GRR Rapid Response

An exercise in failing to replace yourself with a small script.

Darren Bilby - Digital Janitor - Google Tech Lead Incident Response / Forensics
Agenda

● Why GRR?
● What we built
● Demo 1
● Key Design decisions
● Demo 2
● Roadmap
Why GRR?

- Tell me if this machine is compromised
- Joe saw something weird, check his machine
- Why did a packet containing "fooooo" go from A to B?
- Forensically acquire 25 machines for analysis
Why GRR?

● Tell me if this machine is compromised
  ○ (while you're at it, check 20000 of them)

● Joe saw something weird, check his machine
  ○ (p.s. Joe is on holiday in Cambodia and on 3G)

● Why did a packet containing "fooooo" go from A to B?
  ○ (by the way, we're not sure what A was)

● Forensically acquire 25 machines for analysis
  ○ (p.s. they're in 5 continents and none are Windows)
Things We Want

- Make our open tools "enterprise" capable
- Remote access to investigate machines
- Scale to 100K+ machines easily
- Work over the Internet securely
- Work across OSX/Linux/Windows
Things We Want

- Automation should be easy
  - and shouldn't be tied to a vendor's product

- Should be my memory
  - Remember the details about artifacts
  - Know anomalies

- Allow multiple people to work a case at once

- Customizable
What We Wanted
What We Built
What We Built

- Agent based system (Windows, OSX, Linux)
- Communicates over the Internet on HTTP
- Scalable backend
- Ajax UI
- Enables most common IR/Forensics tasks

- Open source (Apache/GPL Dual Licensed)
- Mongo NoSQL backend
- Python compiled to exe/elf/mach-o
- Comms over encrypted, signed protobufs
Demo Time

- Install a new agent
- Collect some artifacts
- Show filesystem view
- View browser history
- List processes extracted from memory
Key Design Decisions

● Thin vs thick client
● Asynchronous Flows
● Axis of Time
Example: Directory Listing

**Server**
- OS
- ListDirectory

**Thick Client**
- Sleuthkit
  - ListDirectory(c:\Windows\System32)

**Thin Client**
- Open(\\PhysicalDrive0)
  - Seek(2232)
  - Read(1024)

**Buffer**
- Read(\\PhysicalDrive0)
  - offset 2232, 1024 bytes
Thin Client vs Thick Client

- No client updates for new functionality
- Raw data stored for future analysis.
- Reduced attacker visibility
- Reduced attacker subversion options

- Decreased network traffic
- Decreased server complexity

Decision: Let's do both
Scale - Asynchronous Flows

- Plan for 500,000+ clients
- Collect 8GB memory from 1k clients at once

- Individual clients cannot "hold" resources
- Only limited by CPU/Memory/Disk available
- Grow as needs grow
Scale - Asynchronous Flows

Initial State
CallClient("HashFile","foobar.iso", next_state="CheckHash")

CheckHash
if not HashInDB(args.hash):
    CallClient("ReadBuffer","foobar.iso",
    offset = 0, length = 4096,
    next_state = "WriteBlock")

WriteBlock
WriteFile(args.data)
if args.offset < file_length:
    CallClient("ReadBuffer","foobar.iso",
    offset = args.offset + 4096, length = 4096,
    next_state = "WriteBlock")
else: goto_state("Done")

Done
VerifyHash("foobar.iso")
Axis of Time

- Live forensics is a smear
- With scalable storage comes snapshots
- Historical record of artifacts
- Enables statistical analysis

- What has changed on this system this week?
- What are the new services in my enterprise?
Axis of Time

- Keep as much history as you have storage
- Files, processes, boot sectors, mutexes, registry keys...
Welcome to the GRR console
Type help<enter> to get help

dbilby@storm3[1]|1> # Open the client.
dbilby@storm3[1]|2> p = aff4.FACTORY.Open("C.793f15613d7251f6/processes", age=aff4.ALL_TIMES)
dbilby@storm3[1]|3> proc_lists = p.GetValuesForAttribute(p.Schema.PROCESSES)
dbilby@storm3[1]|4> # List the snapshots we have
dbilby@storm3[1]|5> for p in proc_lists:
    print p.age, len(p)

2012-09-17 22:19:58 51
2012-09-18 22:18:53 57
2012-09-19 22:18:36 59
dbilby@storm3[1]|6> # Find what is new
dbilby@storm3[1]|7> a = set([m.exe for m in proc_lists[0]])
dbilby@storm3[1]|8> b = set([m.exe for m in proc_lists[1]])
dbilby@storm3[1]|9> print a.difference(b)
set([u'C:\Windows\notepad.exe', u'C:\Windows\System32\evil.exe', u'C:\Windows\System32\ftp.exe'])
dbilby@storm3[1]|10>
Features

- Windows, Linux, OSX clients
- Open source memory drivers Linux, OSX, Windows
- Detailed monitoring of client CPU/Memory impact
- Auto update mechanism
- Volatility integration
- Secure comms infrastructure designed for Internet deployment
- Web UI
- Scriptable console access
- Retrieve files
- Search memory
- Timeline events
- Schedule recurring actions
- Reporting
Demonstration

- Hunt
- Enterprise resource monitoring
Roadmap

A long road ahead....

- Testing testing testing
- Simplification of management
- UI overhaul
- Timelining (log2timeline python)
- Artifact parsers
- Anomaly detection
- Memory analysis
- ...

...
Contributors

Michael Cohen, Andreas Moser, Darren Bilby, Germano Caronni, Joachim Metz, Jordi Sanchez, Kristinn Guðjónsson, Elizabeth Schweinsberg....

Built on the shoulders of giants...
SleuthKit, Volatility, AFF4, Log2timeline...
Questions?


Documentation: http://grr.googlecode.com/git/docs/user_manual.html

Code at: code.google.com/p/grr

Mailing lists: groups.google.com/grr-users
groups.google.com/grr-developers