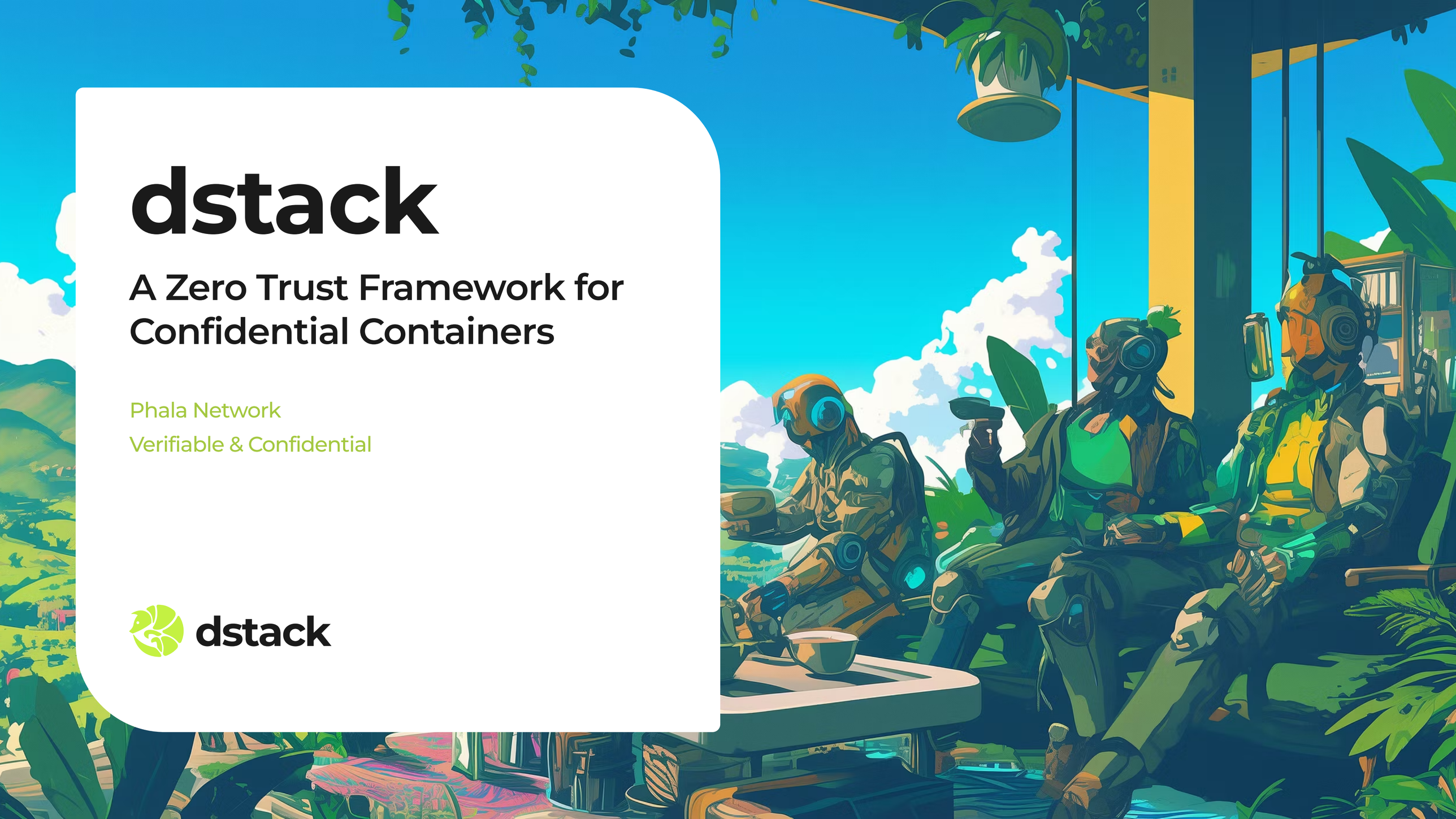
dstack

A Zero Trust Framework for Confidential Containers

Phala Network

Verifiable & Confidential



dstack

Zero Trust Problem

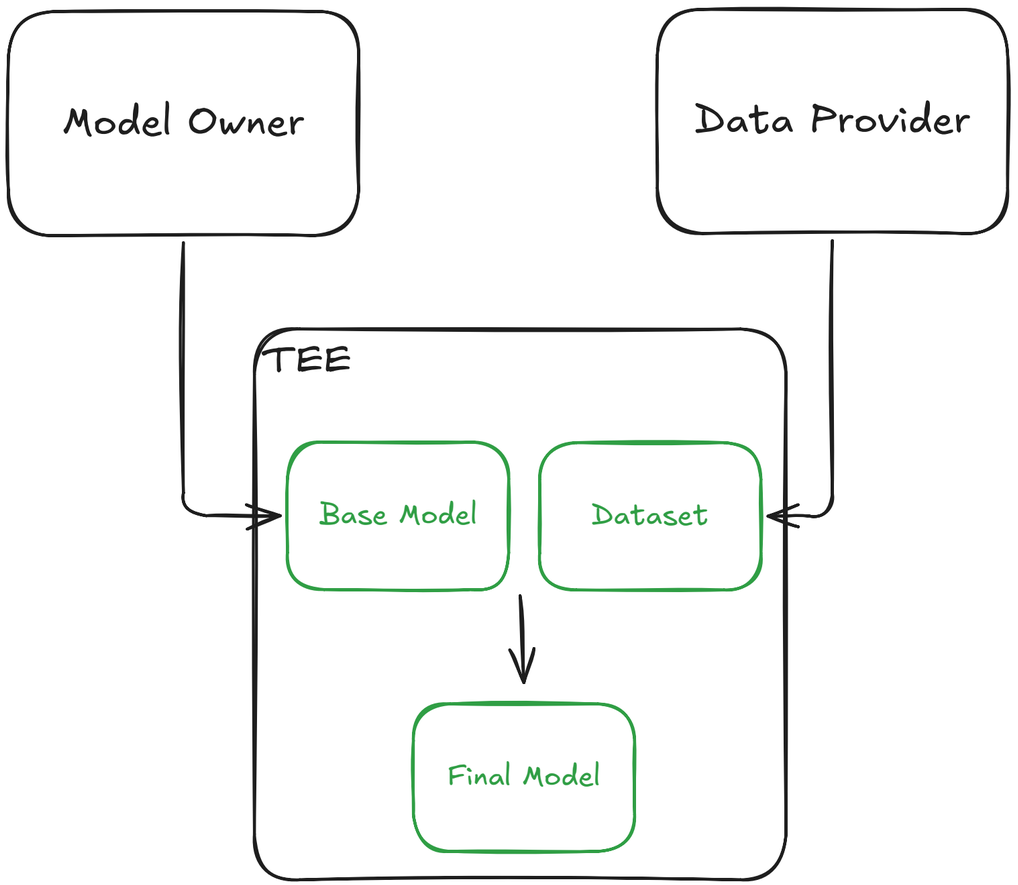
where nobody trusts each other

Requirements

Model parameters must remain confidential to Data Provider

Training data must remain confidential to Model Owner

Security guarantees must be verifiable by both parties



Zero Trust Problem

where nobody trusts each other

Threat Model

Model Owner is not trusted

TEE vulnerabilities - hardware attacks, side channels, e.t.c.

