Putting Your Jobs Under the Microscope using OGRT

FOSDEM 2017

05.02.2017

Brussels, Belgium

https://goo.gl/QmJ1ks

Hello. Who am I?

Georg Rath
@og_rath
Systems Engineer
Freelancer

What is a job?

What is happening inside the job?

program execution

shared libraries

environment

loaded modules

Why would you want to know?

- What software do my users run?
  - Unoptimized Python from the home directory?
  - Binary build of some software?
- Some algebra library had a bug - are my users affected?
  - Which jobs used that library?
- Can I reduce the number of applications?
- Is the environment "sane"?
  - Are there problematic environment variables set?

How would you do it?

Existing Solutions

Asking the users

1. "We use this pipeline: 'mnseq_4_custom_3.Copy 2.sh'."
2. Go through the shell script, check the programs, module loads without versions, hardcoded paths, everything you could and could not imagine.
3. Rinse, repeat

Existing Solutions

Hooking module loads

A sample ~/.profile:

```
module load cd-hit
module load emboss
module load hmmer
module load ncbi-blast
module load ncbi-blast+
module load mafft
module load muscle
```

$$load \neq use$$

What does OGRT do?

- Tracks execution of all programs in a job
- Track every shared object a program loads
- Embeds a signature into programs and shared objects*
- Outputs data to Elasticsearch/Splunk in near-realtime

What makes OGRT unique?

- Works without a launcher
- Lightweight
- Transparent
- Resistant to outside influence
- No runtime dependencies
- Easy to deploy

How does it work?

Tracking Programs

LD_PRELOAD

The loader "preloads" a shared object
when loading a dynamic executable.

...combine with a GCC 'constructor':

No Launcher/Wrapper

Watermarking (Signature)

- Link in an object file at compile time
- Creates a note section in ELF (GCC does this too)
  - gets loaded into memory on execution
  - embeds an UUID
  - can be read by readelf/OGRT

Why the signature?

- same path - different executable
  - recompile of software
- discern user generated programs
- unique identifier

