Commit-Boost

Commit-Boost Client

Security Assessment Report

Version: 2.0

December, 2024

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Commit-Boost source code. The review focused solely on the security aspects of the protocol implementation, though general recommendations and informational comments are also provided.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the protocol. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

Document Structure

The first section provides an overview of the functionality of the source code contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the source code.

Overview

Commit-Boost is a modular sidecar that allows Ethereum validators to safely run community-built commitment protocols. It provides a flexible system for interacting with Ethereum network components, focusing on builder and relayer interactions.

Security Assessment Summary

Scope

The review was conducted on the files hosted on the commit-boost-client repository.

The scope of this time-boxed review was strictly limited to the following crates at commit 6af7e51. Retesting activities were performed on commit 1305b46.

- **`cli/*`**
- **`common/`**
  - `config/*`
  - `pbs/*`
  - `constants.rs`
  - `error.rs`
  - `lib.rs`
  - `types.rs`
  - `utils.rs`
- **`metrics/*`**
- **`pbs/*`**

Note: third party libraries and dependencies were excluded from the scope of this assessment.

Approach

The manual review focused on identifying issues associated with the business logic implementation of the sidecar. This includes internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Blockchain.

To support this review, the testing team also utilised the following automated testing tools:

- `cargo audit`: `https://crates.io/crates/cargo-audit`
- `cargo deny`: `https://github.com/EmbarkStudios/cargo-deny`
- `cargo tarpaulin`: `https://crates.io/crates/cargo-tarpaulin`
- `cargo geiger`: `https://github.com/rust-secure-code/cargo-geiger`
- `clippy`: `https://github.com/rust-lang/rust-clippy`

Output for these automated tools is available upon request.

Coverage Limitations

Due to the time-boxed nature of this review, all documented vulnerabilities reflect best effort within the allotted, limited engagement time. As such, Sigma Prime recommends to further investigate areas of the code, and any related functionality, where majority of critical and high risk vulnerabilities were identified.

Findings Summary

The testing team identified a total of 20 issues during this assessment. Categorised by their severity:

- Medium: 8 issues.
- Low: 6 issues.
- Informational: 6 issues.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Commit-Boost sidecar. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team.
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected component(s) have been made to mitigate the related risk.
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

Summary of Findings

ID	Description	Severity	Status
CBST-01	Denial-of-Service Condition in `PbsService`	Medium	Resolved
CBST-02	No Retries When Submitting Signed Blinded Block	Medium	Resolved
CBST-03	Insufficient Header Validation	Medium	Resolved
CBST-04	Docker Services Run As Root	Medium	Resolved
CBST-05	`204 No Content` Handling Can Cause Discarded Bids	Medium	Resolved
CBST-06	Several Containers Are Exposed To LAN By Default	Medium	Resolved
CBST-07	Grafana Container Uses A Trivial Default Password	Medium	Closed
CBST-08	Missing Error Handling When Loading Keys	Medium	Resolved
CBST-09	`get_header_response.version` Is Unchecked	Low	Closed
CBST-10	Invalid `PbsConfig` Can Result In Skipped Proposals	Low	Resolved
CBST-11	CLI Does Not Support Windows	Low	Closed
CBST-12	Insufficient `MAX_SIZE` For `submit_block()` Responses	Low	Resolved
CBST-13	Precision Loss In Wei To ETH Conversion	Low	Resolved
CBST-14	CLI Does Not Use Non-Zero Exit Code For Errors	Low	Resolved
CBST-15	Loss Of ERC-55 Checksum In Beacon Block `fee_recipient` Serialization Test	Informational	Resolved
CBST-16	Duplicate & Incorrect JWT Tokens In `envs`	Informational	Closed
CBST-17	Docker Images Are Not Version Pinned	Informational	Resolved
CBST-18	Running CLI With `sudo` Suggestion	Informational	Resolved
CBST-19	Incorrect Published Event	Informational	Resolved
CBST-20	Miscellaneous General Comments	Informational	Resolved

CBST-01	Denial-of-Service Condition in `PbsService`		
Asset	`pbs/mev_boost/*`		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: High	Likelihood: Low

Description

When processing the response from a relay, the following pattern is used to check if the payload is too large:

```
let response_bytes = res.bytes().await?;
if response_bytes.len() > MAX_SIZE {
    return Err(PbsError::PayloadTooLarge { payload_size: response_bytes.len() });
}
```

The purpose of this check is to protect against memory Denial-of-Service (DoS) attacks, where a malicious relay could return a large payload to overwhelm the validator's memory.

