tuleap

enalean
software engineering is now

# Continuous Delivery and Secure Code Delivery : impossible ?

SnowCamp - 25/01/2018

## Who Am I

- **Thomas Gerbet (@el_suisse)**
- **@TuleapOpenALM contributor and maintainer**
- **@GreHackConf organizer**
- **Alpaca lover**

## Secure Code Delivery: Definition

- **Getting software to your users in a way they can trust**
  - Userbase Consistency Verification
  - Reproducible Builds
  - Cryptographic Signatures

## Userbase Consistency Verification

**Everyone gets the same deliverable**

**Everyone can verify it**

**Prevent targeted attacks**

**Public append-only ledger**

**(Sorry, blockchains are not needed)**

## Reproducible Builds

**Code ⇔ Deliverable**

**Hard to backdoor the deliverable**

**Help if a maintainer is malicious or attacked**

**https://reproducible-builds.org/**

## Cryptographic Signatures

**Anchor of trust to maintainers**

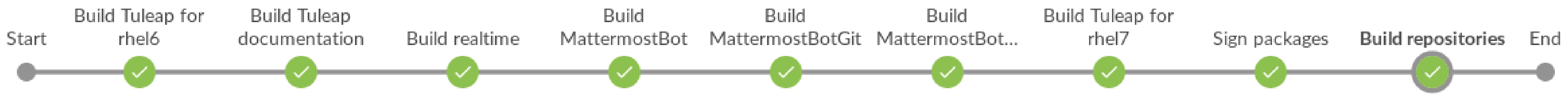
**Software is released by the authorized people**

## Secure Code Delivery: Challenges

- **Userbase Consistency Verification**
  - Mostly not supported by existing delivery software
  - Infrastructure can be hard to maintain
- **Reproducible builds**
  - Heterogeneous build systems
  - Codebase size
- **Cryptographic Signatures**
  - Releases frequency