Example of Signature

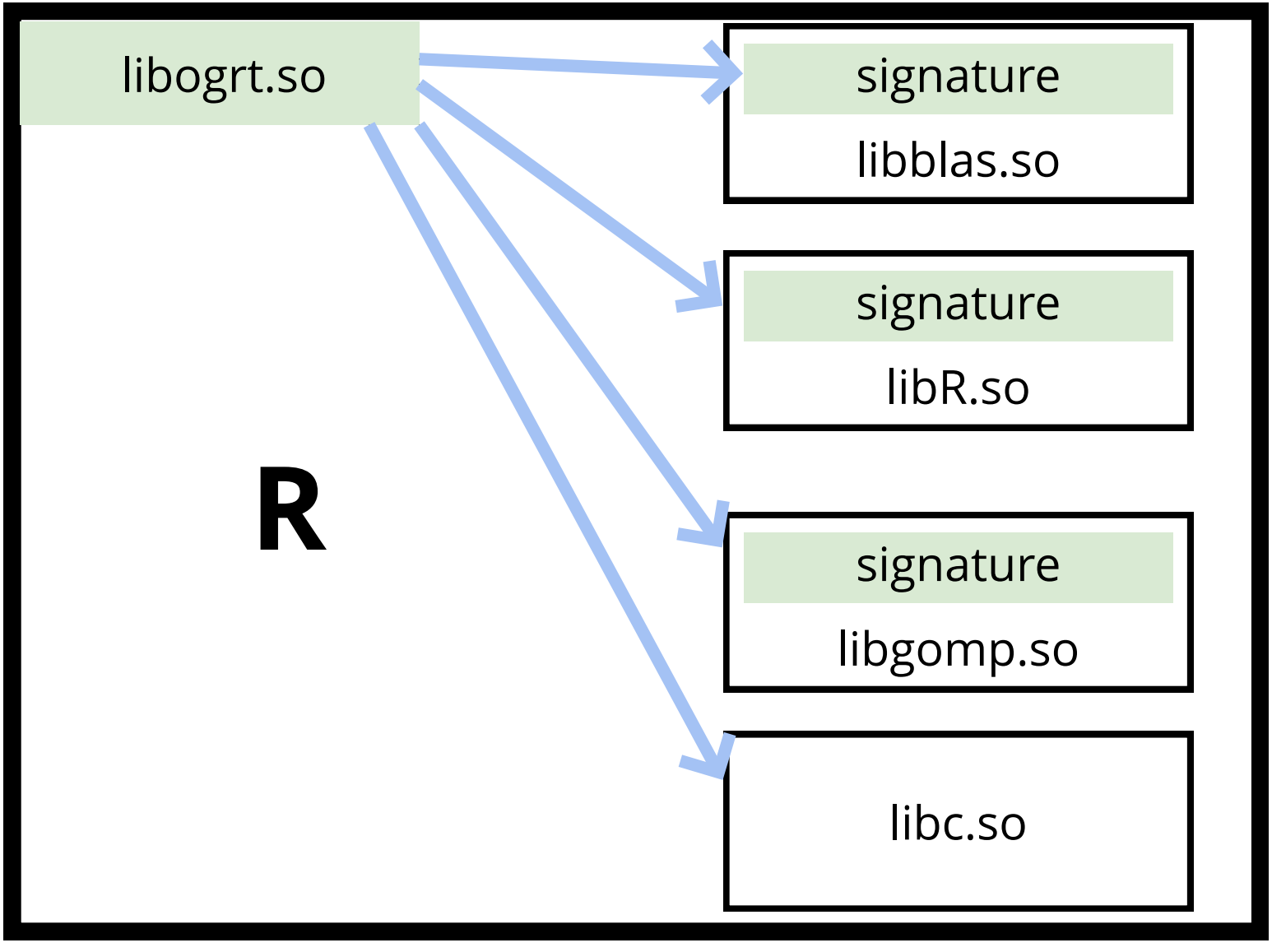
```
georg.rath@pcl-imba-70:~/devel/ogrt/talks/fosdem2017/sw/R$ readelf -n R

Displaying notes found at file offset 0x00000254 with length 0x0000003c:
  Owner                 Data size       Description
  OGRT                 0x00000026       Unknown note type: (0x4f475254)

Displaying notes found at file offset 0x00000290 with length 0x00000020:
  Owner                 Data size       Description
  GNU                  0x00000010       NT_GNU_ABI_TAG (ABI version tag)
    OS: Linux, ABI: 2.6.24

Displaying notes found at file offset 0x000002b0 with length 0x00000024:
  Owner                 Data size       Description
  GNU                  0x00000014       NT_GNU_BUILD_ID (unique build ID bitstring)
    Build ID: ff0a249437ee0f8cfc4c474385746872550 86d71
georg.rath@pcl-imba-70:~/devel/ogrt/talks/fosdem2017/sw/R$ ogrt --show-signature R
OGRT signature in section .note.ogrt.info
allocatable:    yes
version:        1
name:           OGRT
uuid:           b73c604d-c2e9-4057-ae31-586aa5ee29e3
```

Reading Signatures



But wait! There is more...

- support for sending environment
  - some/all variables
- support for sending loaded modules
- send *every* execution
  - or blacklist /usr, /bin, etc.
- configurable at compile-time

How do we persist the gathered data?

The Transport

*for debugging only

ogrt-server

- written in Go
- receive and decode data
- forward to configurable outputs
- buffers data
- embedded web-server

Metrics

OGRT Web Dashboard

Inputs

Inputs implicitly configured

receive

Metric	Value
Processed	276661
Rate 1m	182.65
Rate 5m	451.73
Rate 15m	241.26
Duration Min	0.00 ms
Duration Max	0.18 ms
Duration Median	0.01 ms