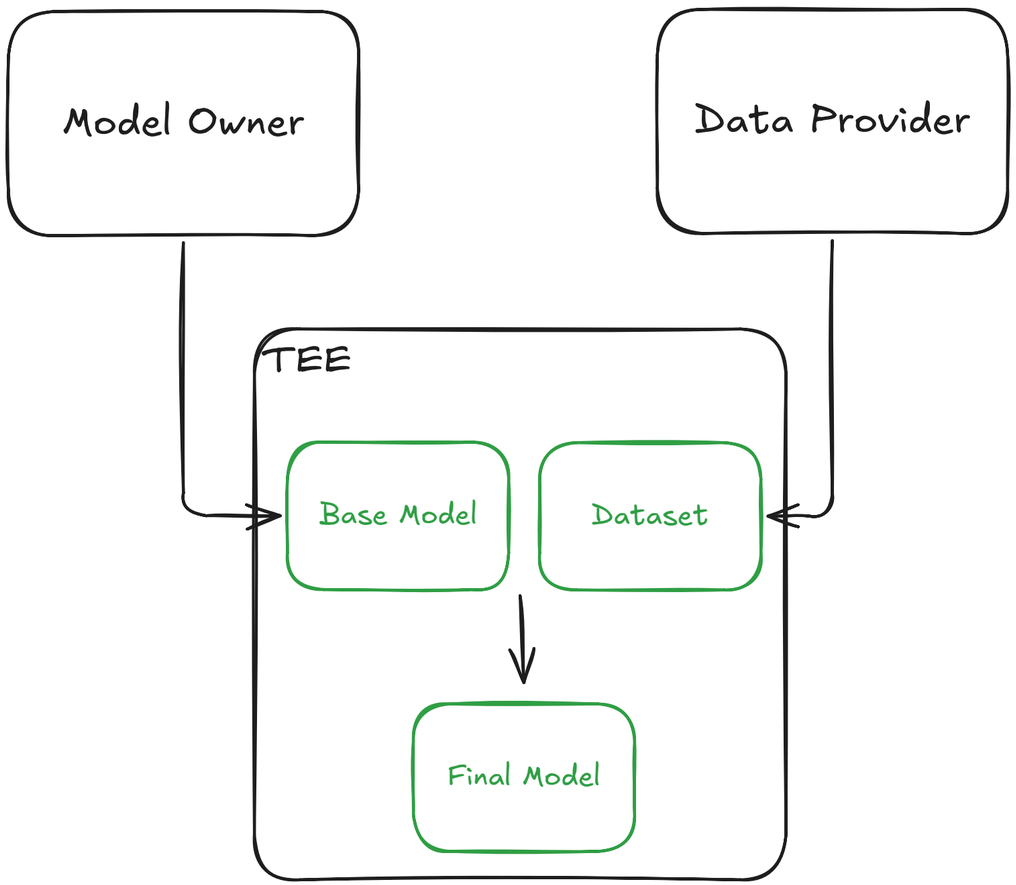
Single point of failure - TEE can be down

What are missing in the current TEE solutions

❌ **Source Code Transparency**

❌ **Robust Key Management**

❌ **Continuous Security Assurance**



TEE Vulnerabilities & Availability Gaps

Limitations of TEE

Known vulnerabilities compromise TEE confidentiality and integrity

Inconsistent key management across different TEE technologies

TEE shutdown = data loss: No cross-hardware recovery mechanism

TEE Solution	Key for Apps
Intel SGX	Hardware-bound keys
Intel TDX	No built-in API
AMD-SEV	No built-in API

TEE Vulnerabilities & Availability Gaps

The Recovery Problem

Core challenge: Restoring encrypted program state and data across different TEEs
Data backup is easy - **Key management is hard**
Need: Secure key recovery mechanism independent of specific hardware

❌ **Extra Cost**: Developers need to handle the encrypted data themselves, including **backup**, **key management** and **migration**.

TEE Solution	Key for Apps
Intel SGX	Hardware-bound keys
Intel TDX	No built-in API
AMD-SEV	No built-in API

Limited Verifiability

What can users learn from such a report?

❌ **Is this really from the app I am interacting with?**
❌ **Does it has backdoors in the code?**
❌ **How will it process my data?**

❌ **No Benefits**: TEE fails to bring **perceivable security and verifiability** to users.

