How secure is your build/server ?

@patrickdebois

Patrick Debois

@patrickdebois

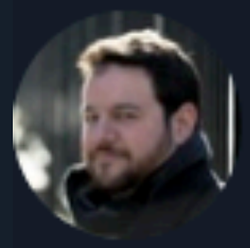
Director of Dev ❤️ Ops Relations at @snyksec - Eternal implementor and pragmatic researcher - released #devops onto the world through @devopsdays

Ghent, Flanders, Belgium jedi.be/blog Joined July 2008

Patrick Debois @patrickdebois 4d

more news - I'll be speaking "How secure is your build/server?" #FOSDEM in the CI/CD room (last day , last slot :) - schedule will be updated soon - cu there!

5 4 44



Mike Fiedler
@mikefiedler

Replying to @patrickdebois

"TL,DR: Not As Secure As You Think"

Patrick Debois @patrickdebois 35m

Eventually consistent

vs

Eventually secure

1 1 4

Tom Marien 🇧🇪 ⚛️

@Tom_Marien

Replying to @patrickdebois

Vs eventually hacked 😛


"Did you build the
build you wanted?"
@patrickdebois


"A story
of Trust"
@patrickdebois

Nerd Alert

@patrickdebois

Talk Scope Compression Algorithm

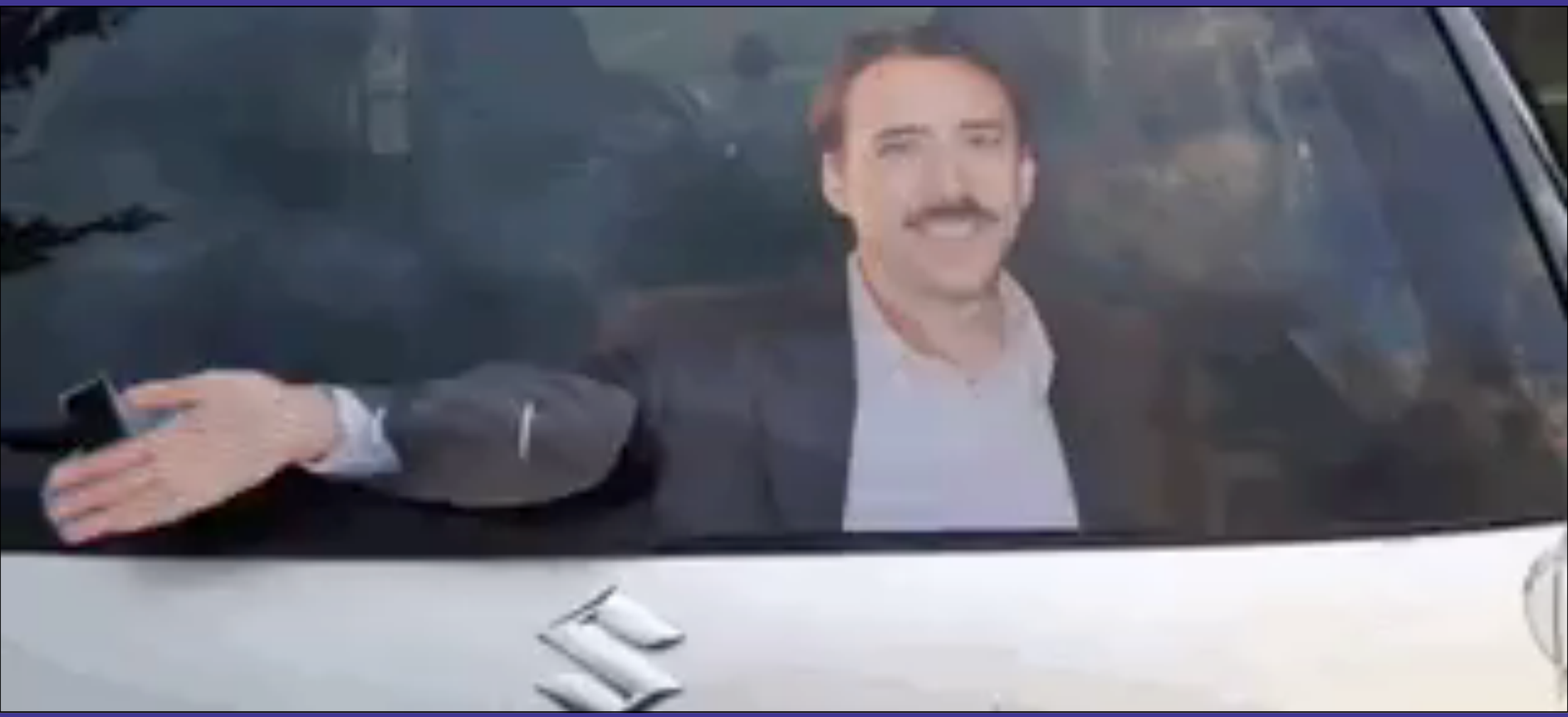
@patrickdebois

delivery from src to prod

@patrickdebois


OK DOOMER
no "scare"
tactics
@patrickdebois

no hardware attacks



@patrickdebois

https://www.youtube.com/watch?v=w1bMEoUR6I4

@patrickdebois


no physical
access
bypasses
@patrickdebois


Not
about
keys mgmt
@patrickdebois

Not about hardening

I just wanna hit something.
I wanna hit it hard!

@patrickdebois

focus on tampering+

STRIDE Threat Model

Spoofing identity

- Illegally accessing and then using another user's authentication information

Tampering with data

- Malicious modification
- Unauthorized changes

Repudiation

- Deny performing an malicious action
- Non-repudiation refers to the ability of a system to counter repudiation threats

Elevation of privilege

- Unprivileged user gains privileged access to compromise the system
- Effectively penetrated and become part of the trusted system

Denial of service

- Deny service to valid users
- Threats to system availability and reliability

Information disclosure

- Exposure of information to individuals not supposed to access

Promise Theory

Promise Theory, in the context of information science, is a model of voluntary cooperation between individual, autonomous actors or agents who publish their intentions to one another in the form of promises. It is a form of labelled graph theory, describing discrete networks of agents joined by the unilateral promises they make.

A 'promise' is a declaration of intent whose purpose is to increase the recipient's certainty about a claim of past, present or future behaviour.[1] For a promise to increase certainty, the recipient needs to trust the promiser, but trust can also be built on the verification (or 'assessment') that previous promises have been kept, thus trust plays a symbiotic relationship with promises. Each agent assesses its belief in the promise's outcome or intent. Thus Promise Theory is about the relativity of autonomous agents.

One of the goals of Promise Theory is to offer a model that unifies the physical (or dynamical) description of an information system with its intended meaning, i.e. its semantics. This has been used to describe configuration management of resources in information systems, amongst other things.

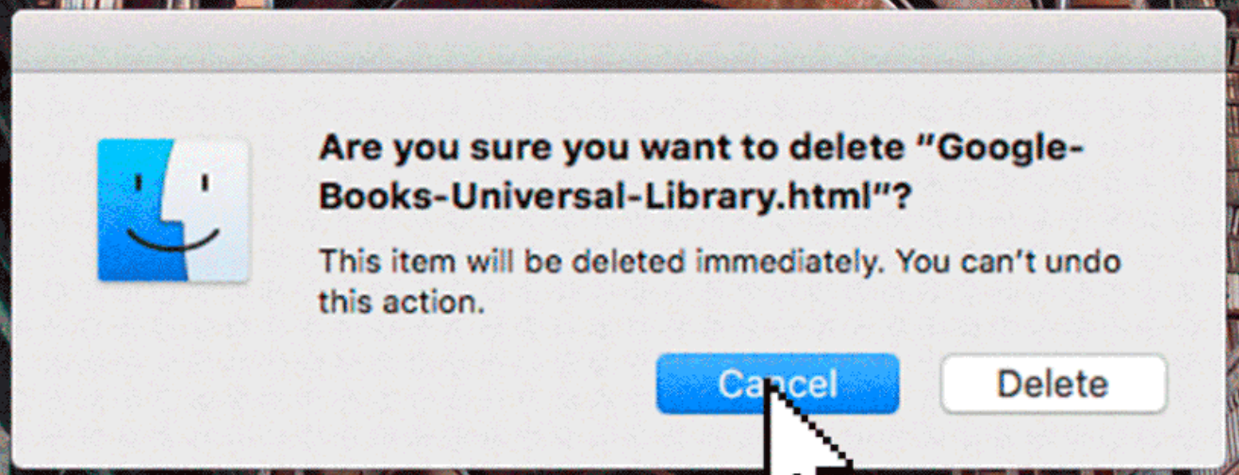
- Secure boot
- Certificate Authorities
- Secure Password
  - Local & iCloud
  - TouchId
- Encrypt Disk
- Harden OS/Firewall
- Install updates

"Treat Laptops as Cattle not Pets"

Developers are the new librarians

Homebrew

DO NOT CURL!!!!!



