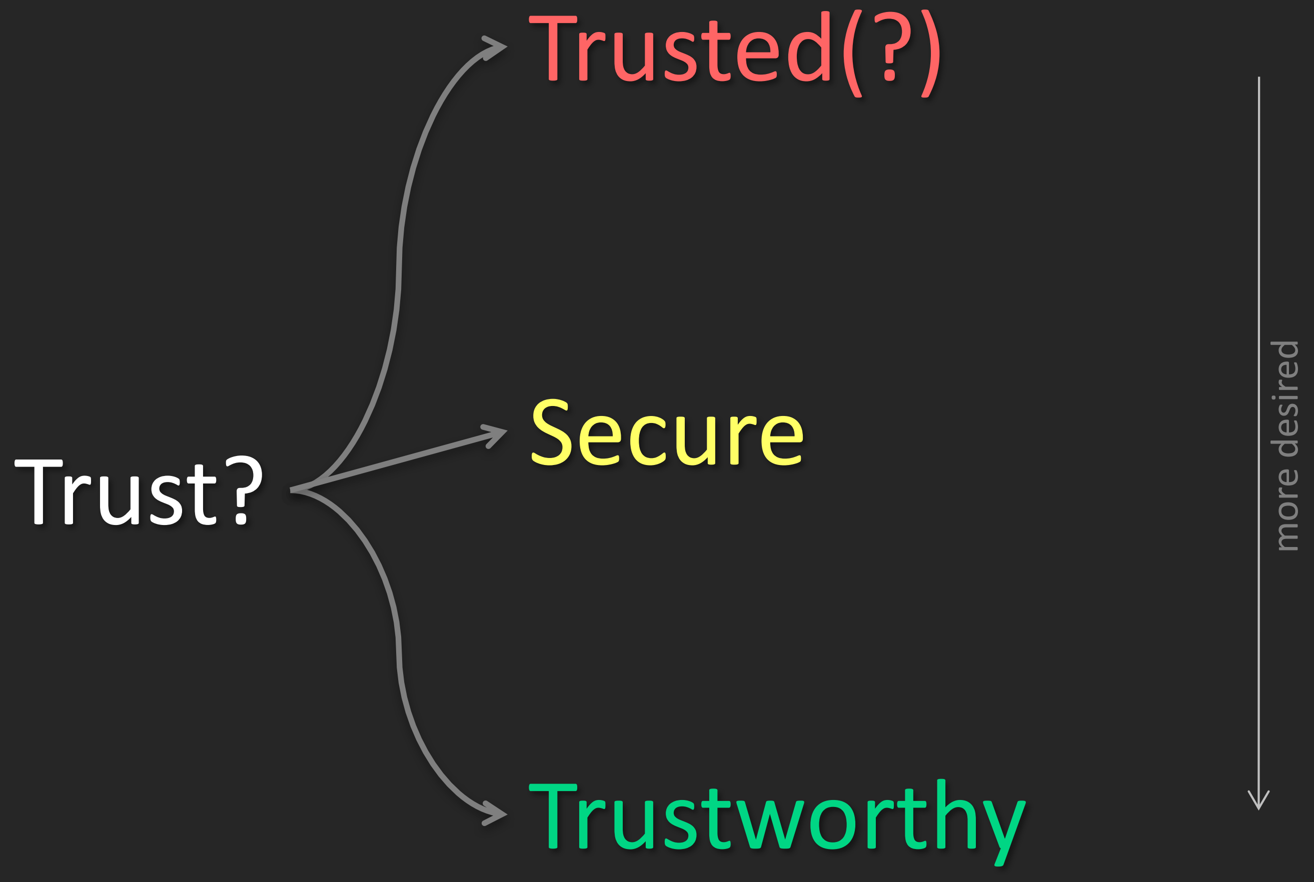
Security through Distrusting

Joanna Rutkowska

Invisible Things Lab & Qubes OS Project

Black Hat EU, London, UK, December 7, 2017

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494



Trust?

Trusted(?)

Secure

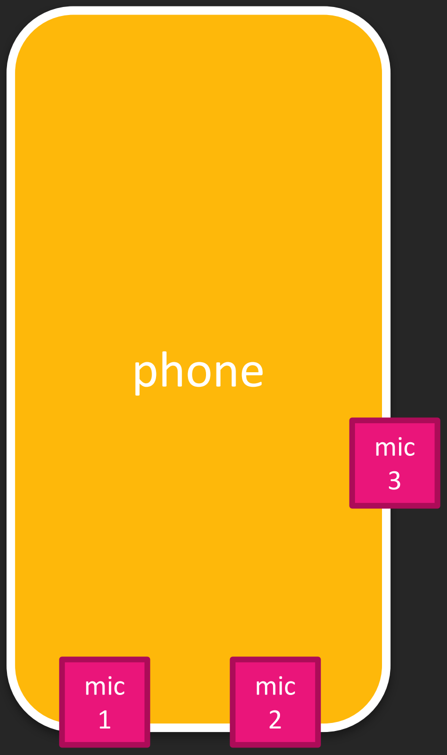
Trustworthy

more desired

Trust consider harmful!

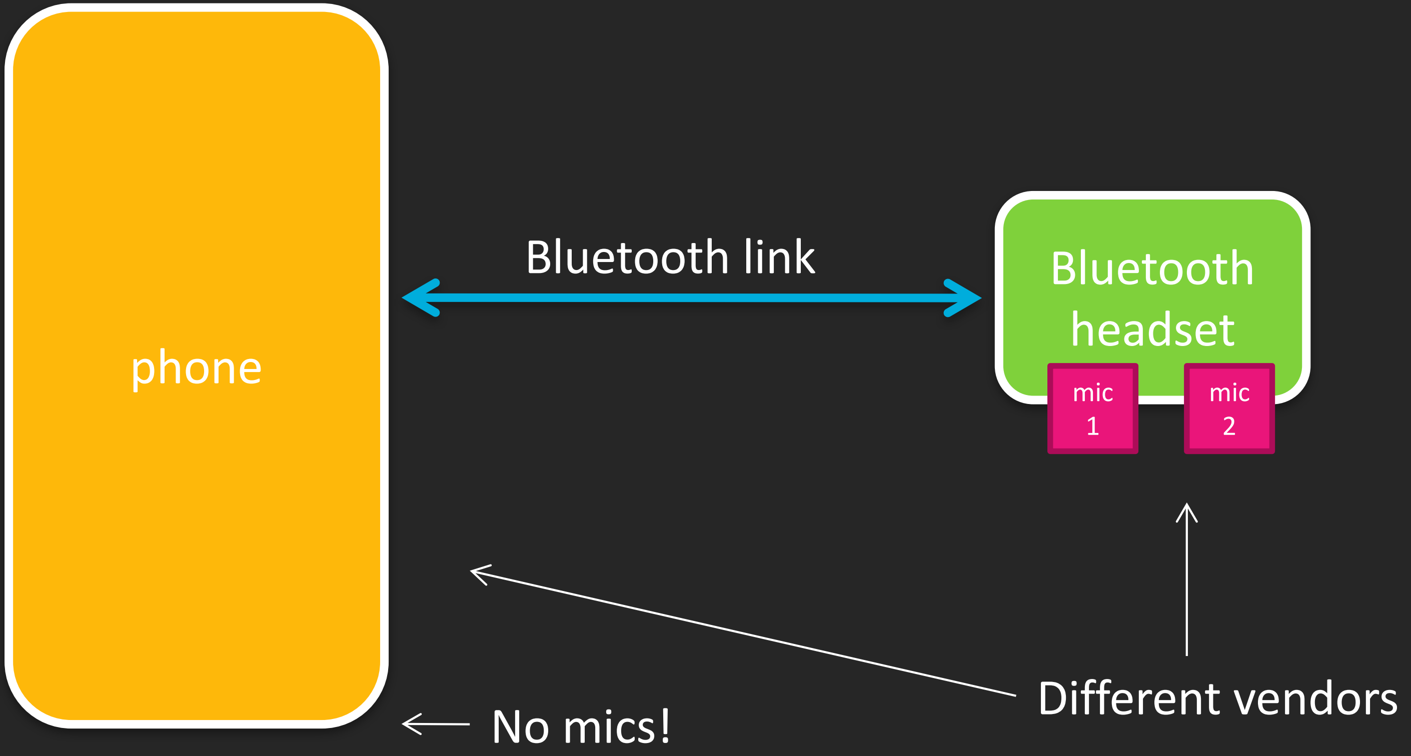
Security through Distrusting examples

Example #1: Pesky microphones



- Mics sniff our activities, including keystrokes, etc.
- Mics are difficult to neutralize
- Mics naturally “cross“ security boundaries

- Mics sniff our activities, including keystrokes, etc.
- Mics are difficult to neutralize
- Mics naturally “cross“ security boundaries



phone

Bluetooth link

Bluetooth headset

mic 1

mic 2

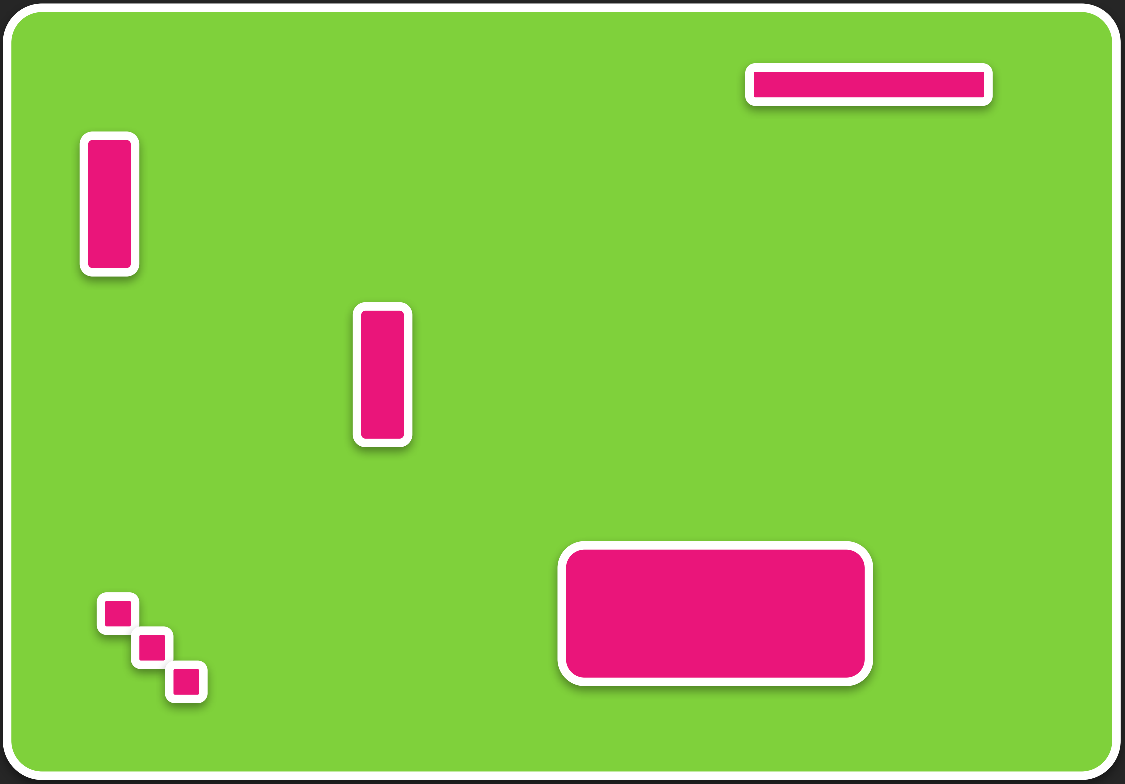
No mics!