Body Details

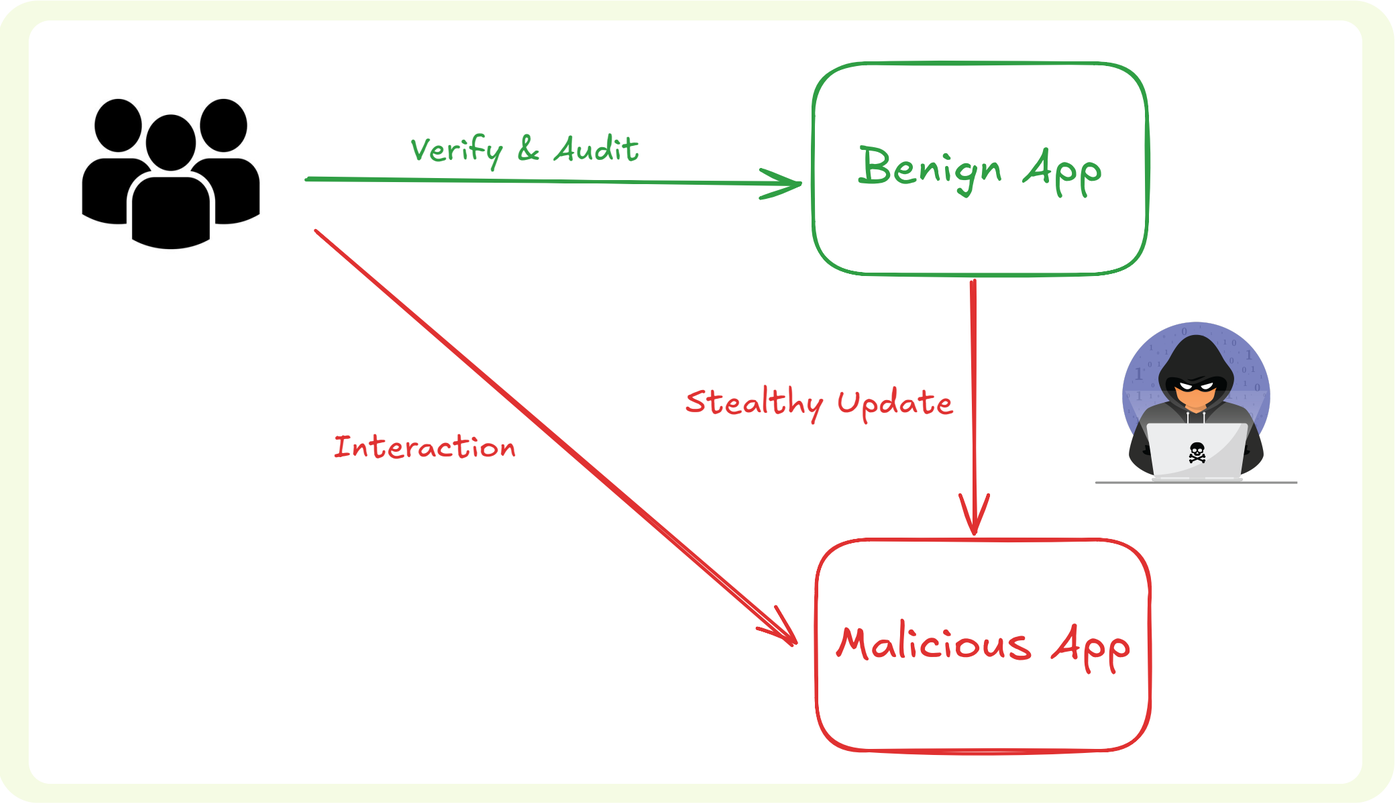
Attribute	Value
tee_tcb_svn	06010300000000000000000000000000
mrseam	0x5b38e33a6487958b72c3c12a938eaa5e3fd4510c51aeeab58c7d5ecee41d7c436489d6c8e4f92f160b7cad34207b00c1
mrsignerseam	0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
seamattributes	0x0000000000000000
tdattributes	0x0000001000000000
xfam	0xe702060000000000
mrtd	0xc68518a0ebb42136c12b2275164f8c72f25fa9a34392228687ed6e9caeb9c0f1dbd895e9cf475121c029dc47e70e91fd
mrconfig	0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
mrowner	0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
mrownerconfig	0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
rtmr0	0x0bb3d6375f94482cdd24b767e4a0d720348527c4f2ab433d77f842b9394fa1638bb6df83fb0a1301f29c71bf60da48bb
rtmr1	0x154e08f5c1f7b1fce4cbfe1c14f3ba67b70044ede2751487279cd1f2e4239dee99a6d45e24ebde6b6a6f5ae49878e0e6
rtmr2	0x9edcd363660e85b71c318324996dda756c372d9f6960edbfa863b1e684822eb48dd95e218ae2b78e51ef97f3b8f5c9dc
rtmr3	0x875ee0bf8b9fb2038f4a12f2dcdbe9b20d173acbfe15a2527009c9b259f8407983e96c1a1f7546b0066d0cbe5fb2f9af
reportdata	0x0000000000000000000000000000000000000000000000000000000000000001f7ca384f9290e470cd729dec852b5dbcea4f1317103e86f0e73d2f78fa0cf4c

Beyond Current Trust Models: Addressing Malicious Administrators

Trust Model Evolved

There should not be "Administrators" any more
A **transparency log** of all application lifecycle changes is needed

❌ If the administrators can update the code **without being noticed**, the protection from TEE becomes meaningless.



Technical Challenges

Security Reliability: TEE vulnerabilities

Lack of Key Management creates Single Point of Failure

Unrestricted Application Lifecycle Control

Attestation report with no context

Target

A Zero Trust Platform

Workaround for Attacks

Data & Service Availability

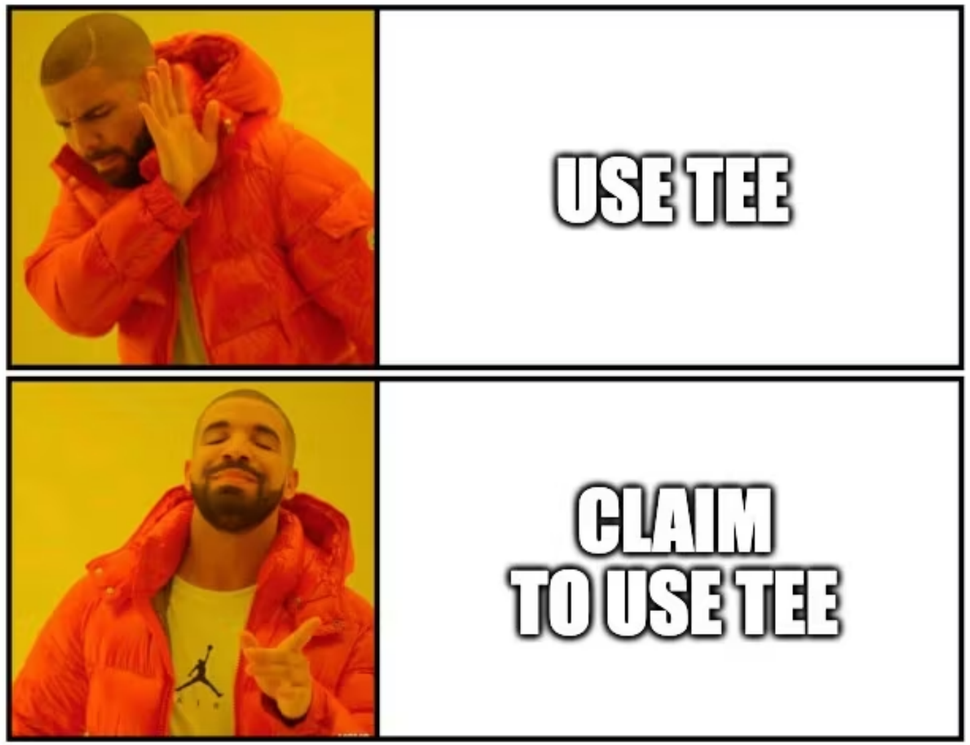
Code is Law

Fully Verifiable Chain-of-Trust

Zero Trust should come with Benefits and No Cost

What prevents developers from using TEE?

- Concerns about vulnerabilities and vendor lock-in
- Programming complexity: don't want to rewrite code
- Performance limitations and overhead
- TEE is invisible to customers



