Flashbots, in collaboration with nuConstruct and rsync-builder, is experimenting with Intel TDX to trustlessly capture bottom-of-block arbitrages without exposing frontrunning risks. nuConstruct, who operates the top atomic arbitrage bot 0x3face, has successfully deployed their entire bot inside TDX without code changes. We are excited to share that their bot inside TDX has equivalent bundle inclusion rates, with 440 bundles landed so far! (Dashboard)

We are looking to onboard one more searcher at this time to experiment with block builders. We are also looking to onboard one other infrastructure partner to help us scale to support more searchers.

The following is a work of art by Artyom Bakhtin, Frieder Erdmann, Angela Lu, and Moe Mahhouk at Flashbots. Special thanks to Phil Daian, Andrew Miller, Mateusz Morusiewicz, and Daniel Sukoneck for their support in our R&D journey and Quintus Kilbourn and Reid Yager for their feedback and review of this post.

Motivation

Earlier this year, orderflow.art showed that a distributed block building network of solvers, searchers, and builders is more precise and resilient than a handful of vertically integrated players in serving the variance of Ethereum's decentralized user base and its hundreds of liquidity pools.

But we can do even better. Our distributed systems are still limited in the value we can create and capture because our systems rely on trust, and reliance on trust inherently centralizes power in the existing dominant few.

Over the last few months, Flashbots has been experimenting with Intel TDX to remove trust in the MEV supply chain without changes to the distributed block building APIs we use today.

What is Intel TDX?

Intel TDX is the latest evolution in trusted execution environments (TEEs), announced by Intel on February 2022 and offered in public preview by cloud providers Microsoft Azure and Google Cloud Platform earlier this year.

Briefly, Intel TDX is a confidential VM, which expands the boundary a user can trust with confidential data from only an application enclave in Intel SGX to an entire VM in Intel TDX (Figure 1). Confidential VMs enable developers to deploy unmodified code that can run with near-native performance in private settings, making TEEs a practical and compelling privacy solution for MEV use cases.

[

1104×381 163 KB

](https://collective.flashbots.net/uploads/default/original/2X/7/787316a4ff22c9b85656accc887da542b619cce7.png)

Figure 1, from Intel TDX Deep Dive by Benny Fuhry of Intel

Specifically, TDX provides two unique properties:

- Privacy

: TDX uses hardware-assisted virtualization of CPU, memory, and I/O and encrypts the entire computer's memory and individual pages to protect the guest VM code, execution, and data from an untrusted Host OS and Hypervisor.

- Integrity

: Data center attestation primitives (DCAP) enable guests to verify the expected VM image is running on a genuine TDX machine by comparing the measurement (you can think of this as a code hash) of the guests' local build to the measurement by the TDX cloud service provider.

For more, see TDX Demystified by IBM Research.

Mutual Privacy

One of the most critical problems for order flow providers today is that sharing user trade intents with searchers who compete to give users the best possible price exposes those trades to frontrunning, which gets users the worst possible price.

The privacy dilemma manifests in different ways at each stage of the trade life cycle:

- intent auctions (eg Uniswap X, Cowswap) share user trade intents in order to outsource routing to many competing solvers, but leave users vulnerable to frontrunning
- order flow auctions (eg MEV-Share, MEV-Blocker) outsource transaction backrunning to arbitrage bots, but must protect against frontrunning by only leaking partial information or creating fake transactions

- block builders could auction off the right to backrun their private order flow, bundles, and blocks, but cannot do so safely without exposing user trades and searcher bundles to high frontrunning risks

However, we can find a creative solution by leaning into Intel TDX's unique privacy and integrity properties.

- First, we build a TDX VM image that only the searcher's SSH key can access. The image must be minimal and auditable so searchers can easily inspect the code and the SSH configuration. It must also be reproducible, generating the same "measurement" (you can think of this as a code hash) across machines with different hardware setups.
- A TDX operator deploys the image, and the searcher performs attestation, which checks that the exact image they audited is running on a genuine TDX VM by comparing their local measurement to the one measured by the TDX cloud provider. If the measurements are the same, the searcher receives the guarantee that their code cannot be accessed, observed, or tampered with by the TDX operator and can safely upload their proprietary code into TDX.
- Yet, the TDX operator can sandbox the searcher VM (eg at the cloud level) so that information can only leave the machine through the order flow provider's endpoint.
- Hence, we now have a simple mutual privacy setup in which proprietary searcher code can compute on sensitive user data.