# Tuleap release process

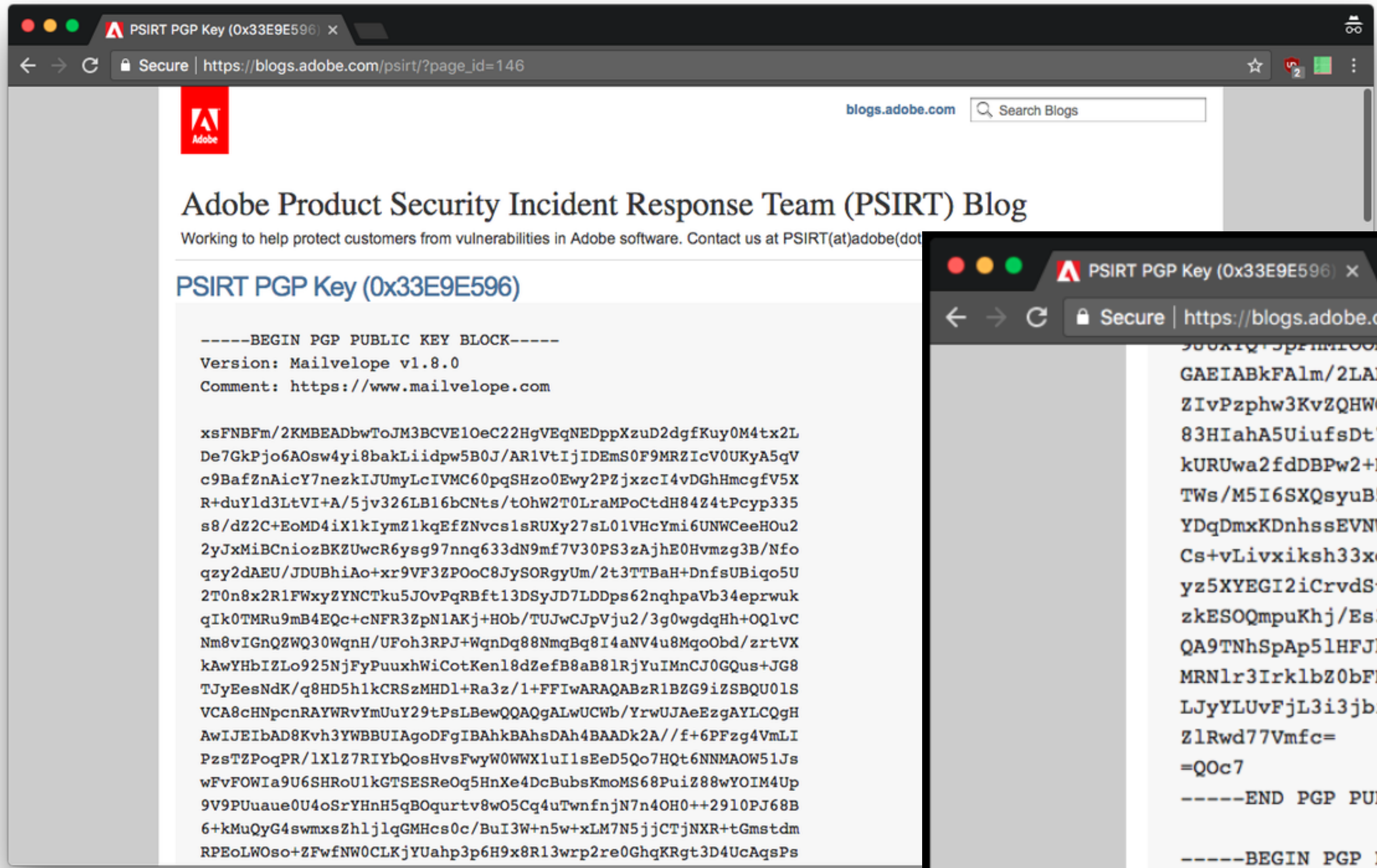
- **What a Tuleap release is?**
  - Delivered as RPM packages (GPG signature)
  - Two kinds:
    - Built out of tags (monthly snapshot)
    - Built out of master
- **Continuous Delivery**
  - Each contributions integrated triggers and publish a new release
  - ~140 monthly releases
  - Heavy use of Jenkins pipeline and Docker containers



## Tuleap packages signing (2012-2016)

- **Only packages built from tags were signed**
- **No automation: process entirely manual**
- **GPG key used to sign the RPM packages:**
  - Stored encrypted with the GPG keys of maintainers
  - Master passphrase of the key also stored encrypted with the GPG keys of the maintainers
  - In the best secret store: a Subversion repository

# Hard to Get Right, Easy to Fail


PSIRT PGP Key (0x33E9E596)

