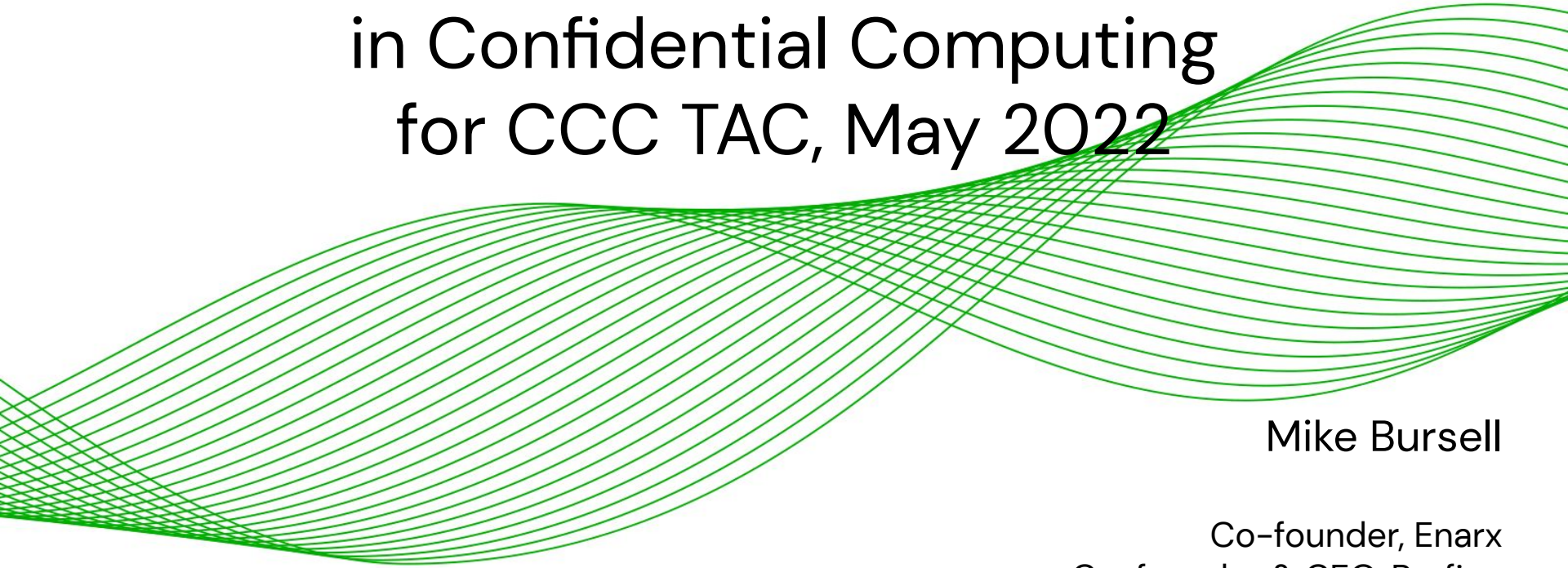
Logging, debugging and error management in Confidential Computing for CCC TAC, May 2022



Mike Bursell

Co-founder, Enarx
Co-founder & CEO, Profian

Agenda

- → Some assumptions
- → Some definitions
- → Lifecycle
- → The problem
- → Logging vs debugging
- → Profiles

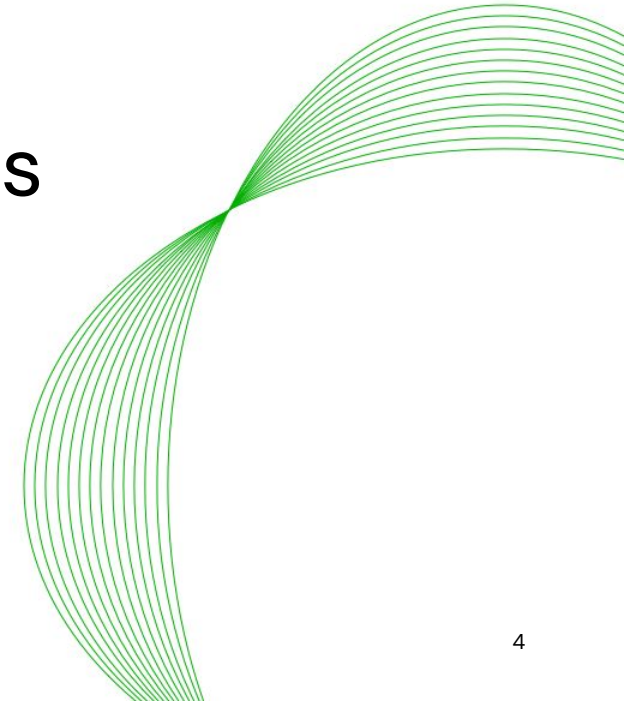
This is an introduction only: for a more detailed treatment, see:

https://blog.enarx.dev/confidential-computing-logging-and-debugging/

Some assumptions

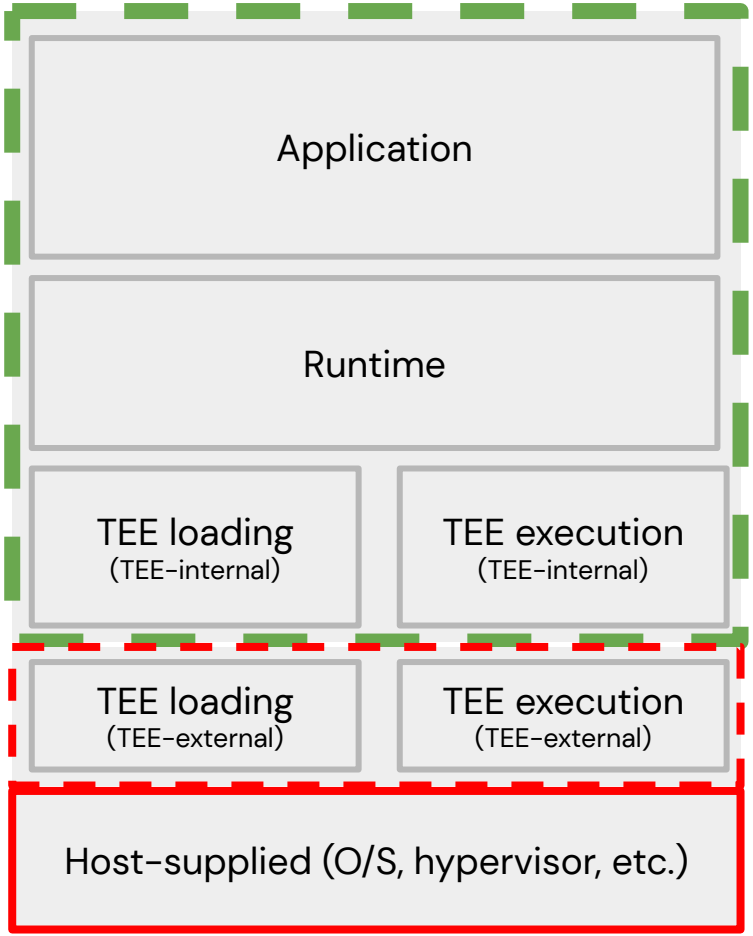
- → We don't trust the host **at all** – it is assumed "malicious"
  - → (except CPU+firmware)
- → Workload (application) and data protection are both important
- → Attestation is out of scope of our conversation
- → We don't write perfect applications first time round
  - → So debugging is important

