

Security through Distrusting

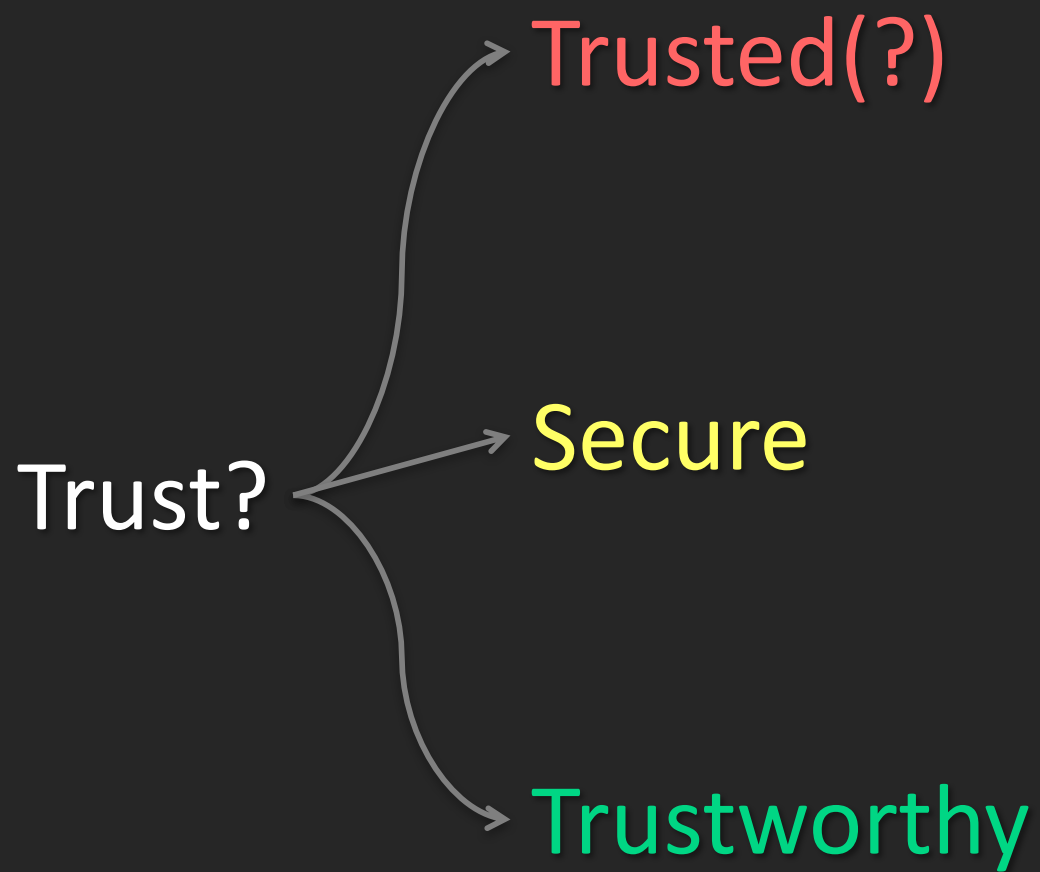
Joanna Rutkowska

Invisible Things Lab & [Qubes OS](#) Project

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

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494





more desired



Trust consider harmful!

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Security through Distrusting examples

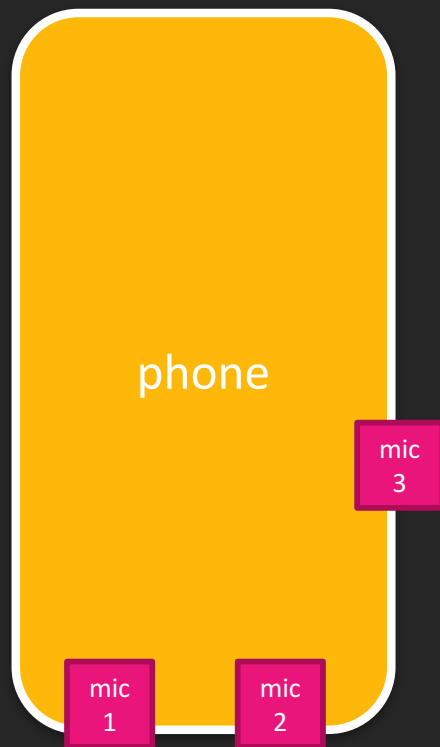
427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Example #1: Pesky microphones

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

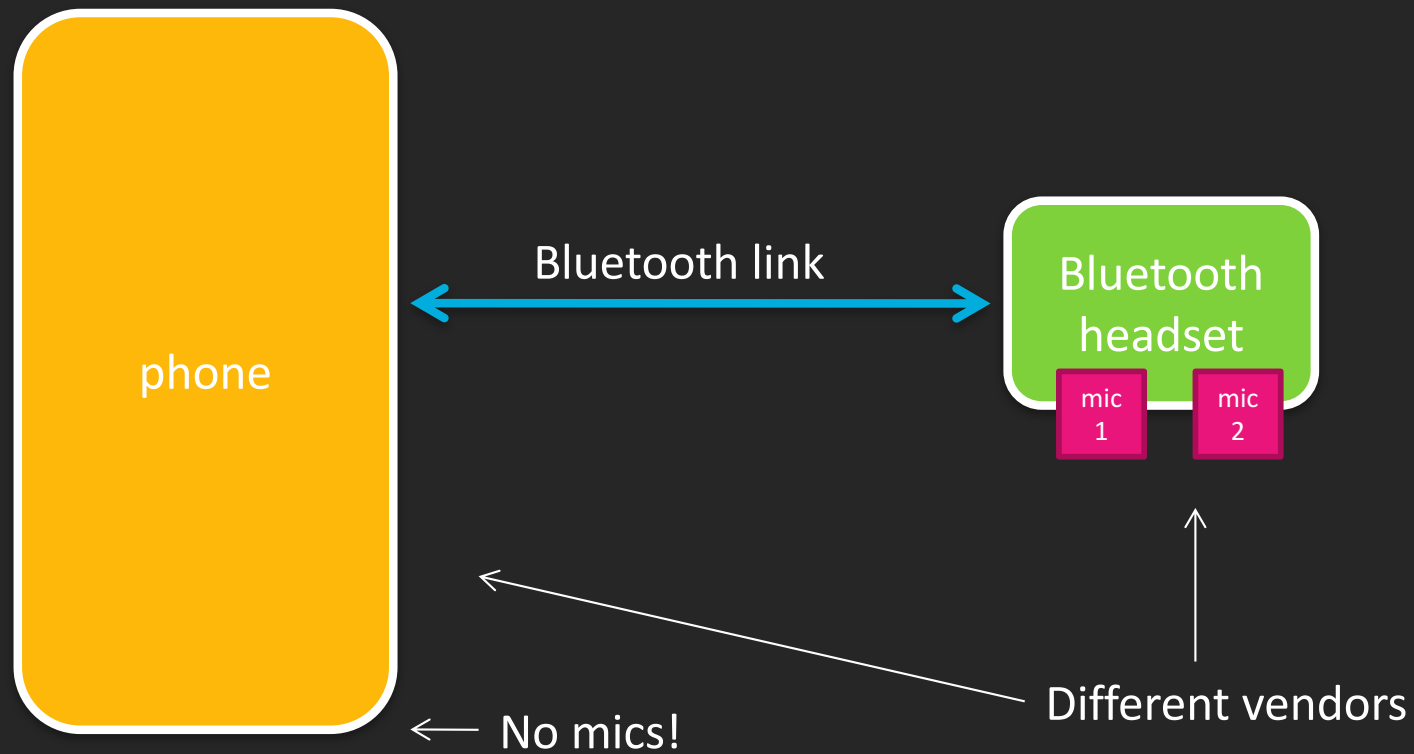




- Mics sniff our activities, including keystrokes, etc.
- Mics are difficult to neutralize
- Mics naturally “cross” security boundaries