Dstack Innovations

Portable Confidential Container

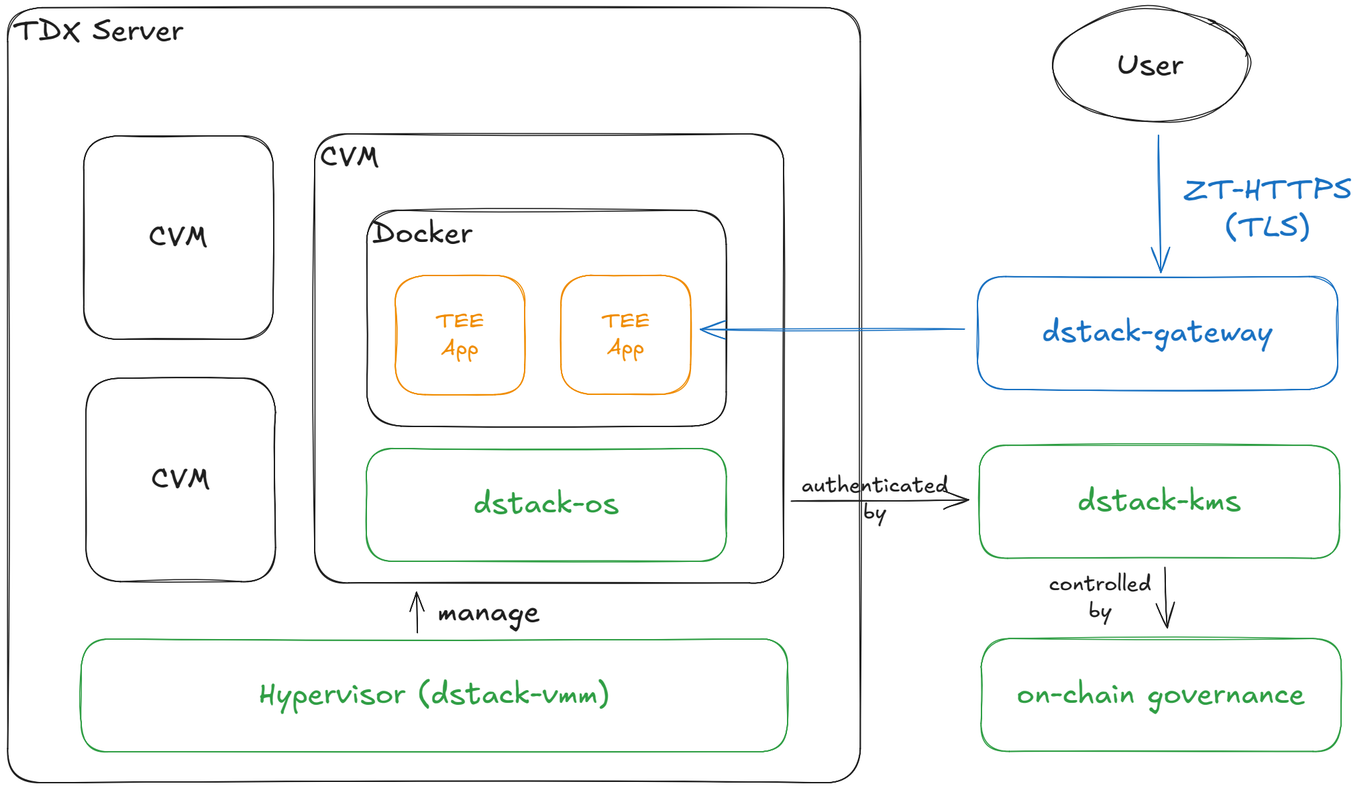
Stateful CVM can be migrated between different TEE instances

Code Management with Distributed Ledger

Transparent logs for all the application lifecycle events

Full Chain of Trust

Verifiable Domain Management with domain and certificate exclusively controlled by TEE
Source Code Verifiability for the whole system image



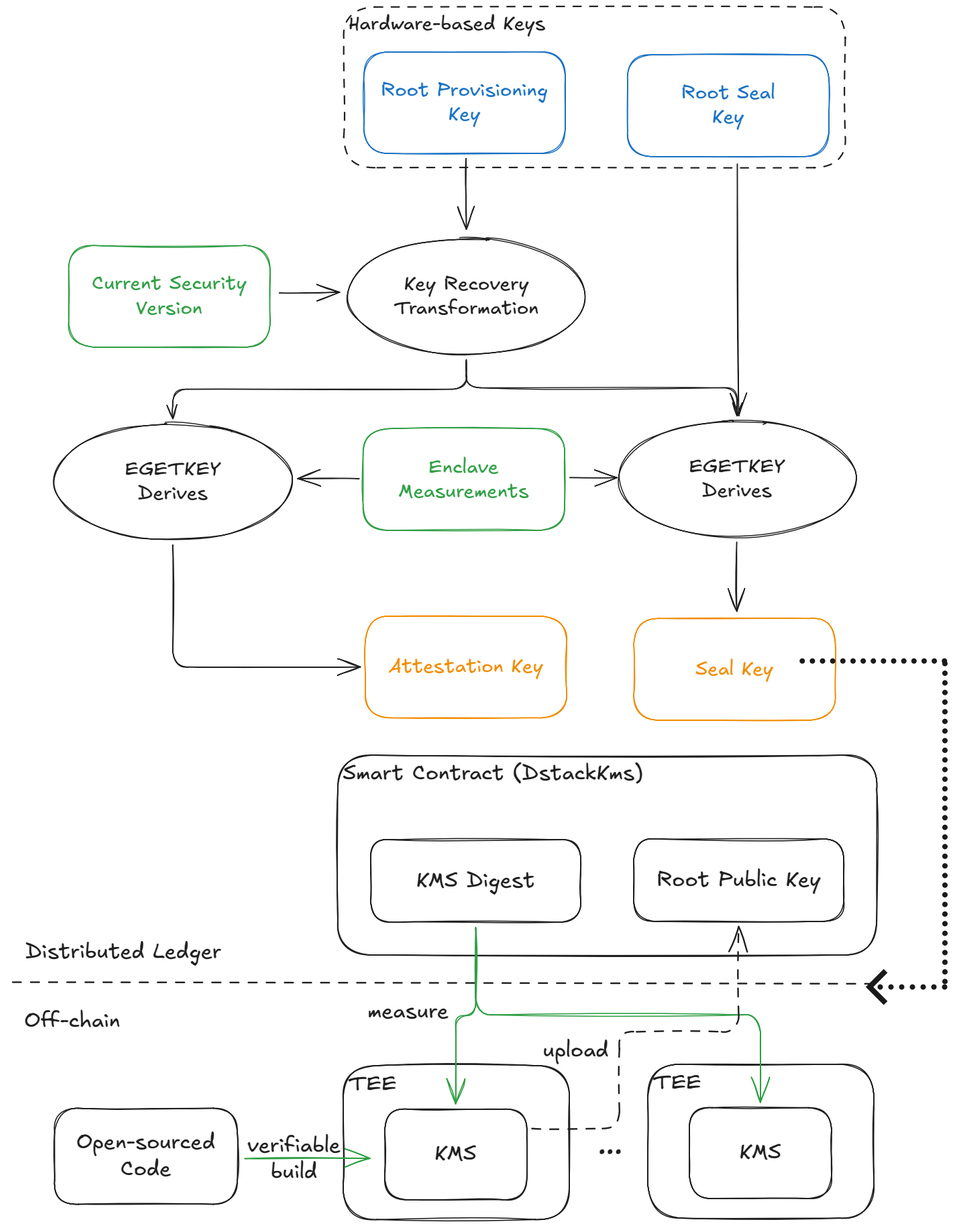
Dstack brings docker support to TEE CPU and GPU with features above