AppStore

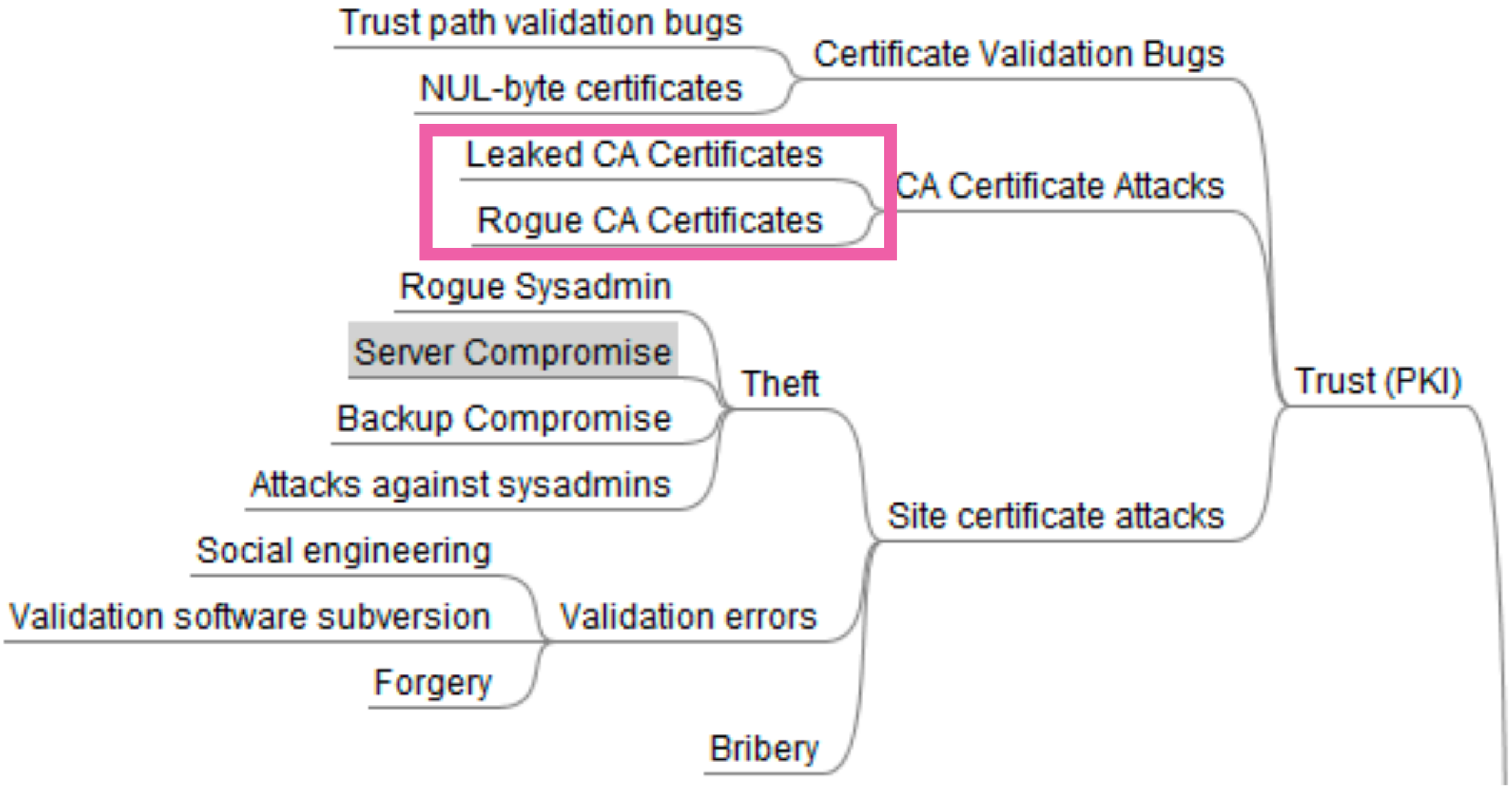
Docker

NPM

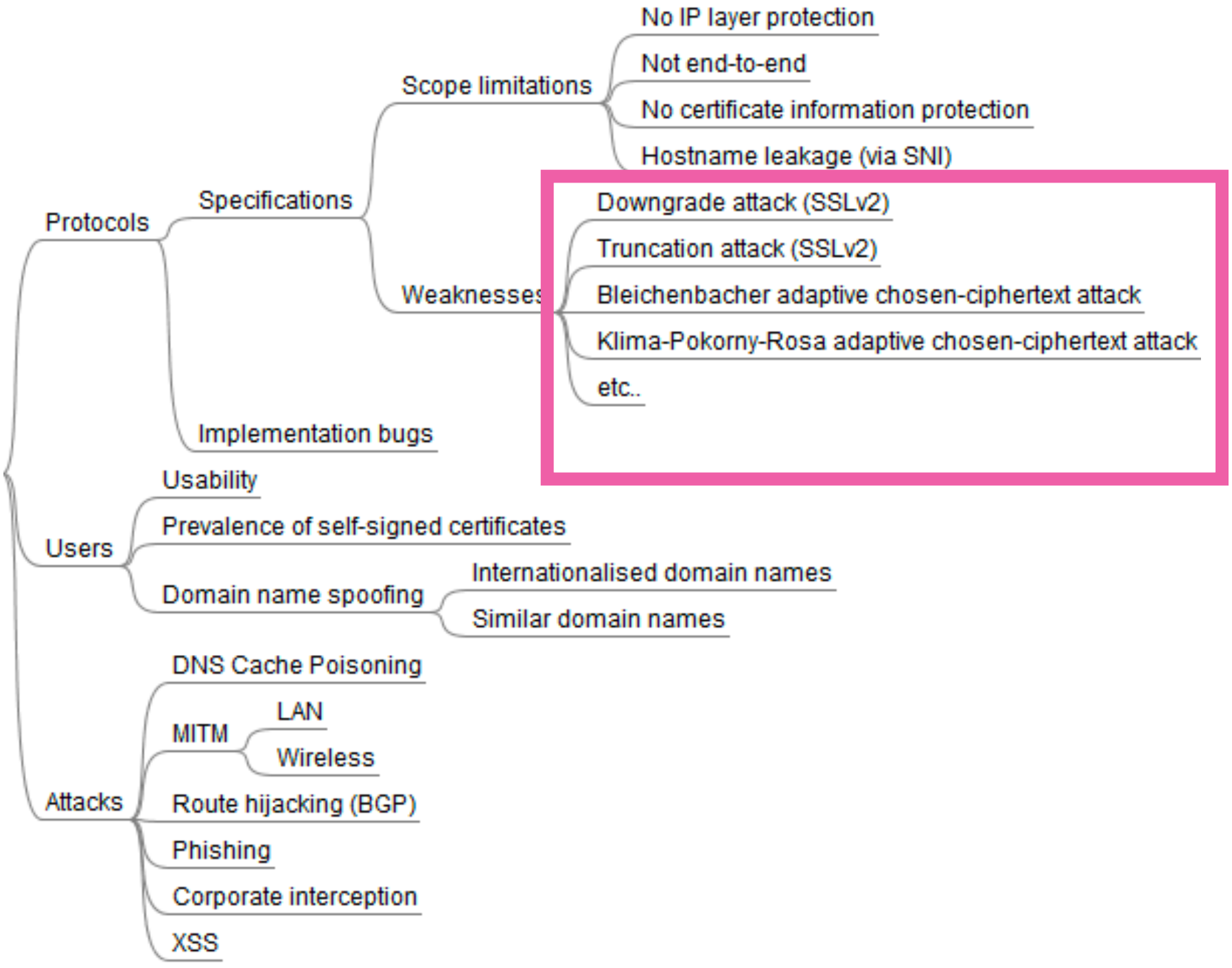
TLS
Chain of Trust

- Server Configuration
  - Server-side
    - End Points
      - Configuration errors
        - Failure to enforce SSL
        - Invalid Certificates
          - Expired certificate
          - Incorrectly configured chain
          - Invalid hostname
          - Not valid for all requried hostnames
          - Insufficient assurance (*)
          - Self-signed Certificates
        - Unprotected Private Key
        - Private Key Duplication (*)
        - Private key reuse
        - Client Authentication
          - Lack of trust validation
          - Validation against other root certs
          - Lack of revocation checking
        - Configuration Weaknesses
          - Use of weak protocols
          - Weak key exchange (*)
          - Weak ciphers (*)
          - Non-FIPS approved ciphers (*)
          - Anonymous key exchange
      - Use of unpactched SSL libraries
    - Site Implementation
      - Mixed SSL/Non-SSL Areas
      - Insecure cookies
  - Client Side
    - User Interface (Usability)
    - Client Configuration
    - Secure Implementation
    - Lack of revocation checking

https://www.ssllabs.com/projects/ssl-threat-model/



https://www.ssllabs.com/projects/ssl-threat-model/



https://www.ssllabs.com/projects/ssl-threat-model/

Modifying the Default TLS Cipher suite

Node.js is built with a default suite of enabled and disabled TLS ciphers. Currently, the default cipher suite is:

```
TLS_AES_256_GCM_SHA384:
TLS_CHACHA20_POLY1305_SHA256:
TLS_AES_128_GCM_SHA256:
ECDHE-RSA-AES128-GCM-SHA256:
ECDHE-ECDSA-AES128-GCM-SHA256:
ECDHE-RSA-AES256-GCM-SHA384:
ECDHE-ECDSA-AES256-GCM-SHA384:
DHE-RSA-AES128-GCM-SHA256:
ECDHE-RSA-AES128-SHA256:
DHE-RSA-AES128-SHA256:
ECDHE-RSA-AES256-SHA384:
DHE-RSA-AES256-SHA384:
ECDHE-RSA-AES256-SHA256:
DHE-RSA-AES256-SHA256:
HIGH:
!aNULL:
!eNULL:
!EXPORT:
!DES:
!RC4:
!MD5:
!PSK:
!SRP:
!CAMELLIA
```

This default can be replaced entirely using the `--tls-cipher-list` command line switch (directly, or via the `NODE_OPTIONS` environment variable). For ins
`AES128-GCM-SHA256:!RC4` the default TLS cipher suite:

curl did not work with *Certificate Revocation Lists* for me either, neither on Windows, nor on Linux. Why should you use **curl**? **Openssl** seems more appropriate:

```
openssl s_client -connect www.google.com:443
```

We get

```
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
   i:/C=US/O=Google Inc/CN=Google Internet Authority G2
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority G2
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
 2 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
   i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
---
```

Then we can inspect some certificate:

```
curl http://pki.google.com/GIAG2.crt | openssl x509 -inform der -text
```

`grep crl` in the output of the above command. The interesting parts are:

```
        X509v3 CRL Distribution Points:
            URI:http://crl.geotrust.com/crls/gtglobal.crl

        Authority Information Access:
            OCSP - URI:http://gtglobal-ocsp.geotrust.com
```

How long is the certificate valid?

SSL certificates generated by Let's Encrypt are valid for 90 days and then renew automatically. This is for two reasons as stated their blog post:

- They limit damage from key compromise and mis-issuance since stolen keys and mis-issued certificates are valid for a sh period of time.
- They encourage automation, which is absolutely essential for ease-of-use. This takes the burden off system administrator manually handle renewals. Once issuance and renewal are automated, shorter lifetimes won't be any less convenient tha longer ones.

What level of encryption is available?

RSA-signed using 2048-bit RSA keys.

- Cryptographic choices

Are wildcard certificates available for use?

No. Although 'Let's Encrypt' offers wildcard certificates, it is currently not possible to use them at DreamHost. If you need SSL certificates on your subdomains, you must enable them individually.

What browsers support Let's Encrypt certs?

Certificates are trusted in all major browsers. View the blog post here:

- https://letsencrypt.org/2015/10/19/lets-encrypt-is-trusted.html

Certificate Transparency Monitoring

Certificate Transparency is an open framework which helps log, audit and monitor publicly-trusted TLS certificates on the Internet. This tool lets you search for certificates issued for a given domain and subscribe to notifications from Facebook regarding new certificates and potential phishing attacks.

Search | Subscriptions

snyk.io **Search**

Domains	Subject	Issuer	Validity	Certificate
mailhooks.prod.snyk.io blackbin.snyk.io piper.snyk.io health.prod.snyk.io	CN=health.prod.snyk.io	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3	Aug 23, 2018 - Nov 21, 2018	**Show Details**
test-2.test.onprem.snyk.io	CN=test-2.test.onprem.snyk.io	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3	Nov 11, 2019 - Feb 09, 2020	**Show Details** (CT Precertificate)

https://developers.facebook.com/tools/ct/search/

If we want to allow both Let's Encrypt and Comodo, we should add 2 CAA record

```
example.com.  CAA 0 issue "comodoca.com"
example.com.  CAA 0 issue "letsencrypt.org"
```

If we want to allow Let's Encrypt and Comodo only for wildcard, then we can use

```
example.com.  CAA 0 issue "letsencrypt.org"
example.com.  CAA 0 issuewild "comodoca.com"
```

Note that the presence of issuewild overrides the `issue`. Therefore, Let's Encry
issue wildcard certificates.

CAA Record

Everything is a Freaking DNS Problem

@patrickdebois

Michael Coté
@cote

If it's always a DNS problem, maybe someone should, like, fix DNS.

3:21pm · 29 Jan 2020 · Twitter Web App

9 Replies **6** Retweets **50** Likes

Kris Buytaert @KrisBuytaert 3d
Replying to @cote
So many have tried ..
2

R.I.Pienaar @ripienaar 3d
Replying to @cote @patrickdebois
Don’t encourage them, last time they tried we got DNSSEC
4

Analyzing DNSSEC problems for debian.org

.	✓ Found 2 DNSKEY records for . ✓ DS=20326/SHA-256 verifies DNSKEY=20326/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=20326 and DNSKEY=20326/SEP verifies the DNSKEY RRset
org	✓ Found 2 DS records for org in the . zone ✓ DS=9795/SHA-1 has algorithm RSASHA1-NSEC3-SHA1 ✓ DS=9795/SHA-256 has algorithm RSASHA1-NSEC3-SHA1 ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=33853 and DNSKEY=33853 verifies the DS RRset ✓ Found 4 DNSKEY records for org ✓ DS=9795/SHA-1 verifies DNSKEY=9795/SEP ✓ Found 3 RRSIGs over DNSKEY RRset ✓ RRSIG=9278 and DNSKEY=9278 verifies the DNSKEY RRset
debian.org	✓ Found 1 DS records for debian.org in the org zone ✓ DS=21832/SHA-256 has algorithm RSASHA256 ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=9278 and DNSKEY=9278 verifies the DS RRset ✓ Found 3 DNSKEY records for debian.org ✓ DS=21832/SHA-256 verifies DNSKEY=21832/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=21832 and DNSKEY=21832/SEP verifies the DNSKEY RRset ✓ debian.org A RR has value 130.89.148.77 ✓ Found 1 RRSIGs over A RRset ✓ RRSIG=38844 and DNSKEY=38844 verifies the A RRset