Different vendors

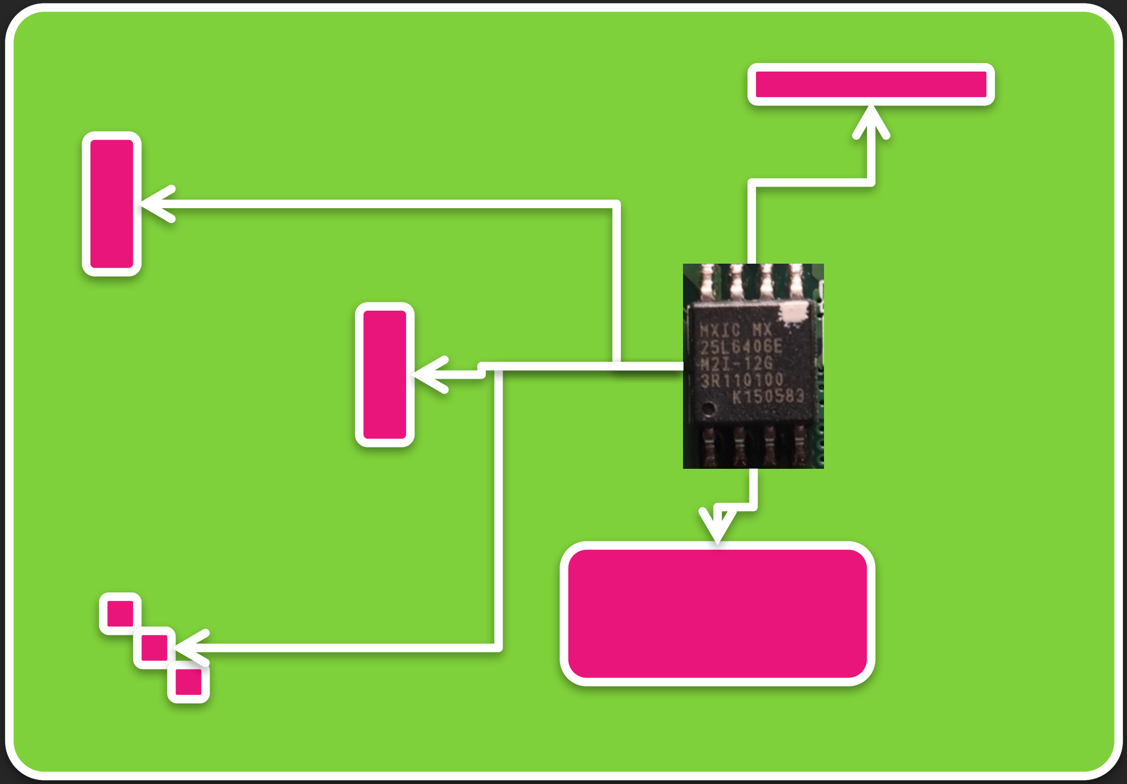
Example #2: Stateless laptop

Persistent laptop compromises…

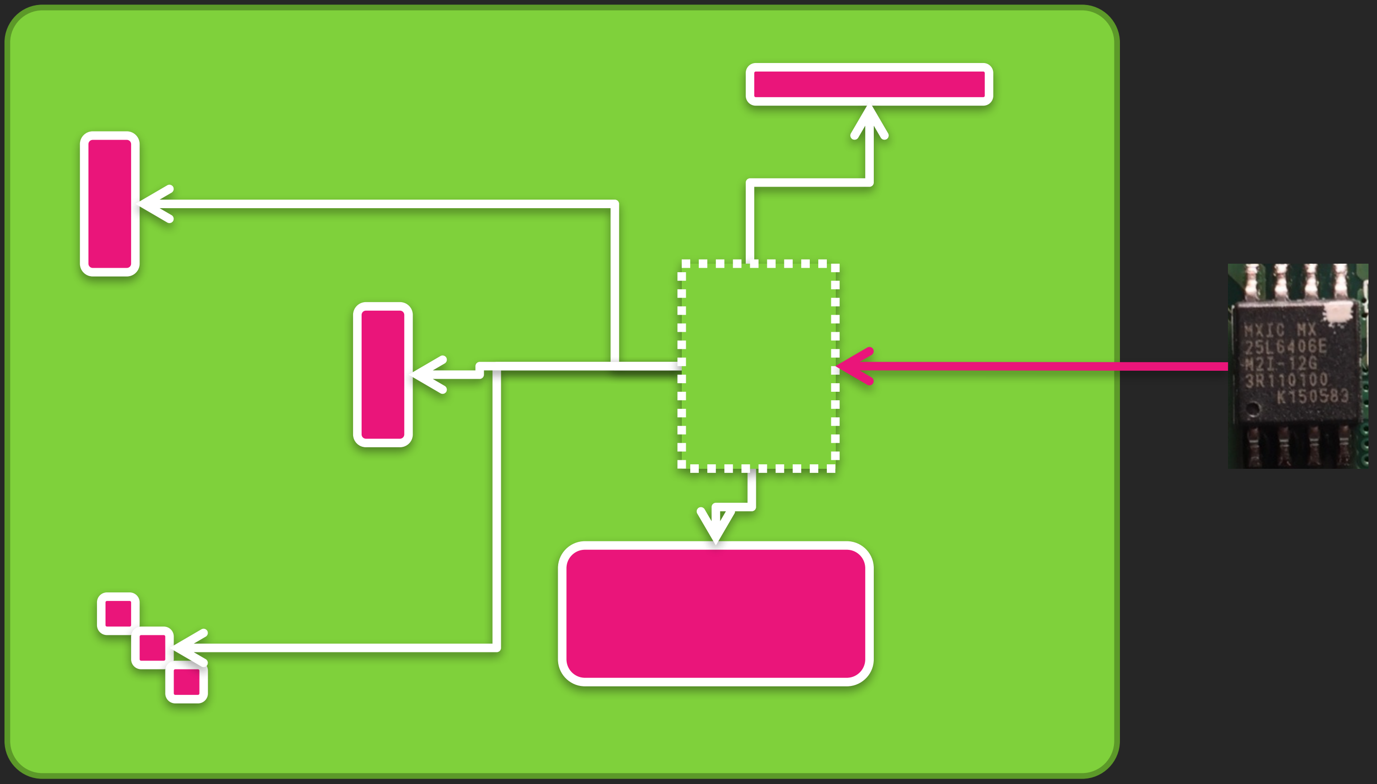
- Persist
- Store secrets
- PII



- Persist
- Store secrets
- PII

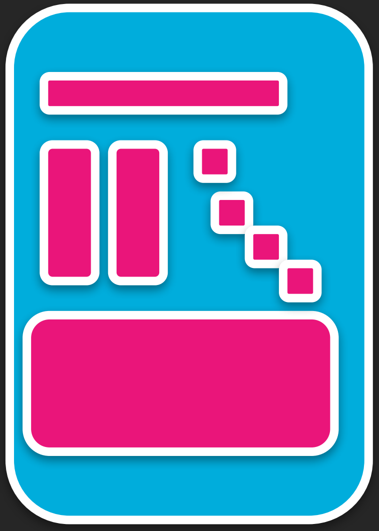


- Persist
- Store secrets
- PII



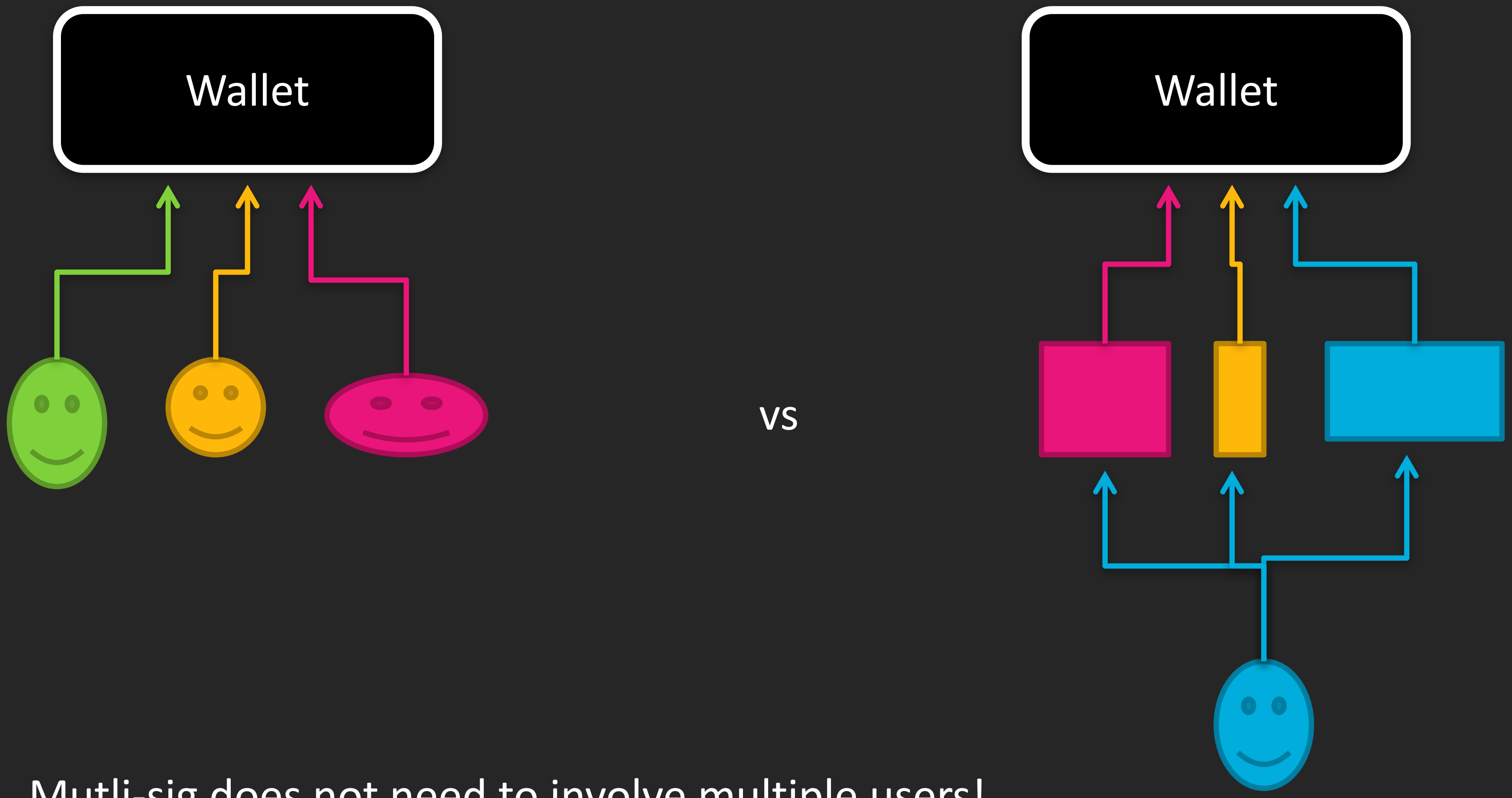
- Firmware infections prevented
- No places to store stolen secrets
- Reliable way to verify firmware
- Reliable way to *choose* firmware
- Boot multiple environments
- Share laptops with others

Stateless Hardware
(persistent state eliminated)


Trusted Stick

Example #3: Multi-party signatures

Photo via Peter Todd (@petertoddbtc)



Wallet

vs

Wallet

Mutli-sig does not need to involve multiple users!

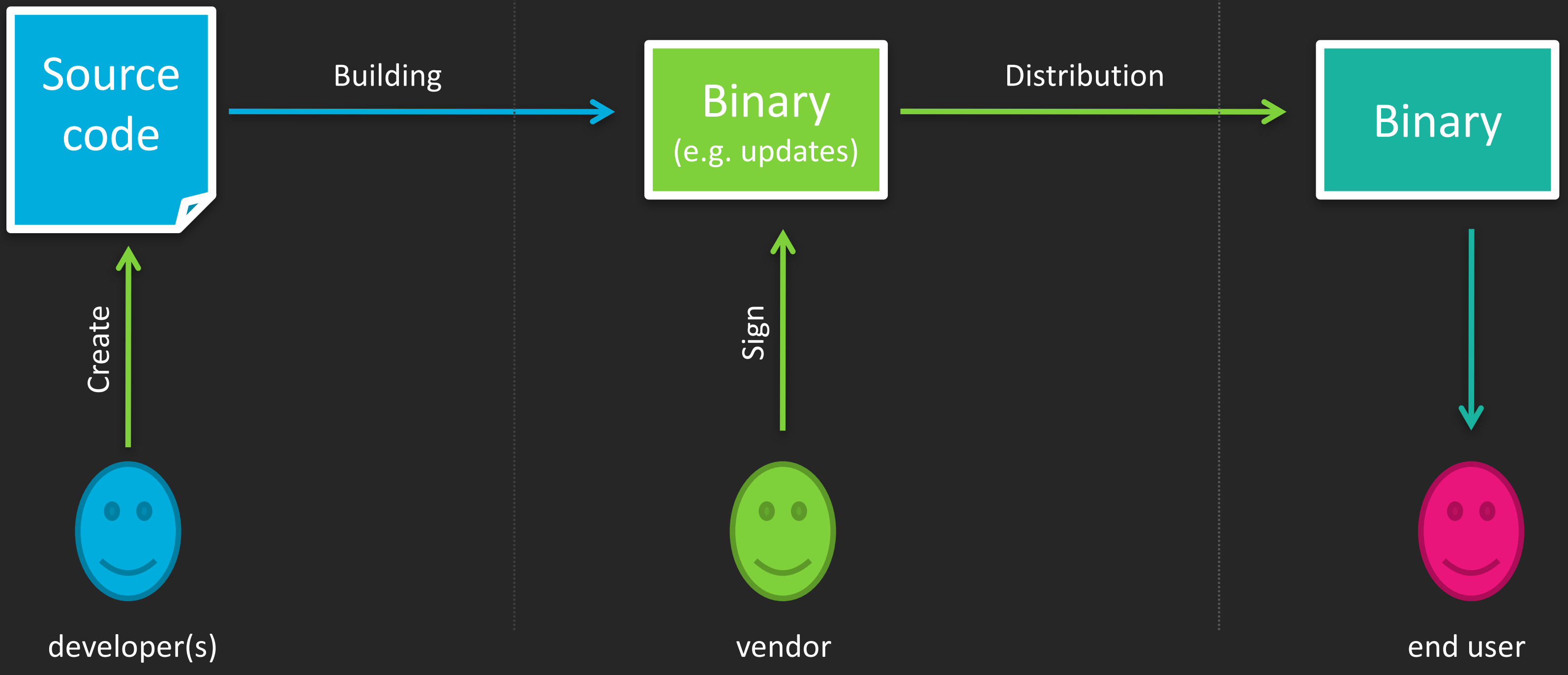
Also: not just Bitcoin wallets…

Example #4: Binary (multi) signing

Why care about binary (multi-) signing?

- OS installation images
- Applications
- Updates
- Firmware

Prime target for backdooring!



Source code

Building

Binary

(e.g. updates)

Distribution

Binary

Create

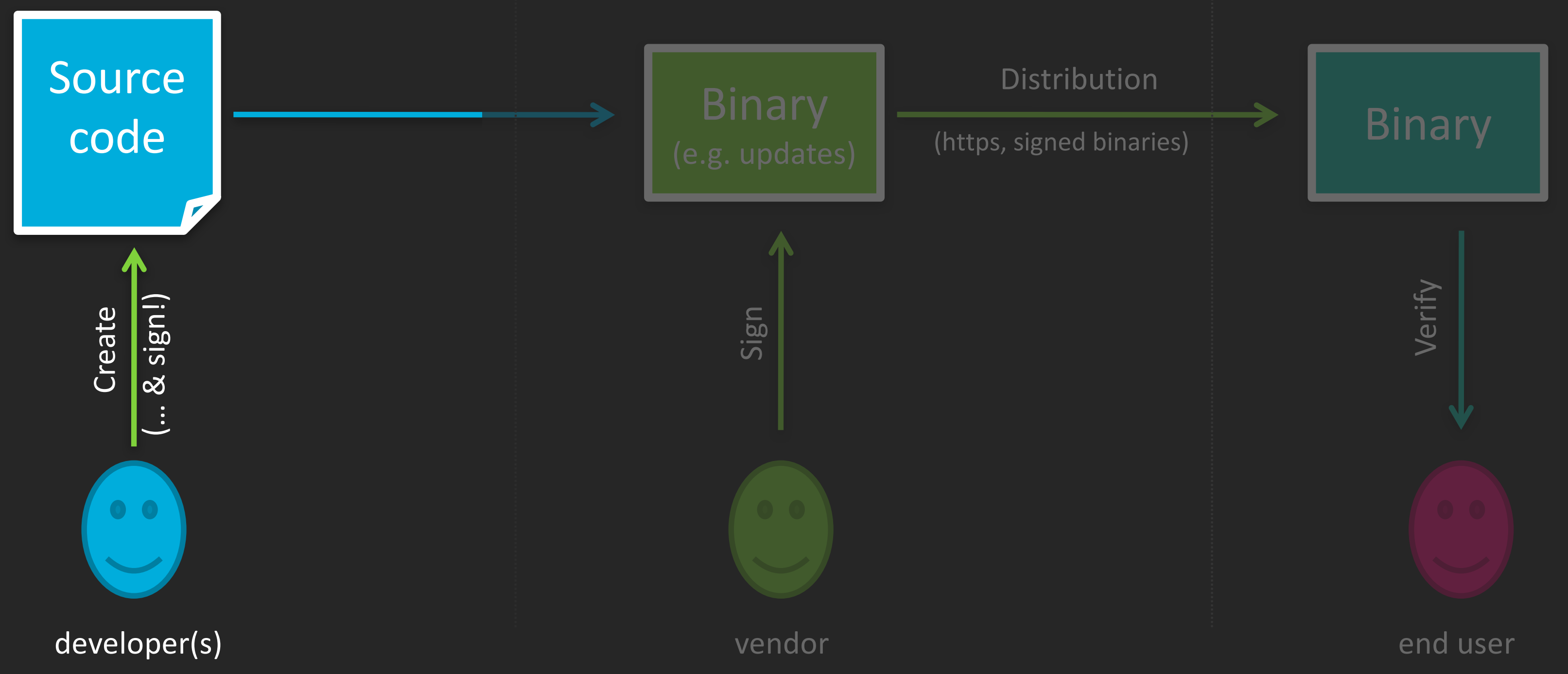
Sign

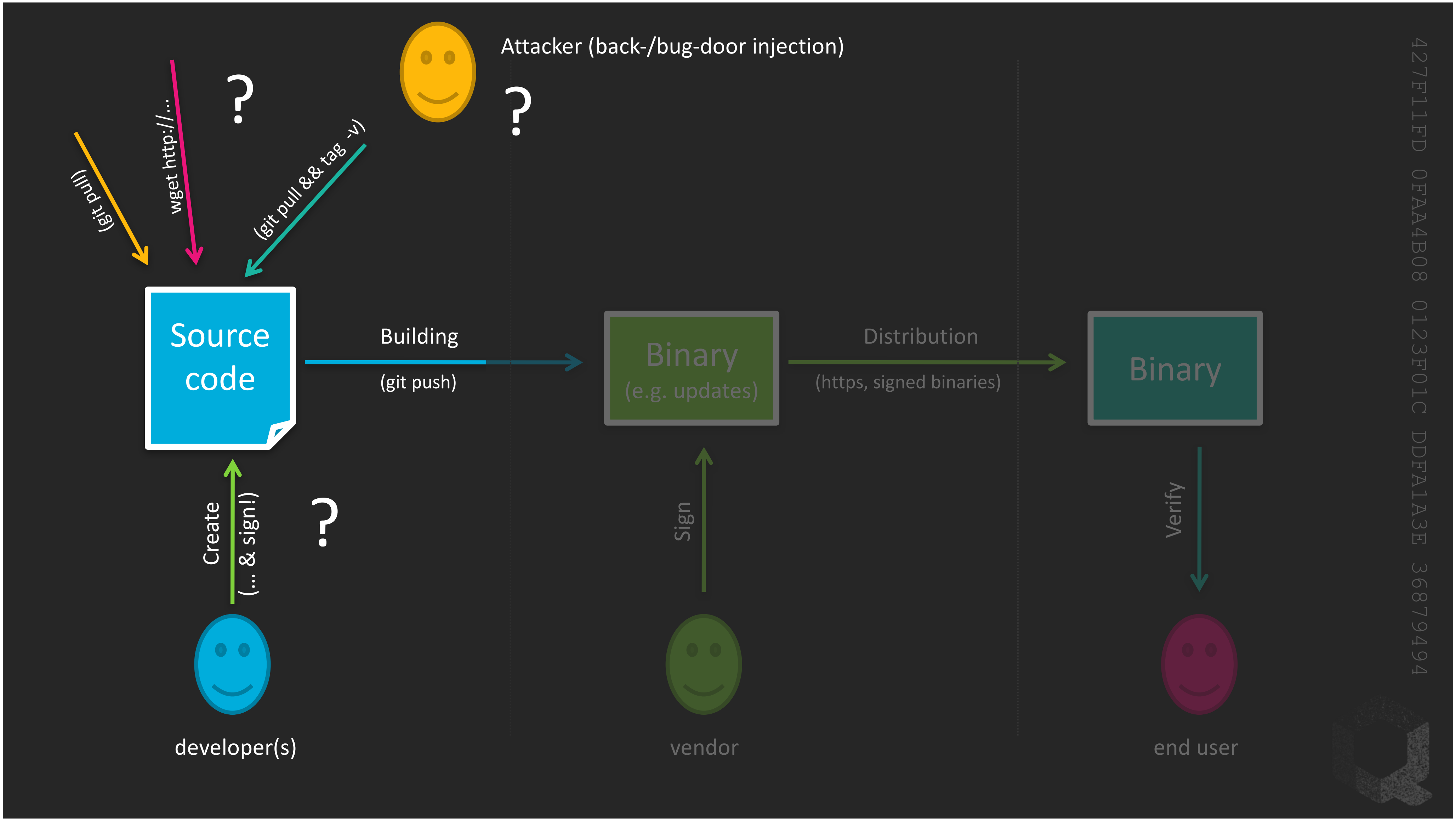
developer(s)

