

REPRODUCE AND VERIFY FILESYSTEMS

Vincent Batts @vbatts

```
$> finger $(whoami)
```

```
Login: vbatts
```

```
Name: Vincent Batts
```

```
Directory: /home/vbatts
```

```
Shell: /bin/bash
```

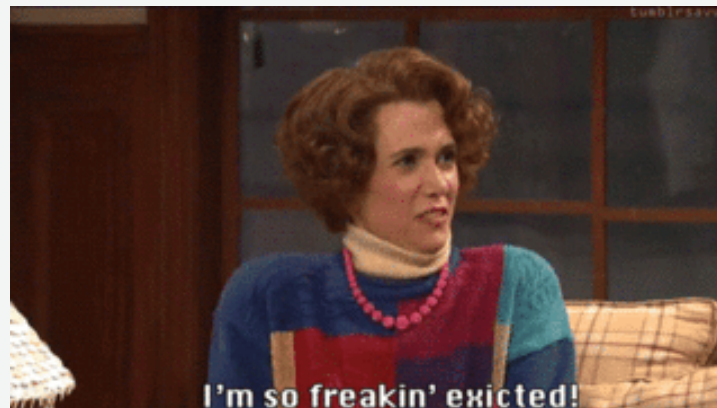
```
Such mail.
```

```
Plan:
```

```
OHMAN
```

```
$> id -Gn
```

```
devel opencontainers docker appc redhat golang slackware
```



AGENDA

- Packaging

AGENDA

- Packaging
- Content Addressability

AGENDA

- Packaging
- Content Addressability
- Compression!

AGENDA

- Packaging
- Content Addressability
- Compression!
- Reproducible Archives

AGENDA

- Packaging
- Content Addressability
- Compression!
- Reproducible Archives
- Verify at rest filesystems

AGENDA

- Packaging
- Content Addressability
- Compression!
- Reproducible Archives
- Verify at rest filesystems



PACKAGING

PACKAGING



PACKAGING

tar archives



PACKAGING

tar archives

Slackware packages ([tar\(1\)](#) archives)



PACKAGING

tar archives

Slackware packages ([tar\(1\)](#) archives)

Debian *.deb ([ar\(1\)](#) archive of [tar\(1\)](#) archives)



PACKAGING

tar archives

Slackware packages ([tar\(1\)](#) archives)

Debian *.deb ([ar\(1\)](#) archive of [tar\(1\)](#) archives)

Red Hat *.rpm (custom key/value binary and [cpio\(1\)](#))



PACKAGING

tar archives

Slackware packages ([tar\(1\)](#) archives)

Debian *.deb ([ar\(1\)](#) archive of [tar\(1\)](#) archives)

Red Hat *.rpm (custom key/value binary and [cpio\(1\)](#))

Java *.jar and *.war ([zip\(1\)](#) archive)



PACKAGING

tar archives

Slackware packages ([tar\(1\)](#) archives)

Debian *.deb ([ar\(1\)](#) archive of [tar\(1\)](#) archives)

Red Hat *.rpm (custom key/value binary and [cpio\(1\)](#))

Java *.jar and *.war ([zip\(1\)](#) archive)

Ruby *.gem ([tar\(1\)](#) archive of [tar\(1\)](#) archives)



PACKAGING

tar archives

Slackware packages ([tar\(1\)](#) archives)

Debian *.deb ([ar\(1\)](#) archive of [tar\(1\)](#) archives)

Red Hat *.rpm (custom key/value binary and [cpio\(1\)](#))

Java *.jar and *.war ([zip\(1\)](#) archive)

Ruby *.gem ([tar\(1\)](#) archive of [tar\(1\)](#) archives)

Container Images ([tar\(1\)](#) archives)



CONTENT ADDRESSIBILITY

CONTENT ADDRESSIBILITY

Opaque Object storage

CONTENT ADDRESSIBILITY

Opaque Object storage
changed object = new object

CONTENT ADDRESSIBILITY

Opaque Object storage
changed object = new object
cryptographic assurance

CONTENT ADDRESSABILITY

Opaque Object storage
changed object = new object
cryptographic assurance



COMPRESSION!

COMPRESSION!

same objects, but variation in compression

COMPRESSION!

same objects, but variation in compression

inflate/deflate (RFC1951)

COMPRESSION!

same objects, but variation in compression

inflate/deflate (RFC1951)

Gzip (RFC1952)

COMPRESSION!

same objects, but variation in compression

inflate/deflate (RFC1951)

Gzip (RFC1952)

`gzip` vs Golang `compress/gzip` vs Zlib

COMPRESSION!

same objects, but variation in compression

inflate/deflate (RFC1951)

Gzip (RFC1952)

`gzip` vs Golang `compress/gzip` vs Zlib

ideally compress for transfer and storage, but not for identity

COMPRESSION!

```
#!/bin/sh
dd if=/dev/urandom of=rando.img bs=1M count=2
cat rando.img | gzip -n > rando.img.gz
cat rando.img | gzip -n -9 > rando.img.9.gz
cat rando.img | xz > rando.img.xz
cat rando.img | xz -9 > rando.img.9.xz
shasum rando.img* > SHA1

cat rando.img | gzip -n > rando.img.gz
cat rando.img | gzip -n -9 > rando.img.9.gz
cat rando.img | xz > rando.img.xz
cat rando.img | xz -9 > rando.img.9.xz
shasum -c ./SHA1
```