Analyzing DNSSEC problems for brew.sh

.	✓ Found 2 DNSKEY records for . ✓ DS=20326/SHA-256 verifies DNSKEY=20326/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=20326 and DNSKEY=20326/SEP verifies the DNSKEY RRset
sh	✓ Found 2 DS records for sh in the . zone ✓ DS=55297/SHA-1 has algorithm RSASHA256 ✓ DS=55297/SHA-256 has algorithm RSASHA256 ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=33853 and DNSKEY=33853 verifies the DS RRset ✓ Found 2 DNSKEY records for sh ✓ DS=55297/SHA-1 verifies DNSKEY=55297/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=51568 and DNSKEY=51568 verifies the DNSKEY RRset
brew.sh	✓ Found 1 DS records for brew.sh in the sh zone ✓ DS=9382/SHA-256 has algorithm RSASHA256 ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=51568 and DNSKEY=51568 verifies the DS RRset ✓ Found 2 DNSKEY records for brew.sh ✓ DS=9382/SHA-256 verifies DNSKEY=9382/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=9382 and DNSKEY=9382/SEP verifies the DNSKEY RRset ✓ brew.sh A RR has value 185.199.110.153 ✓ Found 1 RRSIGs over A RRset ✓ RRSIG=49533 and DNSKEY=49533 verifies the A RRset

Analyzing DNSSEC problems for npmjs.org

.	✓ Found 2 DNSKEY records for . ✓ DS=20326/SHA-256 verifies DNSKEY=20326/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=20326 and DNSKEY=20326/SEP verifies the DNSKEY RRset
org	✓ Found 2 DS records for org in the . zone ✓ DS=9795/SHA-1 has algorithm RSASHA1-NSEC3-SHA1 ✓ DS=9795/SHA-256 has algorithm RSASHA1-NSEC3-SHA1 ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=33853 and DNSKEY=33853 verifies the DS RRset ✓ Found 4 DNSKEY records for org ✓ DS=9795/SHA-1 verifies DNSKEY=9795/SEP ✓ Found 3 RRSIGs over DNSKEY RRset ✓ RRSIG=9278 and DNSKEY=9278 verifies the DNSKEY RRset ⚠ All Queries to a0.org.afilias-nst.info for npmjs.org/DNSKEY timed out or failed
npmjs.org	✗ No DS records found for npmjs.org in the org zone ✗ No DNSKEY records found ✓ npmjs.org A RR has value 104.16.21.35 ✗ No RRSIGs found

Analyzing DNSSEC problems for github.com

.	✓ Found 2 DNSKEY records for . ✓ DS=20326/SHA-256 verifies DNSKEY=20326/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=20326 and DNSKEY=20326/SEP verifies the DNSKEY RRset
com	✓ Found 1 DS records for com in the . zone ✓ DS=30909/SHA-256 has algorithm RSASHA256 ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=33853 and DNSKEY=33853 verifies the DS RRset ✓ Found 2 DNSKEY records for com ✓ DS=30909/SHA-256 verifies DNSKEY=30909/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=30909 and DNSKEY=30909/SEP verifies the DNSKEY RRset
github.com	✗ No DS records found for github.com in the com zone ✗ No DNSKEY records found ⚠ ns4.p16.dynect.net serial (1579651907) differs from ns-520.awsdns-01.net serial (1) ⚠ ns3.p16.dynect.net serial (1579651907) differs from ns-520.awsdns-01.net serial (1) ⚠ ns1.p16.dynect.net serial (1579651907) differs from ns-520.awsdns-01.net serial (1) ⚠ ns2.p16.dynect.net serial (1579651907) differs from ns-520.awsdns-01.net serial (1) ✓ github.com A RR has value 192.30.253.113 ✗ No RRSIGs found

Hvorfor gør du ikke DANE?

Ok, so if DNSSEC / DANE is so great, can we just use this everywhere? Like... your browser?

Mozilla and Google both have been going back and forth on their plans to implement DNSSEC and DANE, and even our otherwise ever reliable `curl(1)` does not have DNSSEC / DANE support: after over 6 years, it remains an open TODO item, although partial work was previously done.

Google initially implemented DNSSEC authenticated HTTPS in Chrome, but later removed it. (There's some irony in Google arguing against DNSSEC in favor of HPKP, only to then remove HPKP from Chrome, shifting entirely towards detection of fraudulent certificates by way of `CAA` records (RFC 6844), even though each of these solutions appear to me to address different, albeit overlapping or intersecting problems and threats.) Mozilla similarly had plans, but the respective bugzilla tickets were either closed `WONTFIX` or left open.

A web browser add-on used to be available from https://www.dnssec-validator.cz/, but that required an external binary application in addition to the add-on and was ultimately abandoned in October of 2018. So as it stands right now (as of 2019-05-07), the only browser add-on I was able to find ways this one; it adds a simple visual marker in the address bar when a site has a valid DNSSEC protected `TLSA` record:

https://www.netmeister.org/blog/dnssec-dane.html

README.md

Homebrew (un)installer

Install Homebrew

DEEP ..... SIGH

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

More installation information and options at https://docs.brew.sh/Installation.html.

Linux and Windows 10 Subsystem for Linux

Install Homebrew on Linux and Windows 10 Subsystem for Linux: https://docs.brew.sh/Linuxbrew.

Uninstall Homebrew

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/uninstall)"
```

Download the uninstall script and run `./uninstall --help` to view more uninstall options.

README.md

pipethis

(I would have called it `stop-piping-the-internet-into-your-shell`, but that seemed too long.)

`tl;dr`

Instead of

```
$ curl -sSL https://get.rvm.io | bash
```

do

```
$ pipethis https://get.rvm.io
```

or (assuming not everybody has adopted my awesome idea but you still want to improve your life)

```
$ pipethis --no-verify --inspect https://get.rvm.io
```

or even

```
$ curl -sSL https://get.rvm.io | pipethis --no-verify | bash
```

https://github.com/ellotheth/pipethis

Verifying Binaries

Download directories contain a `SHASUMS256.txt` file with SHA checksums for the files.

To download `SHASUMS256.txt` using `curl`:

```
$ curl -O https://nodejs.org/dist/vx.y.z/SHASUMS256.txt
```

To check that a downloaded file matches the checksum, run it through `sha256sum` with a command such as:

```
$ grep node-vx.y.z.tar.gz SHASUMS256.txt | sha256sum -c -
```

For Current and LTS, the GPG detached signature of `SHASUMS256.txt` is in `SHASUMS256.txt.sig`. You can use it with `gpg` to verify the integrity of `SHASUM256.txt`. You will first need to import the GPG keys of individuals authorized to create releases. To import the keys:

```
$ gpg --keyserver pool.sks-keyservers.net --recv-keys DD8F2338BAE7501E3DD5AC78C273792F7D83545D
```

See the bottom of this README for a full script to import active release keys.

Next, download the `SHASUMS256.txt.sig` for the release:

```
$ curl -O https://nodejs.org/dist/vx.y.z/SHASUMS256.txt.sig
```

Then use `gpg --verify SHASUMS256.txt.sig SHASUMS256.txt` to verify the file's signature.

Building Node.js

guillaumerose commented on 6 Aug 2019 Member

It appears that `Docker.dmg.sha256sum` link is not in the documentation anymore. I removed the file on S3 to avoid confusion.

If you want to verify the application, I suggest you use codesign tool directly:

```
$ codesign -dv --verbose=4 /Applications/Docker.app
Executable=/Applications/Docker.app/Contents/MacOS/Docker
Identifier=com.docker.docker
Format=app bundle with Mach-O thin (x86_64)
CodeDirectory v=20200 size=64817 flags=0x0(none) hashes=2018+5 location=embedded
VersionPlatform=1
VersionMin=658176
VersionSDK=658944
Hash type=sha256 size=32
CandidateCDHash sha1=687fe9965fd05fece6872da0fd4360e67df507cf
CandidateCDHash sha256=065d2fd2737008357ae12997476723623c288cfc
Hash choices=sha1,sha256
Page size=4096
CDHash=065d2fd2737008357ae12997476723623c288cfc
Signature size=9000
Authority=Developer ID Application: Docker Inc (9BNSXJN65R)
Authority=Developer ID Certification Authority
Authority=Apple Root CA
Timestamp=5 Aug 2019 at 16:14:02
Info.plist entries=45
TeamIdentifier=9BNSXJN65R
Sealed Resources version=2 rules=13 files=91
Internal requirements count=1 size=212
```

https://github.com/docker/for-mac/issues/3800

Trust libraries

```
~ ▶ npq install express jquey

  ✔ Checking package maturity
  ✖ Identifying package author...
  ✔ Checking package download popularity
  ✖ Checking availability of a README
  ✔ Identifying package repository...
  ✔ Checking package for pre/post install scripts
  ✖ Checking for known vulnerabilities

Detected possible issues with the following packages:
  [jquey]
    - the package description has no e-mail associated with author(s). Proceed wi
    - package flagged for security issues and served as place-holder
    - Found 1 vulnerability | Snyk
  [express]
    - Found 3 vulnerabilities | Snyk

? Would you like to continue installing package(s)? (y/N)
```

@patrickdebois

3. Minimize attack surfaces by ignoring run-scripts

The npm CLI works with package run-scripts. If you've ever run `npm start` or `npm test` then you've used package run-scripts too. The npm CLI builds on scripts that a package can declare, and allows packages to define scripts to run at specific entry points during the package's installation in a project. For example, some of these script hook entries may be `postinstall` scripts that a package that is being installed will execute in order to perform housekeeping chores.

With this capability, bad actors may create or alter packages to perform malicious acts by running any arbitrary command when their package is installed. A couple of cases where we've seen this already happening is the popular eslint-scope incident that harvested npm tokens, and the crossenv incident, along with 36 other packages that abused a typosquatting attack on the npm registry.

Apply these npm security best practices in order to minimize the malicious module attack surface:

Always vet and perform due-diligence on third-party modules that you install in order to confirm their health and credibility.

- Hold-off on upgrading blindly to new versions; allow new package versions some time to circulate before trying them out.
- Before upgrading, make sure to review changelog and release notes for the upgraded version.
- When installing packages make sure to add the `--ignore-scripts` suffix to disable the execution of any scripts by third-party packages.
- Consider adding `ignore-scripts` to your `.npmrc` project file, or to your global npm configuration.

https://snyk.io/blog/ten-npm-security-best-practices/

my-cool-project — -zsh ▸ -zsh — 91×22

```
[~/c/my-cool-project ❯❯❯ npm install                                    master * ]

> funding@1.0.6 postinstall /Users/feross/code/my-cool-project
> node bin/funding.js
```