In this setup:

- the searcher's code and execution are private from both the TDX operator and the order flow provider, assured through Intel TDX
- the order flow provider's user trades are private from the searcher, by trusting the TDX operator

While a big ask, the setup is trustless if the order flow provider is the TDX operator and handles the deployment and sandboxing of colocated searcher TDX machines.

For the more common case when the order flow provider and TDX operator are separate parties, Flashbots is working on removing the need to trust the TDX operator by implementing sandboxing within the initial VM image. This way, the block builder and their users can attest to the sandbox and, thus, order flow data privacy.

Bottom of Block Arbitrage

$2.6M* of atomic arbitrage opportunities in May was left uncaptured by Ethereum block builders. At the block level, arbitrage opportunities are created by:

- private searcher order flow (eg sandwiches, CEX-DEX arbitrage)
- private application order flow (eg BananaGun)
- imperfect intent auctions (eg Uniswap X, Cowswap)
- imperfect order flow auctions (eg MEV-Share, MEV-Blocker)
- and the combination of all sources of order flow when ordered in specific ways.

*Methodology: We use LibMEV's atomic arbitrage data set, which tags atomic arbitrages that do not immediately follow a trigger transaction as a naked_backrun. This tag represents a sum of current block blind atomic arbitrages and atomic arbitrages leftover from previous blocks. $2.6M is the sum of the atomic arbitrage opportunity (USD value at the time of the searcher's bid, burn, and profit) from May 1, 2024 to May 31, 2024. Visit their website for more on their methodology.

The ability for block builders to tap into the existing network of atomic arbitrage bots to capture these "bottom-of-block" backruns without trusting them is a net new unlock

by mutual privacy:

- Block builders, who have the searcher bundles, have the most sensitive trade information requirements out of all order flow providers.
- Yet, atomic arbitrage bots can access all the pool states at block building time after each transaction has been ordered,

to be incredibly effective and precise at crafting backruns for block builders.

- This is all made practical by Confidential VM TEEs like Intel TDX, which make it easy for atomic arbitrage bots, which typically run modified Ethereum nodes, to put their entire atomic arbitrage bot inside TEEs without code changes and run at near-native performance.

And this is how we will do it. We call it bob

.

bob

[

1920×1599 171 KB

](https://collective.flashbots.net/uploads/default/original/2X/c/c9ec7639f6b231fed609f121399a8d09d809acf4.jpeg)

To recap, in the first version, we focus on achieving trustless privacy for the searcher; block builders will have to trust the TDX Operator (Flashbots) to protect the privacy of block information.

The initial architecture involves three parties: searchers, block builders, and a TDX operator (Flashbots) that creates the VM images, deploys the images on TDX-enabled servers, maintains the firewall to prevent the leakage of private block information, and provides a few whitelisted auxiliary searcher services. However, we must tweak our initial mutual privacy setup in one key way to make the design practical for atomic arbitrage bots.

adding a delayed log output channel

Visibility in the form of logs is essential for the health and success of atomic arbitrage bots, which are running modified Ethereum nodes with the added complexity of managing various strange behaviors of long-tail tokens throughout simulation and arbitrage calculation. So, we must allow the searcher to output logs, but we can enforce a 2-minute (~10 blocks) time delay so that important transactions can land before searchers can output them.

Because we introduced a log output channel for the searcher, any information ever streamed to the TDX searcher can now be outputted through logs (at a delay!). We can address this issue by being specific about what block builders stream inside.

block builder workflow

Block builders will stream state diffs

into colocated TDX VMs provisioned with private searcher code, which computes arbitrage candidates and submits them back to the block builder's RPC.

To preserve failed bid privacy for searchers, block builders should remove a list of searcher and builder to

and from

addresses from their state diff stream. This way, searchers can only see (and eventually log) the final pool states after all bundles and transactions have been applied, but not the identity, order, or bid of the transactions that touched it.

The state diff streaming API should include the following:

{ "jsonrpc": "2.0", "id": 1, "method": "eth_stateDiffSubscription", "params": { "subscription": 1, "result": { blockNumber, // String, hex encoded block number the block builder is currently building for blockTimestamp, // String, hex encoded seconds since the unix epoch blockUuid, // String, a UUID V4 that is used to identify the current block being streamed pendingState // [address, [slot, postValue]], a nested object of changed addresses to changed storage slot keys and their updated value } } }

searcher workflow

Searchers begin by auditing the initial minimal VM image and running an open-source attestation script to gain confidence that their code cannot be leaked, observed, or tampered with by the TDX operator. An in-depth searcher guide through TDX security assumptions, the BOB TDX image, and attestation with code examples can be found here.