https://blogs.adobe.com/psirt/?page_id=146

blogs.adobe.com

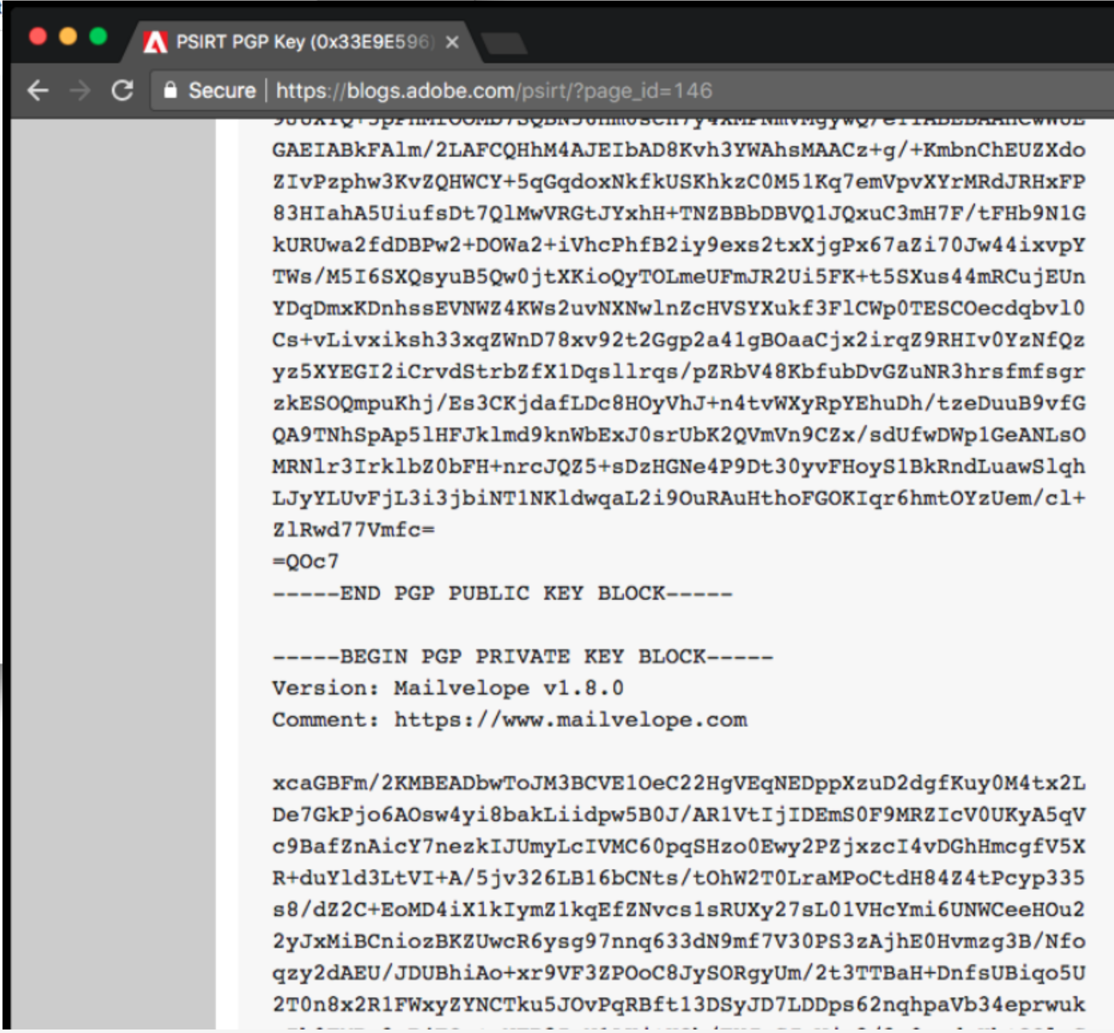
Adobe Product Security Incident Response Team (PSIRT) Blog

Working to help protect customers from vulnerabilities in Adobe software. Contact us at PSIRT(at)adobe(dot

## PSIRT PGP Key (0x33E9E596)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: Mailvelope v1.8.0
Comment: https://www.mailvelope.com