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



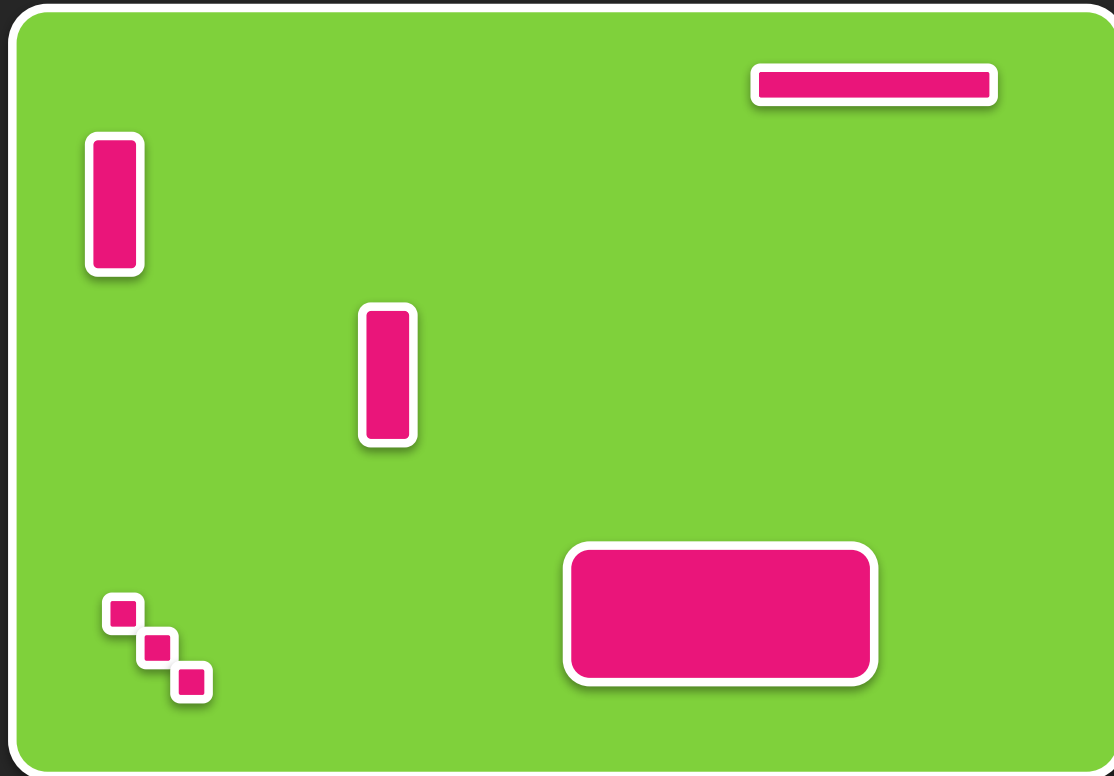
Example #2: Stateless laptop

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

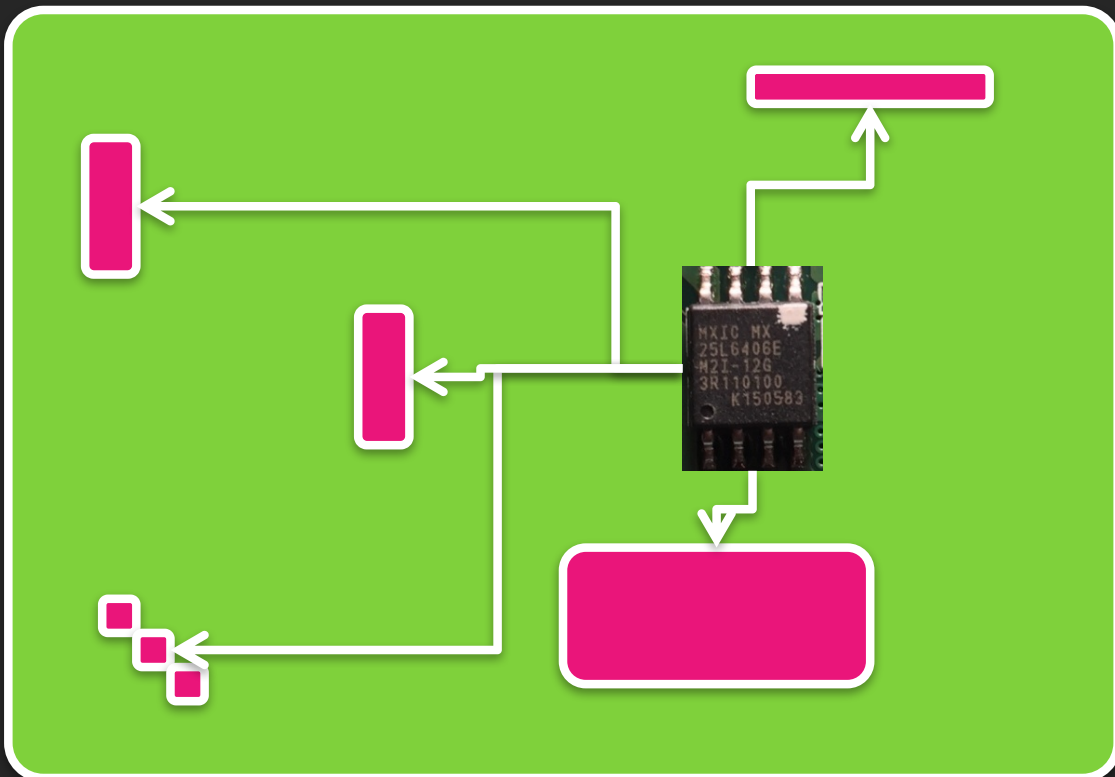


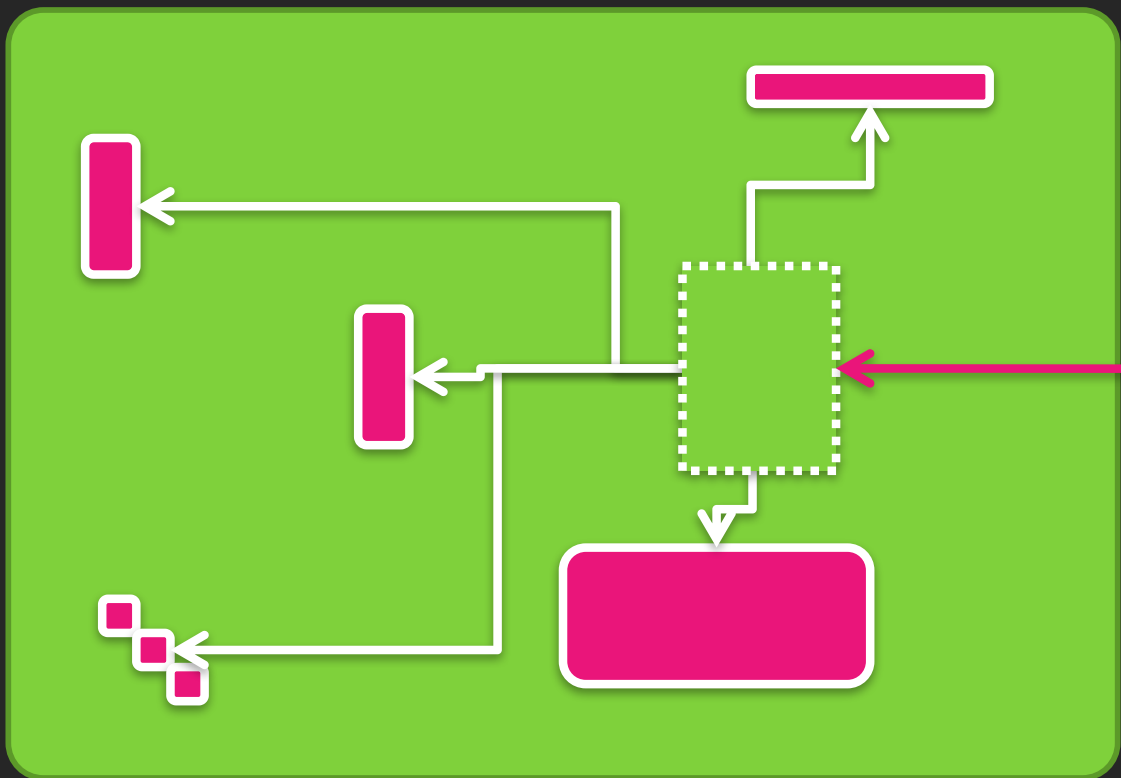
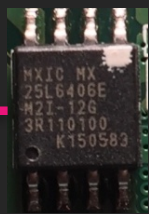
Persistent laptop compromises...

- Persist
- Store secrets
- PII



- Persist
- Store secrets
- PII

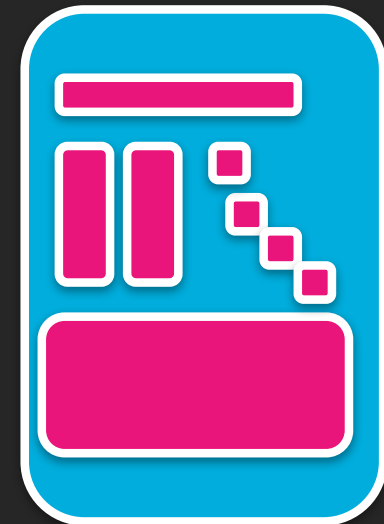




- Persist
- Store secrets
- PII

Stateless Hardware (persistent state eliminated)

- 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



Trusted Stick

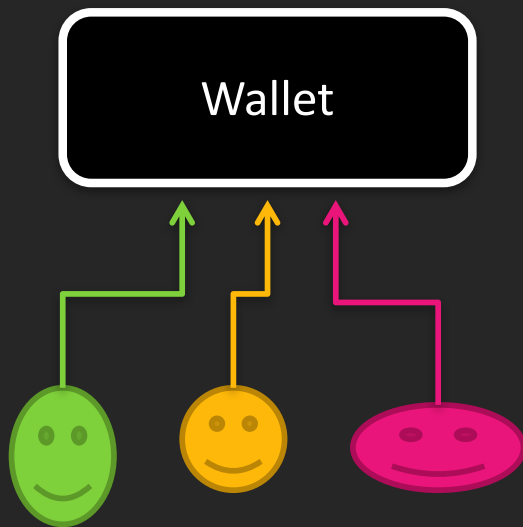


Example #3: Multi-party signatures

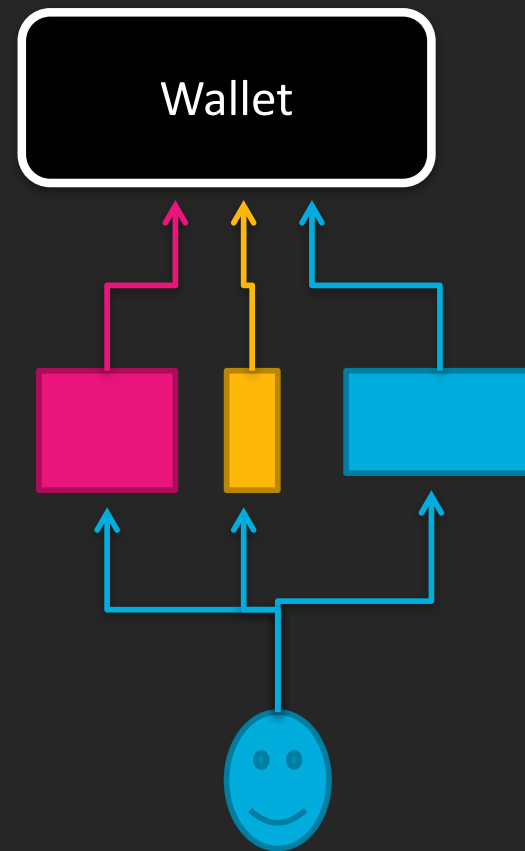


Photo via Peter Todd (@petertoddbtc)





VS



Mutli-sig does not need to involve multiple users!



Also: not just Bitcoin wallets...

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Example #4: Binary (multi) signing

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Why care about binary (multi-) signing?

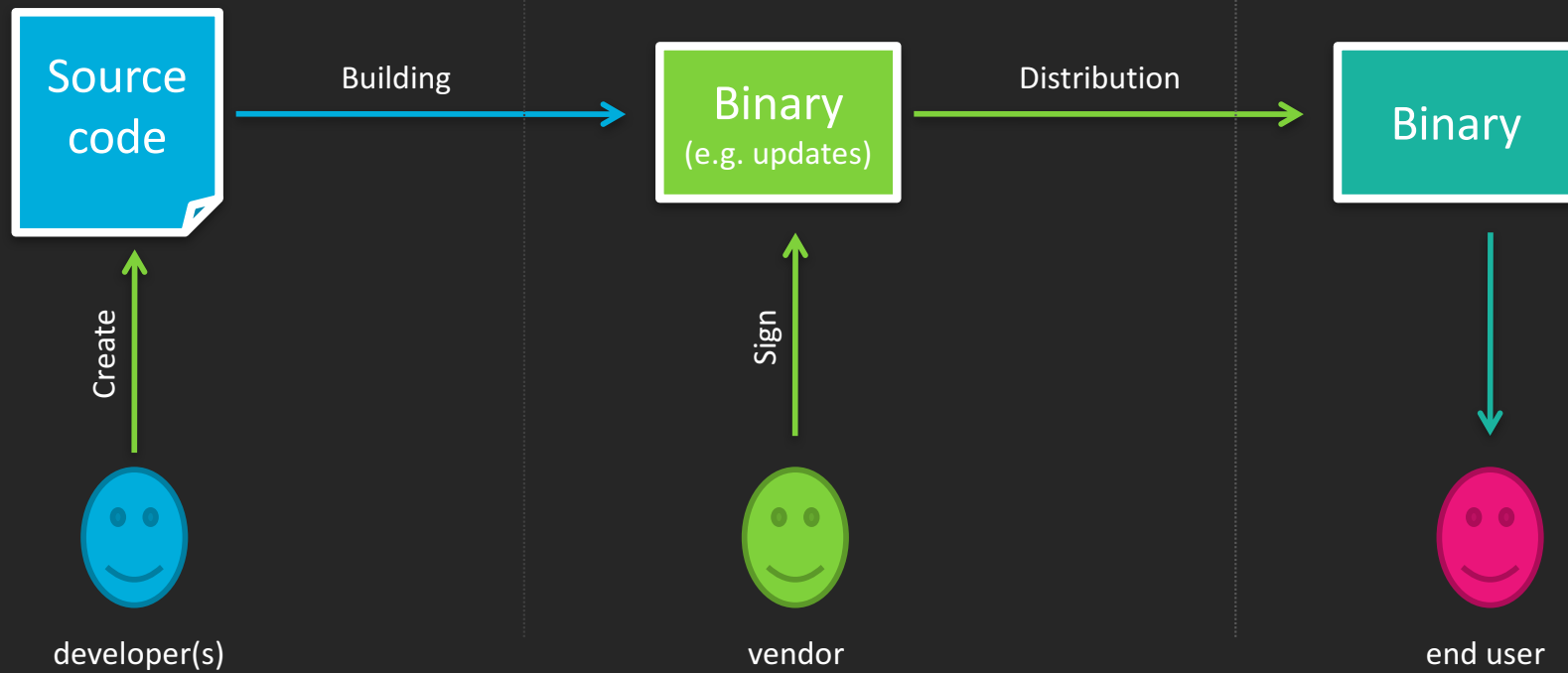
- OS installation images
- Applications
- Updates
- Firmware

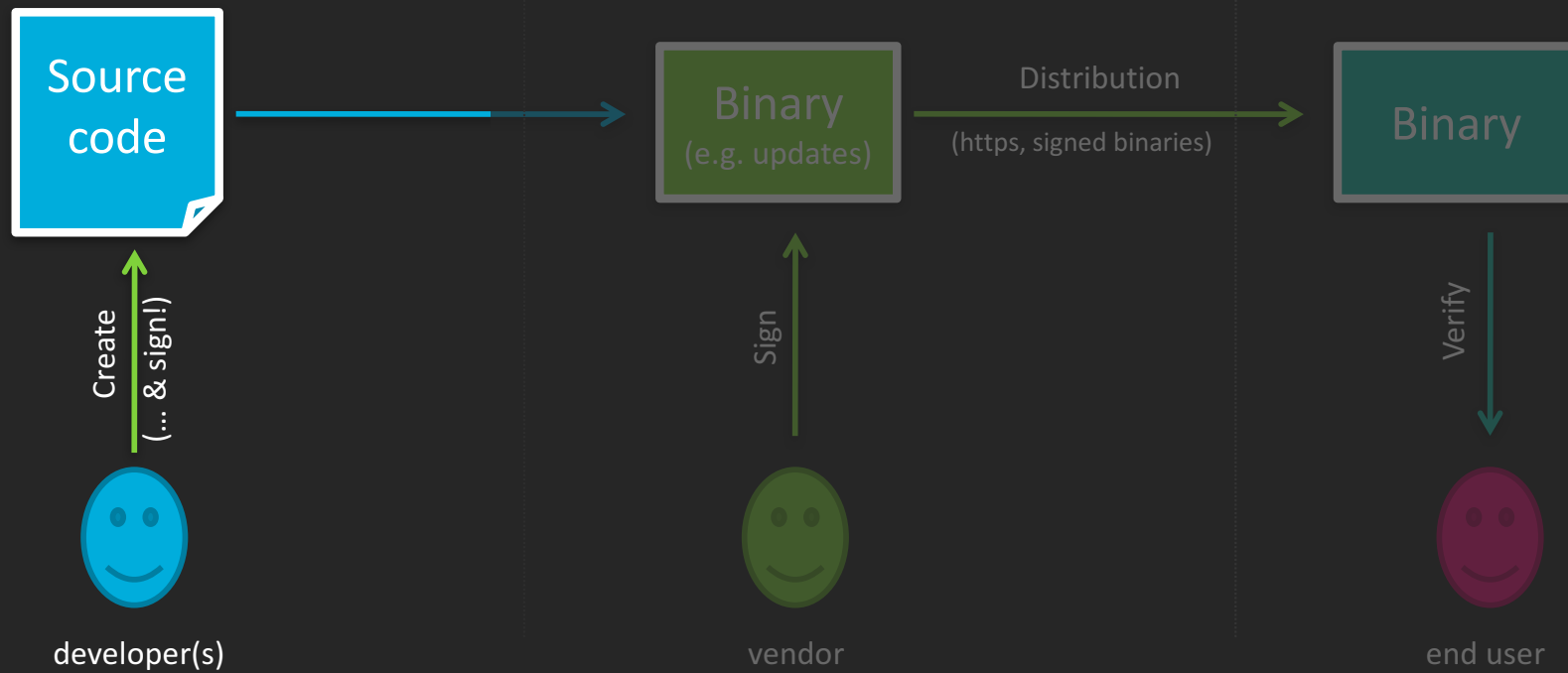


Prime target for backdooring!