COMPRESSION!

```
#!/usr/bin/env ruby

require 'zlib'
include Zlib

input = File.open(ARGV.first)
GzipWriter.open(ARGV.first + '.gz', DEFAULT_COMPRESSION, HUFFMAN_ONLY) do |gz|
  gz.write(IO.binread(input))
end
input.flush()
input.close()
```

COMPRESSION!

```
package main

import (
    "compress/gzip"
    "io"
    "os"
)

func main() {
    input, err := os.Open(os.Args[1])
    if err != nil {
        println(err.Error())
        os.Exit(1)
    }
    output, err := os.Create(os.Args[1] + ".gz")
    if err != nil {
        println(err.Error())
        os.Exit(1)
    }
    gz := gzip.NewWriter(output)
    if _, err := io.Copy(gz, input); err != nil {
        println(err.Error())
        os.Exit(1)
    }
}
```

REPRODUCIBLE ARCHIVE

REPRODUCIBLE ARCHIVE

reproducible-builds.org

REPRODUCIBLE ARCHIVE

reproducible-builds.org

processed checksum of tar archive ([see deprecated Docker TarSum](#))

REPRODUCIBLE ARCHIVE

reproducible-builds.org

processed checksum of tar archive ([see deprecated Docker TarSum](#))

keep around the original *.tar?

REPRODUCIBLE ARCHIVE

reproducible-builds.org

processed checksum of tar archive ([see deprecated Docker TarSum](#))

keep around the original *.tar?

re-assemble the original *.tar

REPRODUCIBLE ARCHIVE

reproducible-builds.org

processed checksum of tar archive (see deprecated Docker TarSum)

keep around the original *.tar?

re-assemble the original *.tar

github.com/vbatts/tar-split

REPRODUCIBLE ARCHIVE

reproducible-builds.org

processed checksum of tar archive (see deprecated Docker TarSum)

keep around the original *.tar?

re-assemble the original *.tar

github.com/vbatts/tar-split



REPRODUCIBLE ARCHIVE

```
go install github.com/vbatts/tar-split/cmd/tar-split
```

```
tar cf demo.tar *.sh  
shasum demo.tar | tee SHA1
```

```
tar-split disasm --no-stdout ./demo.tar  
ls -lh tar-data.json.gz
```

```
rm -f demo.tar  
tar-split asm --output demo.tar --path .  
shasum -c ./SHA1
```

VERIFY AT REST FILESYSTEMS

VERIFY AT REST FILESYSTEMS

Regardless of transport, ensure resulting filesystem

VERIFY AT REST FILESYSTEMS

Regardless of transport, ensure resulting filesystem
(*tar archive, rsync, bittorrent, IPFS, etc)

VERIFY AT REST FILESYSTEMS

Regardless of transport, ensure resulting filesystem
(*tar archive, rsync, bittorrent, IPFS, etc)

```
`rpm -qV <package>` functionality
```

VERIFY AT REST FILESYSTEMS

Regardless of transport, ensure resulting filesystem
(*tar archive, rsync, bittorrent, IPFS, etc)

``rpm -qV <package>`` functionality

Future hopes could be IMA/EVM

VERIFY AT REST FILESYSTEMS

Regardless of transport, ensure resulting filesystem
(*tar archive, rsync, bittorrent, IPFS, etc)

```
`rpm -qV <package>` functionality
```

Future hopes could be IMA/EVM

Passive validation of directory hierarchies

VERIFY AT REST FILESYSTEMS

Regardless of transport, ensure resulting filesystem
(*tar archive, rsync, bittorrent, IPFS, etc)

```
`rpm -qV <package>` functionality
```

Future hopes could be IMA/EVM

Passive validation of directory hierarchies
BSD mtree(8)

VERIFY AT REST FILESYSTEMS

FreeBSD mtree(8)

mtree-port (for linux)

go-mtree (golang cli and library)

libarchive-formats(5)

VERIFY AT REST FILESYSTEMS

with packages: libarchive and python-libarchive-c

```
#!/usr/bin/env python

import libarchive

with libarchive.file_writer('../demo.mtree', 'mtree') as a:
    a.add_files('./')
```

NOTICE: libarchive uses older mtree format

VERIFY AT REST FILESYSTEMS

```
mtree -c -p ./ -K sha256digest | tee /tmp/demo.mtree
```

```
mtree -f /tmp/demo.mtree -p ./
```

```
echo $?
```

```
read
```

```
touch $0 # SCANDALOUS
```

```
mtree -f /tmp/demo.mtree -p ./
```

VERIFY AT REST FILESYSTEMS

Directory Path

```
go get -u github.com/vbatts/go-mtree/cmd/gomtree
gomtree -c -p ./ -K sha256digest | tee /tmp/demo.mtree

gomtree -f /tmp/demo.mtree -p ./
echo $?

read

touch $0 # SCANDALOUS
gomtree -f /tmp/demo.mtree -p ./
```

VERIFY AT REST FILESYSTEMS

Tar Archive Support

```
tar cf /tmp/demo.tar .
gmtree -c -T /tmp/demo.tar -K sha256digest | tee /tmp/demo.mtree

gmtree -f /tmp/demo.mtree -T /tmp/demo.tar
echo $?

read

gmtree -f /tmp/demo.mtree -p ./
echo $?
```

CALL TO ACTION

You have the need to store archives, whole and extracted, check out github.com/vbatts/tar-split

You have the need to verify, or restore, a filesystem regardless of how it was distributed, check out github.com/vbatts/go-mtree or other mtree projects

VINCENT BATTS

@VBATTS | VBATTS@REDHAT.COM

THANK YOU!