xsFNBFm/2KMBEADbwToJM3BCVE1OeC22HgVEqNEDppXzuD2dgfKuy0M4tx2L
De7GkPjo6AOsw4yi8bakLiidpw5B0J/AR1VtIjIDEmS0F9MRZIcV0UKyA5qV
c9BafZnAicY7nezkIJUmyLcIVMC60pqSHzo0Ewy2PZjxzcI4vDGhHmcgfV5X
R+duYld3LtVI+A/5jv326LB16bCNts/tOhW2T0LraMPoCtdH84Z4tPcyp335
s8/dZ2C+EoMD4iX1kIymZ1kqEfZNvcs1sRUXy27sL01VHcYmi6UNWCeeHOu2
2yJxMiBCniozBKZUwcR6ysg97nnq633dN9mf7V30PS3zAjhE0Hvmzg3B/Nfo
qzy2dAEU/JDUBhiAo+xr9VF3ZPOoC8JySORgyUm/2t3TTBaH+DnfsUBiqo5U
2T0n8x2R1FWxyZYNCTku5JOvPqRBft13DSyJD7LDDps62nqhpaVb34eprwuk
qIk0TMRu9mB4EQc+cNFR3ZpN1AKj+HOb/TUJwCJpVju2/3g0wgdqHh+OQlvC
Nm8vIGnQZWQ30WqnH/UFoh3RPJ+WqnDq88NmqBq8I4aNV4u8MqoObd/zrtVX
kAwYHbIZLo925NjFyPuuxhWiCotKenl8dZefB8aB8lRjYuIMnCJ0GQus+JG8
TJyEesNdK/q8HD5h1kCRSzMHDl+Ra3z/1+FFIwARAQABzR1BZG9iZSBQU0lS
VCA8cHNpcnRAYWRvYmUuY29tPsLBewQQAQgALwUCWb/YrwUJAeEzgAYLCQgH
AwIJEIbAD8Kvh3YWBBUIAgoDFgIBAhkBAhsDAh4BAADk2A//f+6PFzg4VmLI
PzsTZPoqPR/lXlZ7RIYbQosHvsFwyW0WWX1uIlsEeD5Qo7HQt6NNMAOW51Js
wFvFOWIa9U6SHRoU1kGTSESReOq5HnXe4DcBubsKmoMS68PuiZ88wYOIM4Up
9V9PUuaue0U4oSrYHnH5qBOqurtv8wO5Cq4uTwnfnjN7n4OH0++2910PJ68B
6+kMuQyG4swmxsZhljlqGMHcs0c/BuI3W+n5w+xLM7N5jjCTjNXR+tGmstdm
RPEoLWOso+ZFwfNW0CLKjYUahp3p6H9x8R13wrp2re0GhqKRgt3D4UcAqsPs
```


PSIRT PGP Key (0x33E9E596)

https://blogs.adobe.com/psirt/?page_id=146

```
GAEIABkFAlm/2LAFCQHhM4AJEIbAD8Kvh3YWAhsMAACz+g/+KmbnChEUZXdo
ZIvPzphw3KvZQHWCY+5qGqdoxNkfkUSKhkzC0M51Kq7emVpvXYrMRdJRHxFP
83HIahA5UiufsDt7QlMwVRGtJYxhH+TNZBBbDBVQ1JQxuC3mH7F/tFHb9N1G
kURUwa2fdDBPw2+DOWa2+iVhcPhfB2iy9exs2txXjgPx67aZi70Jw44ixvpY
TWs/M5I6SXQsyuB5Qw0jtXKioQyTOLmeUFmJR2Ui5FK+t5SXus44mRCujEUn
YDqDmxKDnhssEVNWZ4KWs2uvNXNwlnZcHVSYXukf3FlCWp0TESCOecdqbvl0
Cs+vLivxiksh33xqZWnD78xv92t2Ggp2a41gBOaaCjx2irqZ9RHIv0YzNfQz
yz5XYEGI2iCrvdStrbZfX1Dqsllrqs/pZRbV48KbfubDvGZuNR3hrsfmfsgr
zkESOQmpuKhj/Es3CKjdafLDc8HOyVhJ+n4tvWXyRpYEhuDh/tzeDuuB9vfG
QA9TNhSpAp5lHFJklmd9knWbExJ0srUbK2QVmVn9CZx/sdUfwDWp1GeANLsO
MRNlr3IrklbZ0bFH+nrcJQZ5+sDzHGNe4P9Dt30yvFHoyS1BkRndLuawSlqh
LJyYLUvFjL3i3jbiNT1NKldwqaL2i9OuRAuHthoFGOKIqr6hmtOYzUem/cl+
ZlRwd77Vmfc=
=QOc7
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: Mailvelope v1.8.0
Comment: https://www.mailvelope.com