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

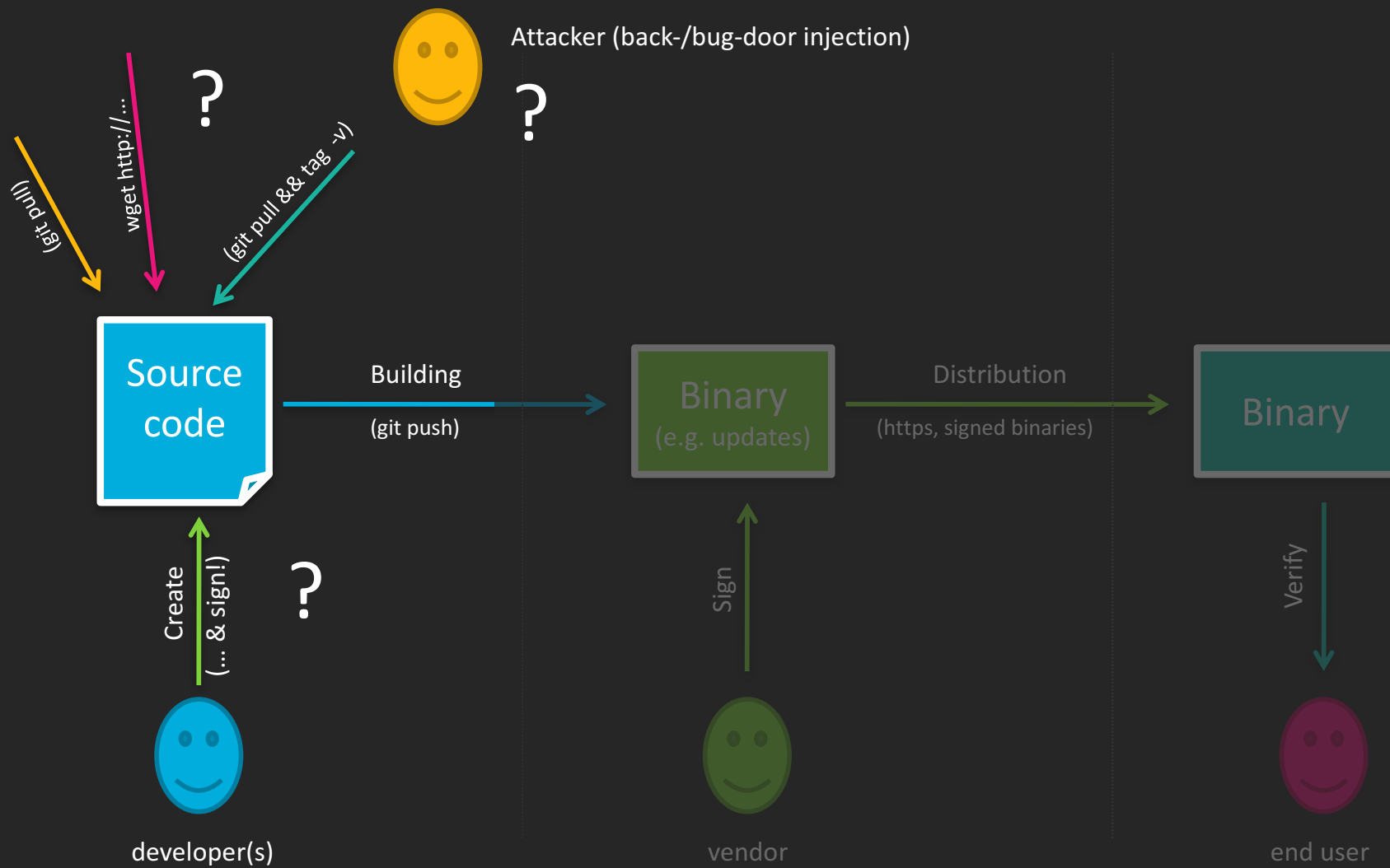






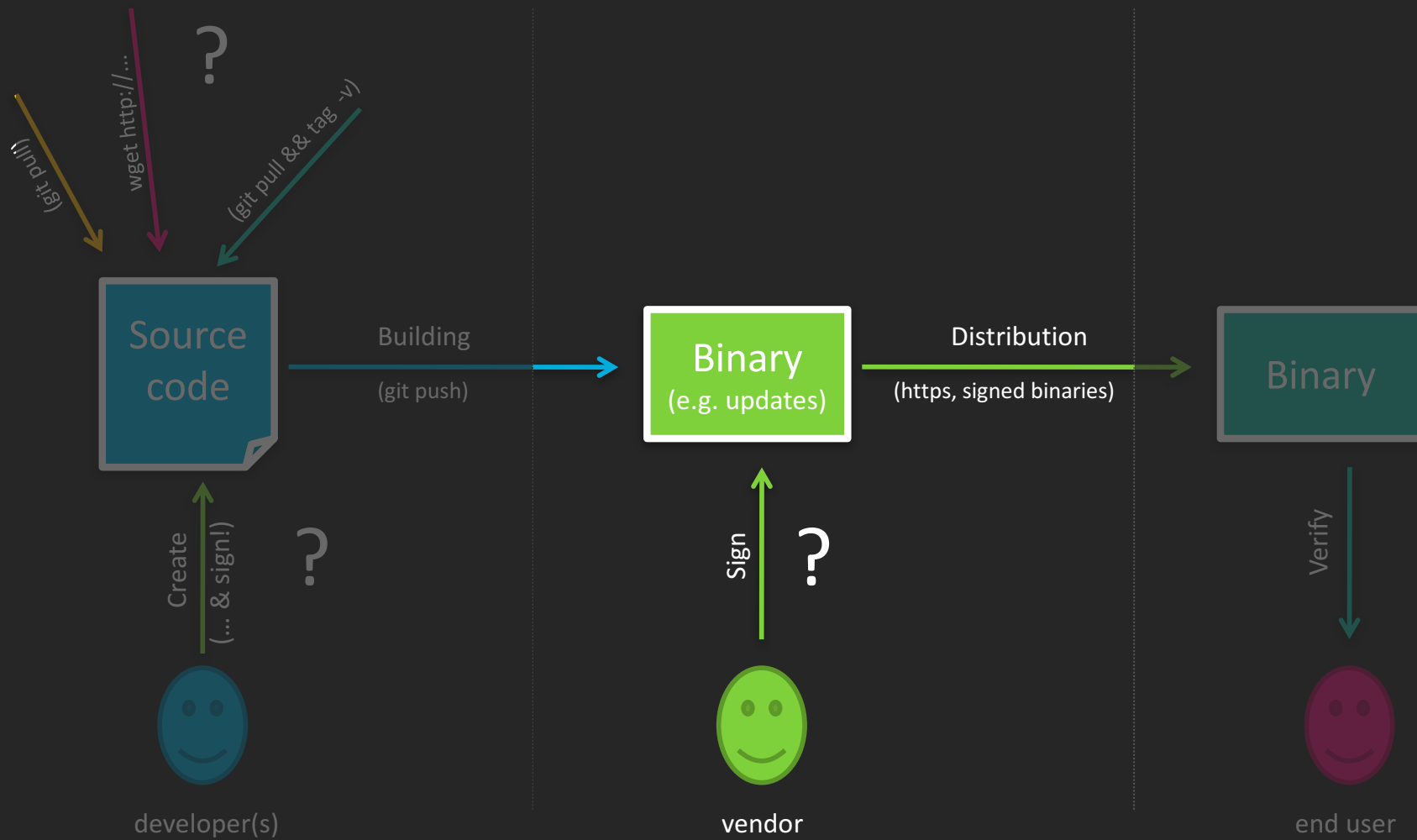
427F11FD 0FAA4B08 0123F01C DDFAlA3E 36879494





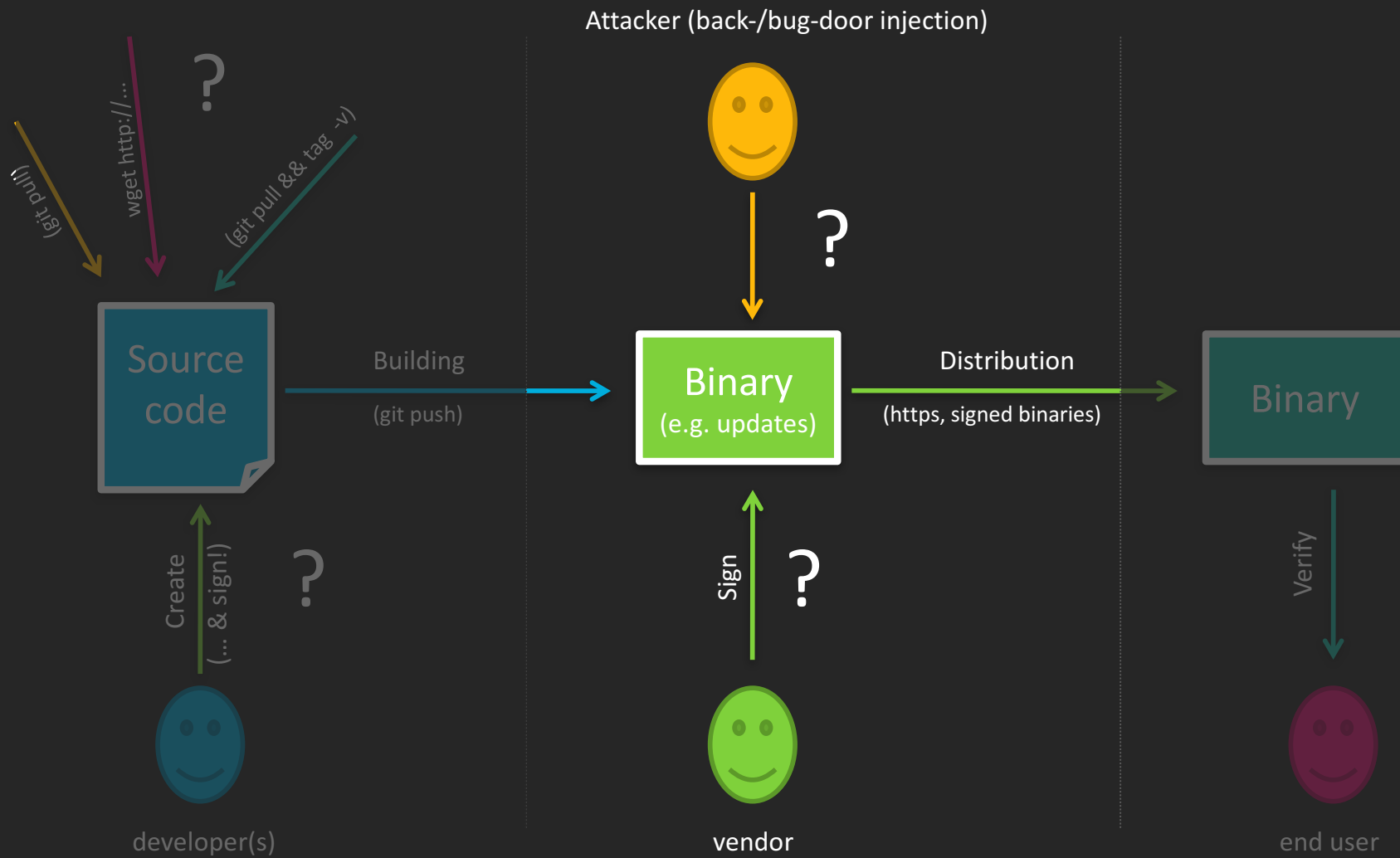
427F11FD 0FAA4B08 0123F01C DDF1A3FE 36879494