Some definitions



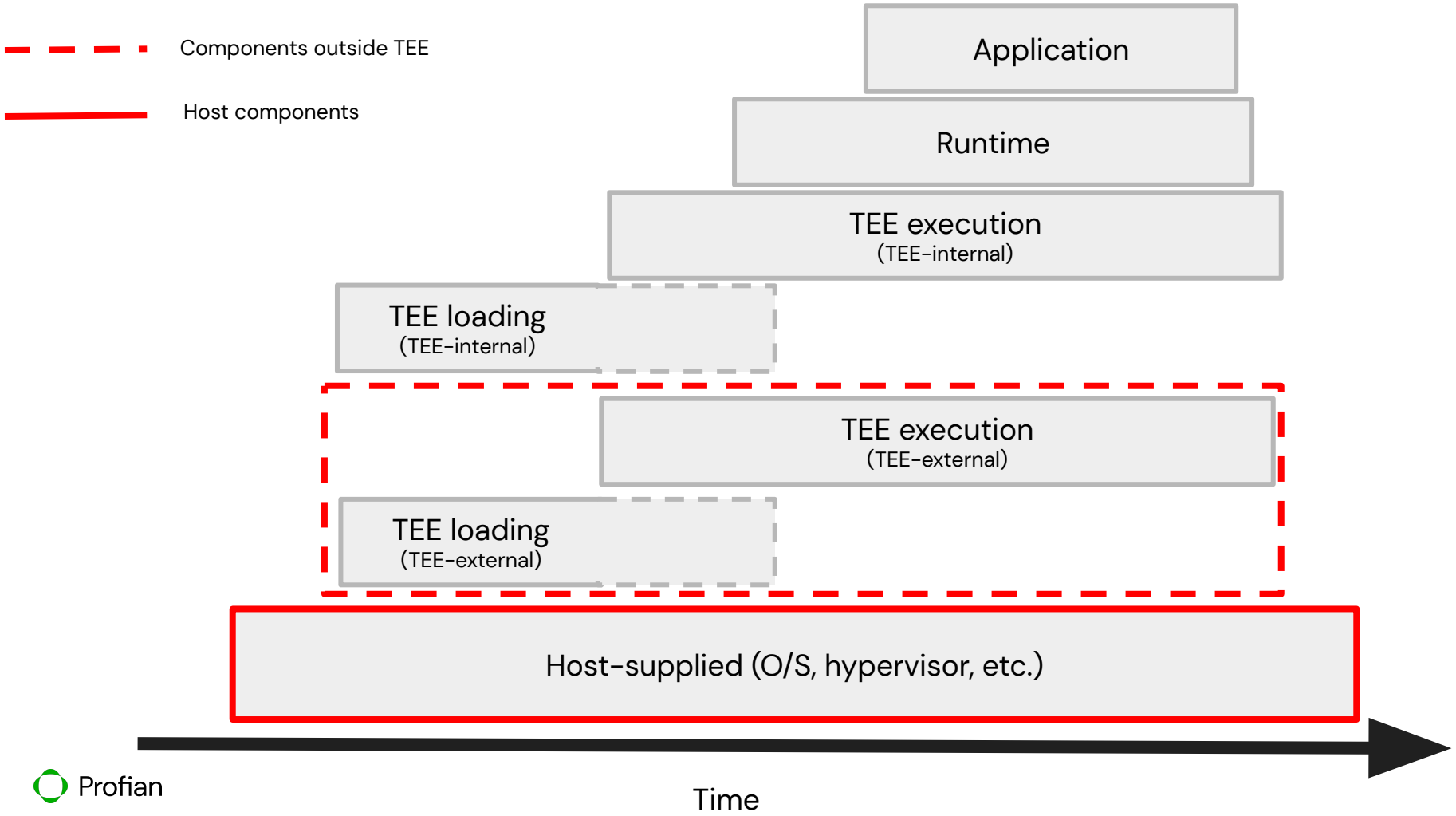
Some definitions

"Client"
external component(s)



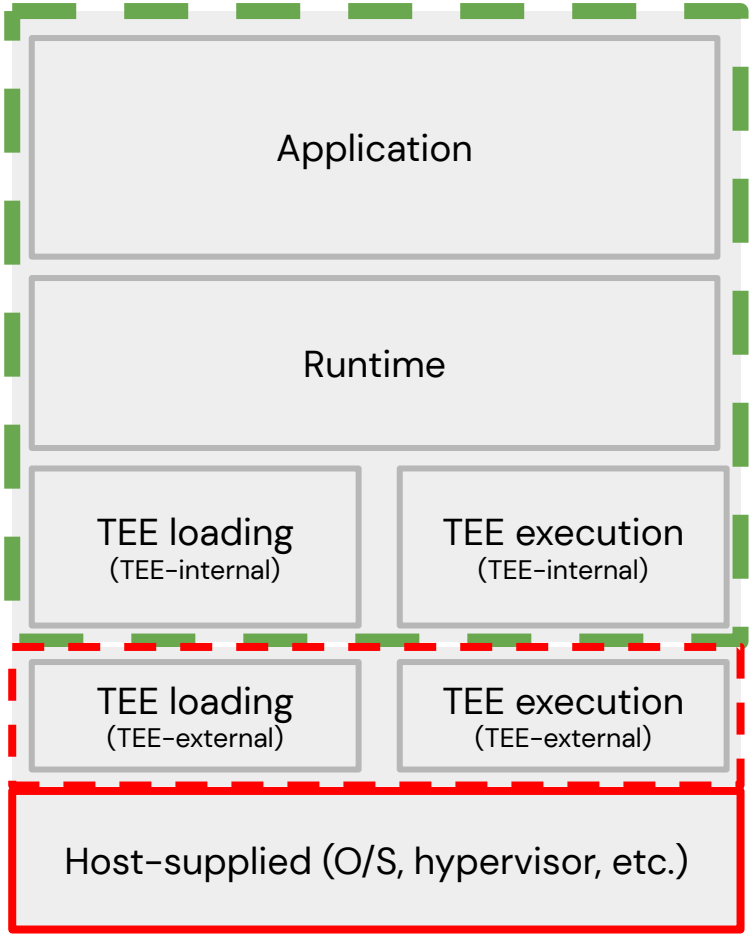
Components within TEE
Components outside TEE
Host components