vendor

end user

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494


Source code
Create
(… & sign!)
developer(s)
Binary
(e.g. updates)
Sign
vendor
Distribution
(https, signed binaries)
Binary
Verify
end user



Attacker (back-/bug-door injection)

?

?

?

(git pull)

wget http://...

(git pull && tag -v)

Source code

Building

(git push)

Binary

(e.g. updates)

Distribution

(https, signed binaries)

Binary

Create

(... & sign!)

Sign

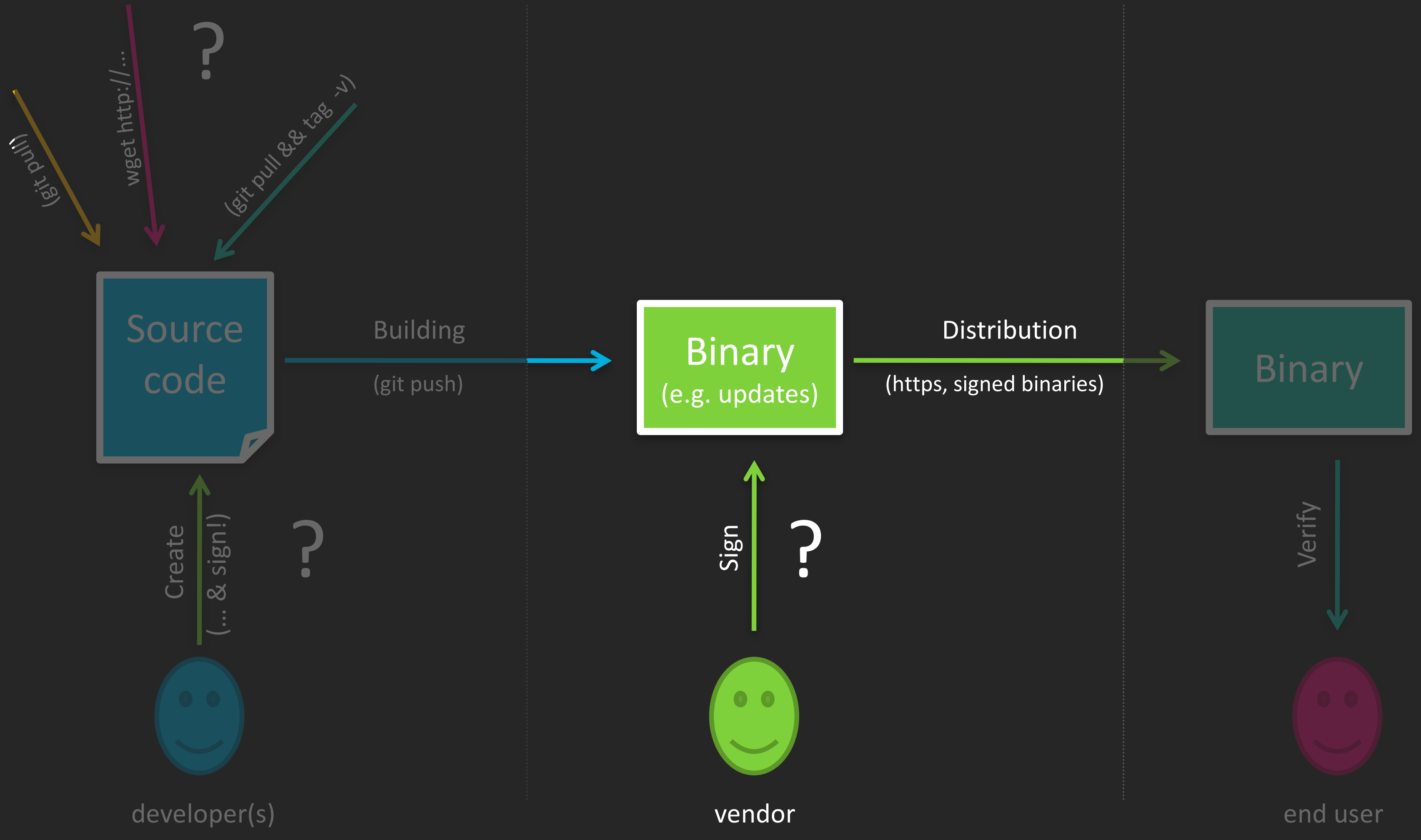
Verify

developer(s)

vendor

end user

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494



(git pull)

wget http://...

?

(git pull && tag -v)

Source code

Building

(git push)

Create

(... & sign!)

?

developer(s)

Binary (e.g. updates)

Distribution

(https, signed binaries)

Sign

?

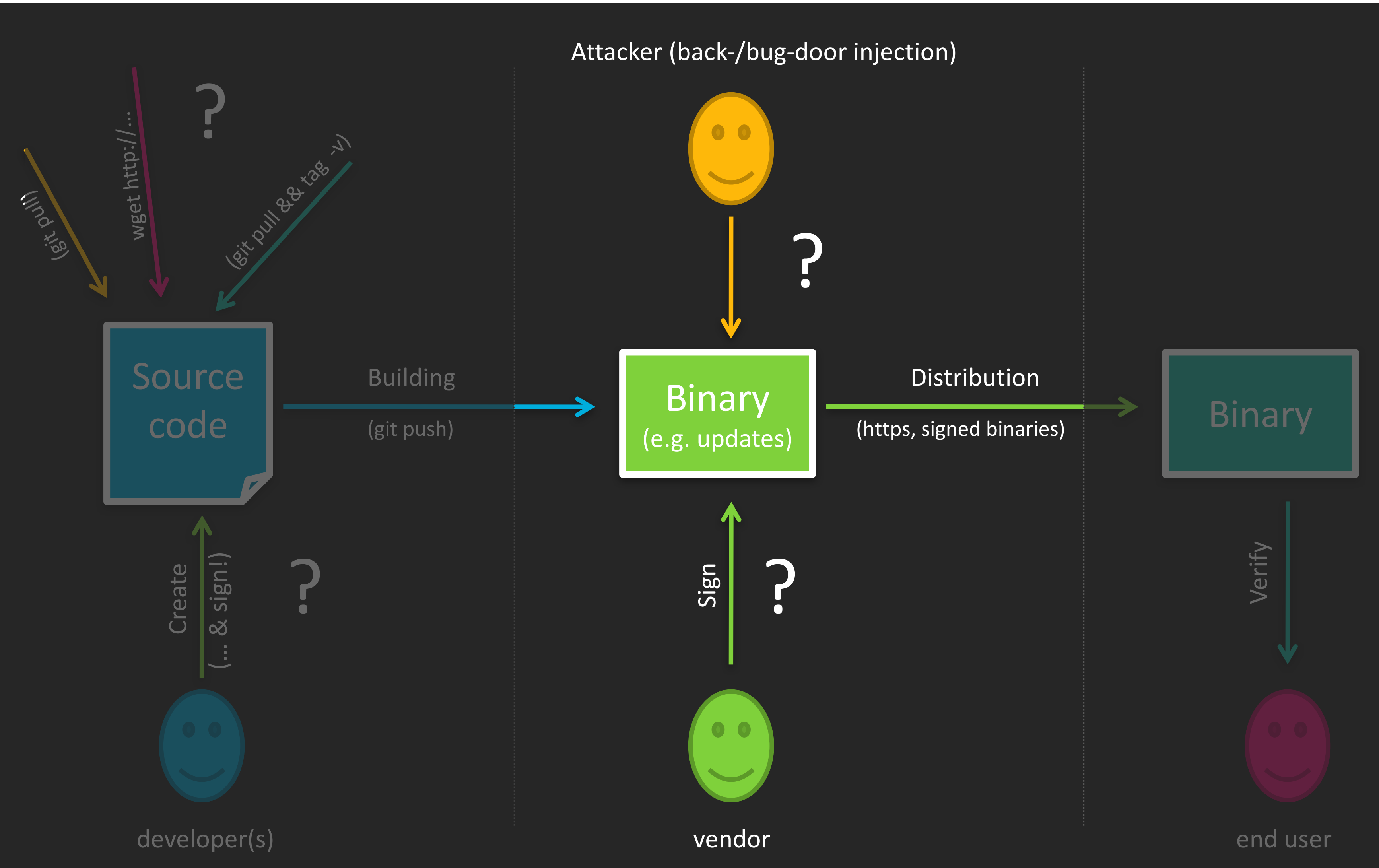
vendor

Binary

Verify

end user

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494



Attacker (back-/bug-door injection)

(git pull)

wget http://...

?

(git pull && tag -v)

Source code

Building

(git push)

Binary (e.g. updates)

?

Distribution

(https, signed binaries)

Binary

Create

(... & sign!)

?

Sign

?

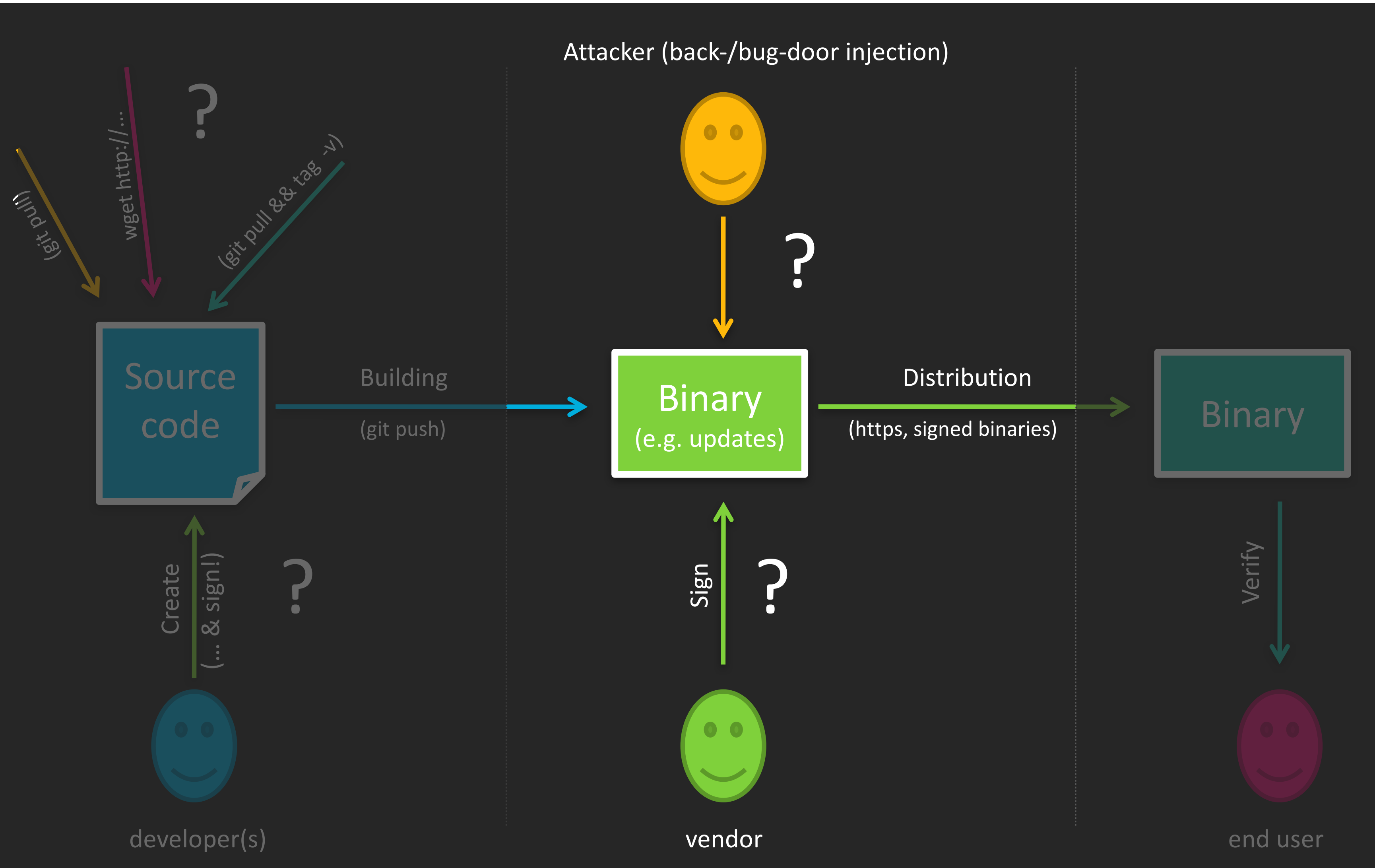
Verify

developer(s)

vendor

end user

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494



Attacker (back-/bug-door injection)

(git pull)

wget http://…

(git pull && tag -v)

?

Source code

Building

(git push)

Binary (e.g. updates)

Distribution

(https, signed binaries)

Binary

Create

(… & sign!)

?

Sign

?