Portable Confidential Container

```
// The kms public RPC service.
service KMS {
  // Request the app key given the app id and tdx quote
  rpc GetAppKey(GetAppKeyRequest) returns (AppKeyResponse);
  // KMS key handover
  rpc GetKmsKey(GetKmsKeyRequest) returns (KmsKeyResponse);
  // Request the app environment encryption public key given the app id
  rpc GetAppEnvEncryptPubKey(AppId) returns (PublicKeyResponse);
  // Request the KMS instance metadata for use as a probe and health check.
  rpc GetMeta(google.protobuf.Empty) returns (GetMetaResponse);
  // Request the temporary CA certificate and key
  rpc GetTempCaCert(google.protobuf.Empty) returns (GetTempCaCertResponse);
  // Sign a certificate
  rpc SignCert(SignCertRequest) returns (SignCertResponse);
  // Clear the image cache
  rpc ClearImageCache(ClearImageCacheRequest) returns (google.protobuf.Empty);
}
```

Generate Secret Keys with Independent Key Management Service

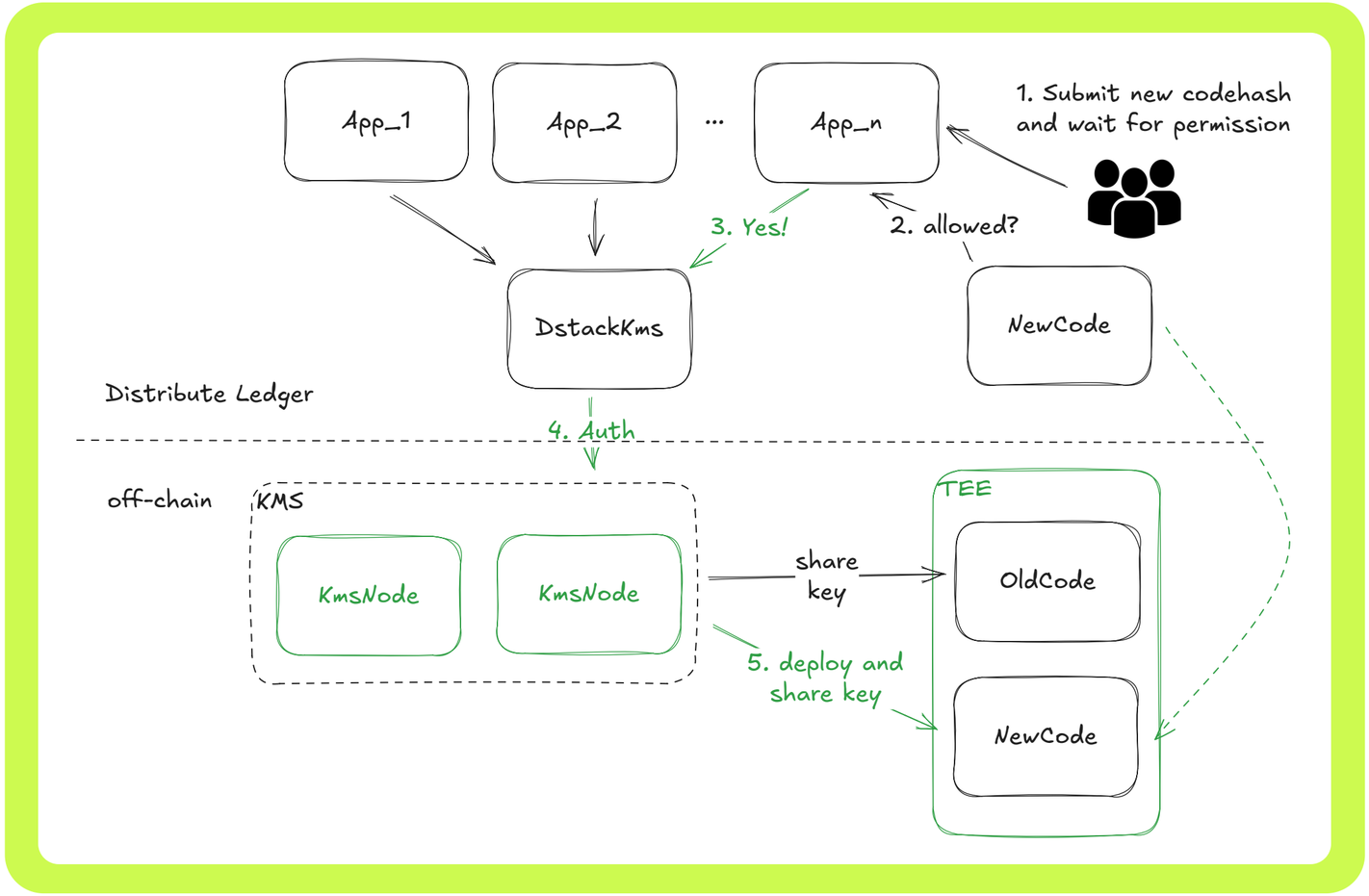
The KMS becomes the new Root-of-Trust of the whole system
By default, KMS is powered by a P2P TEE network



Decentralized Code Management

Customizable management enforced through KMS

Without KMS approval, the TEE hardware will not be about to decrypt all its data



Verifiable Domain Management

with TEE exclusively controls a domain

TEE certificate generation

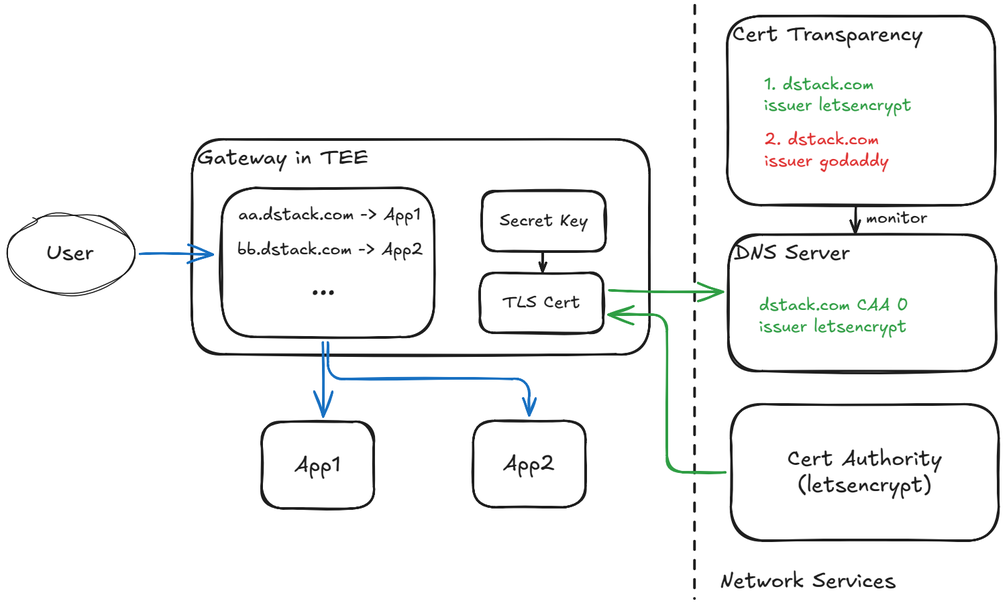
Attestation report will be generate to show an account_uri is bound to **TEE-controlled key**