Profian


Components outside TEE
Host components
Application
Runtime
TEE execution
(TEE-internal)
TEE loading
(TEE-internal)
TEE execution
(TEE-external)
TEE loading
(TEE-external)
Host-supplied (O/S, hypervisor, etc.)
Time
Profian

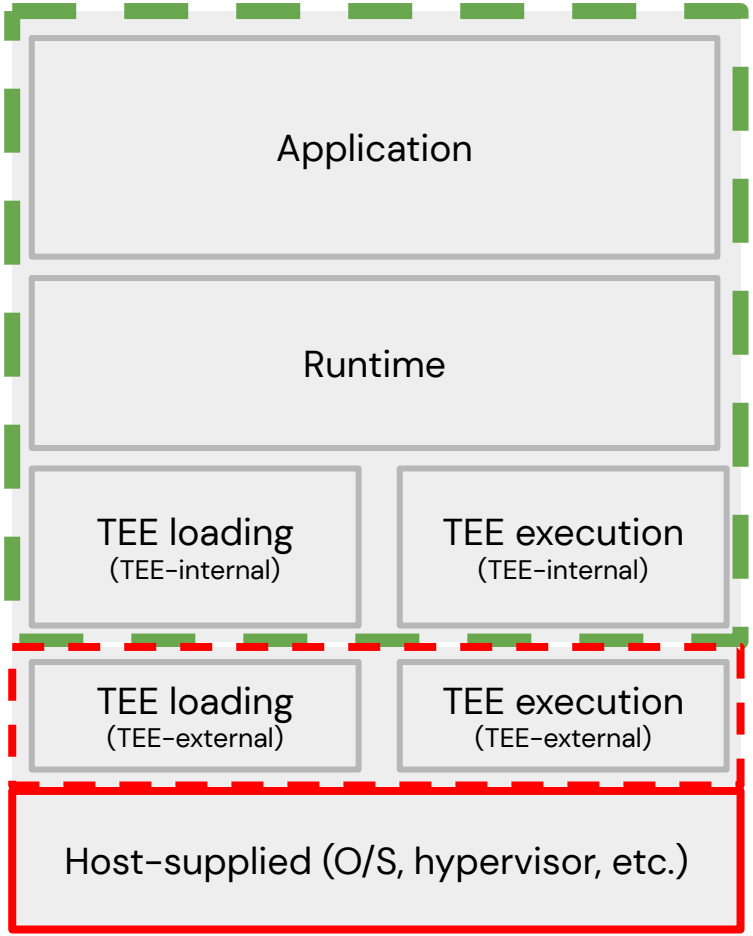
The problem

1. All of the components may need to communicate information
2. Logging data is special
3. Debug data is sometimes needed
4. Untrusted host components can infer information from logging/debugging data