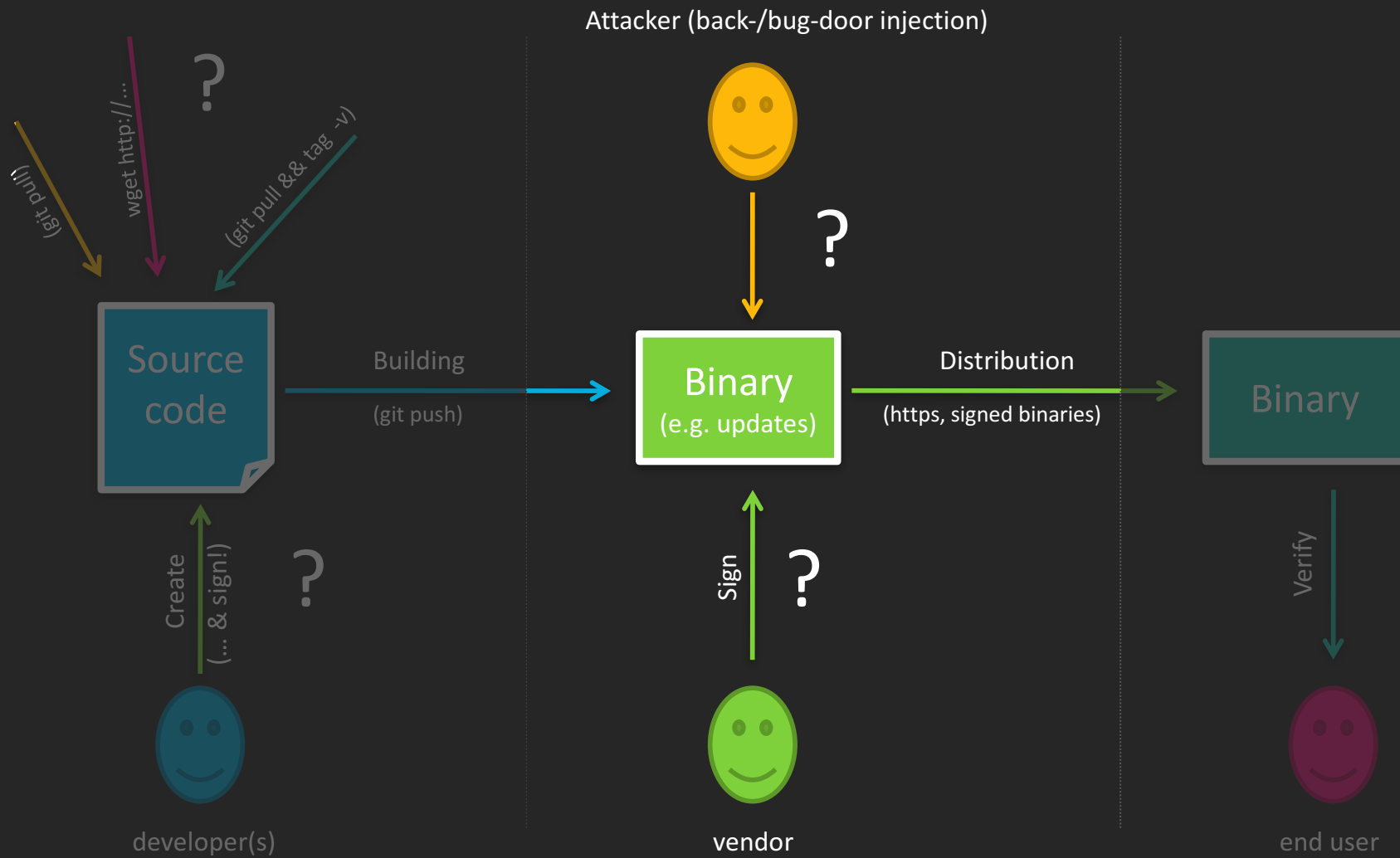
427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494





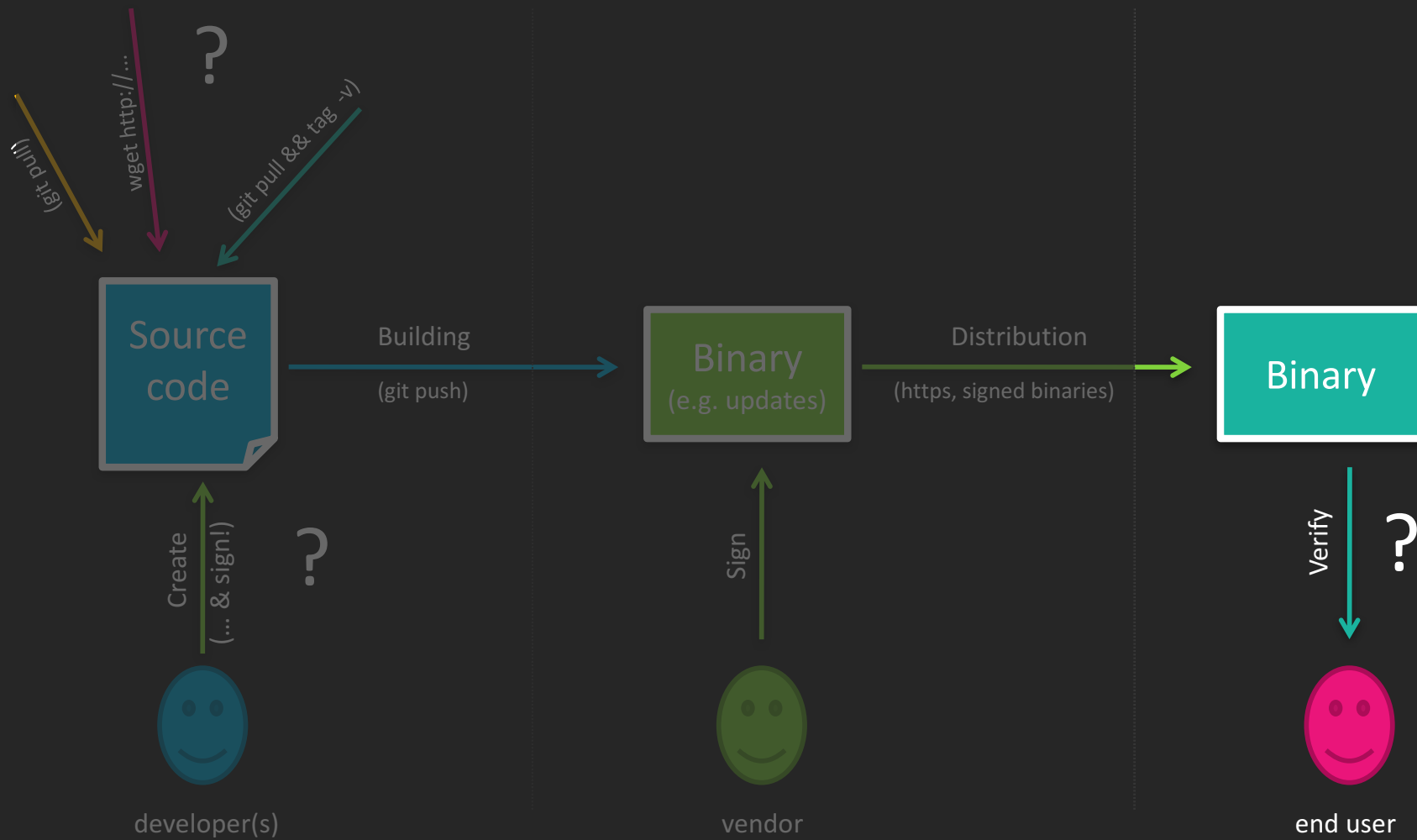
427F11FD 0FAA4B08 0123F01C DDF1A3E 36879494





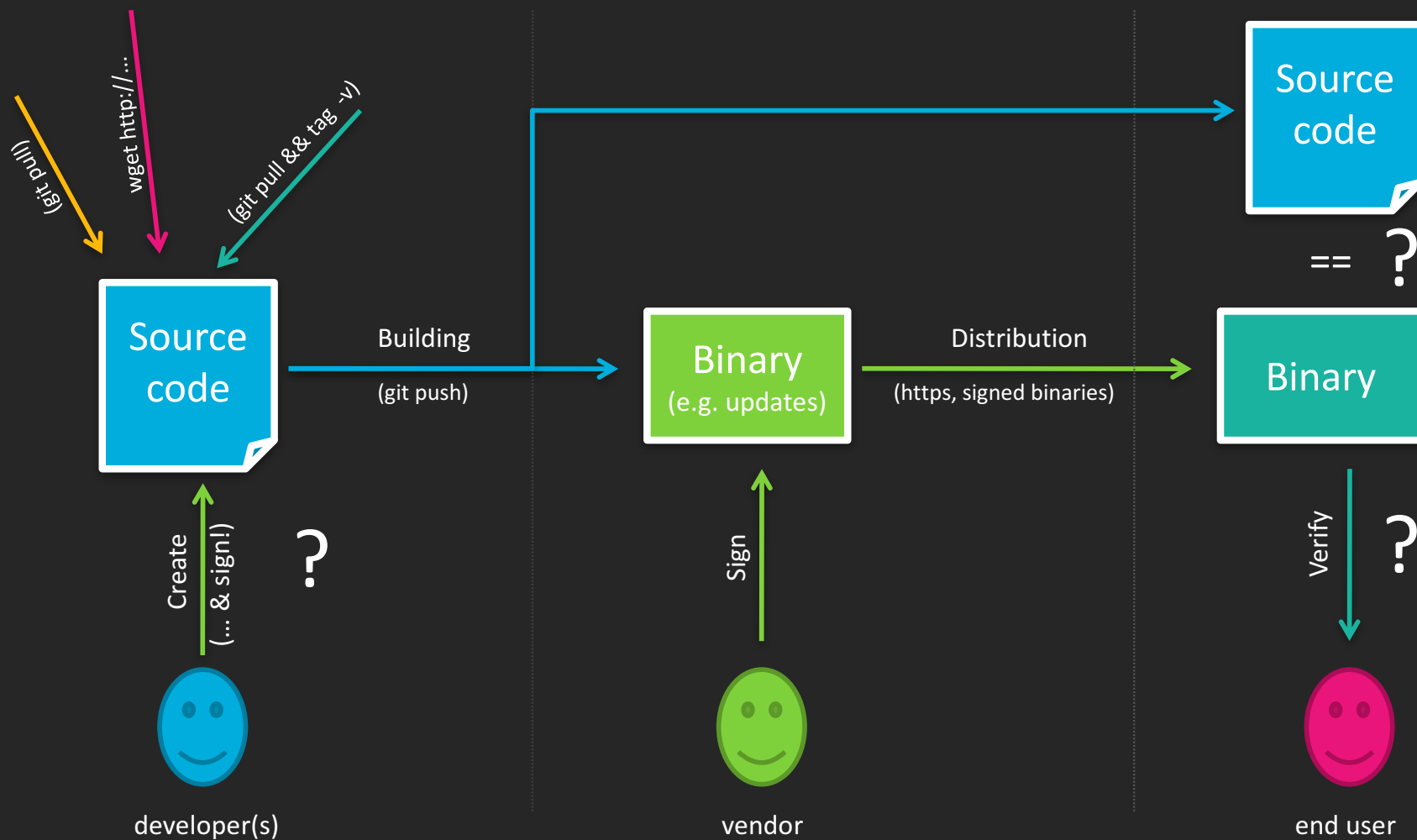
427F11FD 0FAA4B08 0123F01C DDF1A3E 36879494