```
audited 606 packages in 4.945s
found 0 vulnerabilities

~/c/my-cool-project ❯❯❯                                                 master *
```

Code/Dependencies Analysis

snyk/snyk-goof-master:package.json

Snapshot taken a day ago.

Vulnerabilities	39 via 89 paths	**Dependencies**	439
Taken by	Recurring	**Tested with**	package-lock.json,package.json
Manifest	package.json		

NEW ○ Prioritise vulnerabilities by those introduced at runtime. Learn more

Issues | Dependencies

Issue type

- [x] Vulnerabilities 39
- [x] License issues 2

Severity

- [x] High 16
- [x] Medium 17
- [x] Low 8

Choose how to fix these vulnerabilities and open a pull request.

Open a fix PR

HIGH SEVERITY

Prototype Pollution

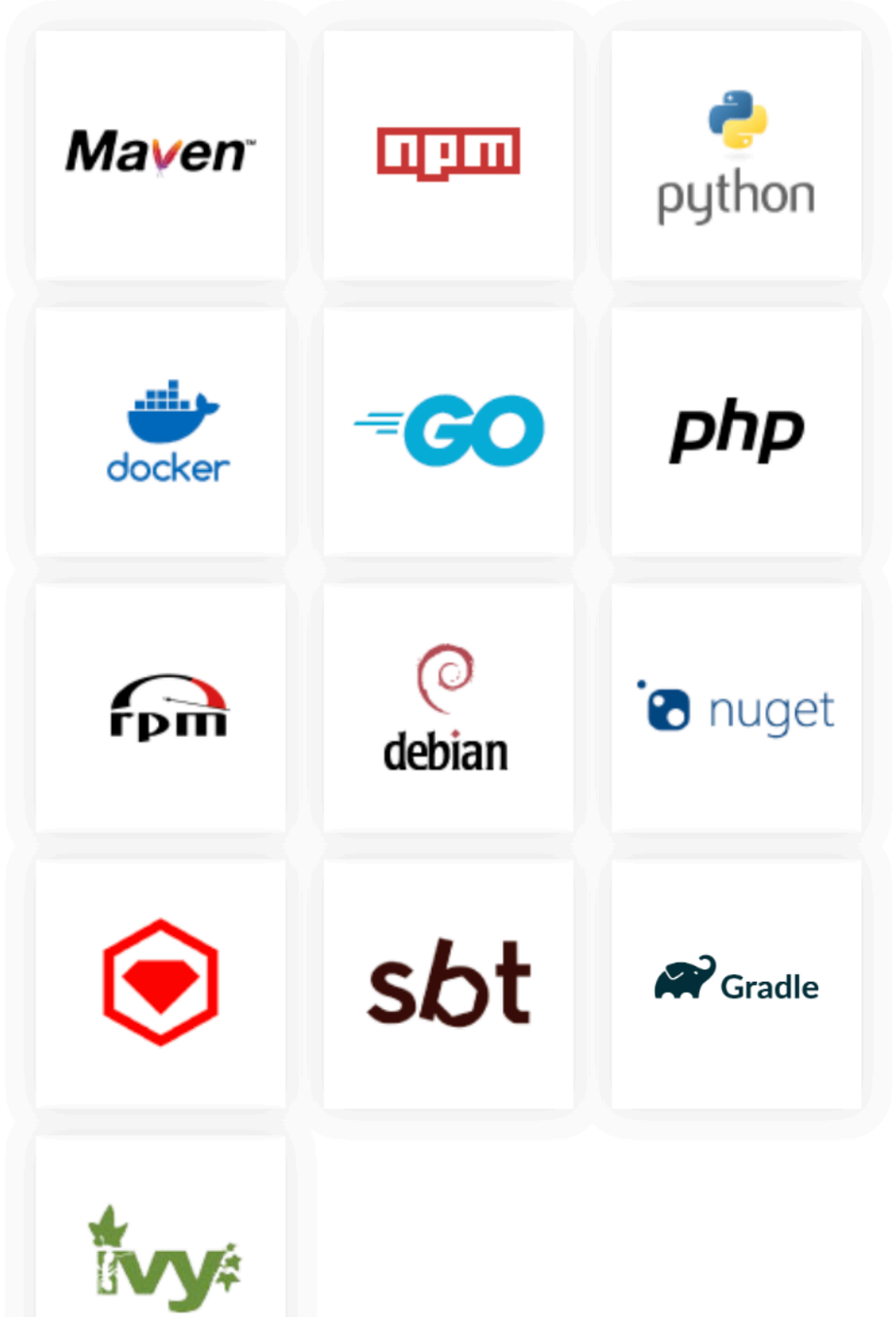
Vulnerable module: handlebars

Introduced through: tap@5.8.0

Detailed paths and remediation

UNIVERSAL
ARTIFACT ANALYSIS

JFrog Xray offers a universal solution that supports all major package types and integrates with various metadata databases such as those related to vulnerabilities, license compliance, component versions and others.



Unlike any other binary analysis product, Xray breaks down artifacts according to their specific packaging. Xray understands each package type, knows how to unpack it and what every underlying layer contains. Each unpacked component is examined individually to uncover potential vulnerabilities and policy violations, mapped out and merged into Xray's universal component graph that represents your entire organization's software structure. This allows you to get maximum visibility into your software dependencies and truly understand the impact of every issue found. Xray provides continuous protection by scanning your components on a regular basis, even though they may have already been found clean and are now exposed to newly discovered vulnerabilities.

Support GPG Signature Verification for Formulas #5852

irlcatgirl opened this issue on 7 Mar 2019 · 15 comments

HOMEBREW

vitorgalvao commented on 8 Mar 2019 — Member

> I'm unsure how you verify that the new package (generally referring to casks) is legitimate

From your earlier comment, I was under the impression you were confusing the two. To clarify, *this issu*
about formulae, not casks. Since they are built differently, they require separate discussions, and we already had this discussion regarding casks.

We had a GPG stanza in casks for years, from before the projects merged. It was just a stub because there was never enough interest from anyone in making it a reality. We eventually decided to not have it

As to your question, the way we ensure the package is legitimate (casks) is by quarantining downloads. other words, we realise there's only so much a small team of volunteers can do, so we shifted the burde to Apple's own system (Gatekeeper).

👍 1

https://github.com/Homebrew/brew/issues/5852

https://blog.npmjs.org/post/172999548390/new-pgp-machinery

```
$ http GET https://registry.npmjs.org/light-cycle | json "versions['1.4.3'].dist.npm-signature" > sig-to-check
```

Next, I get the integrity field for that version:

```
$ http GET https://registry.npmjs.org/light-cycle | json "versions['1.4.3'].dist.integrity"
sha512-sFcuivsDZ99fY0TbvuRC6CDXB8r/ylafjJAMnbSF0y4EMM1/1DtQo40G2WKz1rBbyiz4SLAc3Wa6yZyC4XSGOQ=
```

You can verify that this integrity field matches the tarball you have by calculating the hash yourself. npm uses the ssri module to calculate integrity fields, and it verifies that the hashes match when it adds a tarball to your local cache.

Now that I have an integrity string for my package, I construct the string that ties the two together: package@version:integrity. This is the message I pass to keybase to verify:

```
$ keybase pgp verify --signed-by npmregistry -d sig-to-check -m 'light-cycle@1.4.3:sha512-sFcuivsDZ99fY0TbvuRC6C
▶ INFO Identifying npmregistry
✔ <tracked> public key fingerprint: 0963 1802 8A2B 58C8 4929 D8E1 3D4D 5B12 0276 566A
You last followed npmregistry on 2018-04-10 21:21:57 PDT
✔ <tracked> admin of DNS zone npmjs.com: found TXT entry keybase-site-verification=iK3pjpRBkv-CIJ4PHtWL4TTcF
✔ <tracked> "npmjs" on twitter: https://twitter.com/npmjs/status/981288548845240320 [cached 2018-04-12 13:18:31
✔ <tracked> admin of DNS zone npmjs.org: found TXT entry keybase-site-verification=Ls8jN55i6KesjiX91Ck79bUZ17e
Signature verified. Signed by npmregistry 7 minutes ago (2018-04-13 15:00:37 -0700 PDT).
PGP Fingerprint: 096318028a2b58c84929d8e13d4d5b120276566a.
```

1 When committing changes in your local branch, add the -S flag to the git commit command:

```
$ git commit -S -m your commit message
# Creates a signed commit
```

2 If you're using GPG, after you create your commit, provide the passphrase you set up when you generated your GPG key.

3 When you've finished creating commits locally, push them to your remote repository on GitHub:

```
$ git push
# Pushes your local commits to the remote reposit
```

https://mikegerwitz.com/2012/05/a-git-horror-story-repository-integrity-with-signed-commits

Git extensions for m-of-n signing and verification on commits/tags.

Git Signatures

This repo includes extensions and workflow examples for being able to attach an arbitrary number of GPG signatures to a given commit or tag.

Git already supports commit signing. These tools are intended to compliment that support by allowing a code reviewer and/or release engineer attach their signatures as well.

A CI or build system that enforces m-of-n signature verification on git tags to be released offers strong protection against tampering of the repository by a single bad actor or compromised system.

This approach builds entirely on the git-notes interface which allows the attachment of arbitrary data to an existing commit.

GitHub's SSH key fingerprints

Public key fingerprints can be used to validate a connection to a remote server.

These are GitHub's public key fingerprints (in hexadecimal format):

- `16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48` (RSA)
- `ad:1c:08:a4:40:e3:6f:9c:f5:66:26:5d:4b:33:5d:8c` (DSA)

These are the SHA256 hashes shown in OpenSSH 6.8 and newer (in base64 format):

- `SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8` (RSA)
- `SHA256:br9IjFspm1vxR3iA35FWE+4VTyz1hYVLIE2t1/CeyWQ` (DSA)

Design principles for TUF

1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. Minimized risk through use of offline keys.
5. No key sharing

https://www.youtube.com/watch?v=Gag8RLgvEH4

```
$ docker trust signer add --key cert.pem jeff dtr.example.com/admin/demo
Adding signer "jeff" to dtr.example.com/admin/demo...
Enter passphrase for new repository key with ID 10b5e94:
```

Finally, we will use the delegation private key to sign a particular tag and push it up to the registry.

```
$ docker trust sign dtr.example.com/admin/demo:1
Signing and pushing trust data for local image dtr.example.com/admin/demo:1, may over
The push refers to repository [dtr.example.com/admin/demo]
7bff100f35cb: Pushed
1: digest: sha256:3d2e482b82608d153a374df3357c0291589a61cc194ec4a9ca2381073a17f58e si
Signing and pushing trust metadata
Enter passphrase for signer key with ID 8ae710e:
Successfully signed dtr.example.com/admin/demo:1
```

Docker Notary

Docker Notary

```
$ export DOCKER_CONTENT_TRUST=1

$ docker push dtr.example.com/admin/demo:1
The push refers to repository [dtr.example.com/admin/demo:1]
7bff100f35cb: Pushed
1: digest: sha256:3d2e482b82608d153a374df3357c0291589a61cc194ec4a9ca2381073a17f58e size: 528
Signing and pushing trust metadata
Enter passphrase for signer key with ID 8ae710e:
Successfully signed dtr.example.com/admin/demo:1
```

Remote trust data for a tag or a repository can be viewed by the `$ docker trust inspect` command:

```
$ docker trust inspect --pretty dtr.example.com/admin/demo:1

Signatures for dtr.example.com/admin/demo:1

SIGNED TAG          DIGEST                                                              SIGNERS
1                   3d2e482b82608d153a374df3357c0291589a61cc194ec4a9ca2381073a17f58e    jeff

List of signers and their keys for dtr.example.com/admin/demo:1

SIGNER              KEYS
jeff                8ae710e3ba82

Administrative keys for dtr.example.com/admin/demo:1

  Repository Key:       10b5e94c916a0977471cc08fa56c1a5679819b2005ba6a257aa78ce76d3a1e27
  Root Key:     84ca6e4416416d78c4597e754f38517bea95ab427e5f95871f90d460573071fc
```

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
```

What is Subresource Integrity?

SRI is a new W3C specification that allows web developers to ensure that resources hosted on third-party servers have not been tampered with. Use of SRI is recommended as a best-practice, whenever libraries are loaded from a third-party source.

Learn more about how to use subresource integrity on MDN.

How is Subresource Integrity different to HTTPS?

TLS ensures that the connection between the browser and the server is secure. The resource itself may still be modified server-side by an attacker to include malicious content, yet still be served with a valid TLS certificate. SRI, on the other hand, guarantees that a resource hasn't changed since it was hashed by a web author.

How can I generate Integrity hashes?

Use the generator above or the following shell command:

```
openssl dgst -sha384 -binary FILENAME.js | openssl base64 -A
```

https://www.smashingmagazine.com/2019/04/understanding-subresource-integrity/

Other Things To Note

In this article, I've talked a lot about JavaScript files because that's really where it makes the most sense to defend against hacking attacks. SRI also works with CSS, and so you can use it in exactly the same way there. The risk for malicious CSS is much lower, but the potential to deface a site still exists and who knows what browser bugs could also lead to CSS inadvertently exposing your site to a hacker. So it's work using SRI there too.

Another interesting thing you can do is use a Content Security Policy to specify that any script (or styles) on your page *must* use SRI, and of course that SRI must validate.

```
Content-Security-Policy: require-sri-for script;
```

* AS CODE

@patrickdebois

Ansible

ansible-playbook --check will not make any changes on remote systems. Any module instrumented to support "check mode" will report what changes they would have made rather than making them.