xcaGBFm/2KMBEADbwToJM3BCVE1OeC22HgVEqNEDppXzuD2dgfKuy0M4tx2L
De7GkPjo6AOsw4yi8bakLiidpw5B0J/AR1VtIjIDEmS0F9MRZIcV0UKyA5qV
c9BafZnAicY7nezkIJUmyLcIVMC60pqSHzo0Ewy2PZjxzcI4vDGhHmcgfV5X
R+duYld3LtVI+A/5jv326LB16bCNts/tOhW2T0LraMPoCtdH84Z4tPcyp335
s8/dZ2C+EoMD4iX1kIymZ1kqEfZNvcs1sRUXy27sL01VHcYmi6UNWCeeHOu2
2yJxMiBCniozBKZUwcR6ysg97nnq633dN9mf7V30PS3zAjhE0Hvmzg3B/Nfo
qzy2dAEU/JDUBhiAo+xr9VF3ZPOoC8JySORgyUm/2t3TTBaH+DnfsUBiqo5U
2T0n8x2R1FWxyZYNCTku5JOvPqRBft13DSyJD7LDDps62nqhpaVb34eprwuk
```

## No Audit Trail

- **When was the key used?**
- **By who?**
- **And for what?**

- **Not even a slight chance to detect a compromise**
- **Can tell what happened**

## Does Not Scale

- **~ 10/15mn to sign the packages:**
  - Find the obscure wiki page describing the process
  - Remember how GPG CLI tool works
  - Read 3 times the wiki in the hope of not making mistakes
  - Sign the packages (finally!)
- **140 times a month?**

## Tuleap packages signing (2012-2016)

~ **Packages signed**
  - Most used packages were not signed

✘ **Automated**

✘ **Leak prevention**

✘ **Least privilege**