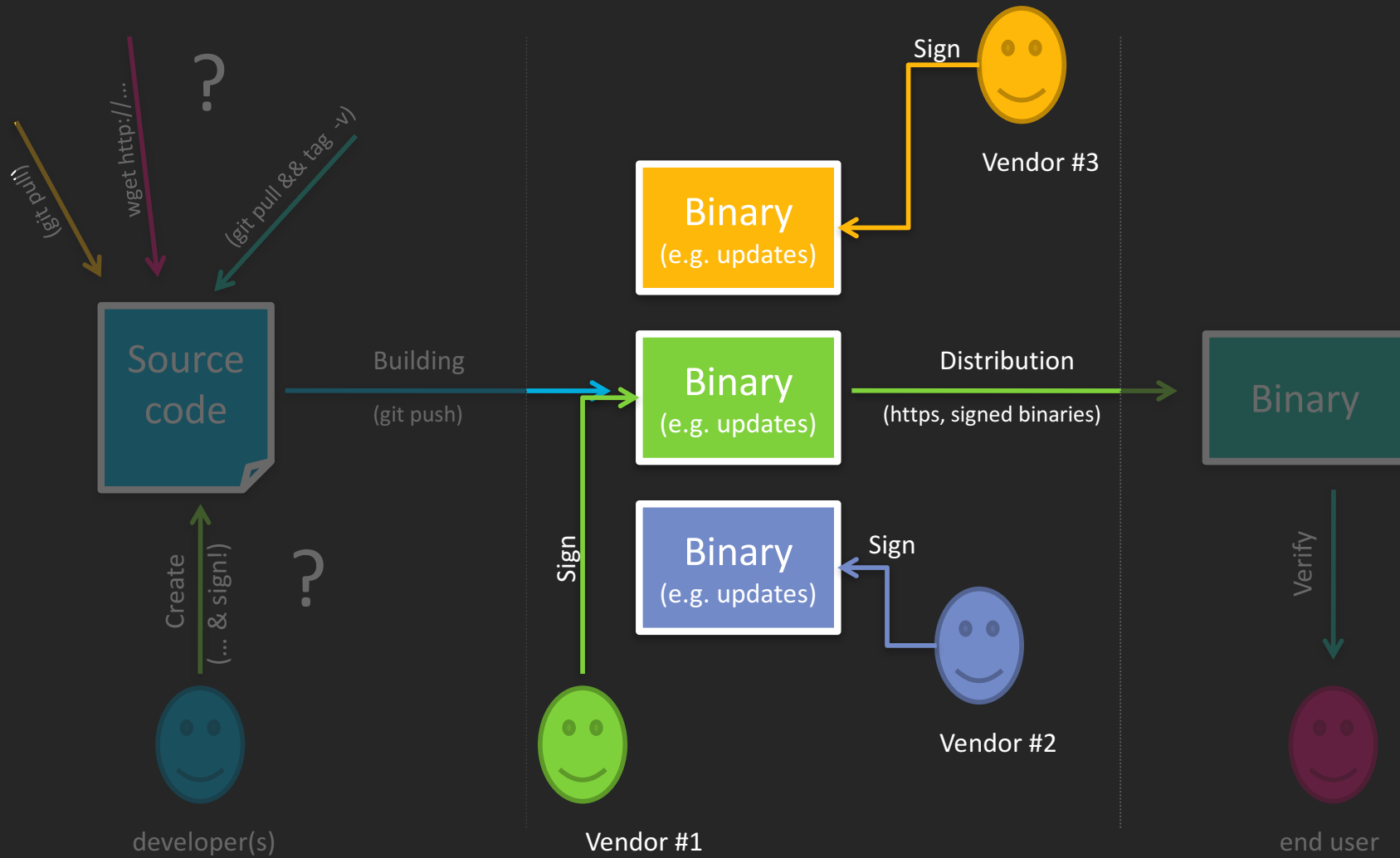




427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

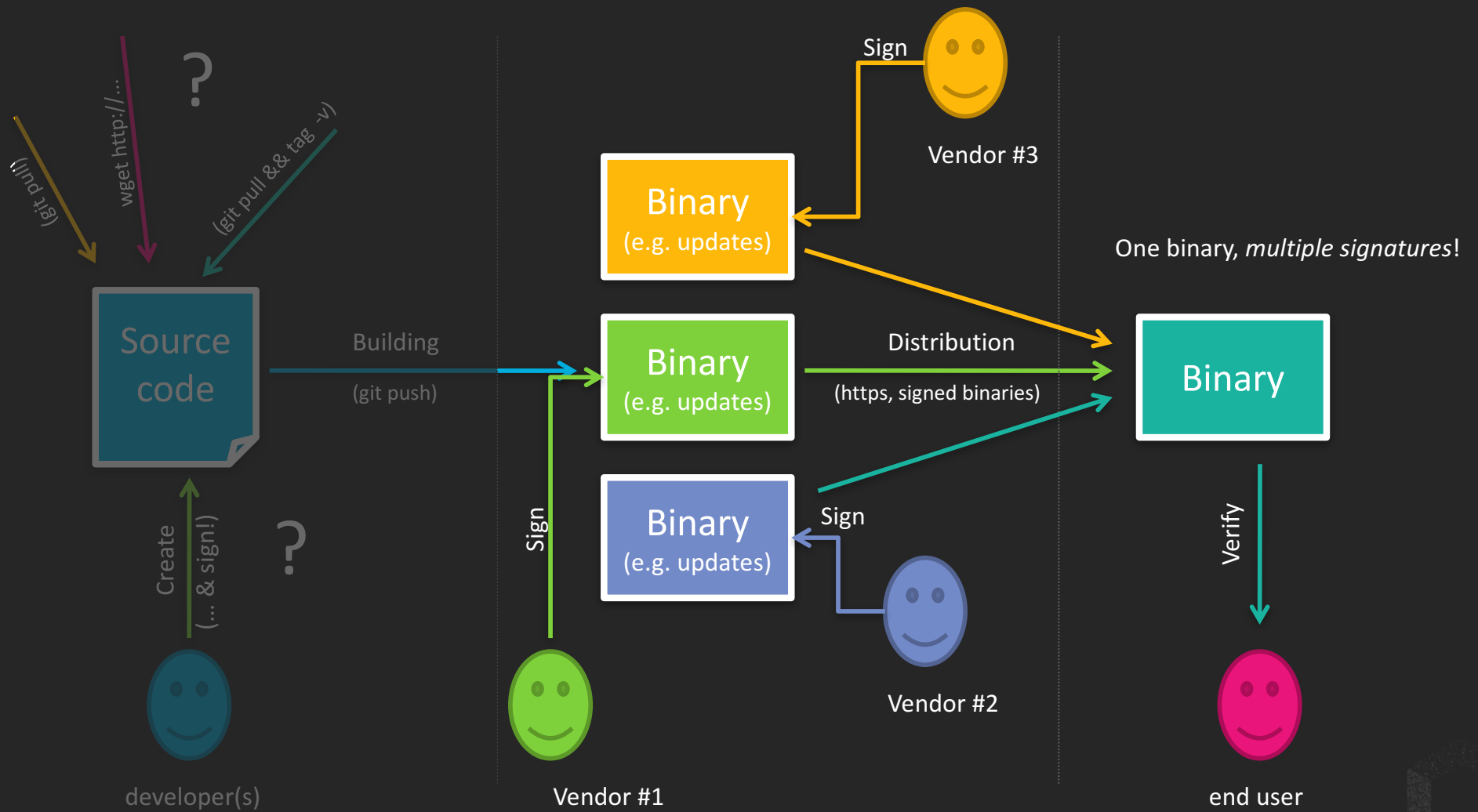






427F11FD 0FAA4B08 0123F01C DDFALASFE 36879494





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>

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Example 5: Preventing data leaks

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Some software
(buggy/backdoored or
otherwise compromised)

Your data...!



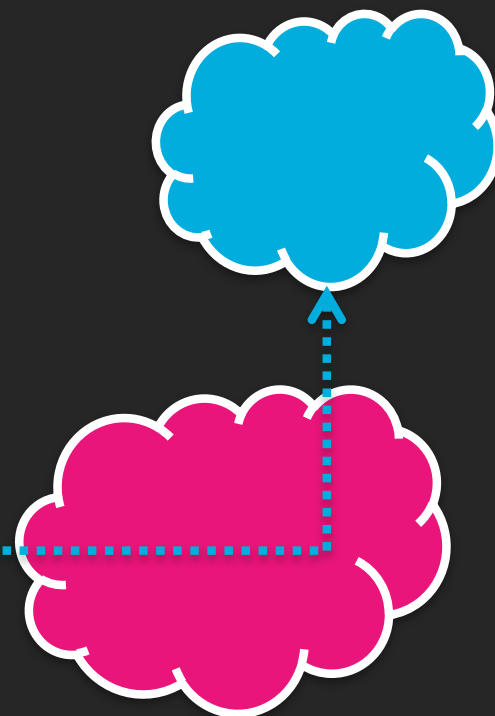
427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Some software
(buggy/backdoored or
otherwise compromised)

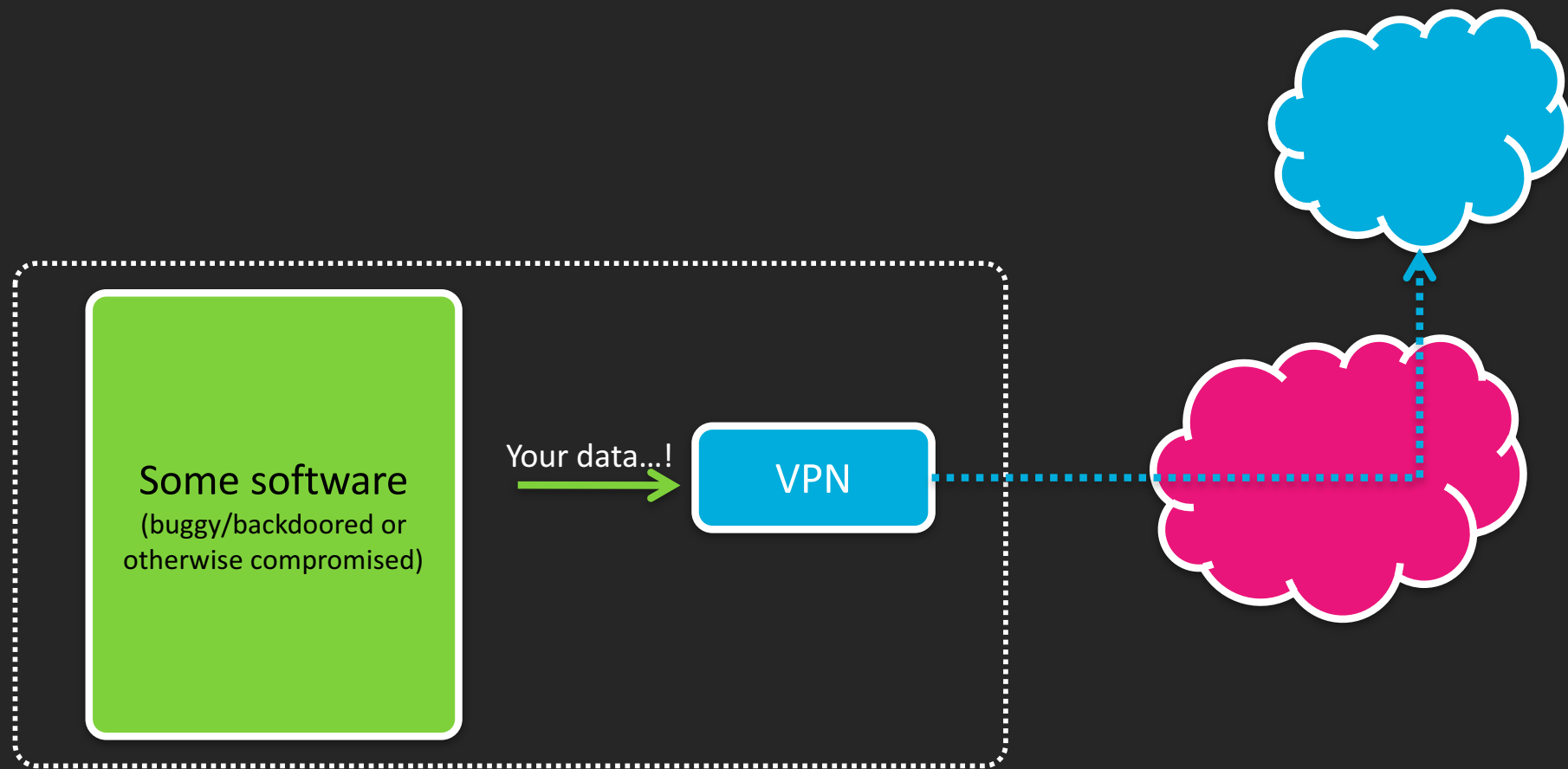
Your data...!

VPN



427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

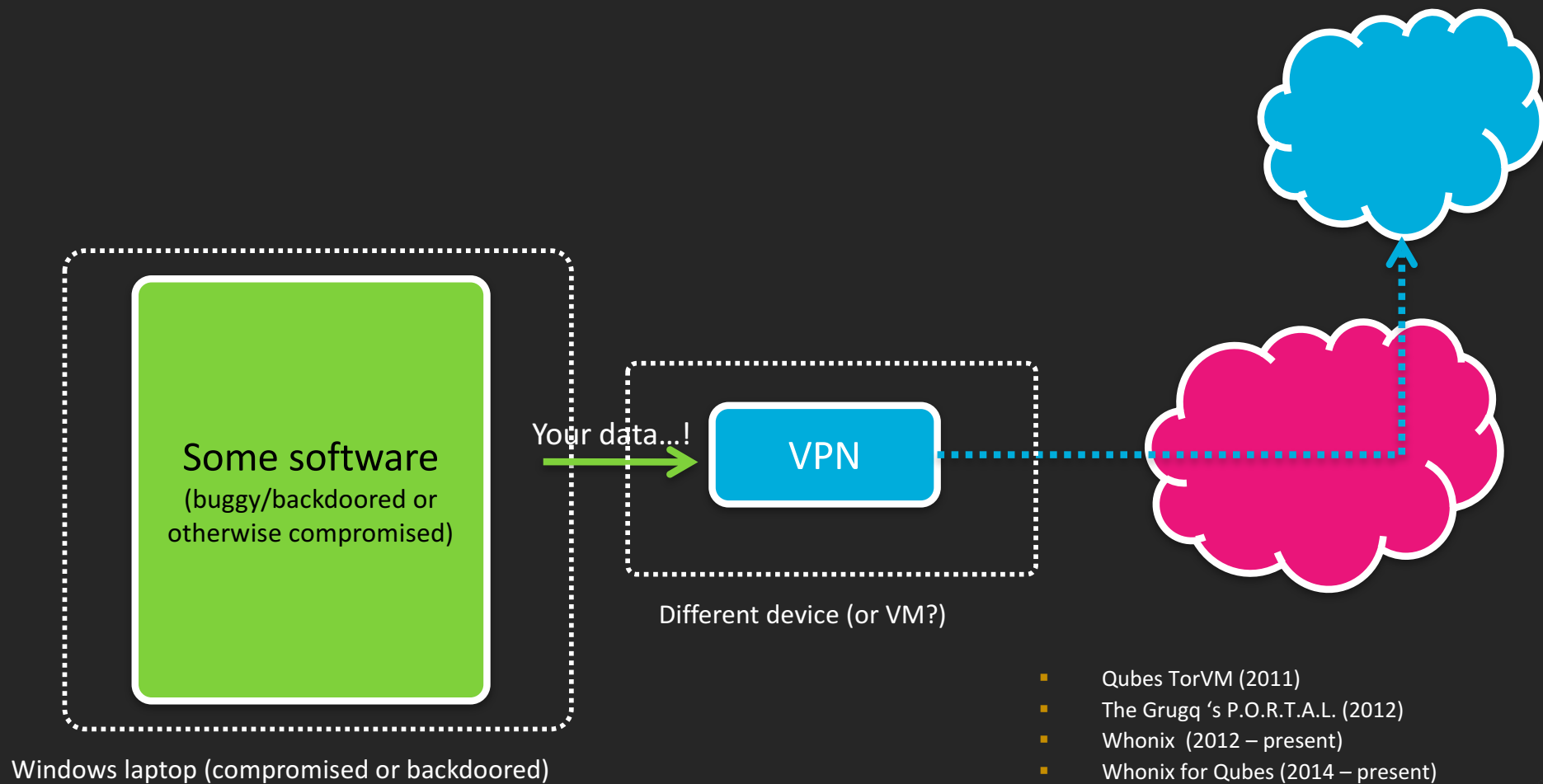




Windows laptop (compromised or backdoored)