Command:

```
ansible-playbook sample.yml --check
```

Documentation: http://docs.ansible.com/ansible/latest/playbooks_checkmode.html

Apache HTTP Server

configtest runs a configuration file syntax test. It parses the configuration files and either reports Syntax Ok or detailed information about the particular syntax error.

Commands:

```
apachectl configtest
apachectl -t
httpd -t
```

Documentation: https://httpd.apache.org/docs/2.4/programs/apachectl.html

BIND

named-checkconf checks the syntax, but not the semantics, of a named configuration file. ***named-checkzone*** checks the syntax and integrity of a zone file.

Commands:

```
named-checkconf
named-checkconf /etc/named.conf
named-checkzone domain.com /var/named/zone.domain.com
```

https://www.configapp.com/2018/01/09/configuration-file-validation-check/

tfsec

https://github.com/liamg/tfsec

tfsec uses static analysis of your terraform templates to spot potential security issues. Now with terraform v0.12+ support.

Example Output

```
~ ↑
▶ tfsec ./example

4 potential problems detected:

Problem 1

  [AWS006] Resource 'aws_security_group_rule.my-rule' defines a fully open ingress security group rule.
  /Users/liamg/example/main.tf:4

      1 |
      2 | resource "aws_security_group_rule" "my-rule" {
      3 |     type        = "ingress"
      4 |     cidr_blocks = ["0.0.0.0/0"]
      5 | }
      6 |
      7 | resource "aws_alb_listener" "my-alb-listener"{

Problem 2

  [AWS004] Resource 'aws_alb_listener.my-alb-listener' uses plain HTTP instead of HTTPS.
  /Users/liamg/example/main.tf:9

      6 |
      7 | resource "aws_alb_listener" "my-alb-listener"{
      8 |     port     = "80"
      9 |     protocol = "HTTP"
     10 | }
     11 |
     12 | resource "aws_db_security_group" "my-group" {

Problem 3

  [AWS003] Resource 'aws_db_security_group.my-group' uses EC2 Classic. Use a VPC instead.
  /Users/liamg/example/main.tf:12-14

      9 |     protocol = "HTTP"
     10 | }
     11 |
     12 | resource "aws_db_security_group" "my-group" {
     13 |
     14 | }
     15 |
     16 | resource "azurerm_managed_disk" "source" {
     17 |     encryption_settings {

Problem 4
```

https://blog.jessfraz.com/post/docker-containers-on-the-desktop/

Most people use Docker for containing applications to deploy into production or for building their applications in a contained environment. This is all fine & dandy, and saves developers & ops engineers huge headaches, but I like to use Docker in a not-so-typical way.

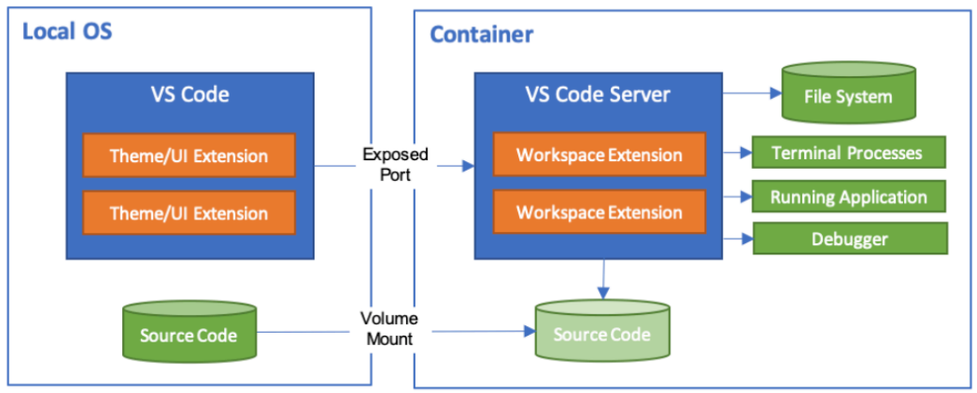
I use Docker to run all the desktop apps on my computers.

Developing inside a Container

Edit

The **Visual Studio Code Remote - Containers** extension lets you use a Docker container as a full-featured development environment. It allows you to open any folder inside (or mounted into) a container and take advantage of Visual Studio Code's full feature set. A `devcontainer.json` file in your project tells VS Code how to access (or create) a **development container** with a well-defined tool and runtime stack. This container can be used to run an application or to sandbox tools, libraries, or runtimes needed for working with a codebase.

Workspace files are mounted from the local file system or copied or cloned into the container. Extensions are installed and run inside the container, where they have full access to the tools, platform, and file system. This means that you can seamlessly switch your entire development environment just by connecting to a different container.



This lets VS Code provide a **local-quality development experience** — including full IntelliSense (completions), code navigation, and debugging — **regardless of where your tools (or code) are located**.

https://code.visualstudio.com/docs/remote/containers

fedora DOCS

Toolbox

As an immutable host, Silverblue is an excellent platform for container-based development and, for working with containers, buildah and podman are recommended.

Silverblue also comes with the *toolbox* utility, which uses containers to provide an environment where development tools and libraries can be installed and used.

Toolbox makes it easy to use a containerized environment for everyday software development and debugging. On immutable operating systems, like Fedora Silverblue, it provides a familiar package-based environment in which tools and libraries can be installed and used. However, toolbox can also be used on package-based systems.

Why use toolbox?

Using toolbox containers to install development tools offers a number of advantages:

- It keeps the host OS clean and stable, and helps to avoid the clutter that can happen after installing lots of development tools and packages.
- Containers are a safe space to experiment: if things go wrong, it's easy to throw a toolbox away and start again.
- Containers are a good way to isolate and organise the dependencies needed for different projects.

https://docs.fedoraproject.org/en-US/fedora-silverblue/toolbox/

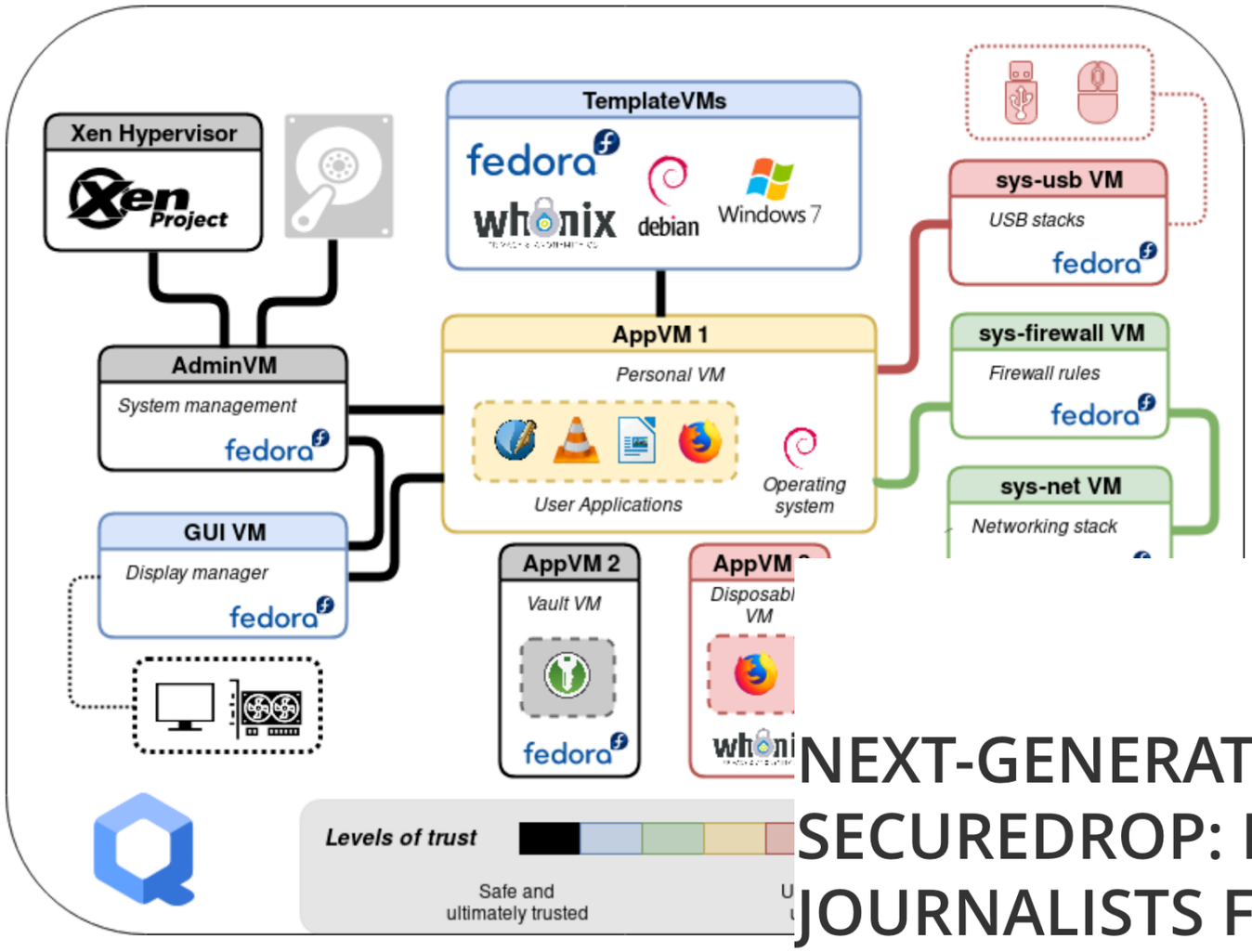
What is Qubes OS?

Qubes OS is a free and open-source security-oriented operating system meant for single-user desktop computing.

Qubes OS leverages **xen-based virtualization** to allow for the creation and management of isolated virtual machines called **qubes**. Qubes, which are also referred to as **domains** or compartments, have specific :

- Purposes : with a predefined set of one or many isolated applications, for personal or professional projects, to manage the **network stack**, **the firewall**, or to fulfill other user-defined purposes.
- Natures : **full-fledged** or **stripped-down** virtual machines which are based on popular operating systems such as **Fedora**, **Debian** or **Windows**.
- Levels of trust : from complete to non-existent. All windows are displayed in a unified desktop environment with **unforgeable colored window borders** so different security levels are easily identifiable.

Qubes OS Overview Example



https://www.qubes-os.org/intro/

https://www.usenix.org/conference/enigma2020/presentation/helsby

ENIGMA

NEXT-GENERATION SECUREDROP: PROTECTING JOURNALISTS FROM MALWARE

Tuesday, January 28, 2020 - 5:00 pm–5:30 pm
Jennifer Helsby, Freedom of the Press Foundation

Abstract:
SecureDrop is a whistleblowing platform originally created in 2012 for journalists to accept leaked documents from anonymous sources. It's now currently in use by dozens of news organizations including NBC News, The Washington Post and The New York Times. The goals of the project are to (1) protect the identity of sources while also to (2) provide a secure environment for journalists to read documents and respond to sources. This talk is about is a new QubesOS-based (Xen) workstation for journalists and other users who need to open potentially malicious documents. The threat of journalists opening malware being submitted through a SecureDrop server is handled via compartmentalization, i.e. opening each potentially malicious document in a separate VM. As journalists are increasingly facing attacks—including those we've observed attempting to phish people through SecureDrop—this can make it significantly safer for them to work with source materials.