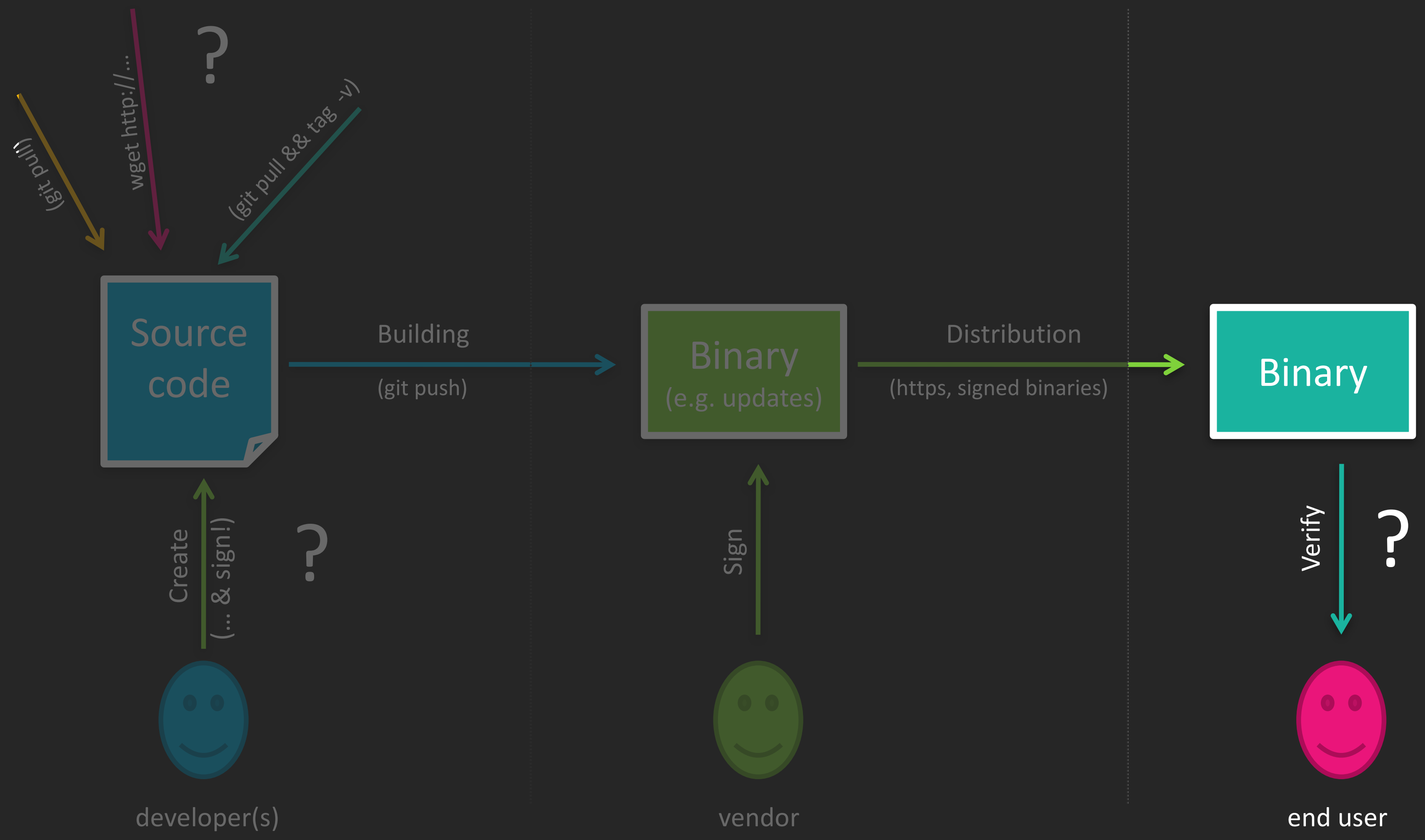
?

Verify

developer(s)

vendor

end user



(git pull)

wget http://...

?

(git pull && tag -v)

Source code

Building

(git push)

Binary

(e.g. updates)

Distribution

(https, signed binaries)

Binary

Create

(... & sign!)

?

Sign

Verify

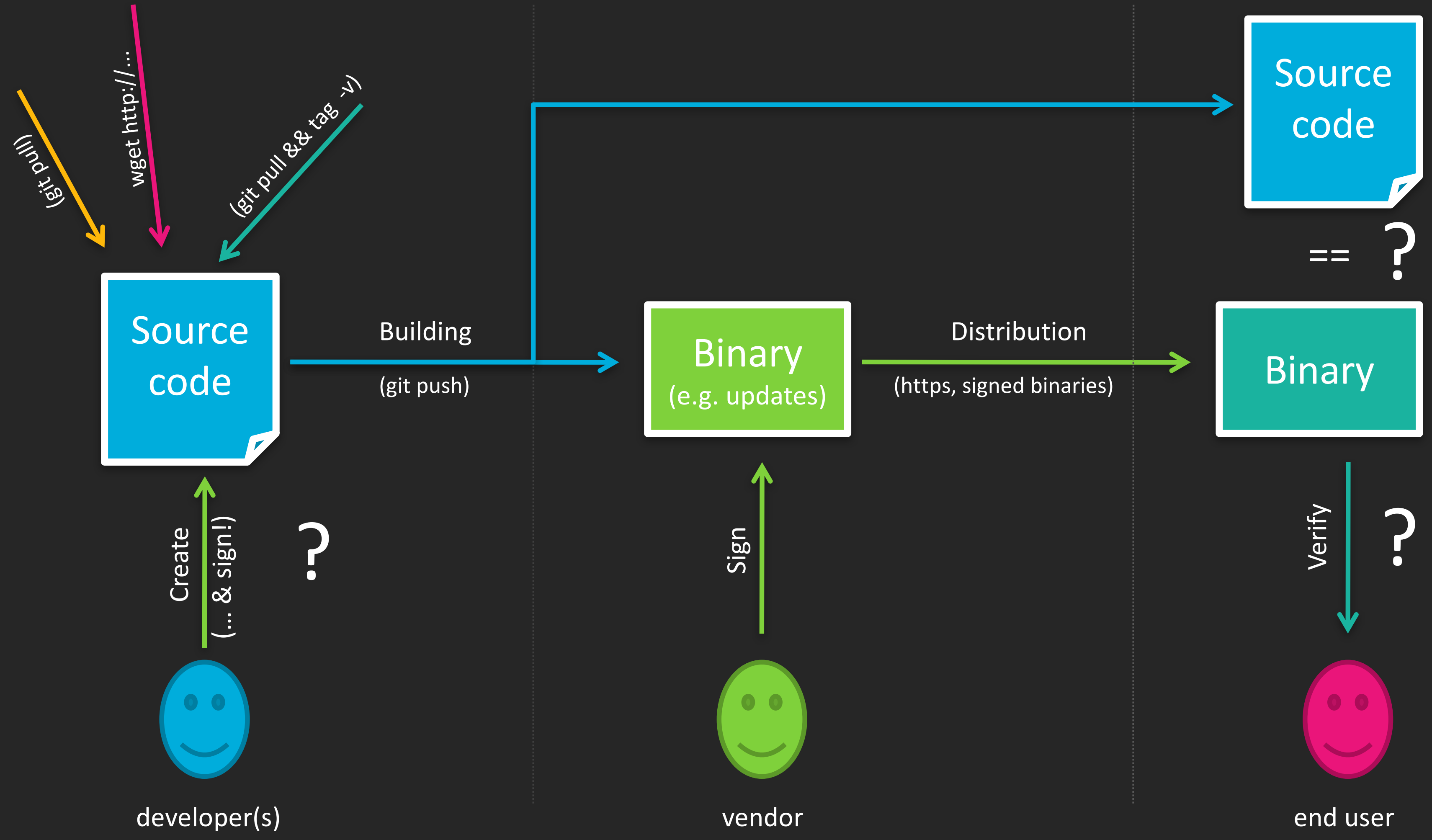
?

developer(s)

vendor

end user

427E11FD 0FAA4B08 0123F01C DDFA1A3E 36879494



(git pull)

wget http://...

(git pull && tag -v)

Source code

Create

(... & sign!)

?

developer(s)

Building

(git push)

Binary (e.g. updates)

Sign

vendor

Distribution

(https, signed binaries)

Source code

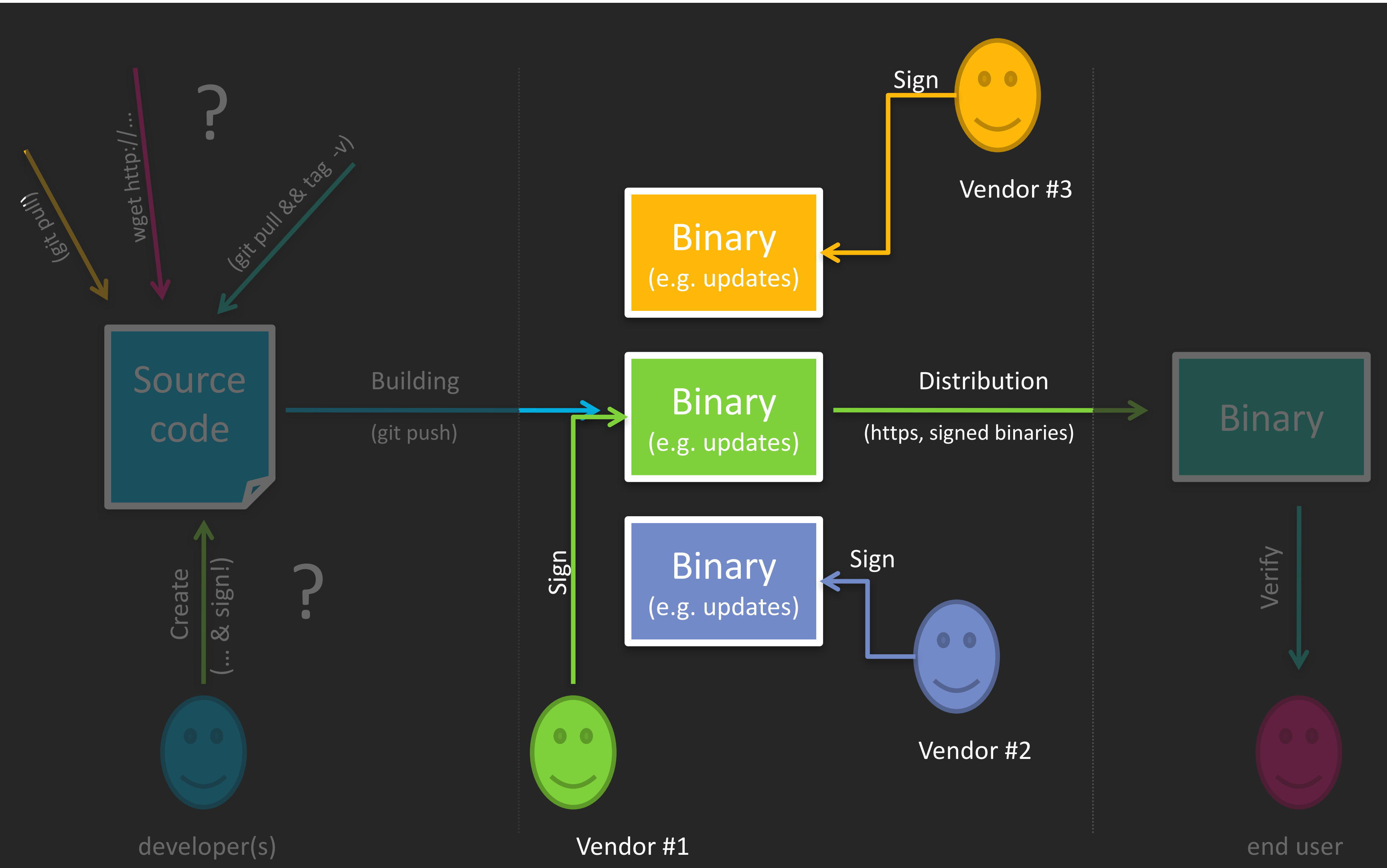
== ?

Binary

Verify

?

end user



(git pull)

wget http://...

?

(git pull && tag -v)

Source code

Create

(... & sign!)

?

developer(s)

Building

(git push)

Binary
(e.g. updates)

Sign

Vendor #3

Binary
(e.g. updates)

Sign

Binary
(e.g. updates)

Sign

Vendor #2

Vendor #1

Distribution

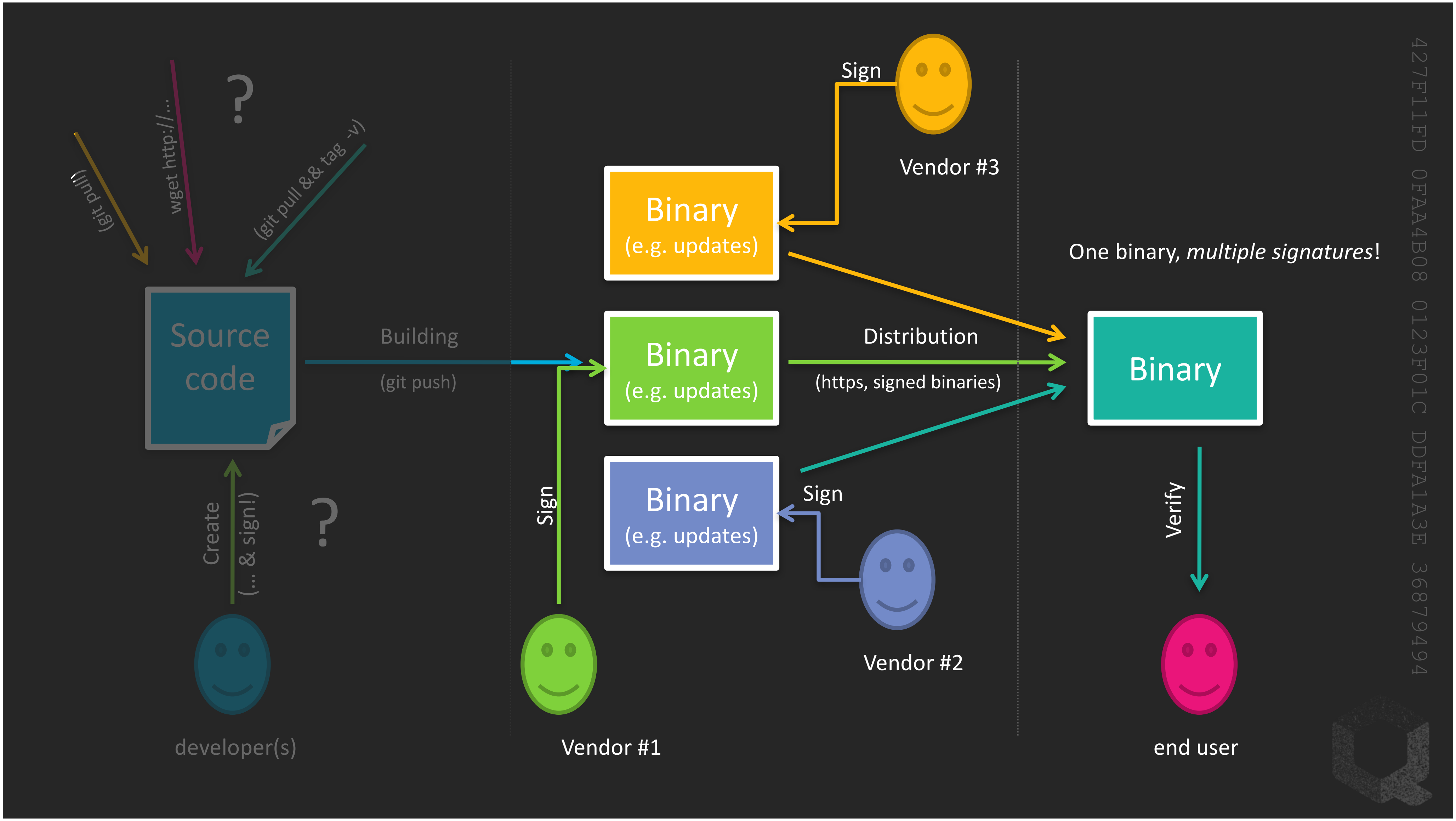
(https, signed binaries)

Binary

Verify

end user

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494



One binary, *multiple signatures*!