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

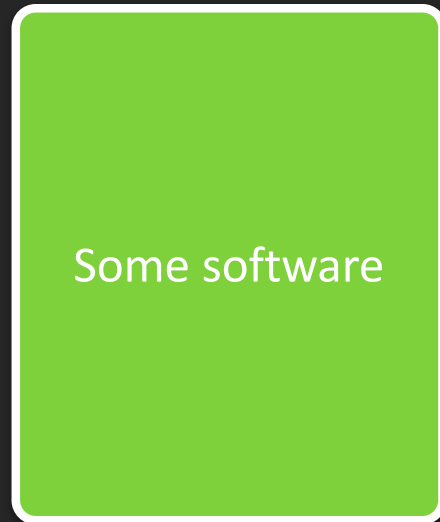




- Qubes TorVM (2011)
- The Grugq 's P.O.R.T.A.L. (2012)
- Whonix (2012 – present)
- Whonix for Qubes (2014 – present)
- Tor-enabled routers (multiple projects/products)



Cut off networking?

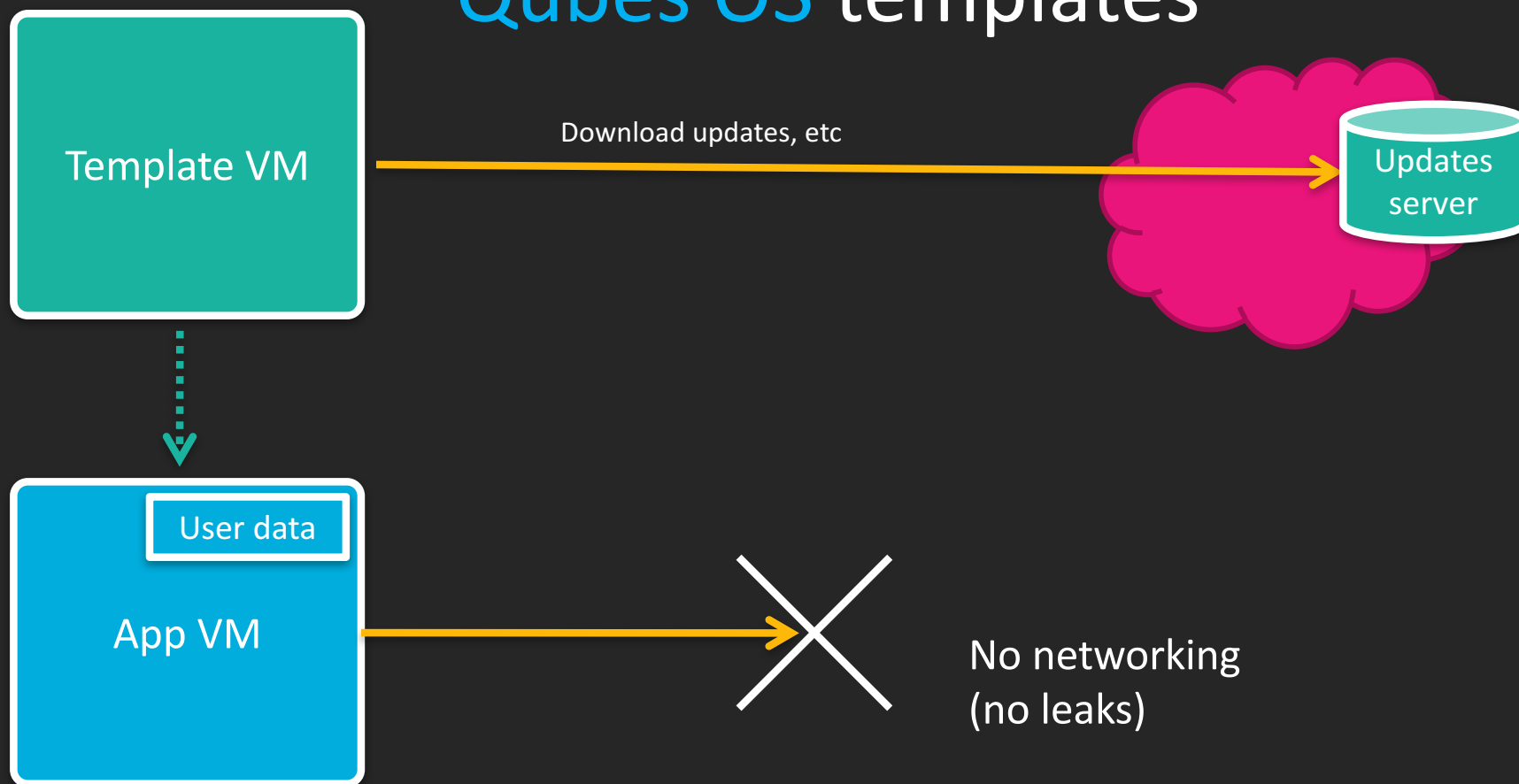


Not very useful...

427F11FD 0FAA4B08 0123F01C DDEA1A3E 36879494



Qubes OS templates



427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Example #6: Compartmentalization

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



“Classic” compartmentalization...



Work VM

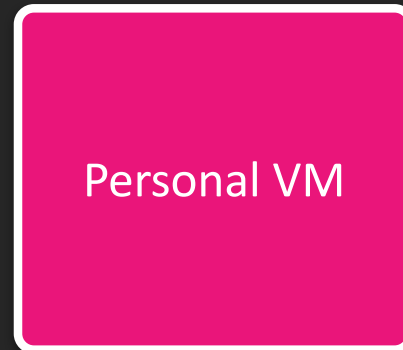
Personal VM



“Classic” compartmentalization...



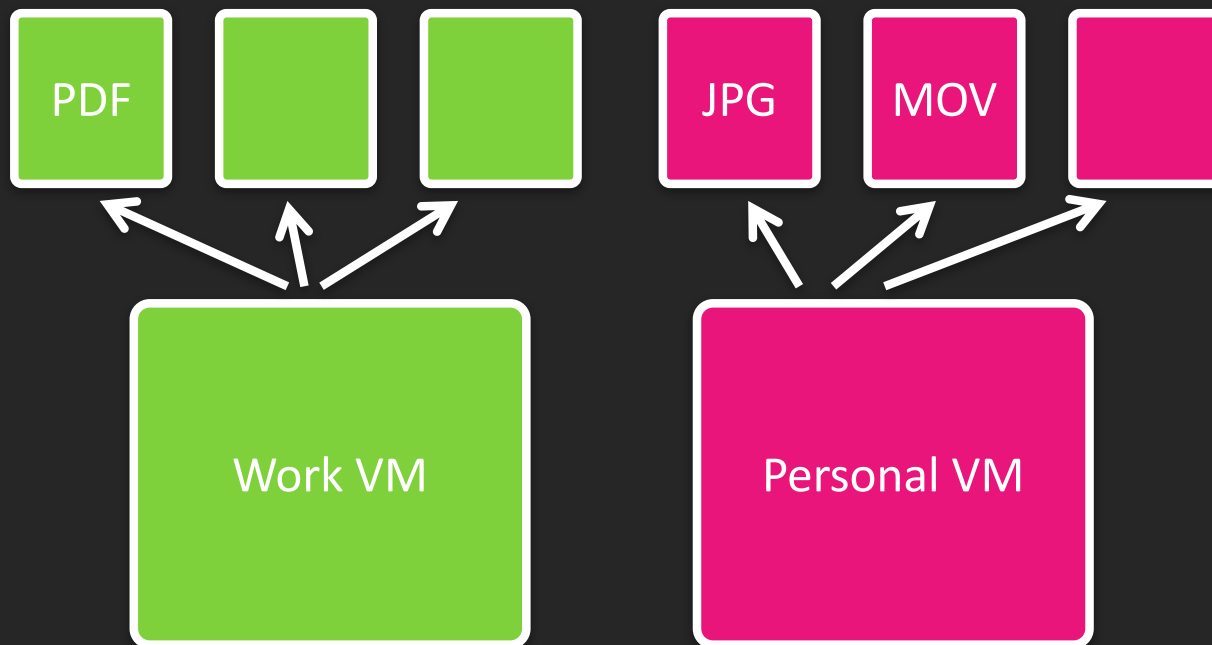
Work VM



Personal VM

...not very useful!





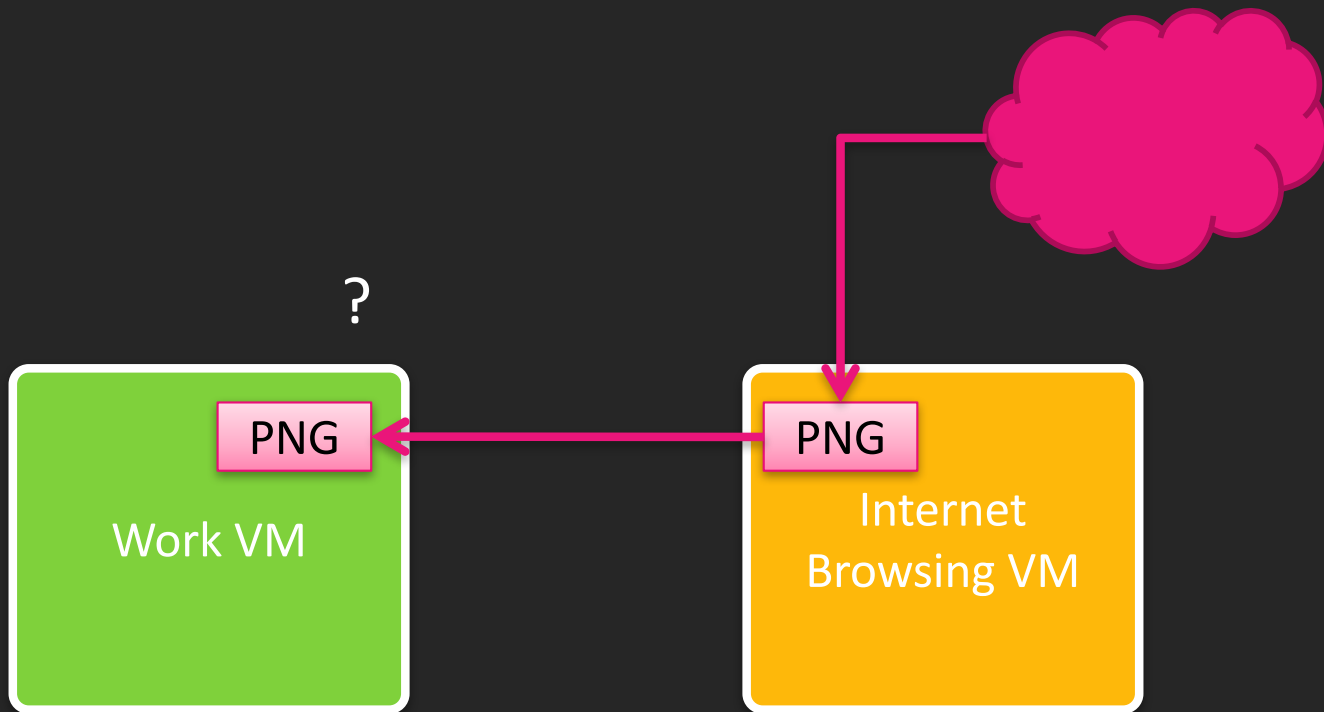
...more useful...