Ownership monitoring with CAA records and CT

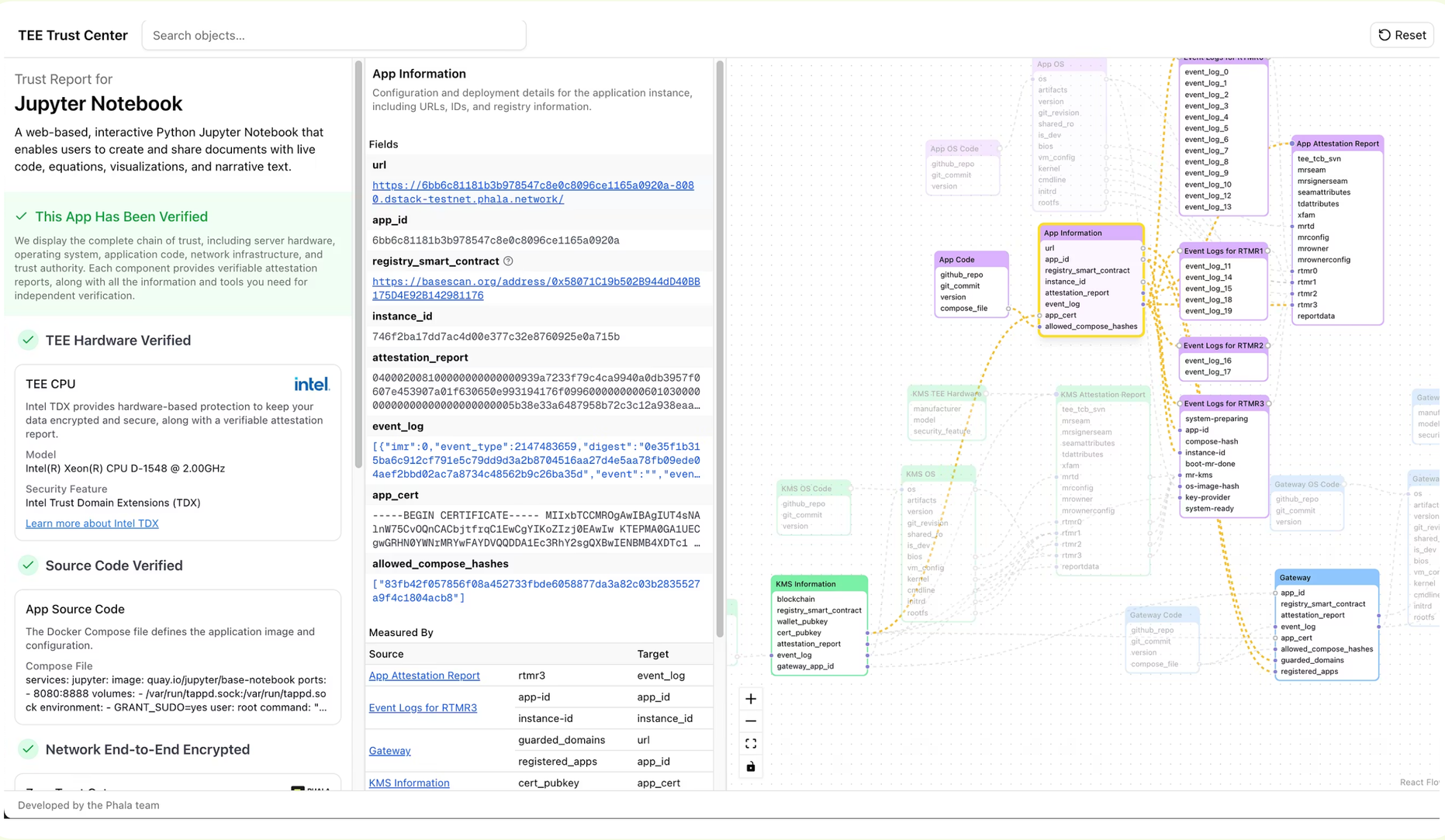
Certificate Authority Authorization restricts **which certificate authorities** can issue certificates for managed domains

Certificate Transparency logs provide an **immutable audit trail of certificate issuance history**

crt.sh ID	Logged At	Not Before	Not After	Common Name	Matching Identities	Issuer Name
18790913612	2025-06-03	2025-06-03	2025-09-01	*.dstack-testnet.phala.network	*.dstack-testnet.phala.network	C=US, O=Let's Encrypt, CN=E6
18790970093	2025-06-03	2025-06-03	2025-09-01	*.dstack-testnet.phala.network	*.dstack-testnet.phala.network	C=US, O=Let's Encrypt, CN=E6
18719669040	2025-05-30	2025-05-30	2025-08-28	*.dstack-testnet.phala.network	*.dstack-testnet.phala.network	C=US, O=Let's Encrypt, CN=E6
18719668685	2025-05-30	2025-05-30	2025-08-28	*.dstack-testnet.phala.network	*.dstack-testnet.phala.network	C=US, O=Let's Encrypt, CN=E6



Walkthrough of Chain-of-Trust



Audit of dstack Components

Scope

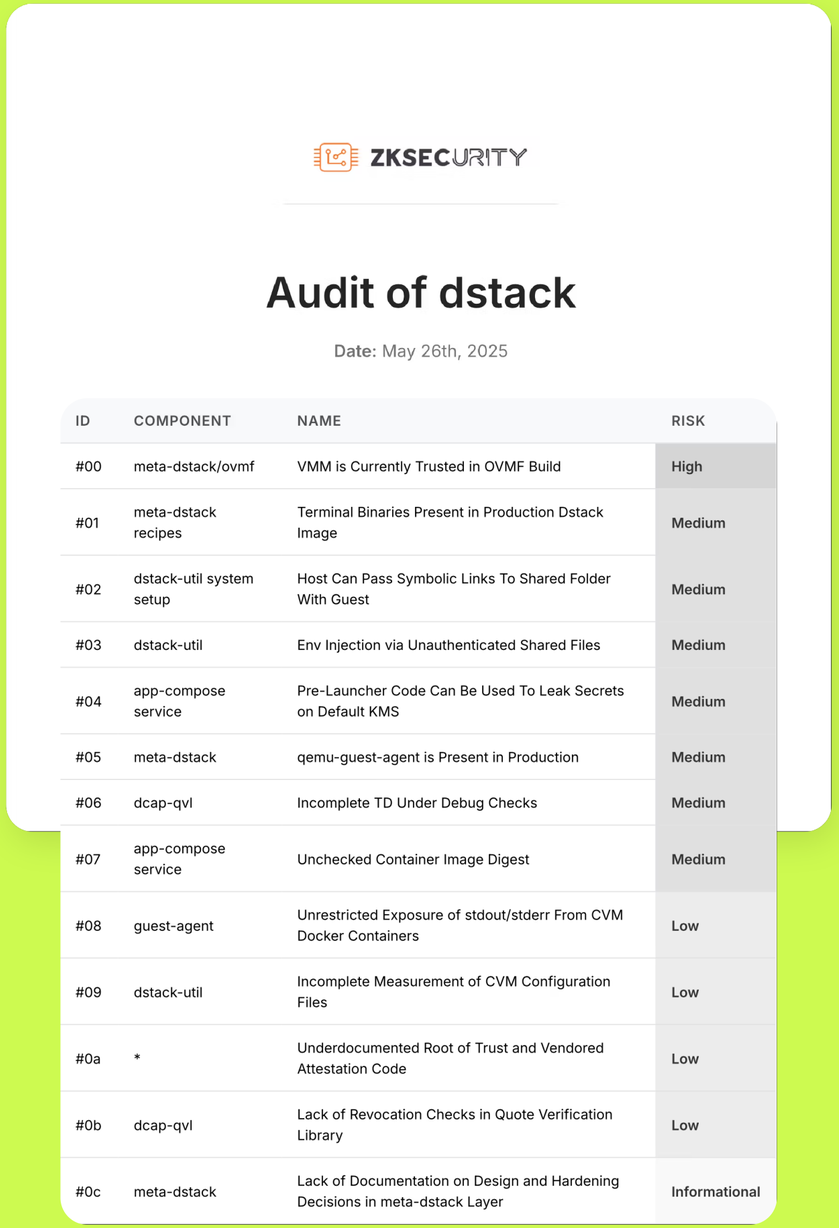
Low-level libraries and tooling, including remote attestation components, guest agent service, and utilities
Image-related files for **dstack-OS**