Servers + Desktop
Cattle not Pets

LECTURE

Reflections on Trusting Trust

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

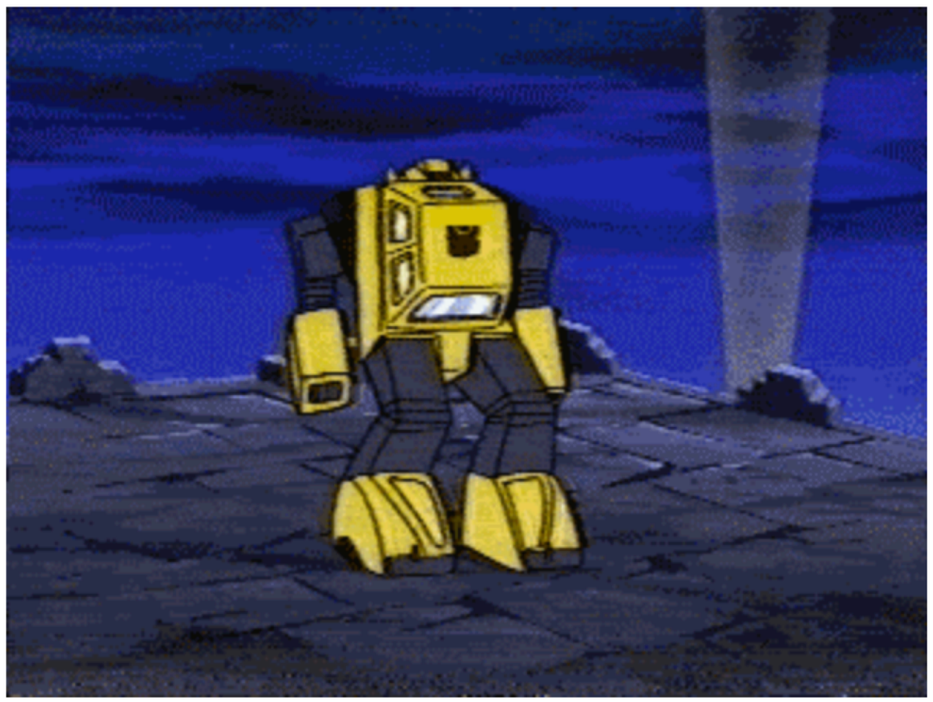
KEN THOMPSON

https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf

volkswagen

Volkswagen detects when your tests are being run in a CI server, and makes them pass.

build passing | code style standard | build passing



Why?

If you want your software to be adopted by Americans, good tests scores from the CI server are very important. Volkswagen uses a defeat device to detect when it's being tested in a CI server and will automatically reduce errors to an acceptable level for the tests to pass. This will allow you to spend *less* time worrying about testing and *more* time enjoying the good life as a trustful software developer.

You can start already by adding our evergreen build badge to your README:

build passing

How to trust SAAS ?

CircleCI-Public / **circleci-dockerfiles**

Watch 31 | Star 296 | Fork 115

Code | Pull requests 2 | Actions | Security | Insights

Dockerfiles for CircleCI's convenience images, built via https://github.com/circleci/circleci-images https://hub.docker.com/r/circleci

circleci docker dockerfile circleci-images

724 commits | **59** branches | **0** packages | **0** releases | **7** contributors

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

cpe-bot Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30629 — Latest commit `1085a11` 9 hours ago

android/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30124	24 days ago
buildpack-deps/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/28833	3 months ago
clojure/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30124	24 days ago
docs/translations	Create README-de.md (#24) [skip ci]	3 months ago
dynamodb/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30608	yesterday
elixir/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30587	2 days ago
erlang/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30566	3 days ago
golang/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30545	4 days ago
jruby/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30124	24 days ago
mariadb/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30566	3 days ago
mongo/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30608	yesterday
mysql/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30251	18 days ago
node/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30419	10 days ago
openjdk/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30608	yesterday
php/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30440	9 days ago
postgres/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/28961	3 months ago
python/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30524	5 days ago
redis/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/29788	last month
ruby/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30124	24 days ago
rust/images	Dockerfiles from https://circleci.com/gh/circleci/circleci-images/30124	24 days ago
README.md	resolved grammatical and spelling errors (#23) [skip ci]	3 months ago

On Prem

layout	section	title	category	order	hide	published	description	sitemap
enterprise	enterprise	LXC-based Installation	advanced-config	5	true	false	How to install CircleCI Enterprise LXC on Ubuntu 14.04	false

The following step-by-step instructions will guide you through the process of installing CircleCI Enterprise LXC-based installation. Note that this is now considered an advanced install option. If you have any questions, please contact us at https://support.circleci.com/hc/en-us.

Prerequisites

1. Deployment environment specifics

CircleCI deployment environment is required to have the following:

- Ubuntu Trusty 14.04 instances. To import the instances into your environment, you can import the image from Ubuntu Cloud releases or plain Ubuntu ISOs. Your probably need the appropriate image named `ubuntu-14.04-server-cloudimg-amd64` for your private cloud environment.
- Access to GitHub Enterprise instance. If running in separate networks, ensure that GitHub Enterprise and CircleCI Enterprise server are whitelisted in the firewall, or VPN connections are setup.
- For ideal setup, CircleCI builders require a second volume, preferably local SSD volumes. This speeds up build runs with maximum effectiveness.

The Audit Log feature is only available for CircleCI installed on your servers or private cloud.

CircleCI logs important events in the system for audit and forensic analysis purposes. Audit logs are separarate from system logs that track performance and network metrics.

Complete Audit logs may be downloaded from the Audit Log page within the Admin section of the application as a CSV file. Audit log fields with nested data contain JSON blobs.

Note: In some situations, the internal machinery may generate duplicate events in the audit logs. The `id` field of the downloaded logs is unique per event and can be used to identify duplicate entries.

Audit Log Events

Following are the system events that are logged. See `action` in the Field section below for the definition and format.

- context.create
- context.delete
- context.env_var.delete
- context.env_var.store
- project.env_var.create
- project.env_var.delete
- project.settings.update
- user.create
- user.logged_in
- user.logged_out
- workflow.job.approve
- workflow.job.finish
- workflow.job.scheduled

CircleCI IP Range

Build Environment

created	last reply	2	3.8k	3	3
A Mar '17	Jun '18	replies	views	users	links

levlaz User — Mar '17

Hello,

We run builds in US-East and US-West so you would need to whitelist all of those ranges 887. If you need more control than this, then the best bet would be to check out our Enterprise 287 offering.

Best,
Lev

✔ Solution

How can I whitelist the IPs addresses for CircleCI for AWS? 18

1 YEAR LATER

Changing your user agent

Now using this concept, I can change my user-agent for the boto library:

```
#!/usr/bin/python
import boto
boto.UserAgent = "MyS3Reader v1.0"
conn = boto.connect_s3()
bucket = conn.lookup('summitroutetest')
for key in bucket:
    print key.name
```

Then I change the condition in the policy:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "StringNotLike": {
                "aws:UserAgent": "MyS3Reader*"
            }
        }
    }
}
```

Notice that I'm using a "StringNotLike" so I'm only checking if the user-agent starts with "MyS3Reader". That way I can change the version number of the app, and this will allow me to debug issues if maybe version 1.0 of the app works but 1.1 does not.

Additionally, I could set policies and user-agents for different code paths, or maybe multiple services run on the same EC2 instance and they should have different permissions. Normally, you would either need to give each service an access key (which is bad) or make your IAM role a superset of all the permissions of the different services running on that host, in cases where you (for whatever reason) have decided to run different services that need different permissions on the same host.

Limit AWS keys by UserAgent

Gitian building

Setup instructions for a Gitian build of Bitcoin Core using a VM or physical system.

Gitian is the deterministic build process that is used to build the Bitcoin Core executables. It provides a way to be reasonably sure that the executables are really built from the git source. It also makes sure that the same, tested dependencies are used and statically built into the executable.

Multiple developers build the source code by following a specific descriptor ("recipe"), cryptographically sign the result, and upload the resulting signature. These results are compared and only if they match, the build is accepted and provided for download.

More independent Gitian builders are needed, which is why this guide exists. It is preferred you follow these steps yourself instead of using someone else's VM image to avoid 'contaminating' the build.

Reproducible Builds

Reproducible builds are a set of software development practices that create an independently-verifiable path from source to binary code. (more)

Contribute | Documentation | Tools | Who is involved? | News | Events | Talks | Continuous tests

Why does it matter?

Whilst anyone can inspect the source code of free and open source software for malicious flaws, most software is distributed pre-compiled with no method to confirm whether they correspond.

This incentivises attacks on developers who release software, not only via traditional exploitation, but also in the forms of political influence, blackmail or even threats of violence.

This is particularly a concern for developers collaborating on privacy or security software: attacking these typically

ReproducibleBuilds About

Got a spare moment? Please migrate this to our new webpages…

With free software, anyone can inspect the source code for malicious flaws. But Debian provide binary packages to its users. The idea of “deterministic” or “reproducible” builds is to empower anyone to verify that no flaws have been introduced during the build process by reproducing byte-for-byte identical binary packages from a given source.

More information about reproducible builds in general are available at reproducible-builds.org.

Open **Reproducable builds** #589

rvagg opened this issue on 5 Jan 2017 · 7 comments

https://github.com/nodejs/build/issues/589

ChALkeR commented on 13 Aug 2019 • edited ▾ Member + 😀 •••

Just tested on Linux (with same restrictions), results:

```
Files node-v12.8.0-1/node and node-v12.8.0-2/node differ
Files node-v12.8.0-1/out/Release/node and node-v12.8.0-2/out/Release/node differ
Files node-v12.8.0-1/out/Release/obj/gen/node_code_cache.cc and node-v12.8.0-2/out/Relea
Files node-v12.8.0-1/out/Release/obj/gen/node_snapshot.cc and node-v12.8.0-2/out/Release
Files node-v12.8.0-1/out/Release/obj.target/node/gen/node_code_cache.o and node-v12.8.0-
Files node-v12.8.0-1/out/Release/obj.target/node/gen/node_snapshot.o and node-v12.8.0-2/
```

Looks like the linker issue could be macOS-specific and does not cause problems on my Linux setup, but we have an issue with unreproducible `node_code_cache.cc` and `node_snapshot.cc` generation. That is a blocker for reproducible builds afaik, and it should be fixed first.

ChALkeR mentioned this issue on 13 Aug 2019

node_code_cache.cc and node_snapshot.cc generation is unreproducible Closed

#29108

ChALkeR commented on 20 Aug 2019 • edited ▾ Member + 😀 •••

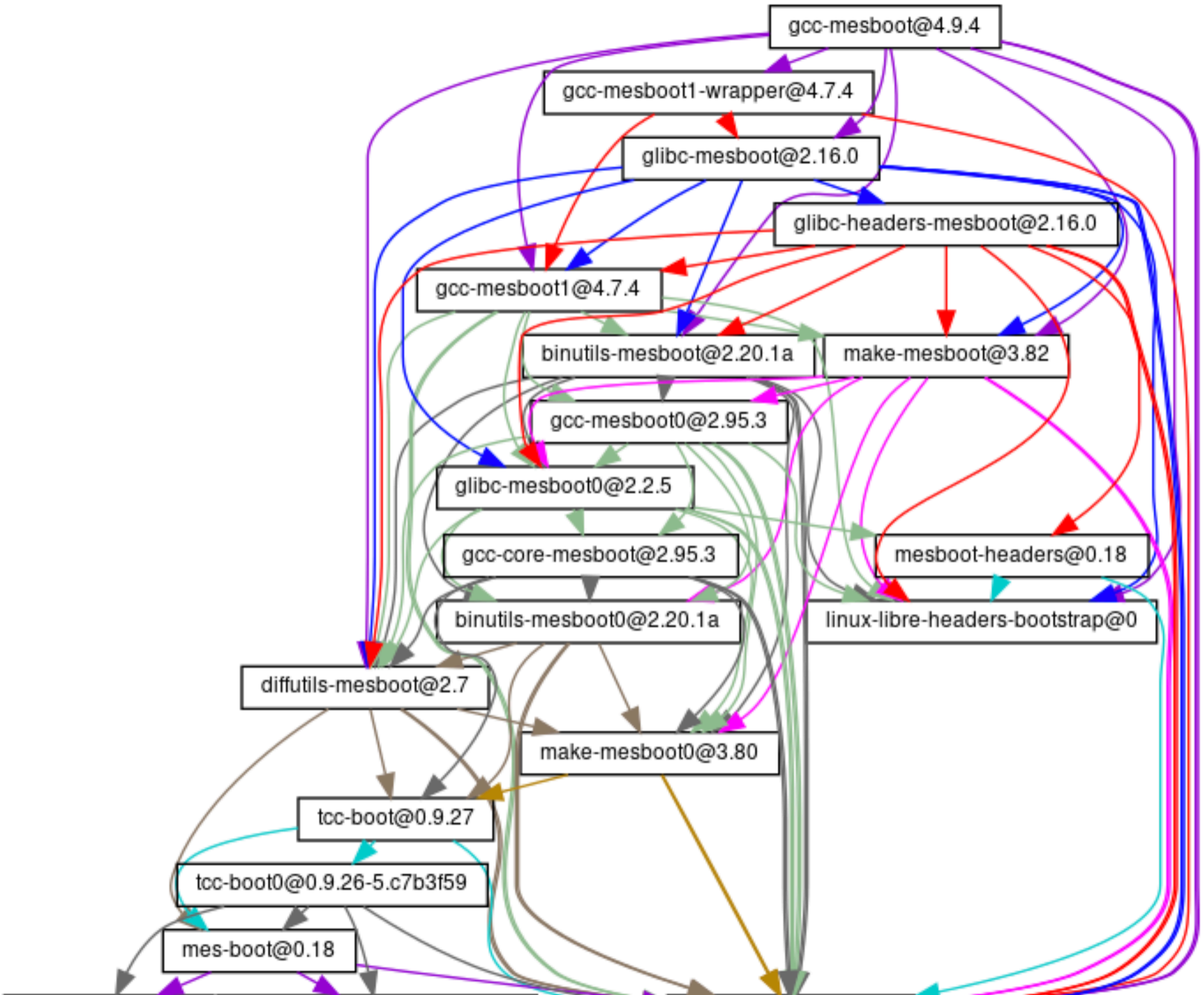
nodejs/node#29108 is not fixed (thanks, **@bnoordhuis**), and two consequent builds from the same

How to Bootstrap: An Old Recipe...

Recipe for yoghurt: Add yoghurt to milk – Anonymous

https://www.youtube.com/watch?v=mhopx8J2Z8s

Below is the generated dependency graph for `gcc-mesboot`, the bootstrap compiler used to build the rest of GuixSD.



1.4 Full Source Bootstrap

There is an obvious solution: we cannot allow any binary seeds in our software stack. Not even in the bootstrap binaries. Maybe that i bit too strong: we want to have the absolute minimum of binary seeds and all binary seeds need to be inspectable and must be reviewe How big would the absolute minimal set be?

1.4.1 The Magical Self-Hosting Hex Assembler

June 2016 I learnt about Stage0. Jeremiah Orians created `hex0` a ~500 byte self-hosting hex assembler. The source code is well documented and the binary is the exact mirror of the source code. I was inspired.

Here is an example of what the `hex0` program looks like; the start of the *hex* function