✘ **Auditing capabilities**

# HashiCorp Vault

HashiCorp
Vault

## HashiCorp Vault: Secret backends

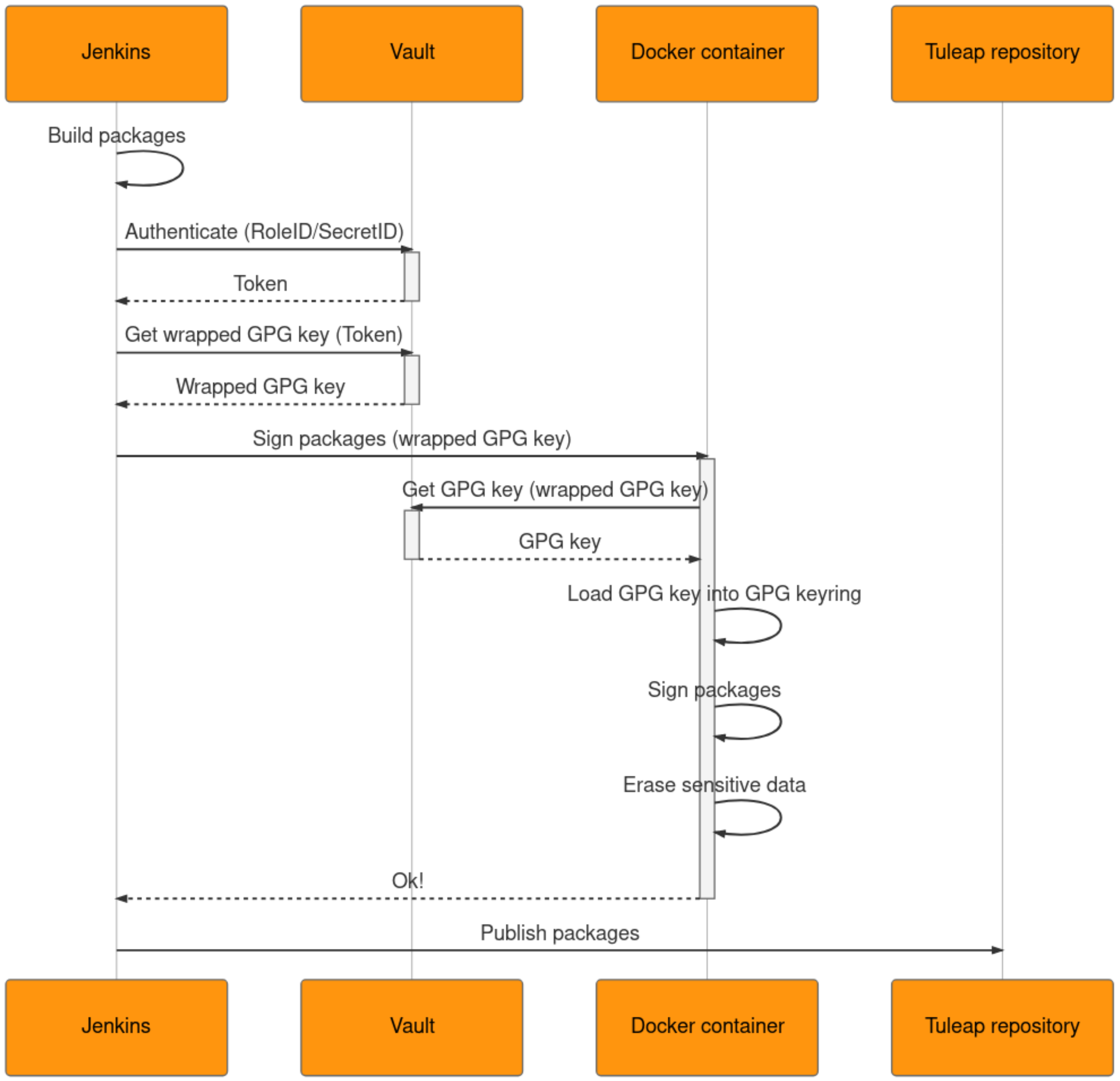
- **Store and generate secrets**
  - AWS
  - Consul
  - Databases
  - Key/Value
  - PKI
  - RabbitMQ
  - SSH
  - TOTP
  - Transit (Cryptography as a Service)
  - ...

**No GPG backend?**

## Tuleap packages signing (early/mid 2017)

- **Workaround for lack of Vault GPG secret backend**
  - Store GPG key in the key/value secret backend
  - Retrieve the GPG key whenever you need it
- **Authenticate Jenkins against Vault**
  - AppRole authentication backend with regular credentials rotation
  - Jenkins credentials plugin
  - Restricted policies
- **Pass the GPG key from Jenkins to the Docker container doing the heavy lifting**
  - Response wrapping: short lived and one use token allows the container to get the key

# Tuleap packages signing (early/mid 2017)

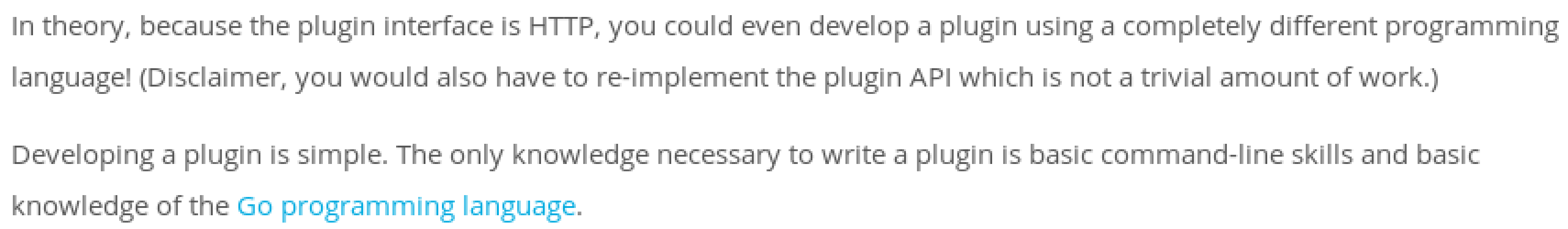


## Tuleap packages signing (early/mid 2017)

- ✓ **Packages signed**
- ✓ **Automated**
- ~ **Leak prevention**
  - GPG key can easily be extracted by an insider
