34.820019-07:00                                                When the flow was created.
  ticket            {'status': u'Started', '__type__': 'HuntStatu ...                               Ticket keeping the state of this flow.
. location          {'GoogleAccessId': u'rekall-test@crypto-prism ...                               Where the ticket should be written.
.. bucket           rekall-temp                                                                     Name of the bucket
.. policy           eyJjb25kaXRpb25zIjogW1sic3RhcnRzLXdpdGgiLCAiJ ...                               The policy document.
.. signature        Yx4jstX9Cy9+ZzVXqwuPqEnttpVRYiyQHqbnkFKuw3vbL ...                               The signature to use when accessing the resource.
.. path_prefix      tickets/HuntStatus/F_224c8bed27                                                 Access is allowed to all paths starting with this prefix.
.. path_template    {client_id}                                                                     A template from which to expand the complete path.
.. GoogleAccessId   rekall-test@crypto-prism-89412.iam.gserviceac ...                               The email form of the service account id
.. expiration       1475978734                                                                      When the url expires.
. client_id         C.4dd70be22bc56fc3
. flow_id           F_224c8bed27
. status            Started
  actions           [{'query': {'mode_linux_memory': 'select proc ...                               The action requests sent to the client.
  session           {'live': u'API', '__type__': 'RekallSession'}                                   The session that will be invoked for this flow.
. live              API                                                                             The Rekall live mode
```

How do I check 100,000 hosts for badness?

- Step 2: Stand back



How do I check 100,000 hosts for badness?

- Step 3: Inspect the results

```
[1] Rekall Agent 19:09:49> inspect_hunt "F_224c8bed27"
-------------------------> inspect_hunt("F_224c8bed27")
-        Field                  Time          Value
  ------------------- -------------------- -----
------------------------------
Flow Object (ListProcessesFlow)
------------------------------
  client_id                                 C.4dd70be22bc56fc3
  queue                                     All
  flow_id                                   F_224c8bed27
  created_time                              2016-10-07T19:05:34.820019-07:00
  session                                   {'live': u'API', '__type__': 'RekallSession'}
  live                                      API
-------
Summary
-------
  Total Clients                             3
  Successful Clients                        3
  Errors Clients                            0
-------
Results
-------
  C.4dd70be22bc56fc3   2016-10-07 19:05:40  rekall-temp/hunts/F_224c8bed27/vfs/analysis/pslist_1475892334/C.4dd70be22bc56fc3
  C.d688903e59d52459   2016-10-07 19:09:19  rekall-temp/hunts/F_224c8bed27/vfs/analysis/pslist_1475892334/C.d688903e59d52459
  C.8409b5bb14099483   2016-10-07 19:09:10  rekall-temp/hunts/F_224c8bed27/vfs/analysis/pslist_1475892334/C.8409b5bb14099483
```

The Rekall Agent is released for comments

- This is an alpha release of the Rekall Agent.
  - We want to hear feedback about the design - I know its radical and very different from existing solutions.
  - Please do not actually deploy this widely yet!
- What works now:
  - Linux support for both API and Memory.
  - Cloud based deployment for Google Cloud Storage.
  - Stand alone HTTP server for own server installs
- Planned for the final release (Real soon now):
  - End to End encryption - no plaintext in cloud.
  - More testing of Windows support.
  - Better packaging (deb, dpkg, msi)
- Contributions needed!
  - Also feature requests

Conclusions

- Rekall's scope has expanded into the complete IR life cycle.
  - Acquisition and Analysis.
  - Memory and Live artifacts.

- We launch the Rekall Agent proof of concept.
  - A scalable and easy to deploy remote agent solution.

http://rekall-forensic.blogspot.com/2016/10/the-rekall-agent-whitepaper.html

http://www.rekall-forensic.com/