However, this check does not provide this protection as the response bytes are loaded into memory before the check can be performed.

Recommendations

Avoid loading the response bytes into memory before checking the payload size.

Instead, convert the response into a `Stream` of `Bytes` using the `res.bytes_stream()` method and incrementally load the bytes into memory, checking the size of the loaded payload at each step.

Resolution

This issue has been addressed in commit a293bdb by converting the response into a stream and reading it in chunks.

CBST-02	No Retries When Submitting Signed Blinded Block		
Asset	`pbs/src/mev_boost/submit_block.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: High	Likelihood: Low

Description

When submitting the signed blinded block in `send_submit_block()`, a single `POST` request is sent to the relay. However, if this request fails it is not tried again, rather, an error is simply returned.

In case of network failures or other issues, this could result in a block being missed since the signed blinded block did not reach the relay.

```
async fn send_submit_block(
    signed_blinded_block: &SignedBlindedBeaconBlock,
    relay: &RelayClient,
    headers: HeaderMap,
    timeout_ms: u64,
) -> Result {
    let url = relay.submit_block_url()?;

    let start_request = Instant::now();
    let res = match relay
        .client
        .post(url)
        .timeout(Duration::from_millis(timeout_ms))
        .headers(headers)
        .send()
        .await
    {
        Ok(res) => res,
        Err(err) => {
            RELAY_STATUS_CODE
                .with_label_values(&[
                    TIMEOUT_ERROR_CODE_STR,
                    SUBMIT_BLINDED_BLOCK_ENDPOINT_TAG,
                    &relay.id,
                ])
                .inc();
            return Err(err.into());
        }
    };
```

Recommendations

Consider retrying the request upon failure, this can be implemented in `reqwest` with `RetryTransientMiddleware`. Additionally, consider if retries are desired for other requests too (e.g. `get_header`).

Note that when using `RetryTransientMiddleware`, an appropriate `RetryableStrategy` should be used such that failures due to an invalid signature are not retried, while failures due to timeouts are retried. Additionally, note that when retries are used, the total request time may exceed the given timeout.

Resolution

This issue has been addressed in commit 605aef1 by adding retries to `submit_block` and `register_validator` requests.

CBST-03	Insufficient Header Validation		
Asset	`pbs/src/meb_boost/get_header.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: High	Likelihood: Low

Description

The `validate_header()` function does not validate the following `ExecutionPayloadHeader` fields:

- `block_number`
- `gas_limit`
- `timestamp`

If the block with the highest bid is invalid due to a discrepancy or invalidation of any of the aforementioned fields, the `validate_header()` function will not reject the block.

Hence, it is possible for Commit-Boost to provide an invalid block to the consensus client, which will reject the block and be forced to fallback to local block proposal without considering other valid blocks with lower bids.

Recommendations

The `timestamp` can be validated by checking if it matches the timestamp of the slot start as follows:

```
timestamp == utils::timestamp_of_slot_start_millis(GetHeaderParams.slot) / MILLIS_PER_SECOND
```

For the other two fields, consider adding an optional feature that can be toggled via a configuration option that allows the user to fetch the `ExecutionPayload` from the parent block by calling `eth_getBlockByHash()` in the *Execution API*, using the `GetHeaderParams.parent_hash` field. The `parent_block_number` and `parent_gas_limit` can be retrieved from the response.

With these values, `block_number` and `gas_limit` fields can be validated as follows:

- `block_number == parent_block_number + 1`
- `gas_limit` can be validated by checking against `parent_gas_limit` and `GAS_LIMIT_ADJUSTMENT_FACTOR` as per the execution specs. Additionally, the validator's `target_gas_limit` should be checked against the current and parent `gas_limit` such that the miner is modifying the limit towards the validator's target. This can be done by maintaining a cache of the `target_gas_limit` for each proposer, which comes from the consensus client via `register_validator` every epoch.