```
00000060: 4883 f830 7c6f 4883 f83a 7c5a 4883 f841  H..0|oH..:|ZH..A
…
000000d0: 48c7 c0ff ffff ffc3 0000 0000 0000 0000  H...............
000000e0: 4883 e830 c300 0000 0000 0000 0000 0000  H..0............
```

All computer programs look like this: an opaque list of computer codes. The initial programs that we take for granted—the bootstrap binaries—are about 250MB of such numbers: think 250,000 pages full of numbers.

Most computers work pretty well so apparently there is not a pressing need to inspect and study all of these codes. At the same time it tricky to fully trust[6] a computer that was bootstrapped in this way.

Here is what the source code of the `hex0` assembler looks like

```
## function: hex
```

```
pip install diffoscope
```

Released: Jan 14, 202

in-depth comparison of files, archives, and directories

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Meta

License: GNU General Public License v3 or later (GPLv3+) (GPL-3+)

Project description

diffoscope

pypi package 135 | build unstable

diffoscope will try to get to the bottom of what makes files or directories different. It will recursively unpack archives of many kinds and transform various binary formats into more human readable form to compare them. It can compare two tarballs, ISO images, or PDF just as easily.

It can be scripted through error codes, and a report can be produced with the detected differences. The report can be text or HTML. When no type of report has been selected, diffoscope defaults to write a text report on the standard output.

diffoscope was initially started by the "reproducible builds" Debian project and now being developed as part of the (wider) "Reproducible Builds" initiative. It is meant to be able to quickly understand why two builds of the same package produce different outputs. diffoscope was previously named debbindiff.

See the `COMMAND-LINE EXAMPLES` section further below to get you started, as well as more detailed explanations of all the command-line options. The same information is also available in `/usr/share/doc/diffoscope/README.rst` or similar.

Exit status

Exit status is 0 if inputs are the same, 1 if different, 2 if trouble.

Open **Update GitHub client and node types for updates** #1

Changes from all commits · File filter... · Jump to... ·

0 / 2 files viewed

Review chang

556 yarn.lock

Viewed

```
@@ -5840,7 +5892,7 @@ mimic-fn@^1.0.0:
5840    resolved "https://registry.npmjs.org/mimic-fn/-/mimic-fn-1.2.0.tgz#820c86a39334640e99516928bd03fca88057d022"
5841    integrity sha512-jf84uxzwiuiIVKiOLpfYk7N46TSy8ubTonmneY9vrpHNAnp0QBt2BxWV9dO3/j+BoVAb+a5G6YDPW3M5HOdMWQ==
5842
5843  -mimic-fn@^2.0.0:
5844    version "2.1.0"
5845    resolved "https://registry.npmjs.org/mimic-fn/-/mimic-fn-2.1.0.tgz#7ed2c2ccccaf84d3ffcb7a69b57711fc2083401b"
5846    integrity sha512-OqbOk5oEQeAZ8WXWydlu9HJjz9WVdEIvamMCcXmuqUYjTknH/sqsWvhQ3vgwKFRR1HpjvNBKQ37nbJgYzGqGcg==

5892    resolved "https://registry.npmjs.org/mimic-fn/-/mimic-fn-1.2.0.tgz#820c86a39334640e99516928bd03fca88057d022"
5893    integrity sha512-jf84uxzwiuiIVKiOLpfYk7N46TSy8ubTonmneY9vrpHNAnp0QBt2BxWV9dO3/j+BoVAb+a5GDPW3M5HOdMWQ==
5894
5895  +mimic-fn@^2.0.0, mimic-fn@^2.1.0:
5896    version "2.1.0"
5897    resolved "https://registry.npmjs.org/mimic-fn/-/mimic-fn-2.1.0.tgz#7ed2c2ccccaf84d3ffcb7a69b57711fc2083401b"
5898    integrity sha512-OqbOk5oEQeAZ8WXWydlu9HJjz9WVdEIvamMCcXmuqUYjTknH/sqsWvhQ3vgwKFRR1HpjvNBKQ37nbJgYzGqGcg==
```

```
@@ -5922,8 +5974,8 @@ ms@2.0.0:
5922
5923    ms@2.1.1:
5924      version "2.1.1"
5925    -  resolved "https://registry.npmjs.org/ms/-/ms-2.1.1.tgz#30a5864eb3ebb0a66f2ebe6d727af06a09d86e0a"
5926    -  integrity sha512-tgp+dl5cGk28utYktBsrFqA7HKgrhgPsg6Z/EfhWI4gl1Hwq8B/GmY/0oXZ6nF8hDVesS/FpnYaD/kOWhYQvyg==
5927
5928    ms@^2.1.1:
5929      version "2.1.2"

5974
5975    ms@2.1.1:
5976      version "2.1.1"
5977    +  resolved "https://github.com/lirantal/ms/tarball/master"
5978    +  integrity sha512-WXEVc7cnWVezFbJ0MYYgjfekrQtKI2R/g+PMhyeiyQtwrFQZLZJDViv7y6a+YO9EYMljtam,Df4sNhYFt5NA==
5979
5980    ms@^2.1.1:
5981      version "2.1.2"
```

```
@@ -6115,28 +6167,19 @@ npm-packlist@^1.1.6:
6115      ignore-walk "^3.0.1"
6116      npm-bundled "^1.0.1"
6117
6118    -npm-path@^2.0.2:
6119    -  version "2.0.4"

6167      ignore-walk "^3.0.1"
6168      npm-bundled "^1.0.1"
6169
```

Our Polymorphism is random, not unstable

We tend to use the word random a little… randomly. Randomness is unpredictability of an outcome, not misunderstanding of the outcome. Randomness derives from having good control, understanding and predictability of a process.

That thing where you don't understand what's happening, why it's happening and what to do about it… that's called "instability."

Each build of Polymorphic Linux is stable. Given the exact same entropy source, with the same input events, it generates a stable, predictable, reproducible binary. It doesn't "screw up" things without understanding or control. It scrambles things based on an unpredictable source of entropy, but with a reproducible source, so is stable and reproducible.

Bazel

Documentation Contribute Blog Search GitHub

{Fast, Correct} - Choose two

Build and test software of any size, quickly and reliably

GET BAZEL GET STARTED

Hermetic builds

Why Bazel?

Speed up your builds and tests

Bazel only rebuilds what is necessary. With advanced local and distributed caching, optimized dependency analysis and parallel execution, you get fast and incremental builds.

One tool, multiple languages

Build and test Java, C++, Android, iOS, Go and a wide variety of other language platforms. Bazel runs on Windows, macOS, and Linux.

Scalable

Bazel helps you scale your organization, codebase and Continuous Integration system. It handles codebases of any size, in multiple repositories or a huge monorepo.

Extensible to your needs

Easily add support for new languages and platforms with Bazel's familiar extension language. Share and re-use language rules written by the growing Bazel community.

Trusted by industry leaders

"Distroless" Docker Images

"Distroless" images contain only your application and its runtime dependencies. They do not contain package managers, shells or any other programs you would expect to find in a standard Linux distribution.

For more information, see this talk (video).

Why should I use distroless images?

Restricting what's in your runtime container to precisely what's necessary for your app is a best practice employed by Google and other tech giants that have used containers in production for many years. It improves the signal to noise of scanners (e.g. CVE) and reduces the burden of establishing provenance to just what you need.

How do I use distroless images?

These images are built using the bazel tool, but they can also be used through other Docker image build tooling.

Entrypoints

Note that distroless images by default do not contain a shell. That means the Dockerfile `ENTRYPOINT` command, when defined, must be specified in `vector` form, to avoid the container runtime prefixing with a shell.

This works:

```
ENTRYPOINT ["myapp"]
```

But this does not work:

Example SBOM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bom xmlns="http://cyclonedx.org/schema/bom/1.1" serialNumber="ur
  <components>
    <component type="library">
      <publisher>Apache</publisher>
      <group>org.apache.tomcat</group>
      <name>tomcat-catalina</name>
      <version>9.0.14</version>
      <hashes>
        <hash alg="MD5">3942447fac867ae5cdb3229b658f4d48</hash>
        <hash alg="SHA-1">e6b1000b94e835ffd37f4c6dcbdad43f4b48a02
        <hash alg="SHA-256">f498a8ff2dd007e29c2074f5e4b01a9a01775
        <hash alg="SHA-512">e8f33e424f3f4ed6db76a482fde1a5298970e
      </hashes>
      <licenses>
        <license>
          <id>Apache-2.0</id>
        </license>
      </licenses>
      <purl>pkg:maven/org.apache.tomcat/tomcat-catalina@9.0.14?pa
    </component>
      <!-- More components here -->
  </components>
</bom>
```


Using Tern

WARNING: The CLI has changed since the last release. Visit Tern's PyPI project page to find the correct CLI options or just run `tern -h`.

Tern creates a report containing the Bill of Materials (BoM) of a container image, including notes about how it collects this information, and files for which it has no information about. Currently, Tern supports only containers built using Docker. This is the most ubiquitous type of container image that exists so the project started with a focus on those. However, it is architected to support other images that closely follow the OCI image spec.

Generating a BoM report for a Docker image

If you have a Docker image pulled locally and want to inspect it

```
$ tern -l report -i debian:jessie
```

Take a look at report.txt to see what packages are installed in the Docker image and how Tern got this information. If you encounter any errors, please file an issue.