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494

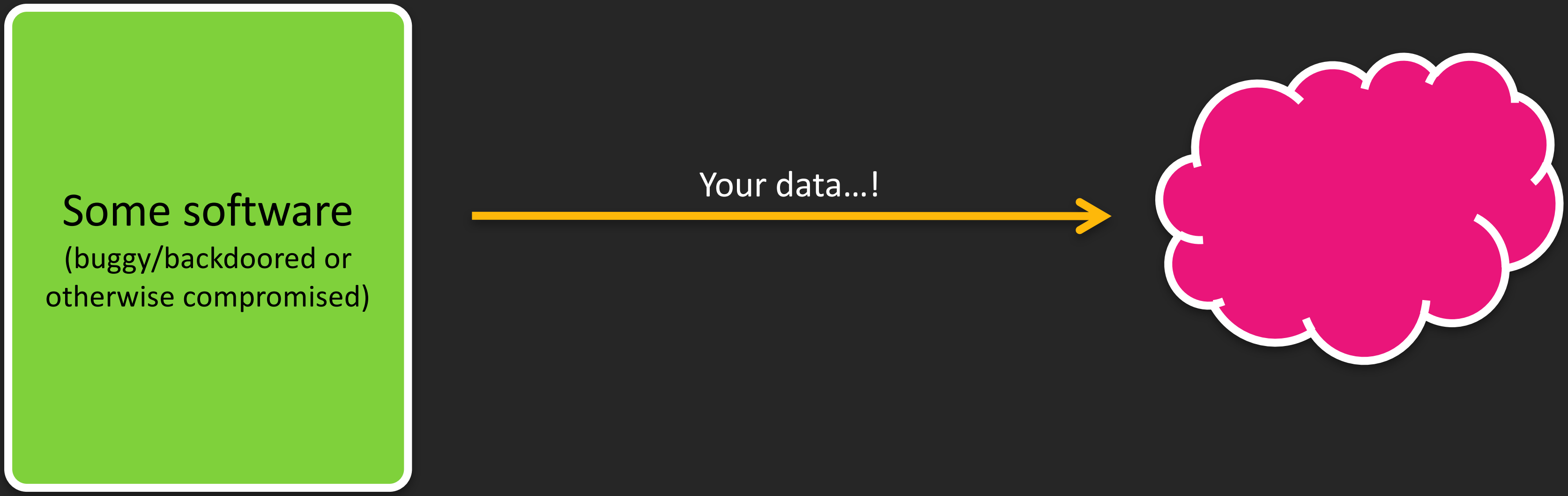
Multi-signed binaries

- Signed by people from different countries
- Different organizations (vendor & auditing)
- Signed by different *machines*
  - In the same organization
  - In different organization

https://reproducible-builds.org

427E11FD 0FAA4B08 0123F01C DDFA1A3E 36879494

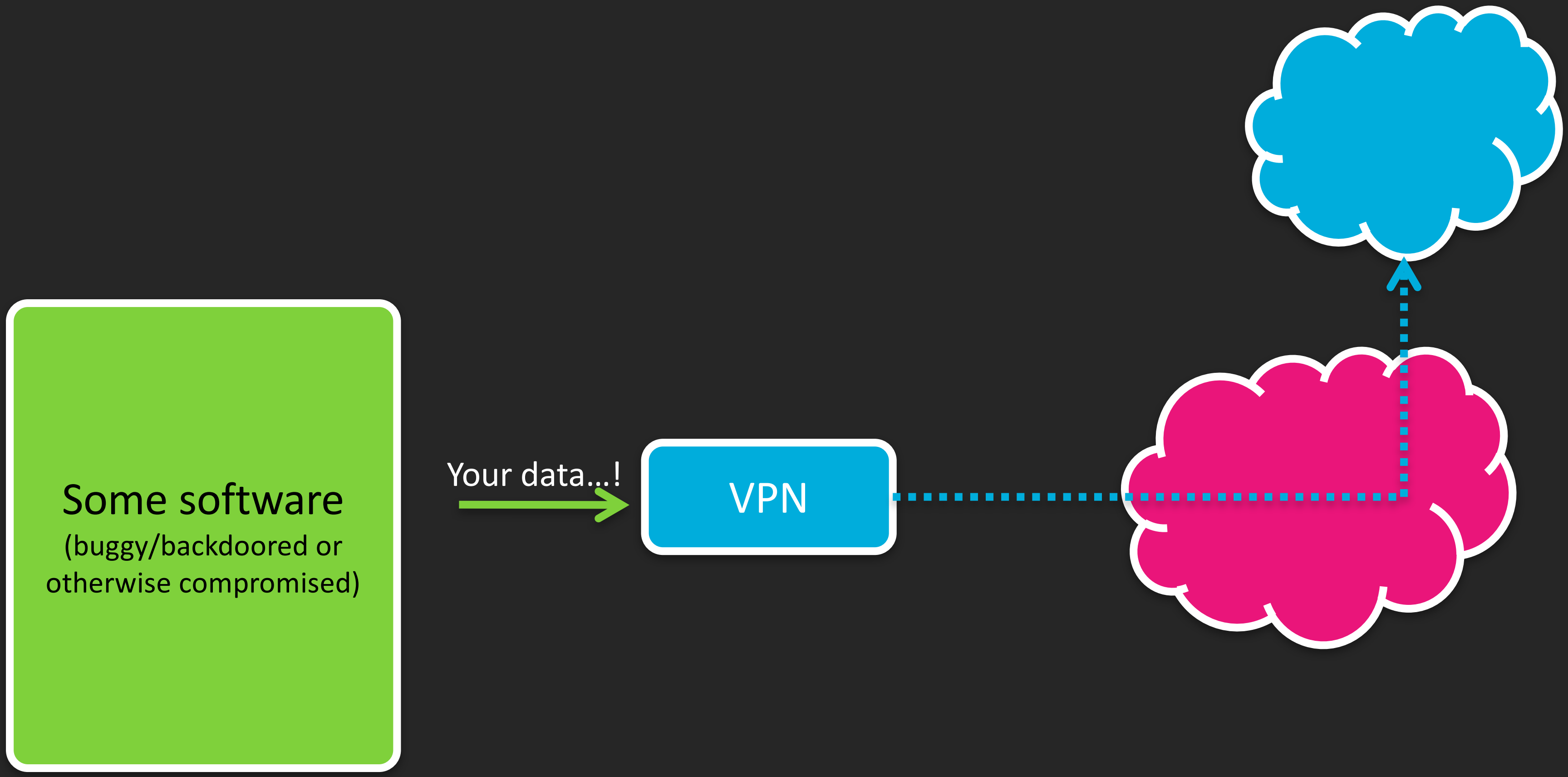
Example 5: Preventing data leaks



Some software

(buggy/backdoored or otherwise compromised)

Your data...!

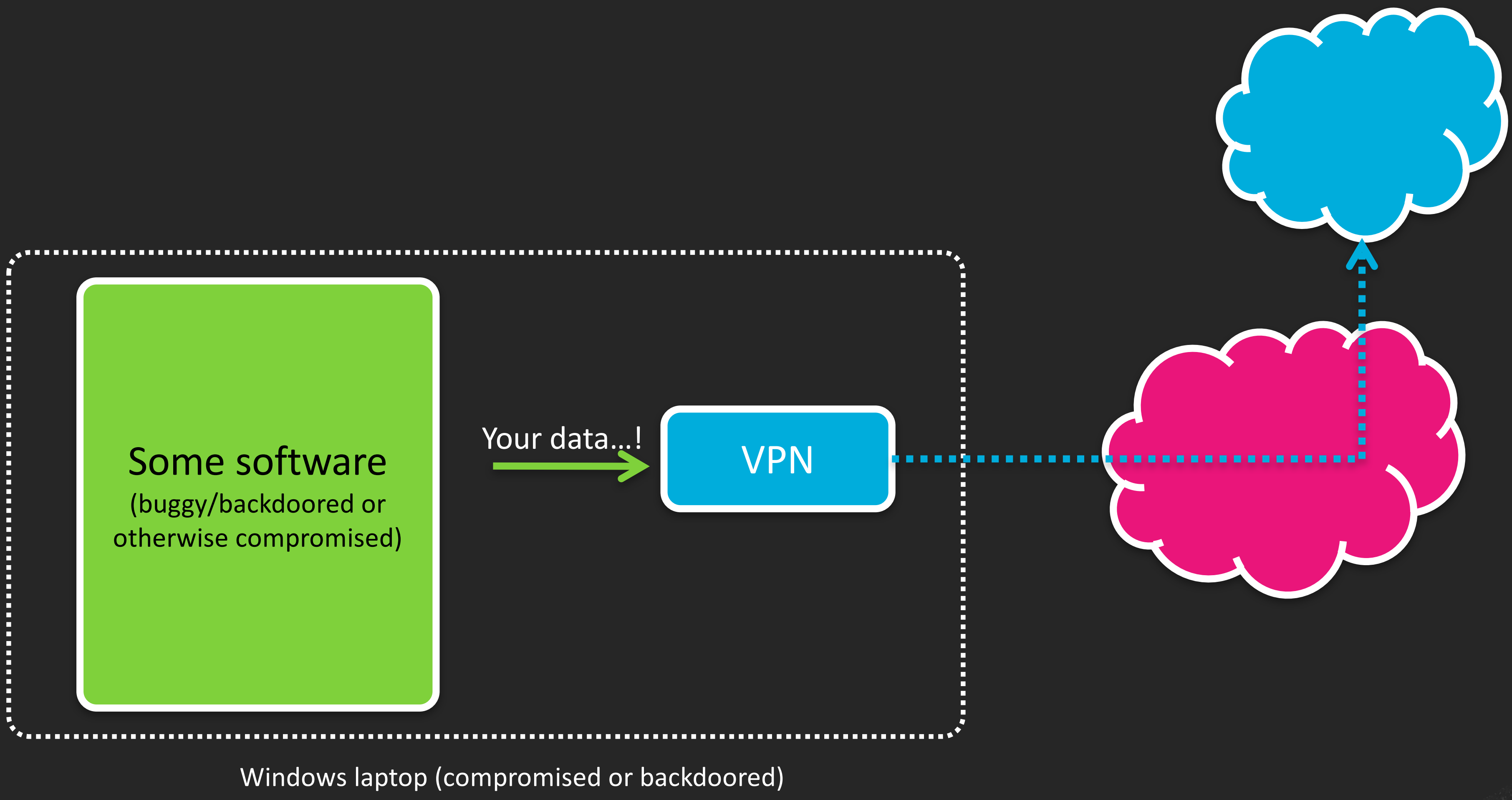


Some software

(buggy/backdoored or otherwise compromised)

Your data…!

VPN



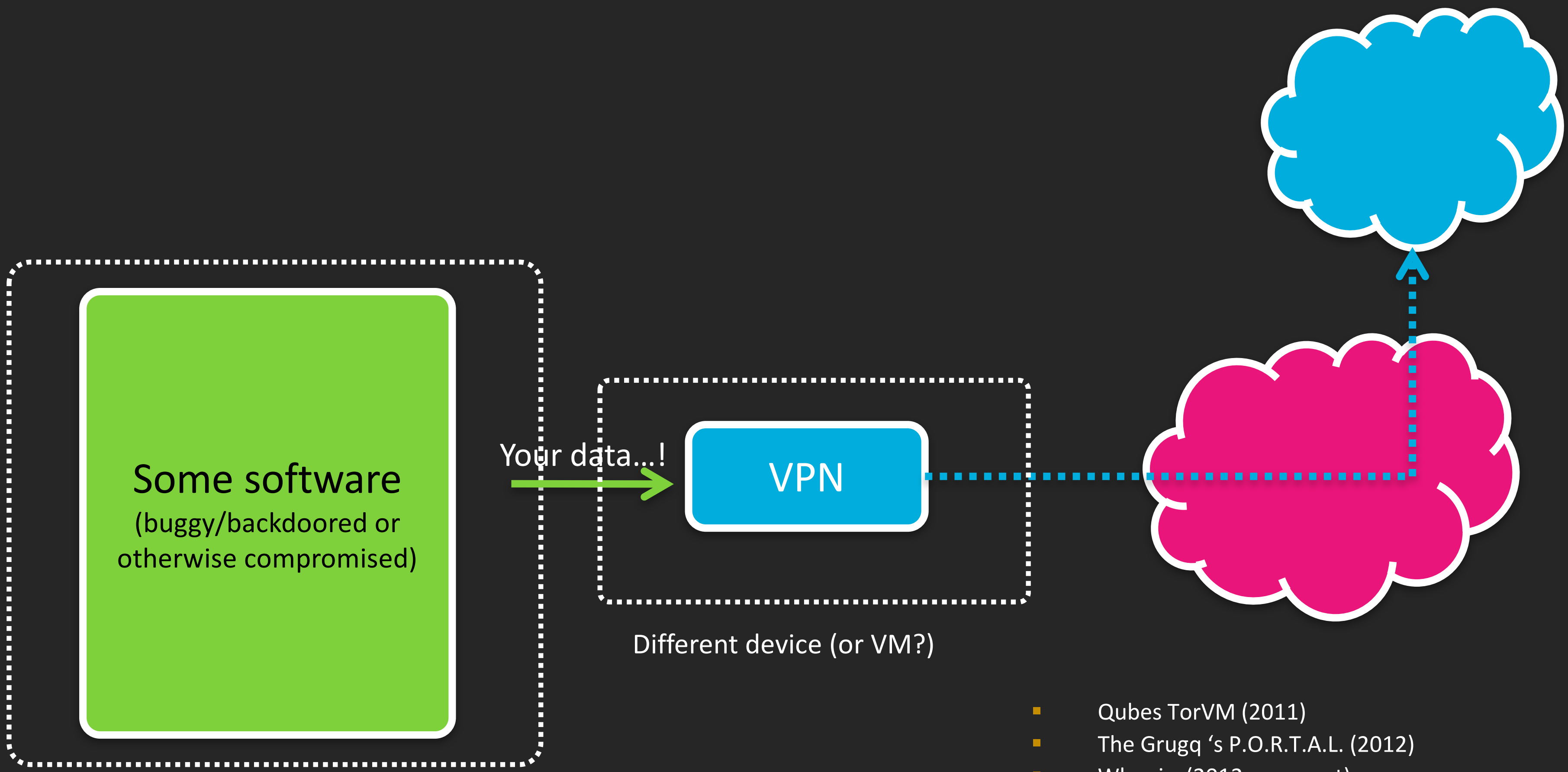
Some software

(buggy/backdoored or otherwise compromised)

Your data...!

VPN

Windows laptop (compromised or backdoored)



Some software

(buggy/backdoored or otherwise compromised)

Your data…!

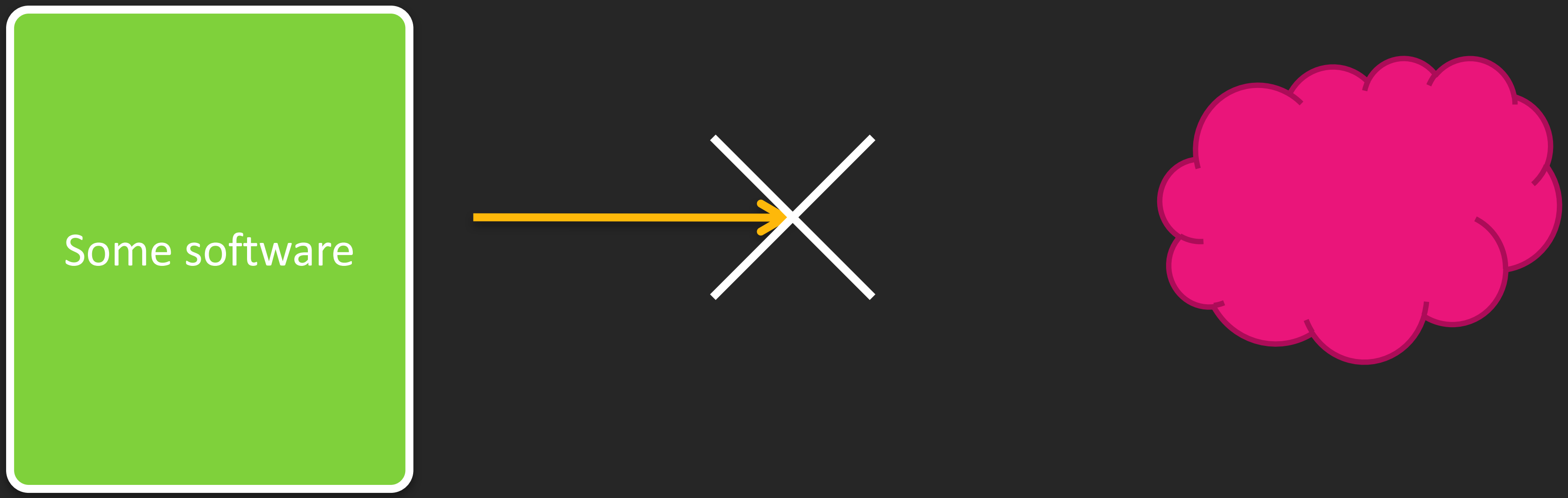
VPN

Different device (or VM?)