We include Podman for searcher ease in building, running, and deploying searcher bots. Searchers can then upload and manage their containers in the TDX VMs via SSH.

TDX VMs can be configured to be large enough torun entire ethereum nodes and block builders. Initial deployment will be on Microsoft Azure's ECedsv5-series confidential VMs available in West Europe, Central US, East US 2 and North Europe with an estimate server cost of ~$800/month up to ~$5000/month, depending on the searcher's workload.

When production mode is initiated, the TDX operator cuts the searcher's SSH connection and begins state diff streaming. The searcher maintains visibility into the health of their bot via a delayed log output channel to their specified remote collector endpoint. When a searcher requests maintenance, the TDX operator pauses state diff streaming and restores the searcher's SSH connection after a delay period.

During searcher setup, production, and maintenance, the TDX operator provides a read-only execution client and consensus client peer to serve the latest headers and chain state, and for state healing after maintenance periods.

Block builders will implement a separate bundle endpoint to merge nonconflicting TDX backruns at the end of the block with the following API:

{ "jsonrpc": "2.0", "id": 1, "method": "eth_sendBottomOfBlockBundle", "params": [{ pendingBlock, // String, UUID V4 of target pending built block txs // array[string] a list of signed transactions to execute in an atomic bundle }] }

Searchers

, although we include the UUID of each state diff, block builders will likely try searcher bundles on their most recently built blocks (not the block that was streamed!) to account for more accumulated order flow.

initial VM image

Flashbots has open-sourced an initial VM image that contains only Dropbear SSH, Podman, and a minimal OS. Per searcher, Flashbots will build the VM image to only allow SSH access to the VM by that searcher's public key.

We use Yocto, an open-source project that allows the creation of custom linux distributions, to create minimal and auditable VM images that can be reproducibly built across machines with different hardware setups for remote attestation.

attestation service

Flashbots has adjusted Edgeless Systems' open-sourced attestation script so that searchers can easily verify their local measurements (hashes) of the initial VM image against the VM measurement provided by Azure.

By checking that each measurement is as expected, searchers verify that the expected, audited VM image runs on a genuine TDX-enabled machine.

Flashbots-trusted firewalling and log delay period

In the initial versions, block builders will trust Flashbots to enforce searcher firewalling and log delay periods.

Flashbots will configure a firewall at the cloud level outside of the TDX machine with the following rules:

- only allow incoming requests from the block builder
- only allow outgoing requests to that block builder's RPC
- read-only access to the consensus and execution client peer

Flashbots will enforce a 2-minute delay for logs outputted from the searcher's TDX machine to the searcher's remote collector endpoint.

infrastructure costs

We expect searchers to pay for their own TDX machine costs. The machine costs are currently high due to some searcher requirements for >2TB local SSDs, which are only offered on machines with 64 vCPUs and 512GB RAM on Azure. We expect the cost to decrease when workloads can be run on bare metal servers and when TDX becomes more generally available.

Future Work

sandboxing inside

Flashbots is currently developing ways toimplement sandboxing within the initial VM image so that the block builder and their users can attest to the sandbox and, thus, data privacy.

We cannot implement log delays inside the TDX VM without sandboxing inside the image because the searcher can overwrite any delay with SSH access. Once sandboxing is implemented inside the image, Flashbots can include a lightweight logs processor such as Fluent Bit inside the VM so that the remote collector endpoint and log delay script can also be attested to by order flow providers.

beyond the bottom of the block

We are exploring other APIs for atomic arbitrageurs to collaborate with order flow providers, such as routing CEX-DEX searcher intents and driving more MEV-Share backruns.

We are also excited about making the private mempool feel "public" again inside TDX, where builders stream private order flow to searchers who can submit back mega-bundles with custom transaction ordering that constructs the best arbitrage.

Integrate

Atomic arbitrage is about to go nuclear!

[

1840×586 75 KB

](https://collective.flashbots.net/uploads/default/original/2X/3/3b2361928e3bb201d8c379f400973a7cb026fe42.jpeg)

We are looking to onboard one more searcher at this time, send us your Flashbots signing address on Discord (@astarinmymind

)!

We are also looking to partner with an infrastructure team to codevelop and scale to support more searchers. We will share our TDX expertise and work together to create value for order flow providers. Please email angela@flashbots.net if interested.

We are also hiring for a senior DevOps engineer and TEE (Linux & Distributed Systems) engineers!