Inter-compartments data transfers

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

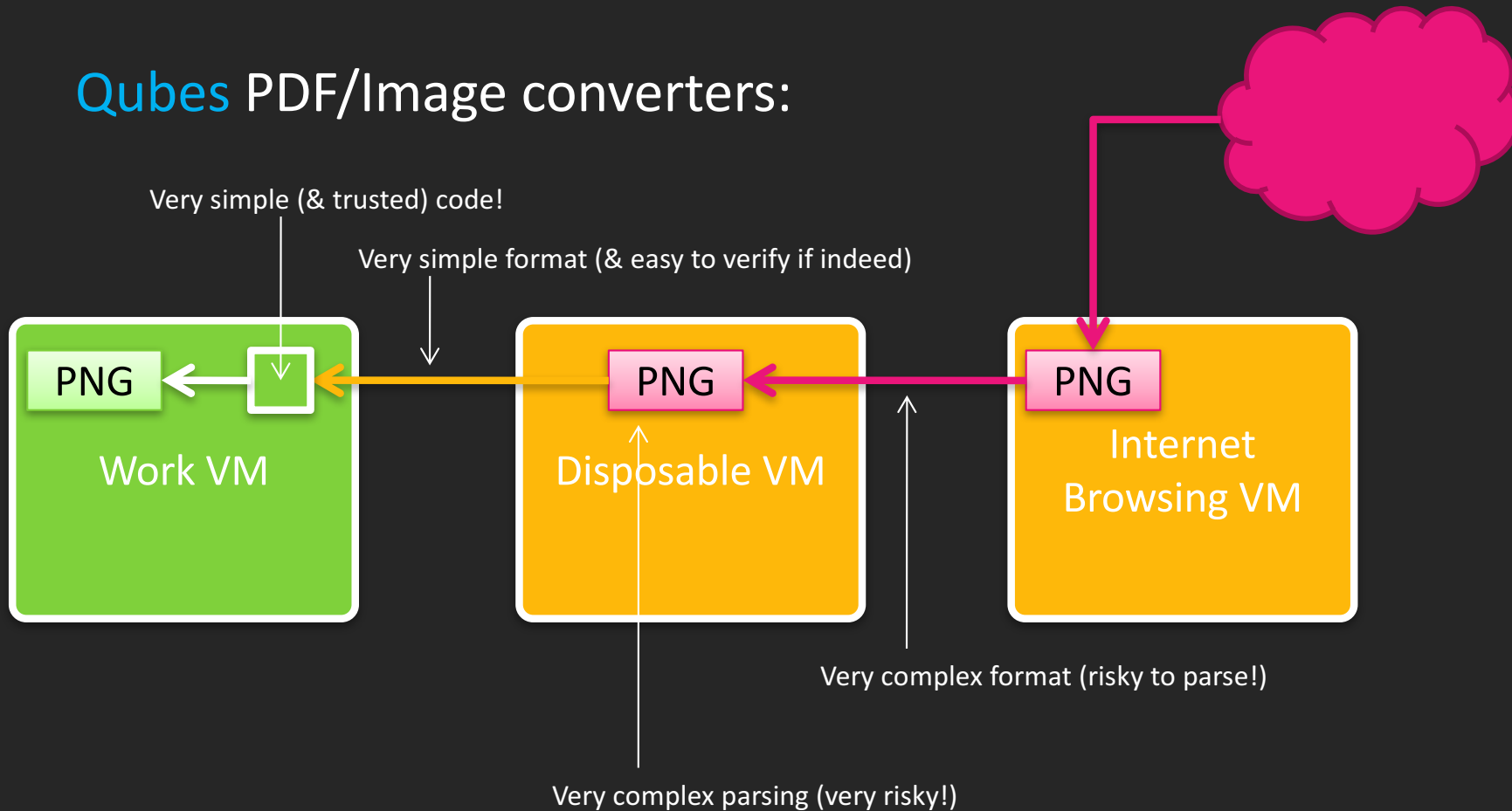




427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Qubes PDF/Image converters:



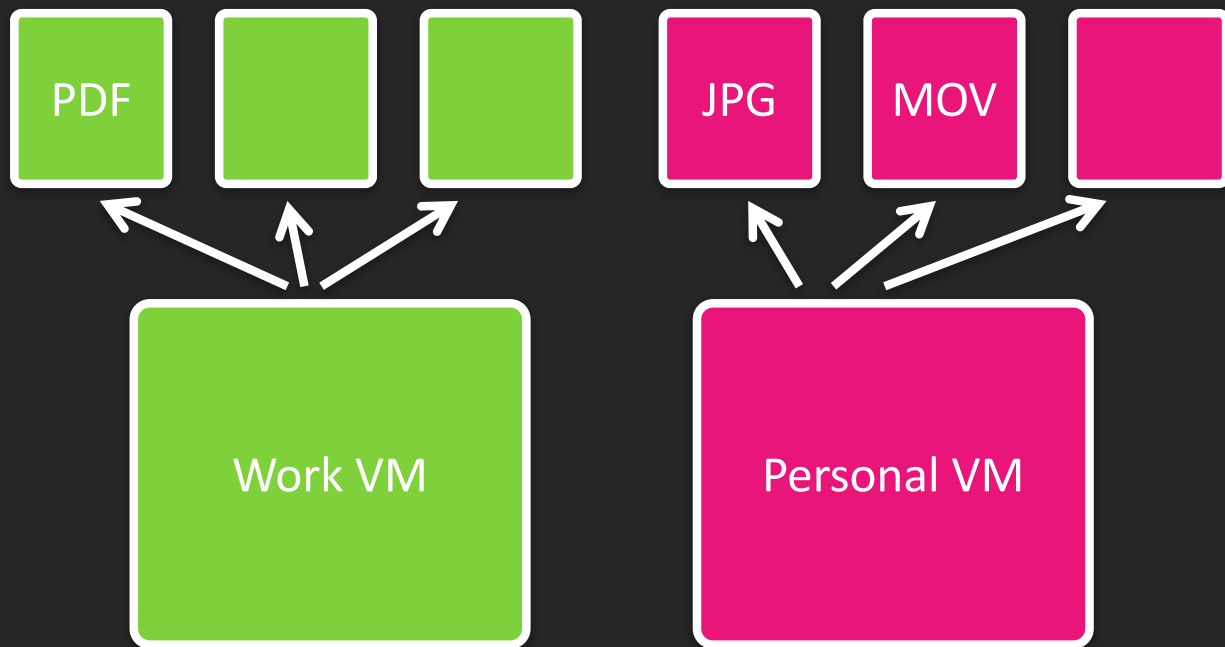
427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



App sandboxing is just part of the story...

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

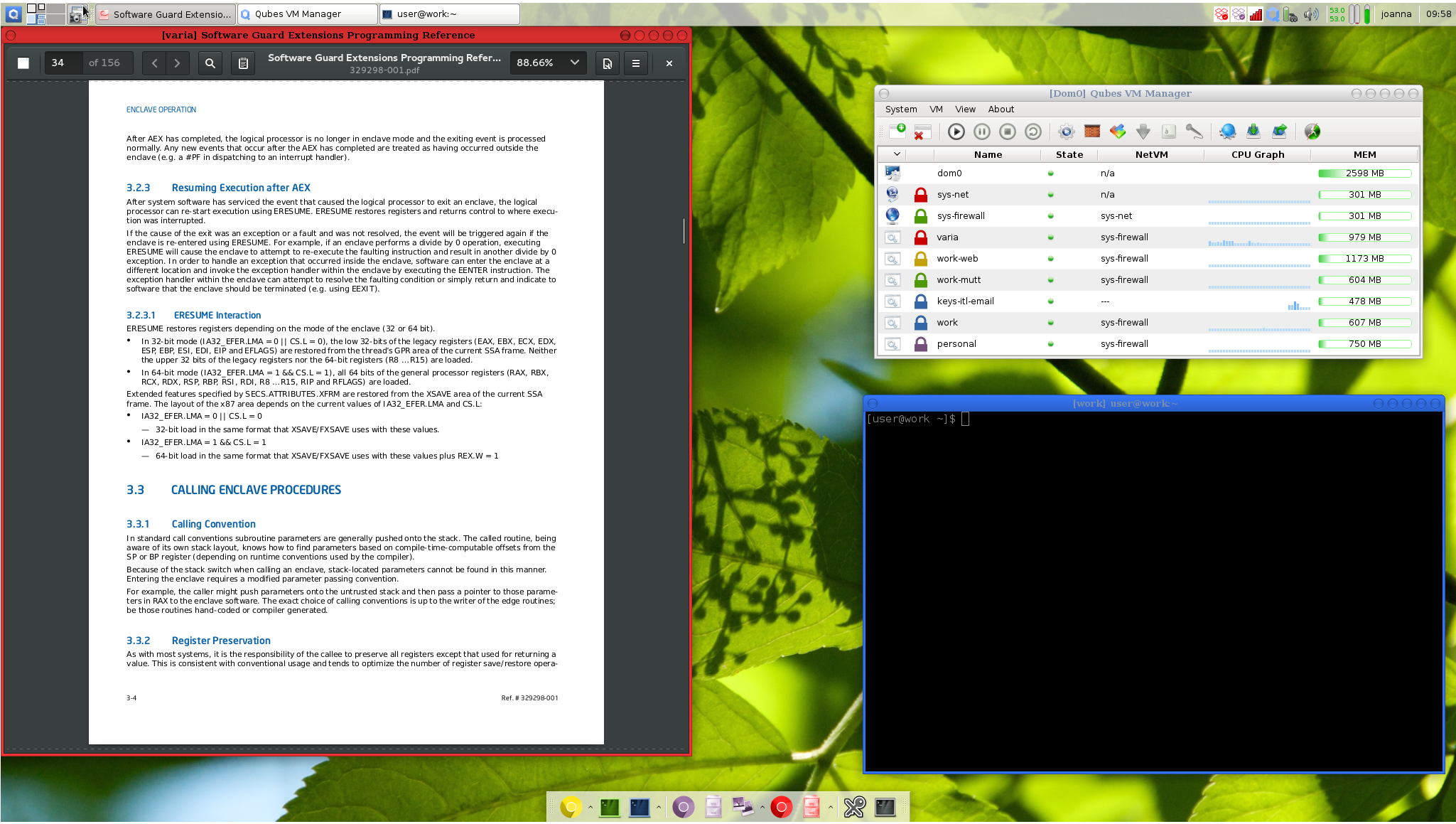




Isolation is just part of the story!

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494





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-

[Dom0] Qubes VM Manager

System VM View About

</

[work] user@work:~

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

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



Example #7: Almighty admins?

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



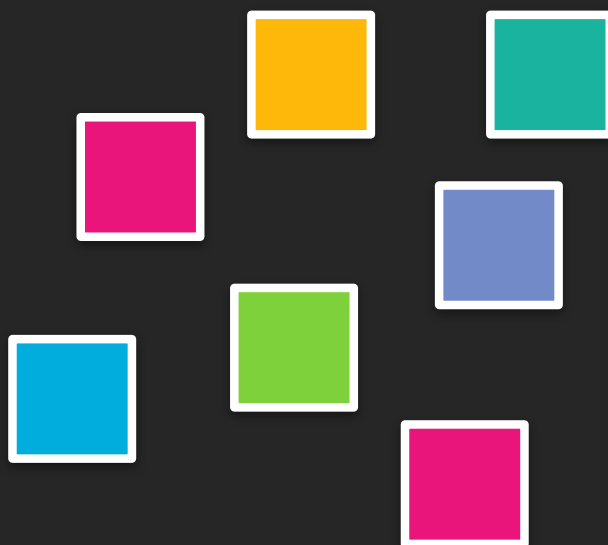
Admins can steal all our data :(

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494





Admin



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



User

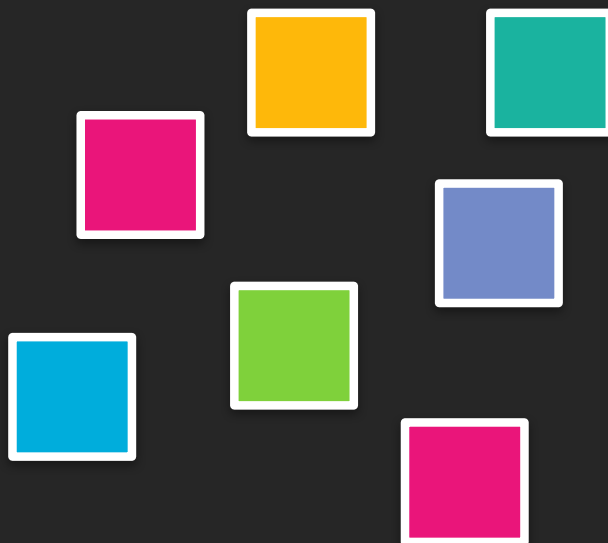




Admin



Can access and do
everything she wants!