Findings

A **high-severity issue** with the OVMF build configuration, now fixed with a more secure implementation

Medium-severity issues related to terminal binaries, symbolic link handling, and environment variable protection

Several **low-severity findings** and documentation improvement recommendations



ZKSECURITY

Audit of dstack

Date: May 26th, 2025

ID	COMPONENT	NAME	RISK
#00	meta-dstack/ovmf	VMM is Currently Trusted in OVMF Build	High
#01	meta-dstack recipes	Terminal Binaries Present in Production Dstack Image	Medium
#02	dstack-util system setup	Host Can Pass Symbolic Links To Shared Folder With Guest	Medium
#03	dstack-util	Env Injection via Unauthenticated Shared Files	Medium
#04	app-compose service	Pre-Launcher Code Can Be Used To Leak Secrets on Default KMS	Medium
#05	meta-dstack	qemu-guest-agent is Present in Production	Medium
#06	dcap-qvl	Incomplete TD Under Debug Checks	Medium
#07	app-compose service	Unchecked Container Image Digest	Medium
#08	guest-agent	Unrestricted Exposure of stdout/stderr From CVM Docker Containers	Low
#09	dstack-util	Incomplete Measurement of CVM Configuration Files	Low
#0a	*	Underdocumented Root of Trust and Vendored Attestation Code	Low
#0b	dcap-qvl	Lack of Revocation Checks in Quote Verification Library	Low
#0c	meta-dstack	Lack of Documentation on Design and Hardening Decisions in meta-dstack Layer	Informational

First Benchmark for Running LLM on TEE GPU

Conclusions

Minimal computational overhead within the GPU, the overall performance penalty is primarily attributable to **data transfer**.

For most of the LLM queries, the overhead remains **below 7%**.

Larger models and longer sequences experiencing **nearly zero overhead**.

Confidential Computing on NVIDIA Hopper GPUs: A Performance Benchmark Study

Jianwei Zhu, Hang Yin, Peng Deng[†], Aline Almeida[‡], Shunfan Zhou
Phala Network, [†]Fudan University, [‡]io.net
{jianweiz, hangyin, shelvenzhou}@phala.network,
[†]pdeng21@m.fudan.edu.cn, [‡]aline@io.net

November 6, 2024

Abstract

This report evaluates the performance impact of enabling Trusted Execution Environments (TEE) on NVIDIA Hopper GPUs for large language model (LLM) inference tasks. We benchmark the overhead introduced by TEE mode across various LLMs and token lengths, with a particular focus on the bottleneck caused by CPU-GPU data transfers via PCIe. Our results indicate that while there is minimal computational overhead within the GPU, the overall performance penalty is primarily attributable to data transfer. For the majority of typical LLM queries, the overhead remains below 7%, with larger models and longer sequences experiencing nearly zero overhead.

Acknowledgments

We would like to express our gratitude to the io.net [io.] and IOG Foundation [Fou] for their generous grant, which made this research possible. We also extend our thanks to Engage Stack [Sta], the cloud service provider, for providing the necessary hardware and technical support.

1 Introduction

Trusted Execution Environments (TEEs) are increasingly important in machine learning and AI due to growing security requirements in both enterprise and decentralized applications [SAB15, MSM+18, AKKH18]. The introduction of TEE-enabled GPUs, such as the NVIDIA H100 and H200, adds an extra layer of protection for sensitive data but may impact performance. Understanding these trade-offs, particularly for large-scale machine learning tasks, is crucial for adopting TEE in high-performance AI applications [YMY+22, WO24].

This report quantifies the performance overhead of enabling TEE on the NVIDIA Hopper architecture GPUs during LLM inference tasks, identifying where the overhead arises and under what conditions it can be minimized.

2 Background

2.1 Trusted Execution Environment

A TEE is a hardware-based security feature that isolates computations, preventing unauthorized access and tampering, even from the operating system or the physical hardware owner. As the core technology enabling Confidential Computing, TEEs create secure enclaves where sensitive data and code are processed with encryption, ensuring confidentiality and integrity even if the broader system is compromised [SAB15]. Traditionally implemented in CPUs, TEE technology was extended to GPUs by NVIDIA in 2023, enabling tamper-proof and confidentiality-preserving computation inside the GPU with minimal performance penalty [DGK+23].

Usage Statistics

Phala Cloud Beta

Join the waitlist today: https://bit.ly/TEECloud

Enjoy FREE Beta Credits: 4 vCPUs, 8GB memory, and 1 instance—available for a limited time!