  - Need to be very careful to not leak the key accidentally
- ~ **Least privilege**
- ~ **Auditing capabilities**
  - Retrieval of the key is logged
  - What’s done with the key is not logged

## HashiCorp Vault: Plugins

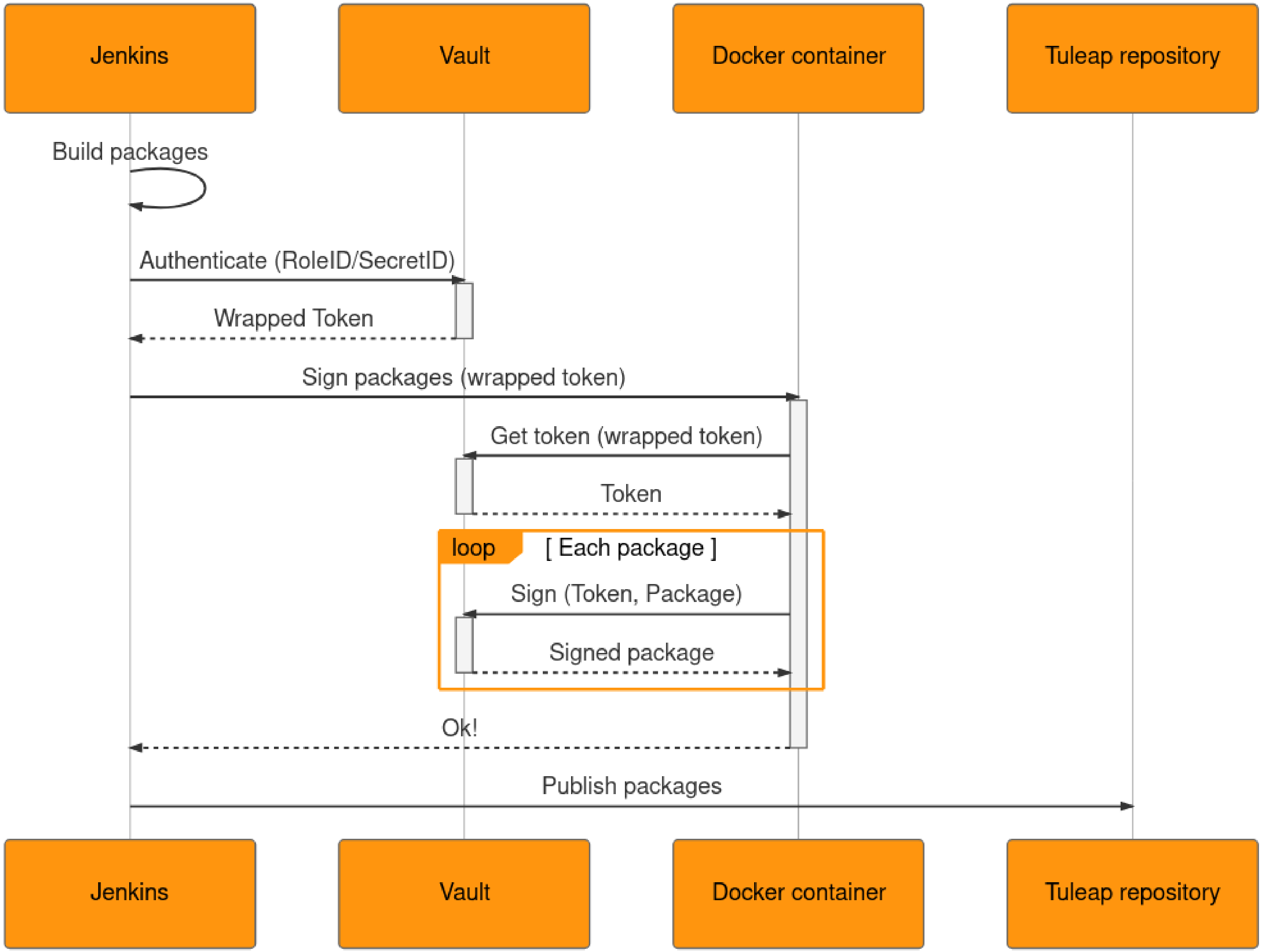
- **Possible since Vault 0.8.0 (August 2017)**
- **We can write our GPG secret backend**
- **Golang**
  - (Almost) No prior knowledge of the language
  - High level operations on GPG operations is already there


In theory, because the plugin interface is HTTP, you could even develop a plugin using a completely different programming language! (Disclaimer, you would also have to re-implement the plugin API which is not a trivial amount of work.)

Developing a plugin is simple. The only knowledge necessary to write a plugin is basic command-line skills and basic knowledge of the Go programming language.

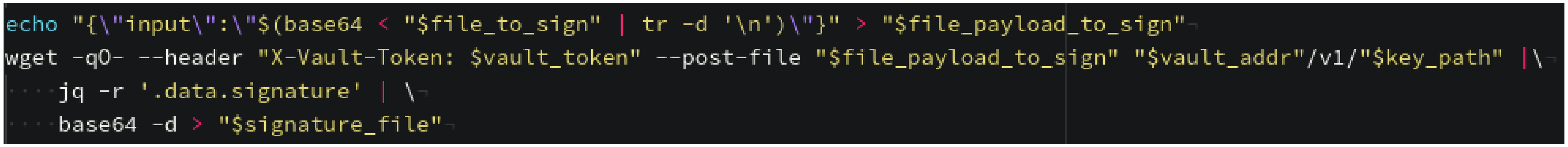
**Challenge accepted!**

# Tuleap packages signing (end 2017 - 2018)



# Call Vault from rpmsign(8)

## Switch GPG binary by a small shell script (rpmsign option)



