Here's a prototype of making P2P self-replicating confidential VMs (CVMs). Submitted for inclusion (or at least to influence) the T/acc movement. It is in the same cinematic universe as Sirrah, but oriented around CVMs instead.

- Github: GitHub - amiller/dstack-vm
- Visit the greeting page: https://dstack-mockup.ln.soc1024.com/
- Orchestration contract (on Sepolia)

The full code in the repository is meant to be short enough to read and understand in entirety (under 1000 lines).

You can play along using any linux environment with qemu. You do not need a real TDX machine since this supports TDX DCAP attestations from a dummy service.

Dstack Overview

Each Kettle node on the network is a CVM that runs a payload application container. For this testnet, there is an orchestration contract on Sepolia that manages the onboarding process for new kettles and tracks the currently-deployed version of the application. Here's a high level illustration of the components that we'll go over:

[

image

1174×532 55.9 KB

](https://collective.flashbots.net/uploads/default/original/2X/e/ed5b196a5f8a6cf6a577aff24a140964ff6a7d0a.png)

Replicatoor

When you first start your VM, it will try to join the network and get a copy of the shared key. This requires posting a Register

message on Sepolia, and passing along a quote. The existing node checks your quote before giving you a copy of the key.

For the first node, we simply check the quote generated at "bootstrap" time into the source code repository. It becomes part of the security auditor's scope of attention, just like smart contract constructor parameters.

[

image|450

1168×635 70 KB

](https://collective.flashbots.net/uploads/default/original/2X/0/0ba5468e42d03b3a63e4c9f7f8dc4ec2ccbadeb3.png)

Notice that all of the DCAP quote handling is kept off-chain, keeping gas costs very low.

This implementation is using real TDX DCAP attestations, but if you run the VM outside of TDX, it uses a dummy service to produce the attestation (a real CVM that produces a remote attestation for any report_data

you ask for).

The replicatoor key is also used to provide EVM-friendly remote attestations, simply by ecdsa-signing a message.

Unstoppable TLS

The replicatoor key is used to derive a TLS private key, so all the nodes can serve HTTPS under the same certificate.

This approach supports both proactive checking (like RATLS and aTLS) as well as browser-friendly optimistic auditing with Certificate Transparency.

For proactive checking, simply inspect the public key for the certificate and check it has been signed by the replicatoor key. This suffices as evidence the session is with an enclave that has passed the onboarding flow.

Since we can't make ordinary browsers do this proactive check, we can instead do our best with Certificate Transparency. This is where browsers accept certificates as long as they know they're published on a transparency log. Auditors should check that every public key contained in a certificate appearing here has been signed b

KubernEthes

The smart contract keeps track of the current version of the app, including its sha256 hash.Here's the explorer link to see the current app: Call "container()"

A monitor script on the CVM simply checks the blockchain for changes and reloads the application container, pulling from the network or untrusted host volume as needed.

Try it! The demo should looks like this

[

image

1902×902 302 KB

](https://collective.flashbots.net/uploads/default/original/2X/0/0629ce4ae1e0d68f6b95267e59aaf156f894e16b.png)

Discussion

This is meant to be a provocative prototype. Everything written in Python or Bash (100% of the repo) is meant to be discardable and replaceable with Rust. Eventually this will be a pull request to "Tstack" the production version SUAVE might use.

Here are some of the main takeaways:

- permissionless onboarding with smart contracts doesn't have to be hard
- encourage moving complexity out of the TEE and into the untrusted host where it's more manageable
- the same certificates can be used for both ordinary (optimistic) TLS and proactive checking
- encapsulating the raw DCAP remote attestations in the onboarding process means EVM-friendly attestations can be used everywhere else

Other things left to do:

- DCAP verification is not yet in the CVM yet
- Smart contract is missing on-chain PCCS / TCBInfo / MRTD.
- We did not cover key rotation yet
- Handling of keys and key derivation needs a look over
- Still to integrate with a real TDX build
- Reproducibility of the CVM using virt-customize is poor
- The untrusted host service is not so robust yet
- The on-chain container orchestration is pretty simplistic (just one app) but fun to think about further