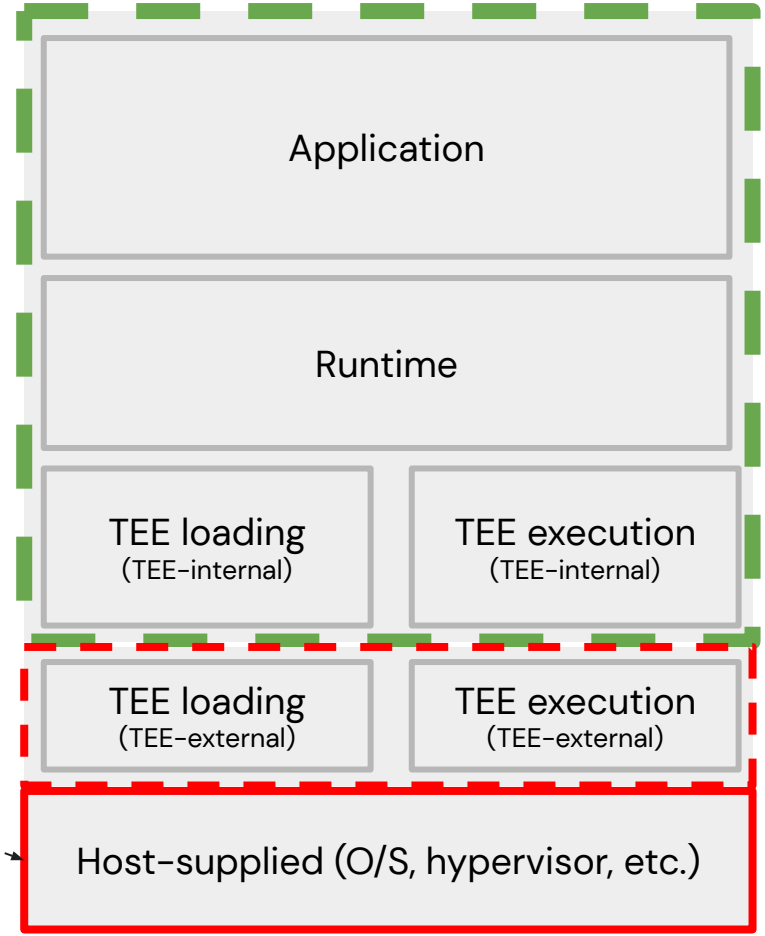


Tools/options

1. Minimising messages
2. Restricting cross-TEE boundary messages
3. Encryption (for integrity/confidentiality)
4. Hashing/signing (for integrity)