```
echo "{\"input\":\"$(base64 < "$file_to_sign" | tr -d '\n')\"}" > "$file_payload_to_sign"
wget -qO- --header "X-Vault-Token: $vault_token" --post-file "$file_payload_to_sign" "$vault_addr"/v1/"$key_path" |\
    jq -r '.data.signature' | \
    base64 -d > "$signature_file"
```

## Tuleap packages signing (end 2017 - 2018)

- ✓ **Packages signed**
- ✓ **Automated**
- ✓ **Leak prevention**
- ✓ **Least privilege**
- ✓ **Auditing capabilities**

## Outcome

- **No universal solution for signing deliverable**
- **HashiCorp Vault**
  - Integrates easily with your Jenkins pipeline and the rest of your infrastructure
  - Flexible secrets management
- **Secure Code Delivery is a mandatory step to bring security into your SDLC process**
  - Bringing it small steps by small is fine: it’s not a black and white situation

## Play with it at home

### RPM signing with HashiCorp Vault K/V backend and Jenkins pipelines

All resources (Jenkinsfiles, Dockerfiles, scripts…) are linked in the following blogpost

https://blog.tuleap.org/delivering-rpm-packages-securely-and-continuously-jenkins-and-hashicorp-vault

### Vault GPG secret backend

https://github.com/LeSuisse/vault-gpg-plugin

Ongoing work to integrate it into Vault

## Questions?

## Why Not Sigul?

- **Lots of unknown (to us) moving parts**
- **We already had HashiCorp Vault up and running**
- **No deployment outside of the Fedora infrastructure?**
- **Audit capacities unclear**

## Vault GPG secret backend plugin performance

- **~1mn30 for our 84 packages (~150MB)**
- **No optimization have been done yet**
  - Signing packages is done sequentially in our pipeline
  - Network limitation
- **Raw performance of the plugin will be improved once integrated directly into Vault**