Users Cloud

6,393
Users

@phala_network ··· 58min

Total VMs Cloud

2,385
Total VMs

@phala_network ··· 58min

Running VMs Cloud

1,018
Running VMs

@phala_network ··· 58min

Total vCPU Cloud

4,557
Total vCPU

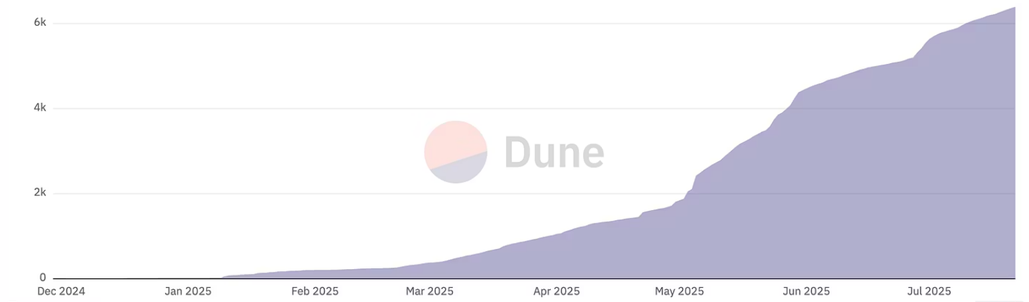
@phala_network ··· 58min

Annual revenue in PHA Cloud

21,951,328
Annual revenue in PHA

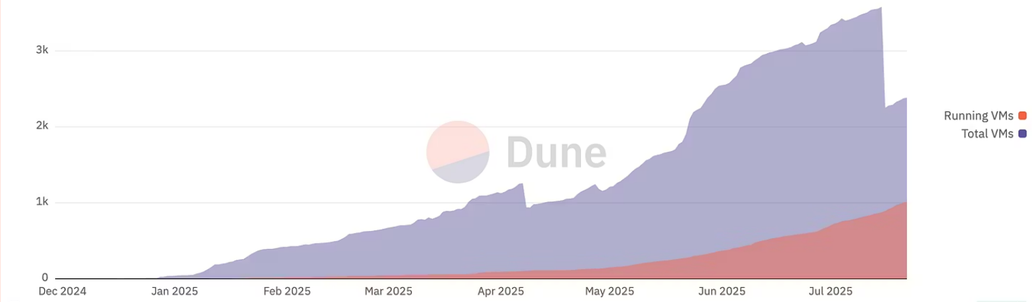
@phala_network ··· 58min

Users Cloud



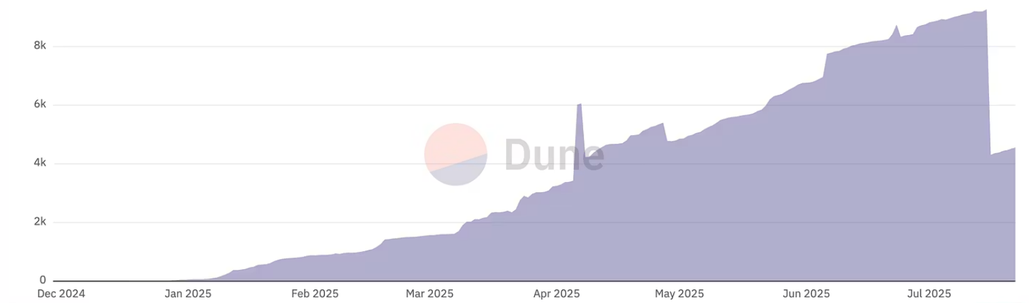
@phala_network ··· 58min

VMs Cloud



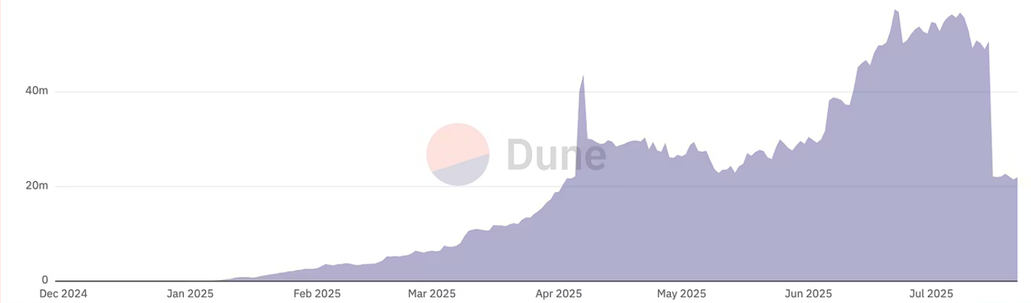
@phala_network ··· 58min

Total vCPUs Cloud



@phala_network ··· 58min

Annual revenue in PHA Cloud



@phala_network ··· 58min

Project Status

Contributors

Phala Network

Core Contributor

Near AI

Private ML SDK

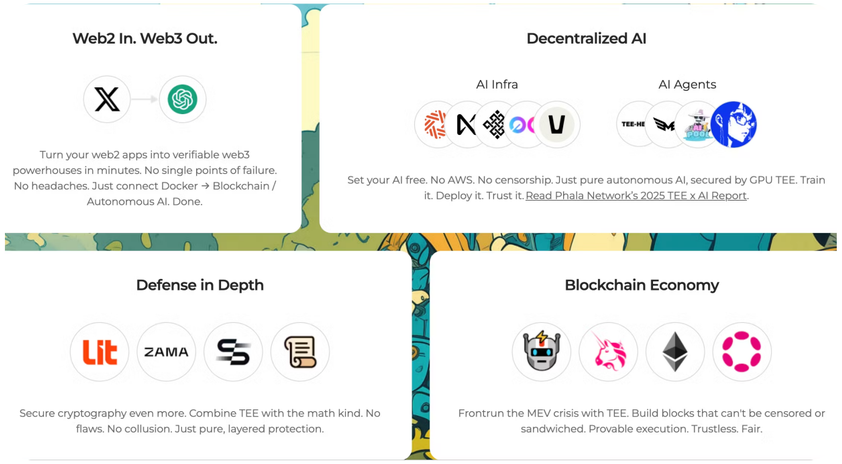
Prof. Andrew Miller

Core Contributor

Flashbots

Contributor

Use cases



dstack

Deploy containerized apps to TEE with end-to-end security in minutes.

stars 223 | license Apache-2.0 | Community | Ask DeepWiki

59 Contributions in the Last 30 Days

0 Opened/Closed Issue Ratio	16 Pull Requests Opened	73 Commits
−1 (−100%) ↓ past month	−14 (−46.67%) ↓ past month	−36 (−33.03%) ↓ past month

Issues — Opened / Closed

Pull Requests — Opened / Closed

Pushes & Commits — Pushes / Commits

A Live Community dstack Applications explorer is available

Future Plan

Distributed System

K8s support
High availability
E2EE network

Compatibility

AMD SEV-SNP support
Nitro support

Security

Safer KMS with MPC + TEE
Attack detection with honeypot
Sandboxing

A comprehensive version is available

PHALA

Thank you !

Dr. Shunfan (Shelven) Zhou

Lead Researcher

shelvenzhou@phala.network

https://cloud.phala.network/

Comparisons between dstack and Confidential Containers

Similar Security Design Principle

Independent KMS to manage the secrets for running TEE workers / pods

Different Requirements on Verifiability

Dstack is designed with verifiability as a core requirement

Different Approaches to Usability

Dstack prioritizes **ease of use** with built-in security features
CoCo adopts a **minimalist approach**, leaving these to application developers

	dstack	**Coco**
Reproducible Build	All components: os images, kms and gateway	No
Code Audit	Core components	No
Measurement Target	Docker Image / Source Code	Sandbox and Security Policy
Attestation Generation	On request	Static during Sandbox Creation
Encrypted Network Connection	Built-in Support	Instruction only
Encrypted Disk Support	Built-in Support	Fs API available
Trust Center	Yes	No
Lifecycle Transparency Log	Built-in Support	No