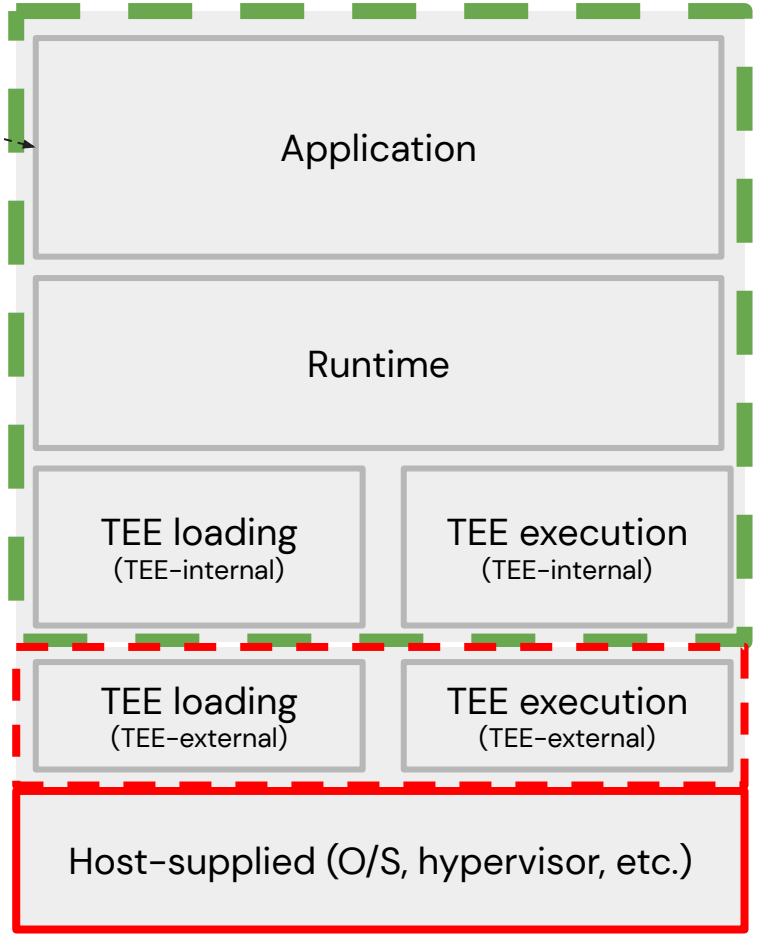


Issues by component

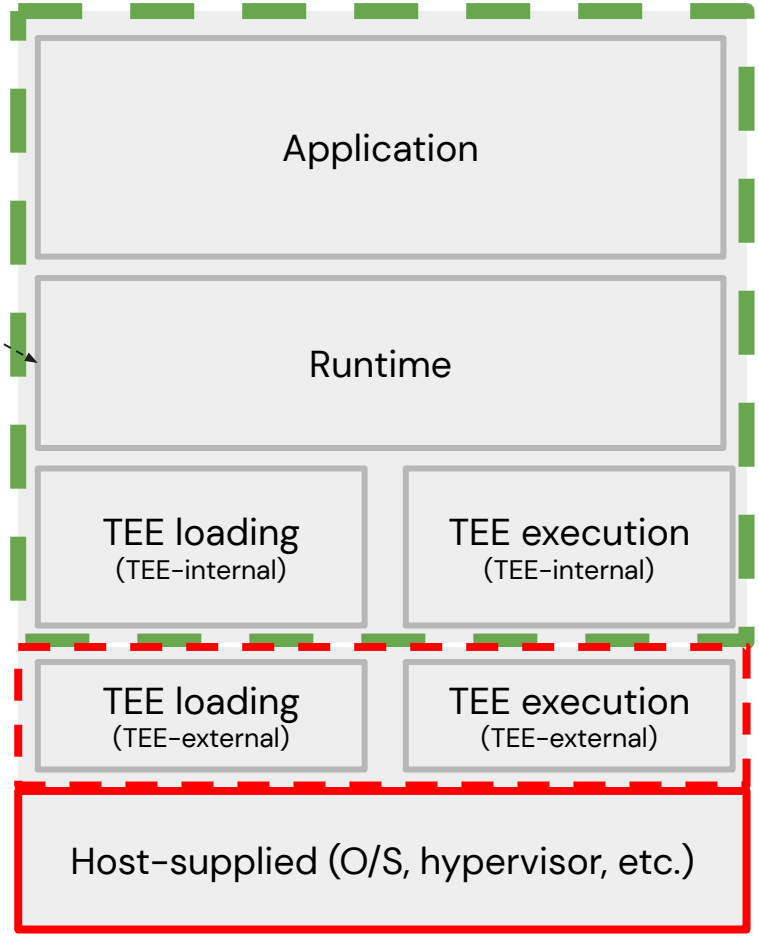


- Can change and interfere with all logging and error messages to which it has access
- May use them to infer information about the workload (application and associated data)

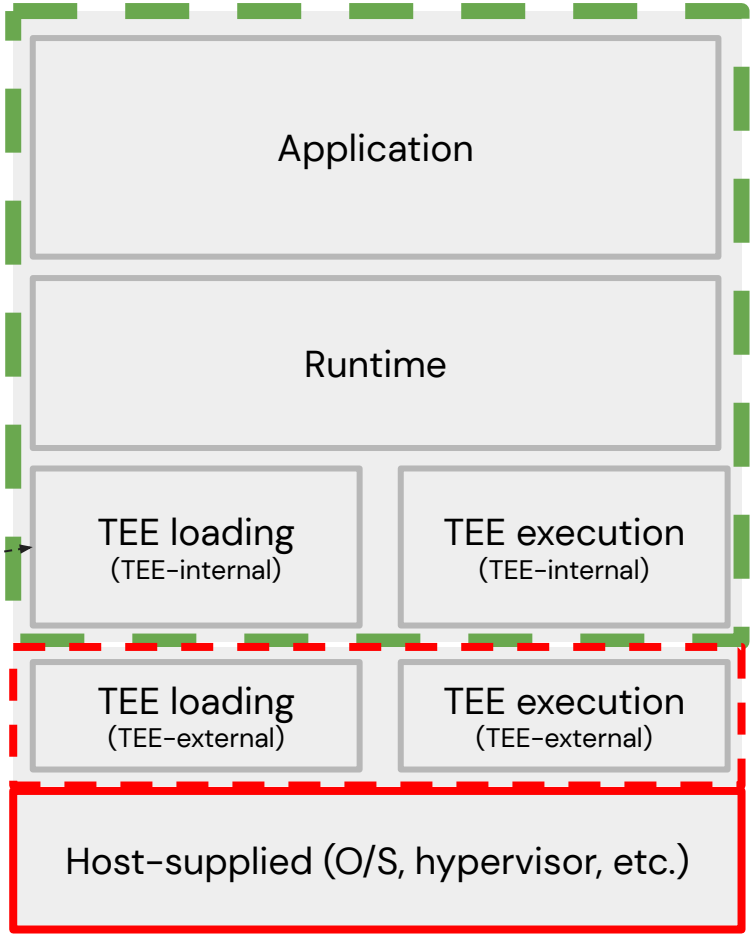
- Can communicate messages over data layer to client (assuming networking + keying)
- May choose to provide error messages to Runtime