Generating a BoM report from a Dockerfile

You can provide a Dockerfile to Tern to figure out the Bill of Materials and other information. Tern will build the image, analyze it with respect to the Dockerfile and discard the image. This is useful to engineers who are developing a Dockerfile for their app or in a container build and release pipeline.

```
$ tern -l report -d samples/photon_git/Dockerfile
```

Take a look at report.txt to see what packages you would be shipping if you were to use the given Dockerfile. Feel free to try this out on the other sample Dockerfiles in the samples directory or on Dockerfiles you may be working with. If it doesn't work for you, please file an issue.

Report Formats

Tern creates BoM reports suitable to read over or to provide to another tool for consumption.

Human Readable Format

The default report Tern produces is a human readable report. The object of this report is to give the container developer a deeper understanding of what is installed in a container image during development. This allows a developer to glean basic information about the container such as what the true base operating system is, what the app dependencies are, if the container is using an official or personal repository for sources or binaries, whether the dependencies are at the correct versions, etc.

```
$ tern -l report -i golang:1.12-alpine -o output.txt
```

Forensics inspired:

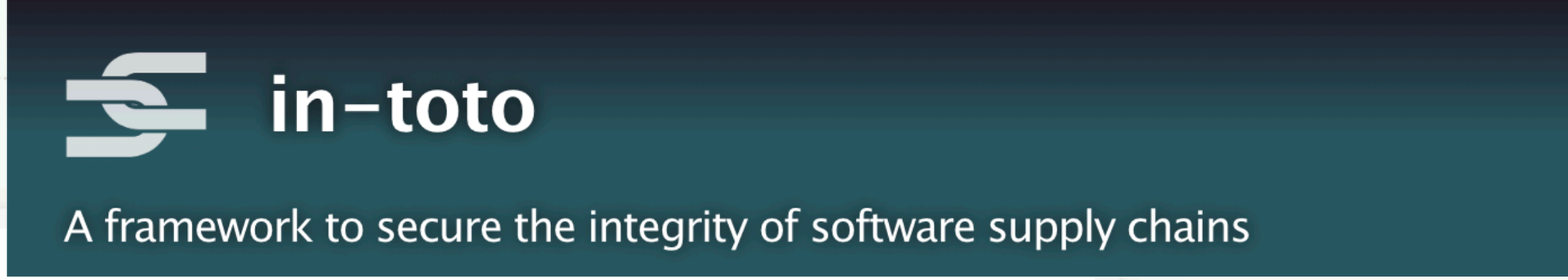
- docker-explorer: A tool to help forensicate offline docker acquisitions
  - https://github.com/google/docker-explorer
- docker-forensics: Uses mac-robber to see what's in there
  - https://github.com/att/docker-forensics
- dive: A tool for exploring each layer in a docker image
  - https://github.com/wagoodman/dive
- openscap: OpenSCAP is a NIST-certified scanner.
  - https://hub.docker.com/r/openscap/openscap

Reverse Engineer inspired:

- DfImage: Reverse Engineer a Dockerfile from a Docker image
  - https://github.com/LanikSJ/dfimage
- Whaler:
  - https://github.com/P3GLEG/Whaler
- Docker Historian
  - https://github.com/alexivkin/docker-historian

Layout

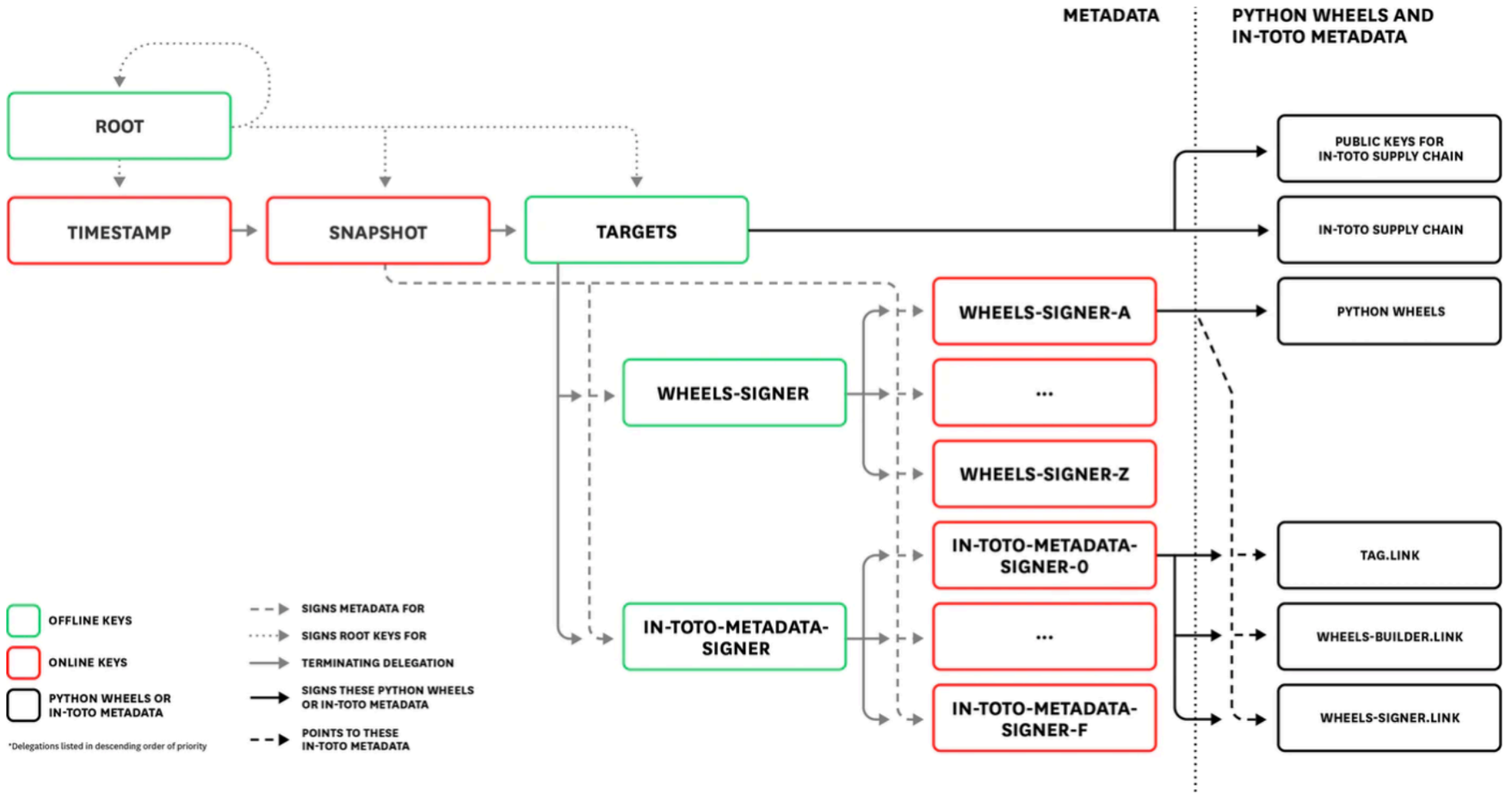
debian/root.layout




```
{
  "signed": {
    "_type": "layout",
    "readme": "",
    "expires": "2017-09-02T12:57:02Z",
    "steps": [
      {
        "_type": "step",
        "name": "fetch",
        "expected_command": [
          "dget",
          "http://cdn.debian.net/debian/pool/main/g/grep/grep_2.12-2.dsc"
        ],
        "pubkeys": [
          "12c55e46654c682d3ffd3b63492adf422e6812eb1ac41574d083b9e770d1e4c2"
        ],
        "threshold": 1,
        "expected_materials": [
          [
            "DISALLOW",
            "*"
          ]
        ],
        "expected_products": [
          [
            "ALLOW",
            "*"
          ]
        ]
      },
      {
        "_type": "step",
        "name": "extract",
        "expected_command": [
```

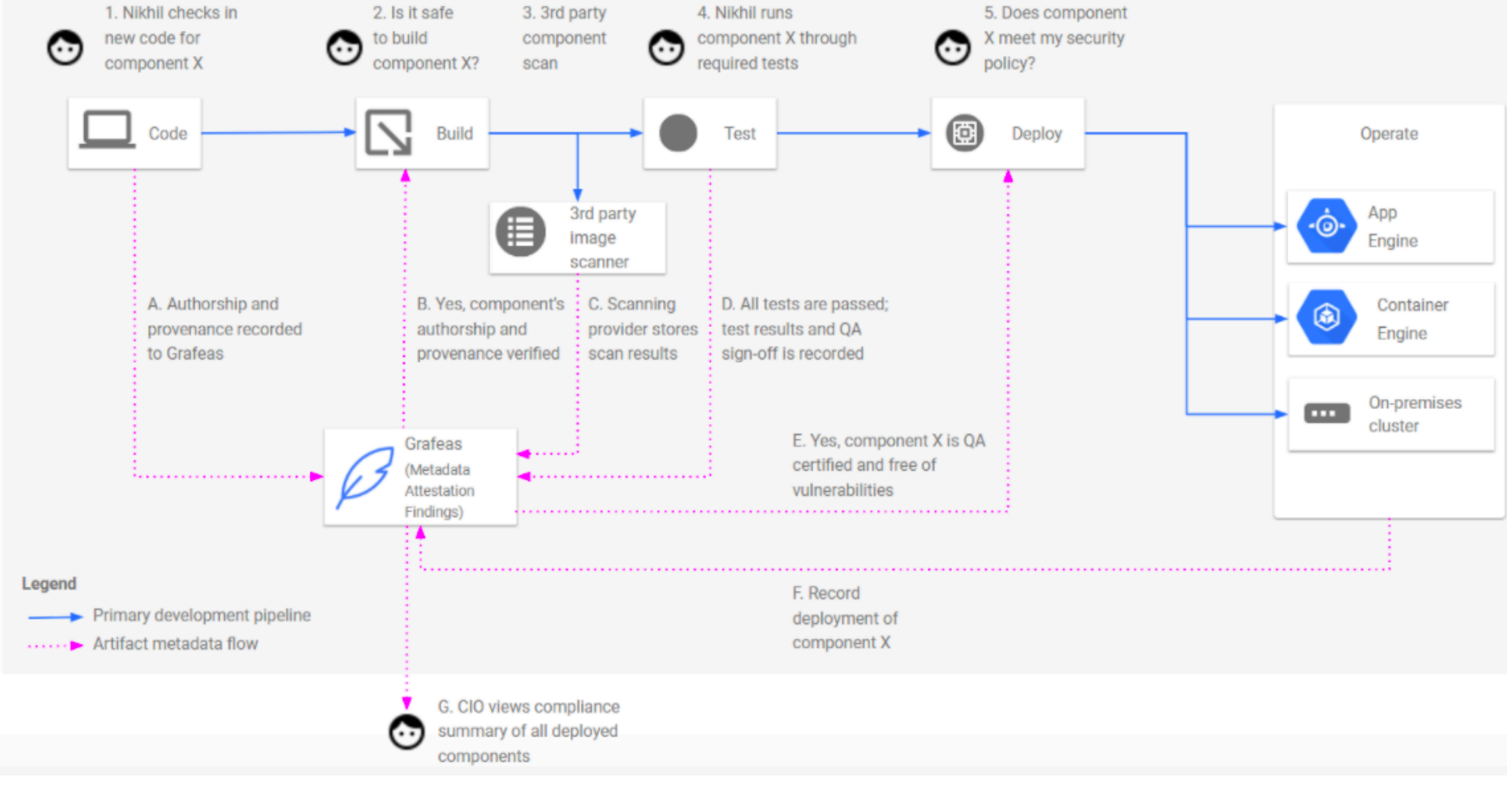
Secure transport with TUF

While in-toto provides end-to-end verification of a software supply chain, it does not solve a crucial problem that arises in practice: how to securely distribute, revoke, and replace the public keys used to verify the supply chain. This mechanism must be compromise-resilient itself.



Grapheas

Audit and govern your software supply chain with Grafeas



Needs continuously monitoring

If it's hard
do it
more often

"Building trust in a server is no different to a person. We grow trust through transparency and communication "

@chrisbuxton

@patrickdebois


Thank you
for thinking
with me
@patrickdebois