Windows laptop (compromised or backdoored)

- Qubes TorVM (2011)
- The Grugq ‘s P.O.R.T.A.L. (2012)
- Whonix (2012 – present)
- Whonix for Qubes (2014 – present)
- Tor-enabled routers (multiple projects/products)

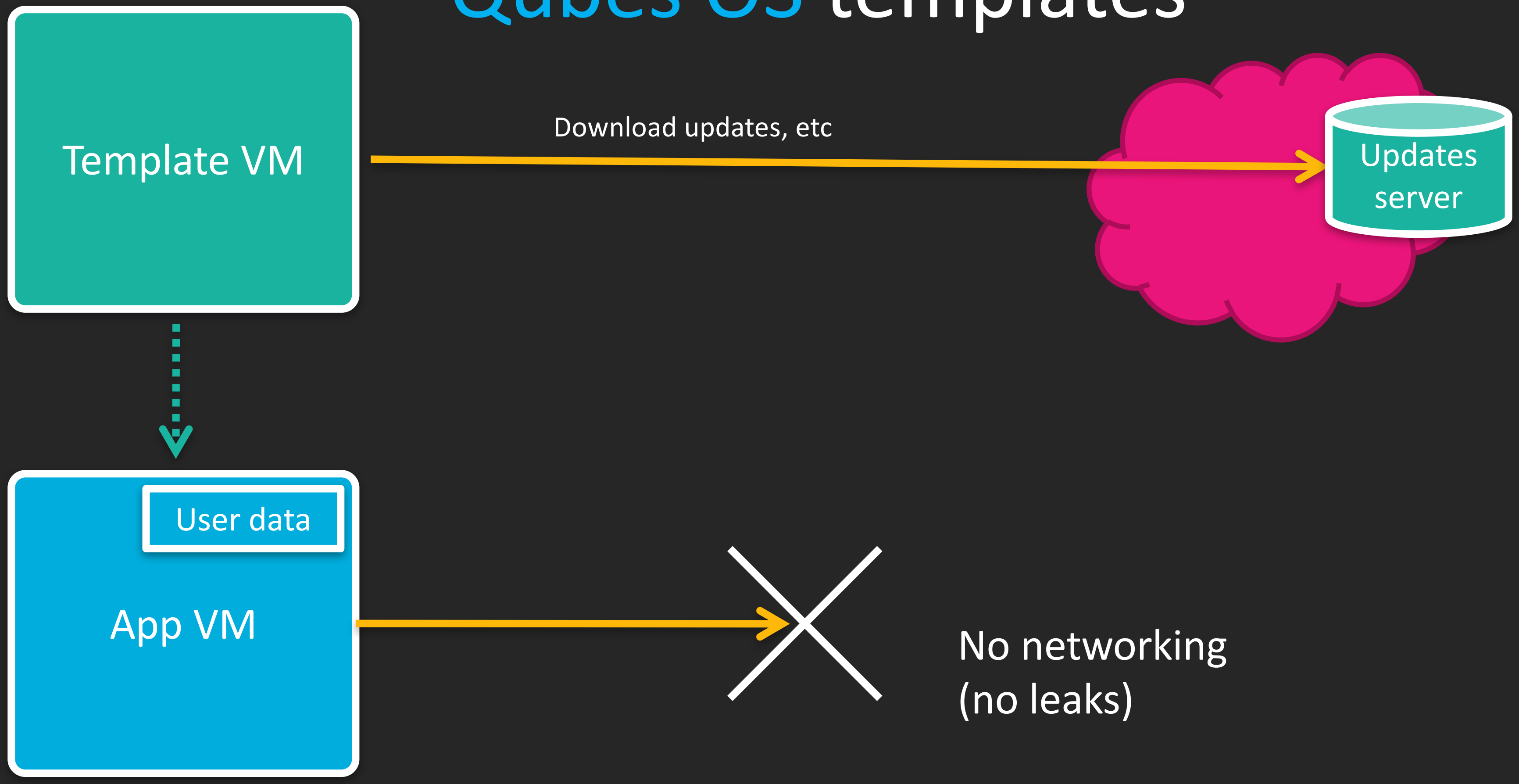
Cut off networking?



Some software

Not very useful…

Qubes OS templates



No networking
(no leaks)

Example #6: Compartmentalization

“Classic” compartmentalization…

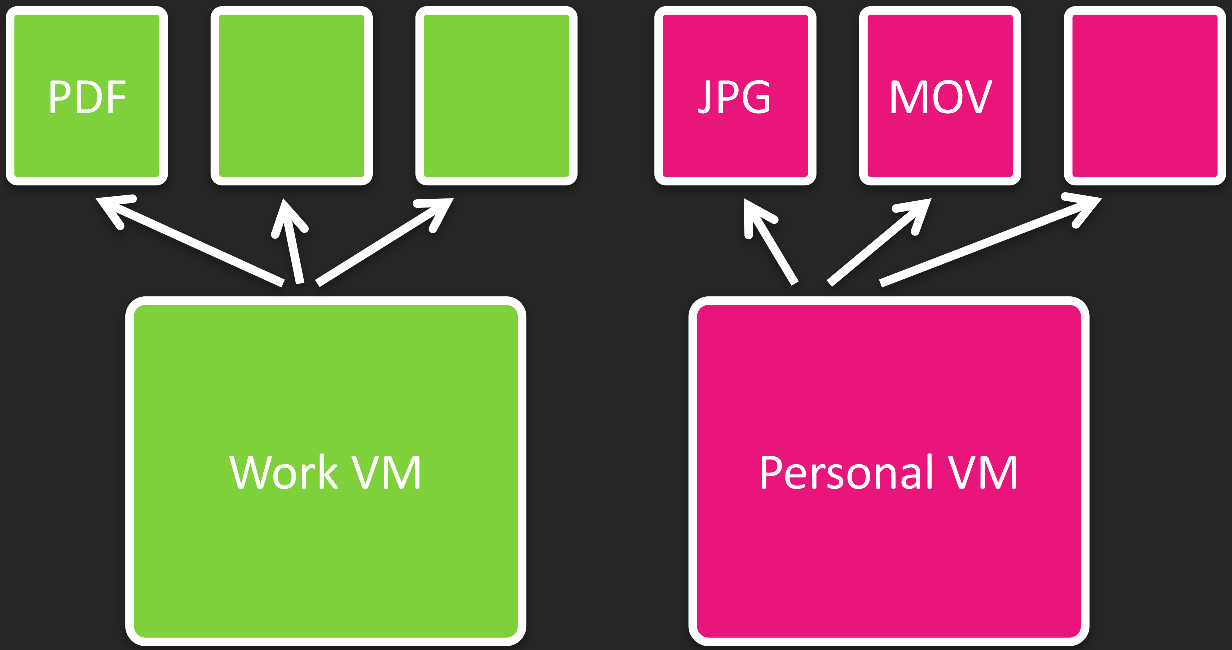
Work VM
Personal VM

“Classic” compartmentalization…

Work VM

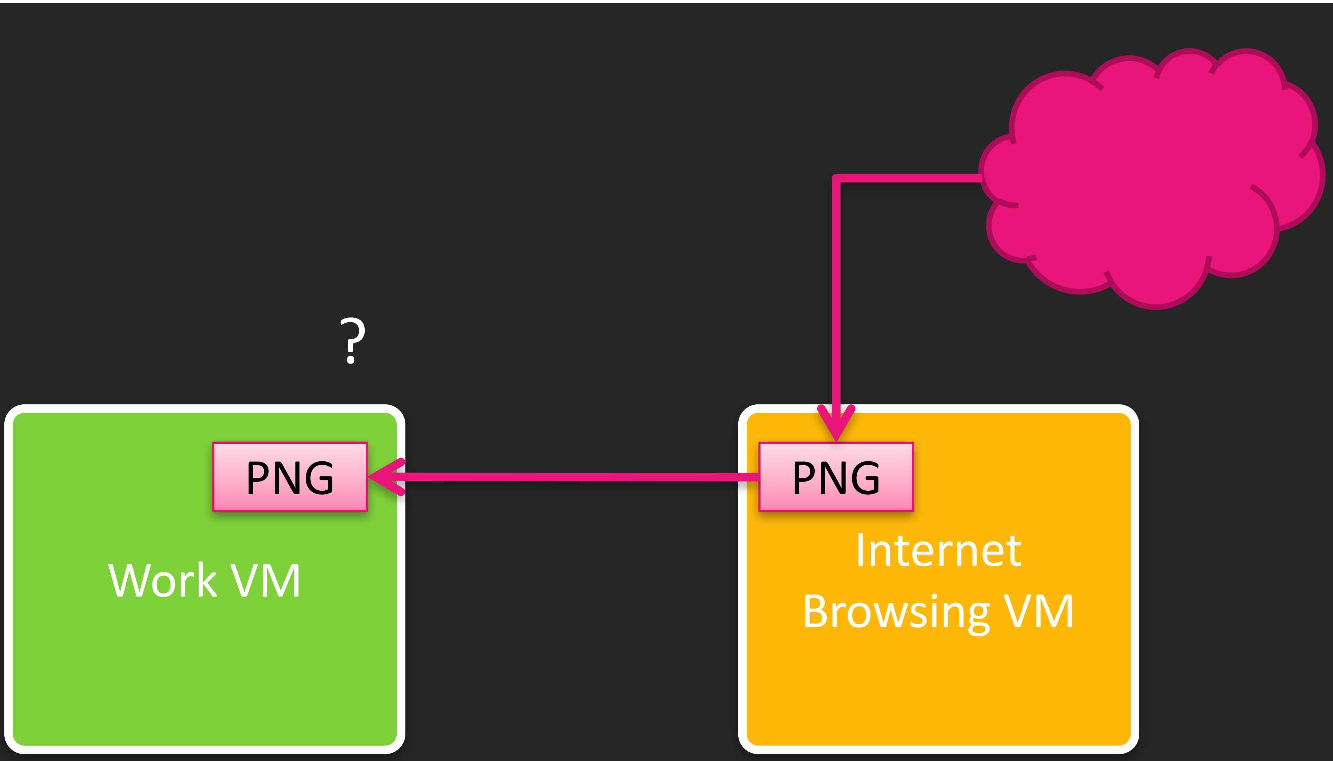
Personal VM

…not very useful!

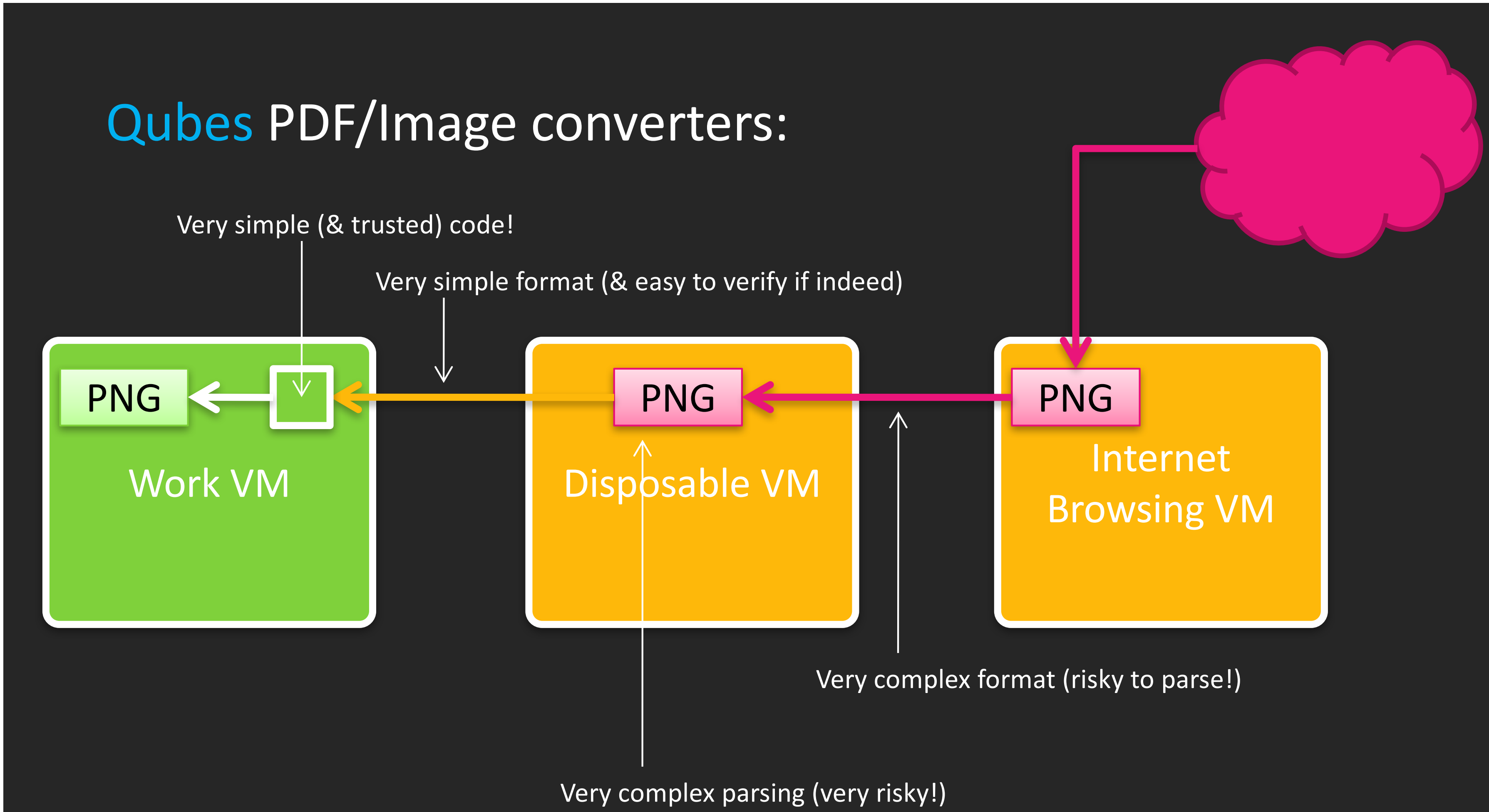


…more useful…

Inter-compartments data transfers


?
PNG
Work VM
PNG
Internet Browsing VM

Qubes PDF/Image converters:



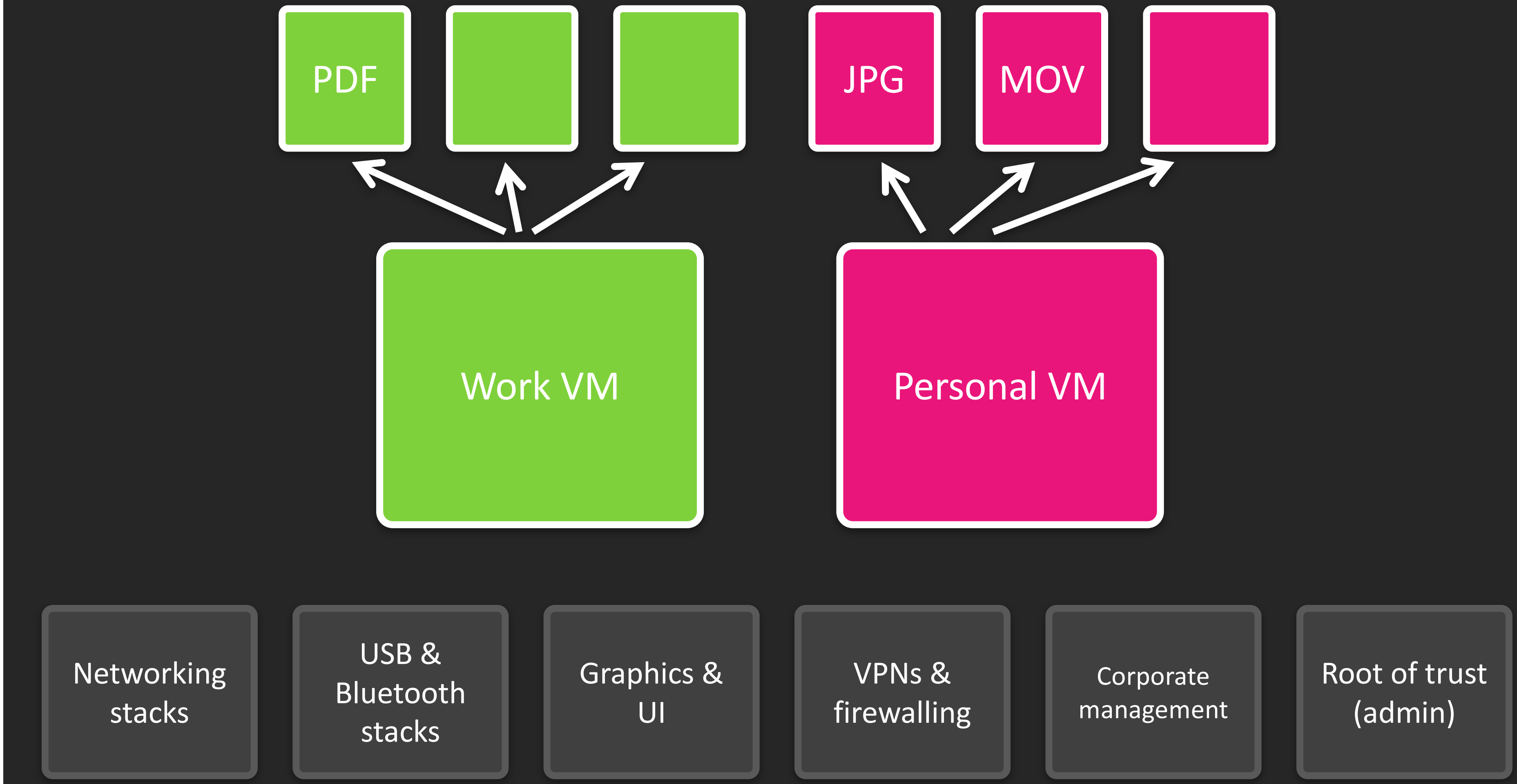
Very simple (& trusted) code!

Very simple format (& easy to verify if indeed)

Very complex format (risky to parse!)

Very complex parsing (very risky!)

App sandboxing is just part of the story…



PDF

JPG

MOV

Work VM

Personal VM

Networking stacks

USB & Bluetooth stacks

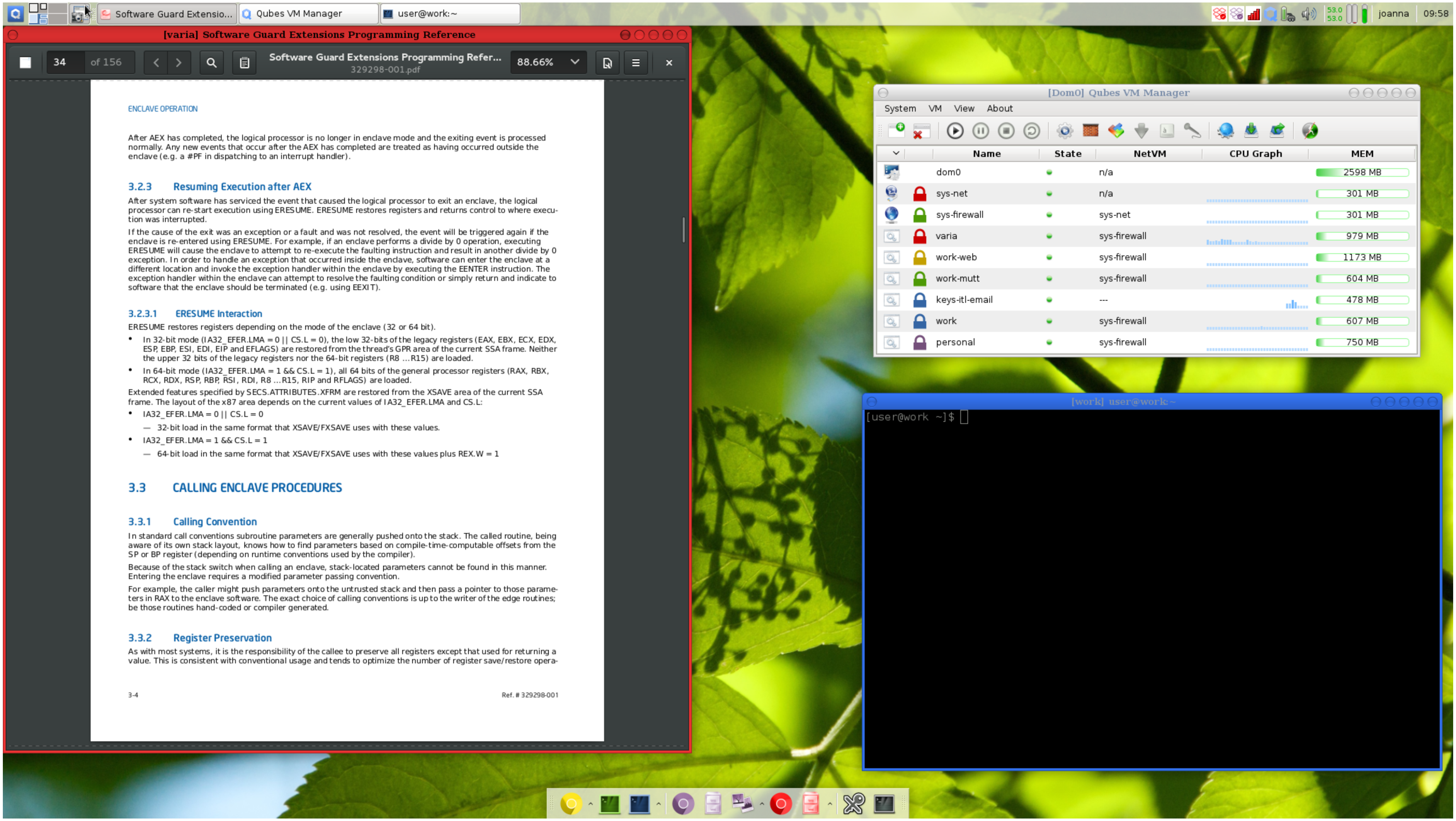
Graphics & UI

VPNs & firewalling

Corporate management

Root of trust (admin)

Isolation is just part of the story!


Software Guard Extensio...
Qubes VM Manager
user@work:~
joanna
09:58
[varia] Software Guard Extensions Programming Reference
34
of 156
Software Guard Extensions Programming Refer...
329298-001.pdf
88.66%
[Dom0] Qubes VM Manager
System
VM
View
About
[work] user@work:~

ENCLAVE OPERATION

After AEX has completed, the logical processor is no longer in enclave mode and the exiting event is processed normally. Any new events that occur after the AEX has completed are treated as having occurred outside the enclave (e.g. a #PF in dispatching to an interrupt handler).

3.2.3 Resuming Execution after AEX

After system software has serviced the event that caused the logical processor to exit an enclave, the logical processor can re-start execution using ERESUME. ERESUME restores registers and returns control to where execution was interrupted.

If the cause of the exit was an exception or a fault and was not resolved, the event will be triggered again if the enclave is re-entered using ERESUME. For example, if an enclave performs a divide by 0 operation, executing ERESUME will cause the enclave to attempt to re-execute the faulting instruction and result in another divide by 0 exception. In order to handle an exception that occurred inside the enclave, software can enter the enclave at a different location and invoke the exception handler within the enclave by executing the EENTER instruction. The exception handler within the enclave can attempt to resolve the faulting condition or simply return and indicate to software that the enclave should be terminated (e.g. using EEXIT).

3.2.3.1 ERESUME Interaction

ERESUME restores registers depending on the mode of the enclave (32 or 64 bit).

- In 32-bit mode (IA32_EFER.LMA = 0 || CS.L = 0), the low 32-bits of the legacy registers (EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI, EIP and EFLAGS) are restored from the thread's GPR area of the current SSA frame. Neither the upper 32 bits of the legacy registers nor the 64-bit registers (R8 ... R15) are loaded.
- In 64-bit mode (IA32_EFER.LMA = 1 && CS.L = 1), all 64 bits of the general processor registers (RAX, RBX, RCX, RDX, RSP, RBP, RSI, RDI, R8 ... R15, RIP and RFLAGS) are loaded.

Extended features specified by SECS.ATTRIBUTES.XFRM are restored from the XSAVE area of the current SSA frame. The layout of the x87 area depends on the current values of IA32_EFER.LMA and CS.L:

- IA32_EFER.LMA = 0 || CS.L = 0
  - — 32-bit load in the same format that XSAVE/FXSAVE uses with these values.
- IA32_EFER.LMA = 1 && CS.L = 1
  - — 64-bit load in the same format that XSAVE/FXSAVE uses with these values plus REX.W = 1

3.3 CALLING ENCLAVE PROCEDURES

3.3.1 Calling Convention

In standard call conventions subroutine parameters are generally pushed onto the stack. The called routine, being aware of its own stack layout, knows how to find parameters based on compile-time-computable offsets from the SP or BP register (depending on runtime conventions used by the compiler).

Because of the stack switch when calling an enclave, stack-located parameters cannot be found in this manner. Entering the enclave requires a modified parameter passing convention.

For example, the caller might push parameters onto the untrusted stack and then pass a pointer to those parameters in RAX to the enclave software. The exact choice of calling conventions is up to the writer of the edge routines; be those routines hand-coded or compiler generated.

3.3.2 Register Preservation

As with most systems, it is the responsibility of the callee to preserve all registers except that used for returning a value. This is consistent with conventional usage and tends to optimize the number of register save/restore opera-

3-4 Ref. # 329298-001

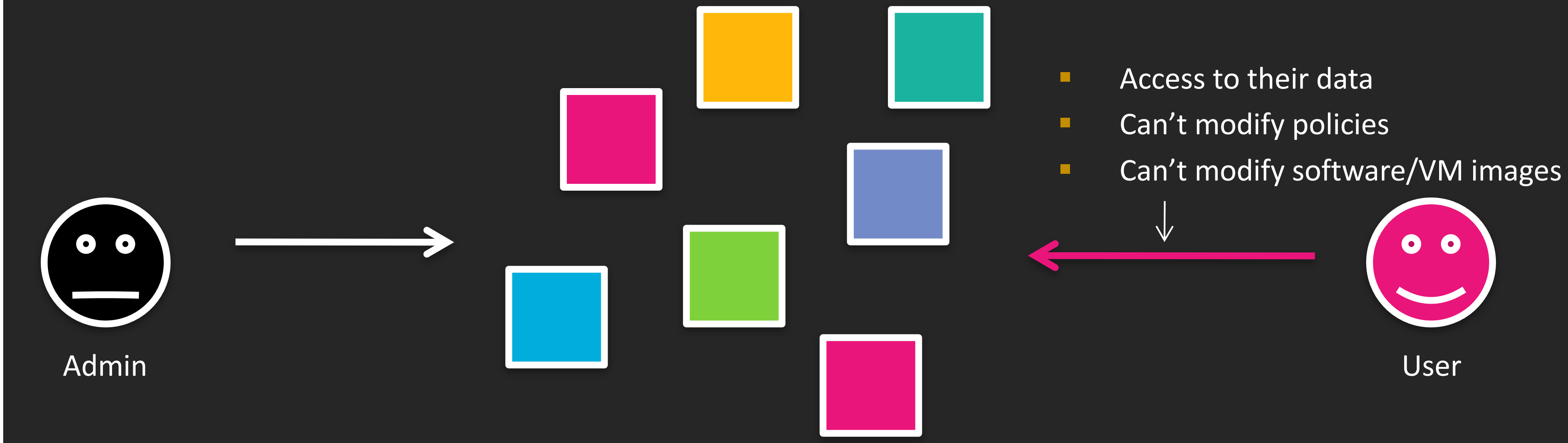
Name	State	NetVM	CPU Graph	MEM
dom0	●	n/a		2598 MB
sys-net	●	n/a		301 MB
sys-firewall	●	sys-net		301 MB
varia	●	sys-firewall		979 MB
work-web	●	sys-firewall		1173 MB
work-mutt	●	sys-firewall		604 MB
keys-itl-email	●	---		478 MB
work	●	sys-firewall		607 MB
personal	●	sys-firewall		750 MB

```
[user@work ~]$ 
```

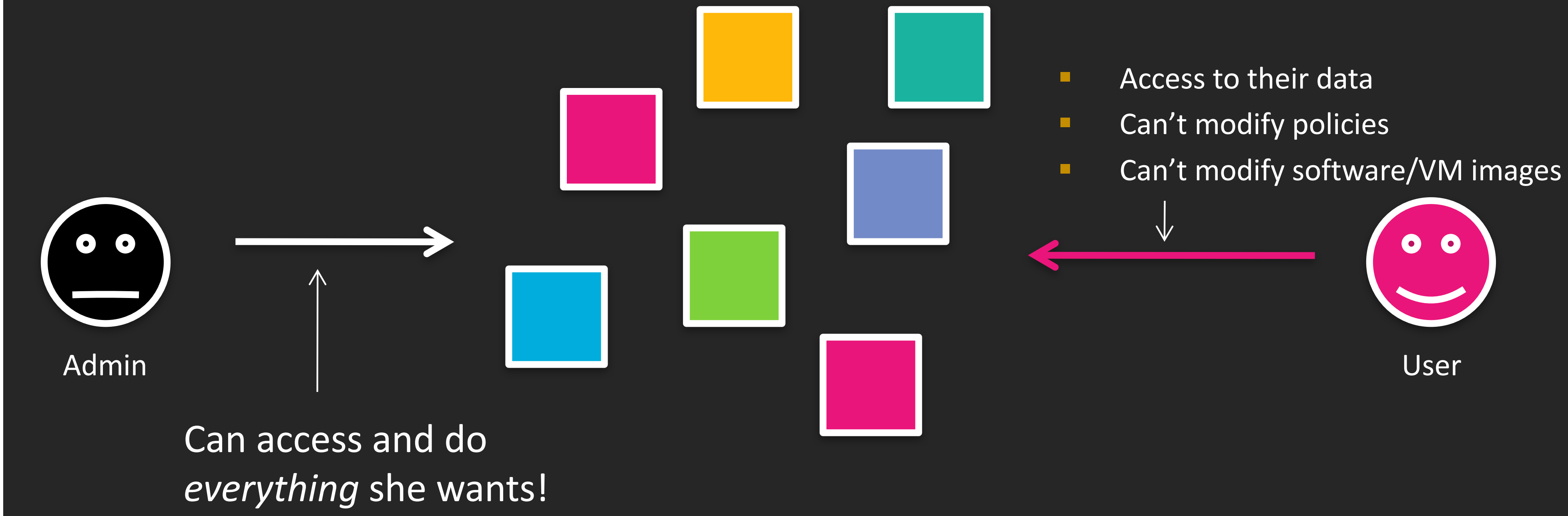
Biggest challenge for Qubes OS is
how to do *desktop integration* (seamless UX)
without compromising isolation!