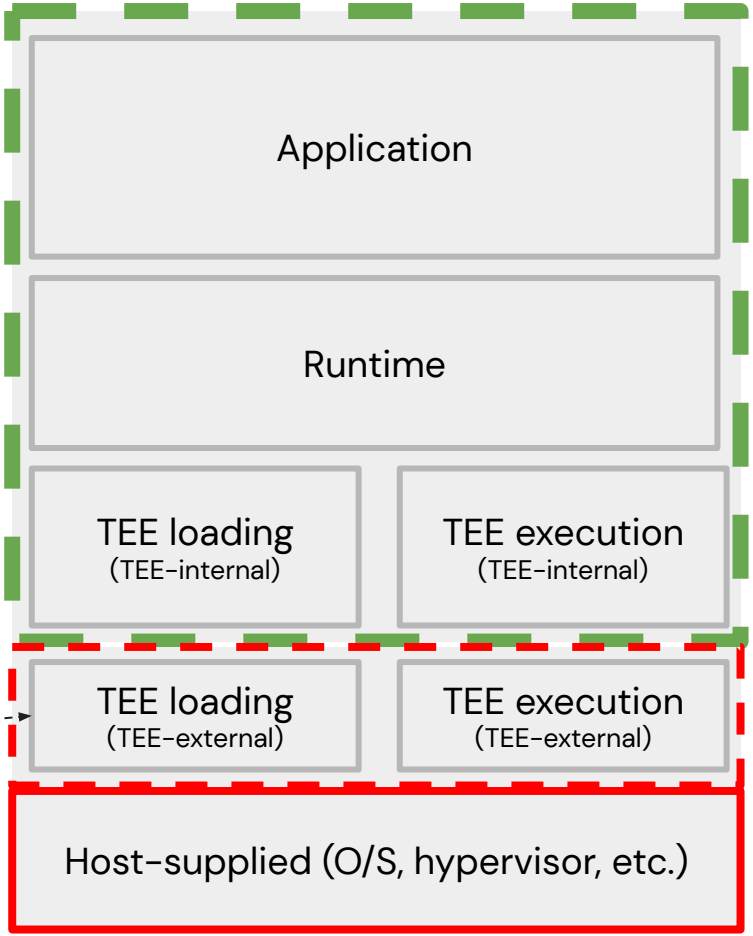
- May be able to communicate to client via control plane
- May have access to application-sourced messages
- Should generally not report information to host components (safer to use control plane)



- May be sensitive to forced error attacks
- Can pass errors to Runtime on event of eventual success
- May encrypt, hash and/or sign messages to be passed via host to external components

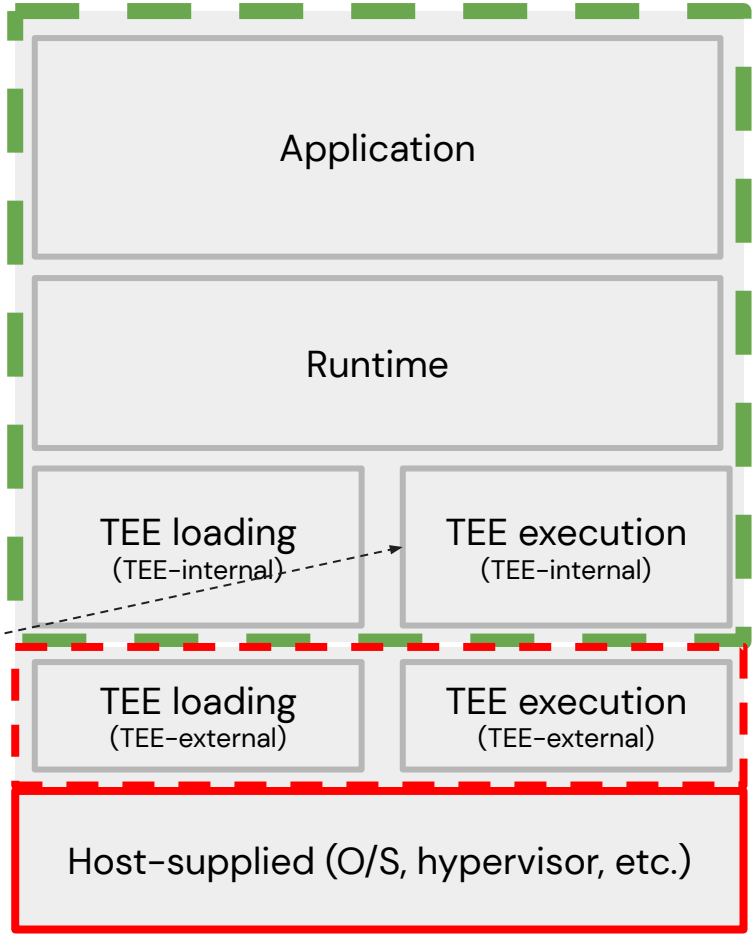


- Subject to malicious manipulation
- May choose to store or transmit to:
  - Host
  - TEE loading (TEE-internal)
  - TEE execution (TEE-internal)
  - TEE execution (TEE-external)
  - Client