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



User

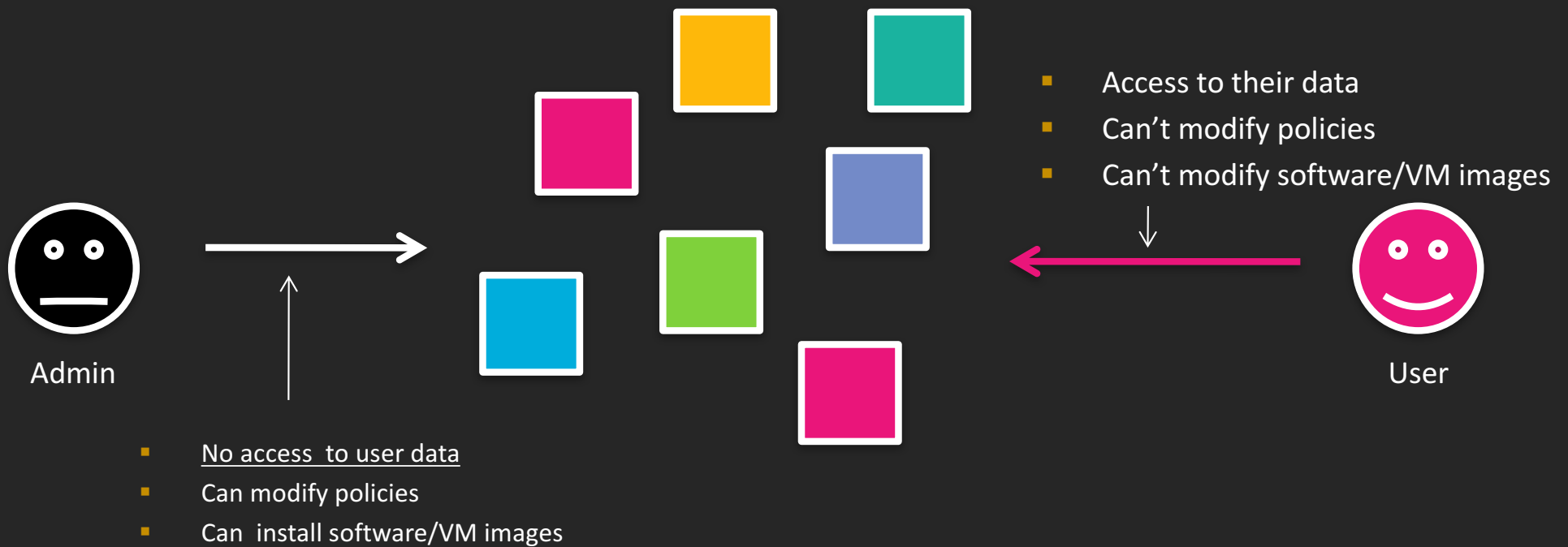


427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494

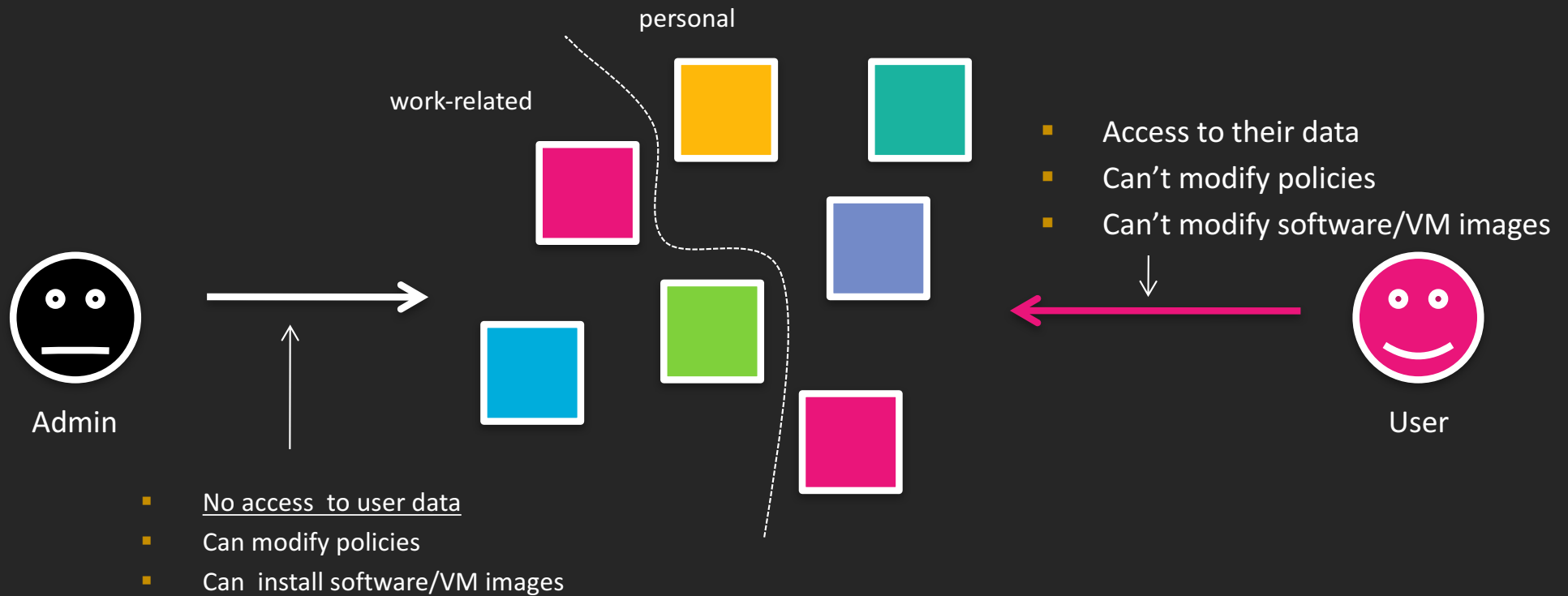


Hmm...

What we want instead:



What we want instead:



Check our [Qubes OS](#) new Admin API for implementation details



Occasionally mishaps happen still...

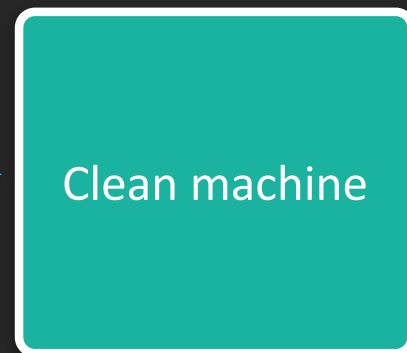
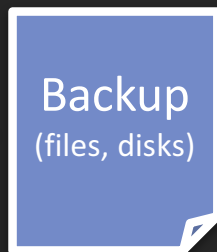
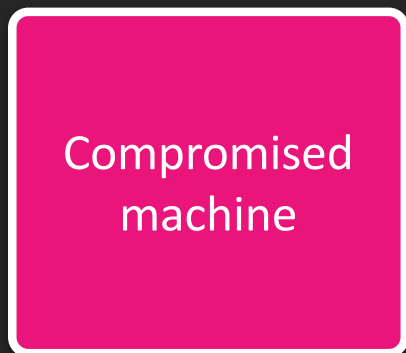
427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



Example #8: Plan B

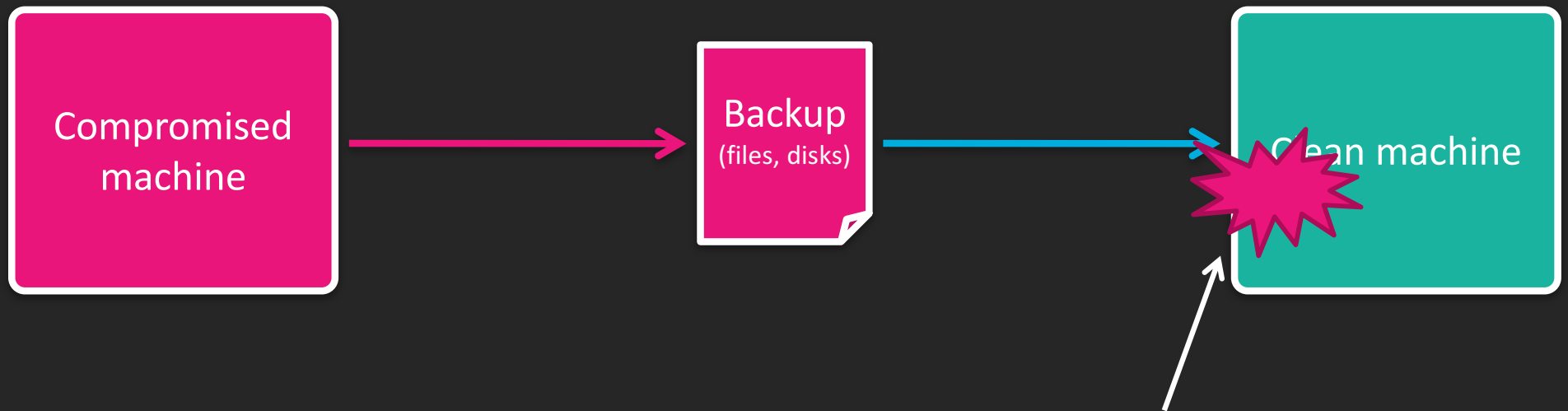
427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494





427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



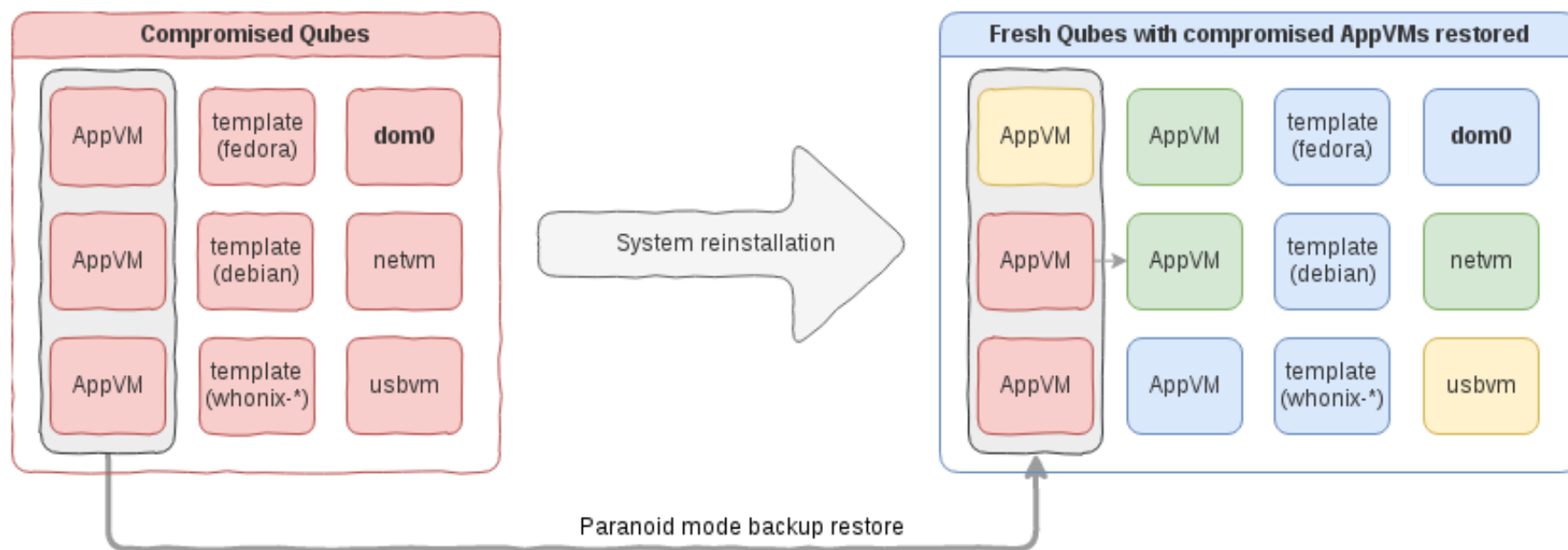


Restoring compromised backup is risky!

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



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)

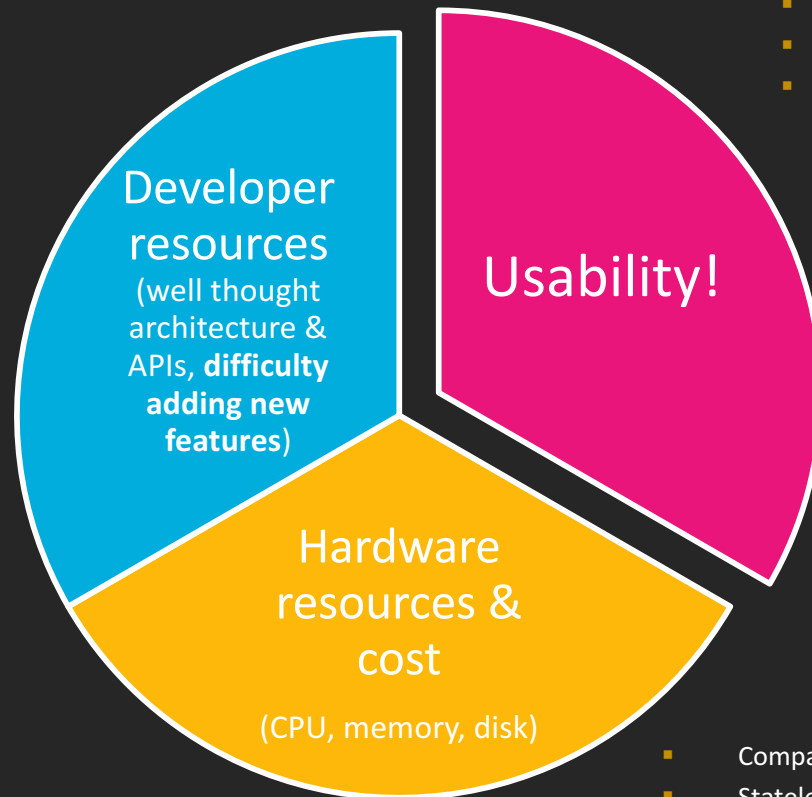


Tradeoffs?

427F11FD 0FAA4B08 0123F01C DDEFA1A3E 36879494



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



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 DDEFA1A3E 36879494