Example #7: Almighty admins?

Admins can steal all our data :(



- Access to their data
- Can’t modify policies
- Can’t modify software/VM images



- Access to their data
- Can’t modify policies
- Can’t modify software/VM images

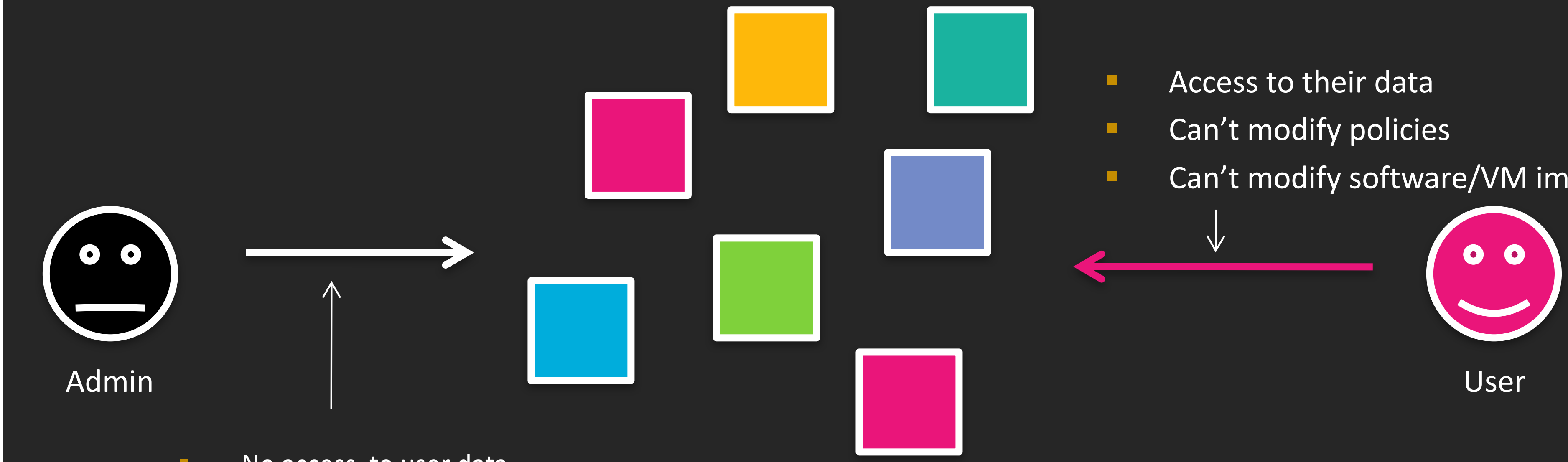
Admin

User

Can access and do *everything* she wants!

Hmm…

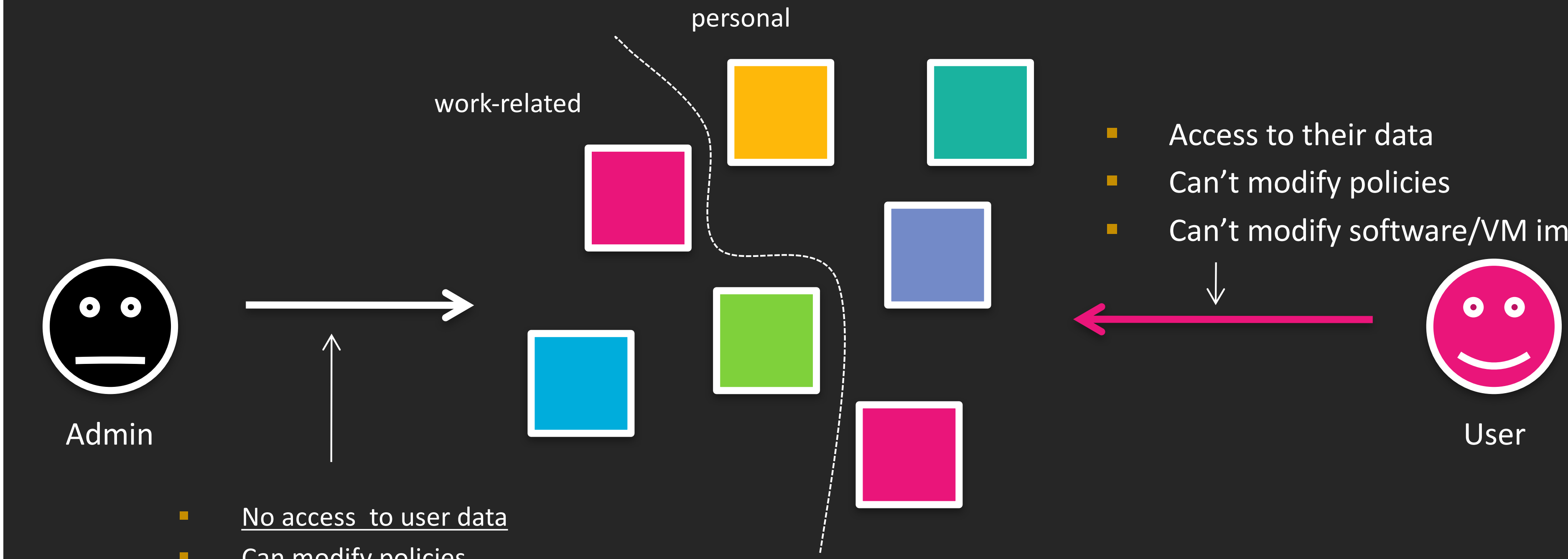
What we want instead:



- Access to their data
- Can’t modify policies
- Can’t modify software/VM images

- <u>No access to user data</u>
- Can modify policies
- Can install software/VM images

What we want instead:



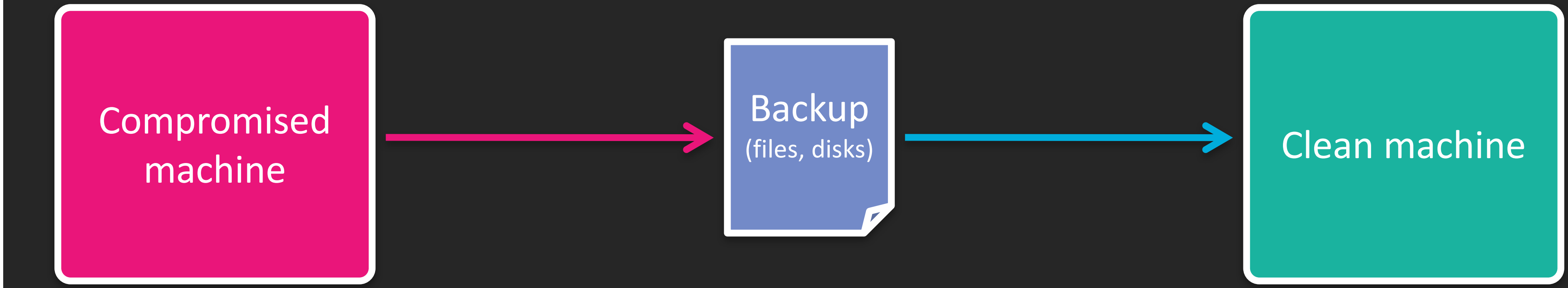
- Access to their data
- Can’t modify policies
- Can’t modify software/VM images

- No access to user data
- Can modify policies
- Can install software/VM images

Check our Qubes OS new Admin API for implementation details

Occasionally mishaps happen still…

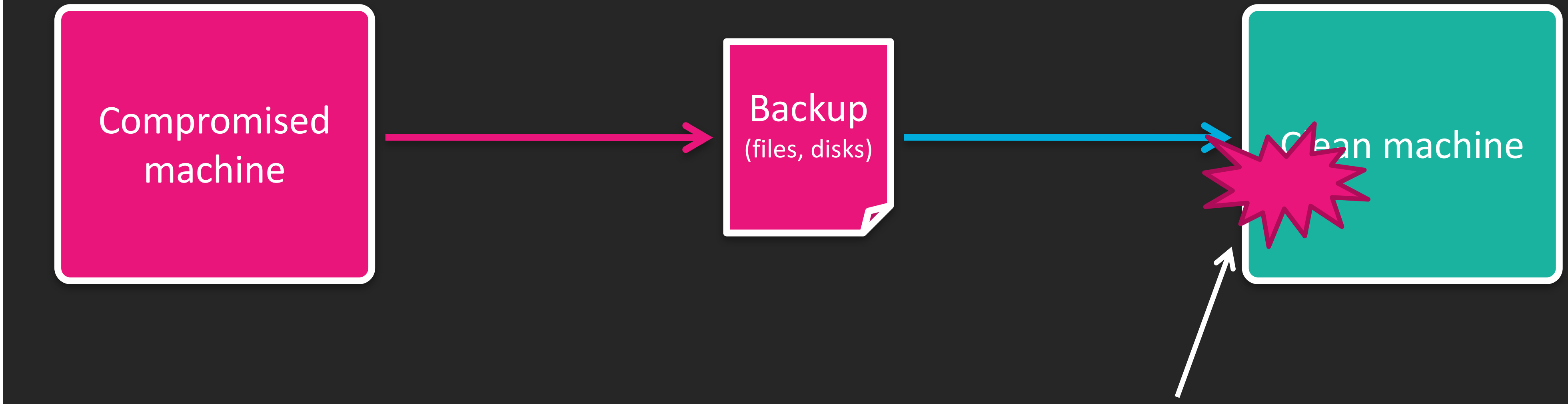
Example #8: Plan B



Compromised machine

Backup
(files, disks)

Clean machine



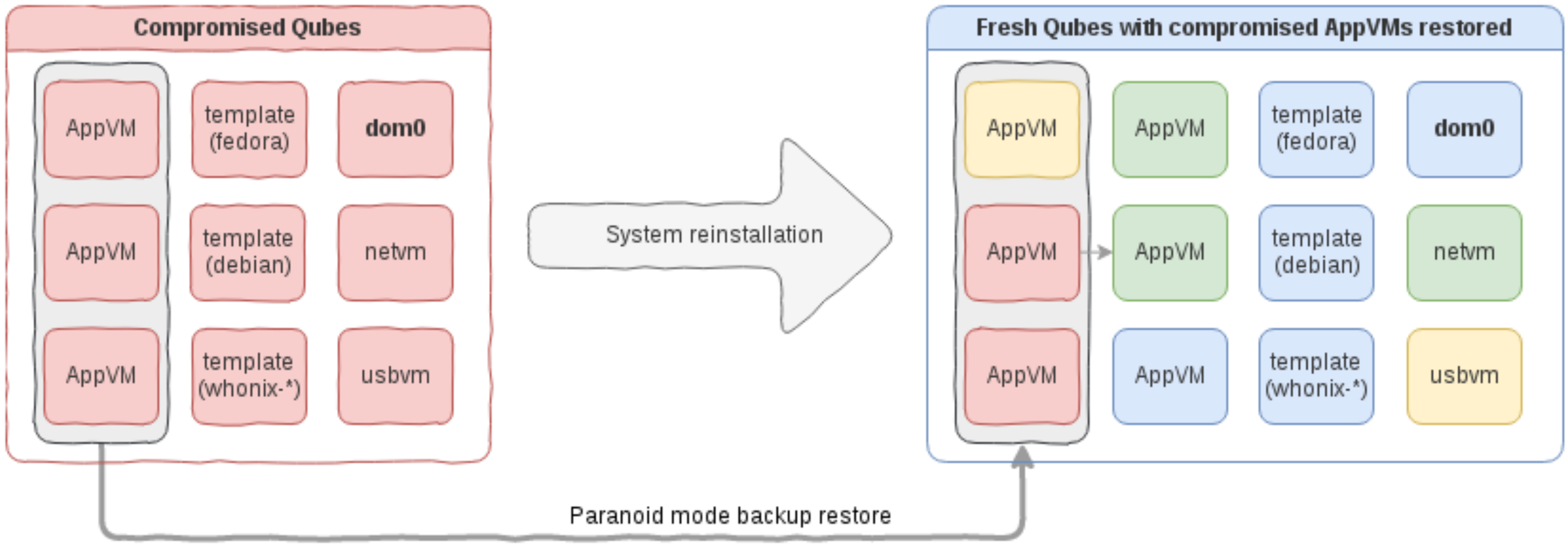
Compromised machine

Backup
(files, disks)

Clean machine

Restoring compromised backup is risky!

Qubes (Paranoid) Backup Restore



Security through Distrusting

Division of Duty

- Mics (#1)
- Stateless laptop (#2)
- Multi signatures (#3/4)
- Tunneling (#5)

Compartmentalization

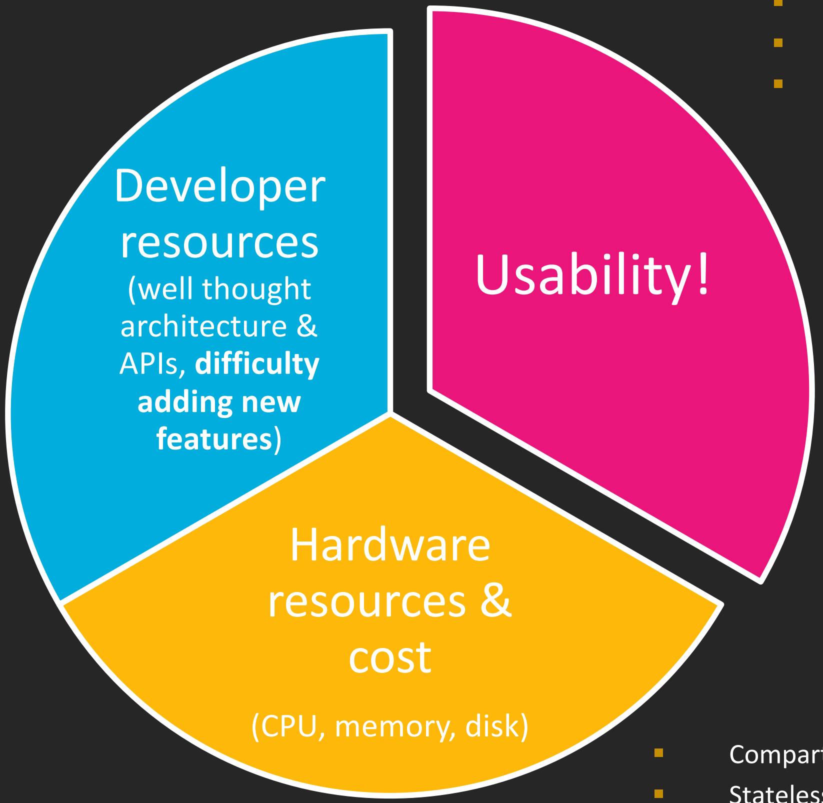
- Qubes (#6/7)
- Tunneling (#5)
- Qubes Backup Restore (#8)

Plan B having

- Qubes Backup Restore (#8)

Tradeoffs?

- Compartmentalization
- Multisigs for binaries
- Stateless laptop (BOM costs)



Developer resources (well thought architecture & APIs, **difficulty adding new features**)

Usability!

Hardware resources & cost

(CPU, memory, disk)

- Mics
- Multisigs for wallets
- Compartmentalization (?)

- Compartmentalization
- Stateless laptop (BOM costs)

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494

Thanks!

***

https://qubes-os.org
https://invisiblethingslab.com
https://blog.invisiblethings.org
https://github.com/rootkovska
@QubesOS // Twitter for Qubes OS
@rootkovska // Personal Twitter
427F11FD0FAA4B080123F01CDDFA1A3E36879494 // Qubes Master Key
ED727C306E766BC85E621AA65FA6C3E4D9AFBB99 // Personal Master Key

This presentation made in MS Office on Qubes OS 3.2

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494