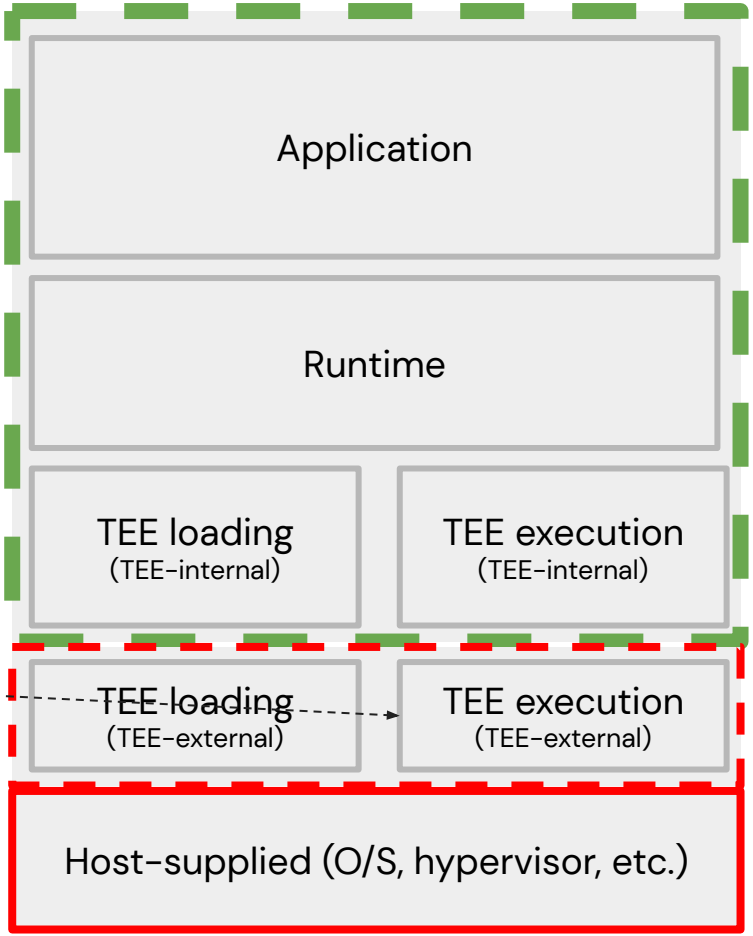


Profian

- Can communicate via:
  - Application (for information, or transmission via) data plane
  - Runtime (for transmission via control plane)
  - TEE runtime external component
- Must restrict communication with TEE external component to standardised messages



- Similar to TEE loading (TEE-external)
- Sensitive to manipulation of messages (e.g. syscalls) from host component



Logging vs debugging

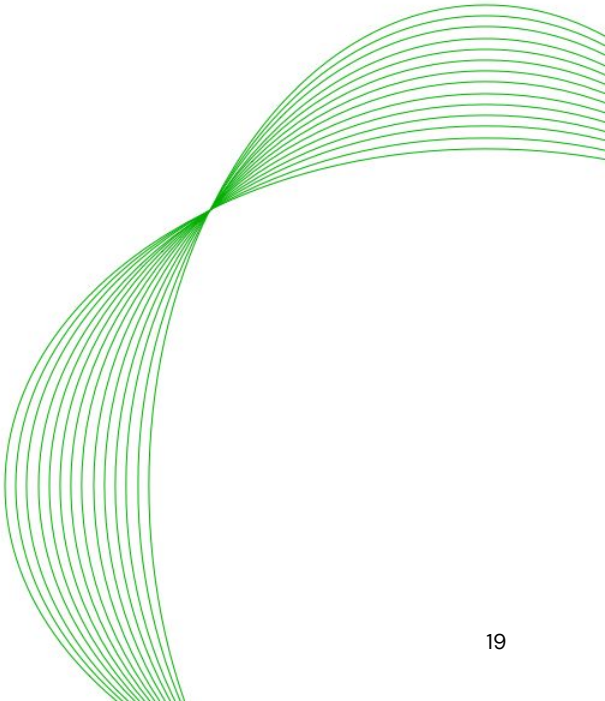
Why make a distinction?

Because a environment which supports both logging and debugging must:

- → Be clear about state
  - → **Where** is it clear? To what entity? Over what channel? With what authority?
- → Not be able to move (or be moved) to a less secure state

Life would be much easier if we could ignore debugging, but that's unrealistic.

Profiles



Conclusion

Confidential Computing environments MUST:

- → Always consider the host malicious
- → Support different runtime profiles
  - → These may be broader than logging
  - → But logging must be part of them
- → Minimise information flow between/outside components
- → Consider leakage via logging/debugging as part of threat models

Confidential Computing environments SHOULD

- → Restrict messages to defined set, with no “plain text” fields
- → provide hooks to help manage profiles through (organisational) processes.