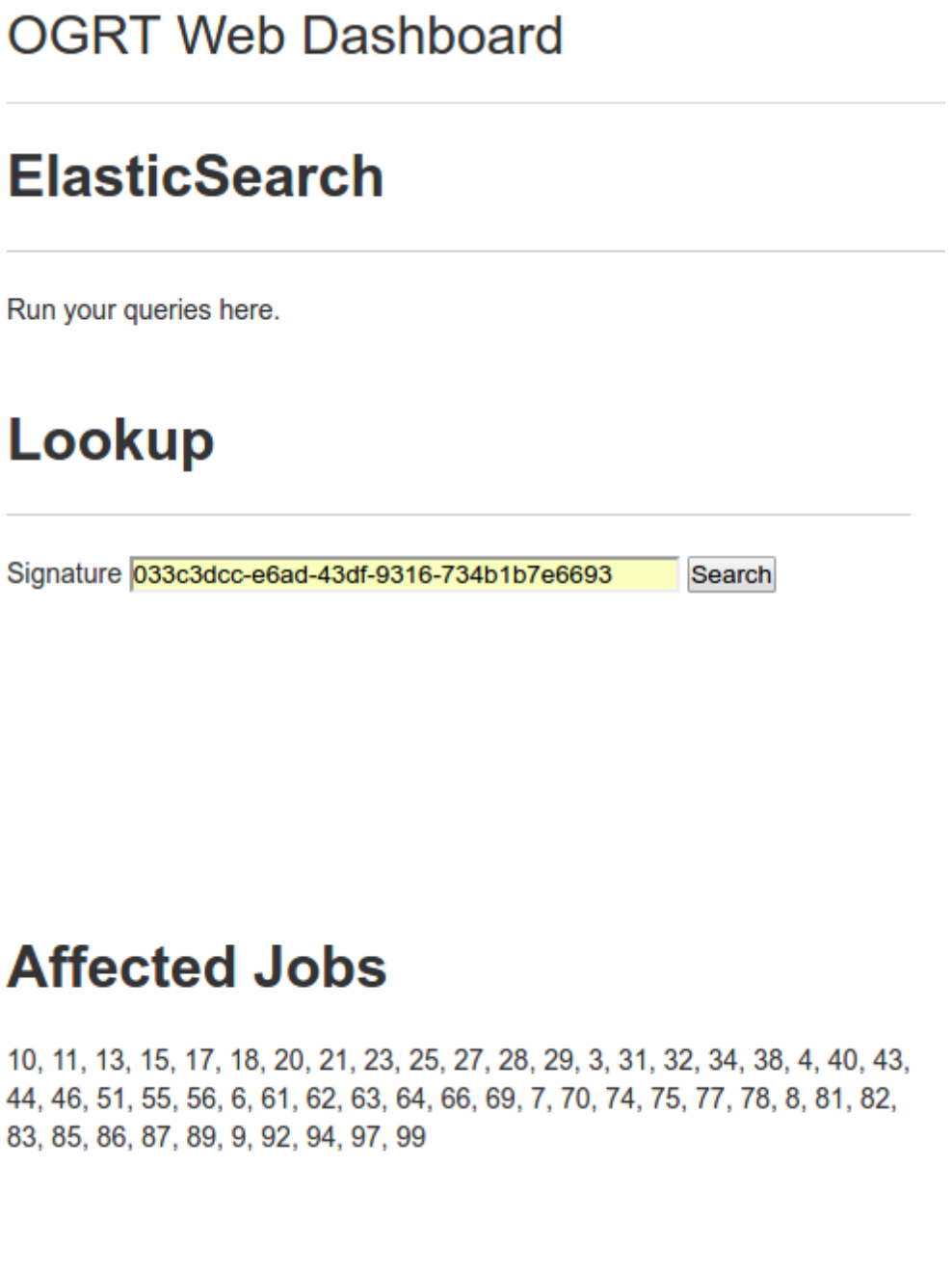
Outputs

Outputs configured in ogrt.conf

ElasticSearch

Metric	Value
Processed	27891
Rate 1m	93.99
Rate 5m	58.02
Rate 15m	26.12
Duration Min	8.61 ms
Duration Max	639.72 ms
Duration Median	21.74 ms

A broken library...



Benefits of using ElasticSearch

- you get access to the whole ecosystem
- analytics with Kibana
- you can combine data with Grafana
- stability and performance

Deployment in a slide

Server

```
wget -q https://github.com/georg-rath/ogrt/releases/download/v0.4.1/ogrt-server-v0.4.1.tar.bz2
tar xf ogrt-server-v0.4.1.tar.bz2
cd ogrt-server-v0.4.1
./ogrt-server
```

Client

```
git clone https://github.com/georg-rath/ogrt.git
cd ogrt/client
./vendorize
./configure --prefix=/tmp/ogrt
make install
LD_PRELOAD=$(find /tmp/ogrt/ -name libogrt.so) OGRT_ACTIVE=1 bash
# every command you run in the spawned bash gets sent to the server
ls
```

Conclusion

OGRT is

- giving you deep insight into what runs on your machine
- a versatile tool for the sysadmins toolbox
- configurable to your needs
- very easy to deploy (literally in 10 minutes)

With OGRT you can

- provide a census of used software (including user-built)
- troubleshoot problems with user's programs picking up unexpected shared libraries
- retroactively inform users about buggy libraries
- contribute to reproducibility of application runs

Outlook

- Built-in Queries
- Syslog transport (format?)
- eBPF evaluation
- Symbol level tracking (has the function x() been used)

Questions?

Thank you

https://github.com/georg-rath/ogrt