Resolution

This issue has been addressed in commit 9fdb987 by adding validation for `timestamp`, `block_number` and `gas_limit`.

CBST-04	Docker Services Run As Root		
Asset	`docker/pbs.Dockerfile`, `docker/signer.Dockerfile` *(out-of-scope)*		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: High	Likelihood: Low

Description

The PBS and Signer services' Dockerfiles do not specify a non-root user, causing these containers to run with root privileges by default. This configuration violates security best practices and the principle of least privilege.

Running Docker containers with root privileges poses significant security risks, as any compromise of the containerised application could grant an attacker root-level access to the container. If an attacker manages to escape the container, they may gain root privileges on the host system.

Recommendations

Consider adding a dedicated non-root user and group in each Dockerfile using the `USER` instruction. The user should be created with a specific UID/GID, as shown in the following example:

```
RUN groupadd -g 10001 commitboost && \
    useradd -u 10001 -g commitboost -s /sbin/nologin commitboost
USER commitboost
```

Resolution

This issue has been addressed in commit b999b38 by adding a new user and group.

CBST-05	`204 No Content` Handling Can Cause Discarded Bids		
Asset	`pbs/src/meb_boost/get_header.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: High	Likelihood: Low

Description

It is possible for Commit-Boost to discard valuable bids from relays which return `204 No Content`.

If timing games are enabled, the `send_timed_get_header()` function will call `send_one_get_header()` at a configured frequency of `frequency_get_header_ms` milliseconds. `send_one_get_header()` will request headers from a relay until it is late in the slot configured by `late_in_slot_time_ms`.

The last `maybe_header` from the relay is returned to the `get_headers()` function, representing the relay's best bid as shown below:

```
if let Some((_, maybe_header)) = results
    .into_iter()
    .filter_map(|res| {
        // ignore join error and timeouts, log other errors
        res.ok().and_then(|inner_res| match inner_res {
            // @audit This does not filter out `None` headers
            Ok(maybe_header) => {
                n_headers += 1;
                Some(maybe_header)
            }
            Err(err) if err.is_timeout() => None,
            Err(err) => {
                error!(?err, "TG: error sending header request");
                None
            }
        })
    })
    .max_by_key(|(start_time, _)| *start_time)
{
    debug!(n_headers, "TG: received headers from relay");
    return Ok(maybe_header);
}
```

However, it is possible for a relay to return `204 No Content` if the `getHeader` request cut-off has passed. These responses are treated as valid:

```
if code == StatusCode::NO_CONTENT {
    debug!(
        ?code,
        latency = ?request_latency,
        response = ?response_bytes,
        "no header from relay"
    );
    return Ok((start_request_time, None));
}
```

This means that if a header is requested after the relay's configured cut-off, then all previous bids from the relay are discarded, resulting in the relay not being considered when selecting the best bid.

This issue has a low likelihood rating as the Commit-Boost default configuration value of `late_in_slot_time_ms` is set to 2 seconds, whereas the default MEV-Boost relay configuration uses a `getHeaderRequestCutoffMs` of 3 seconds. This means that it is unlikely for Commit-Boost to request headers after the relay's cut-off. However, these default values can be overridden, potentially causing valuable bids to be discarded.

Recommendations

Consider storing all bids from a relay into state instead of just the latest `maybe_header`.

Alternatively, filter out `maybe_header` values where the `header` is `None` before returning the latest bid as shown below:

```
if let Some((_, maybe_header)) = results
    .into_iter()
    .filter_map(|res| {
        // ignore join error and timeouts, log other errors
        res.ok().and_then(|inner_res| match inner_res {
            Ok(maybe_header) => {
                if let (start_time, Some(header)) = maybe_header {
                    n_headers += 1;
                    return Some((start_time, Some(header)));
                } else {
                    // `None` headers from 204 responses are filtered out
                    return None;
                }
            }
            Err(err) if err.is_timeout() => None,
            Err(err) => {
                error!(?err, "TG: error sending header request");
                None
            }
        })
    })
    .max_by_key(|(start_time, _)| *start_time)
{
    debug!(n_headers, "TG: received headers from relay");
    return Ok(maybe_header);
}
```

Resolution

This issue has been addressed in commit 16eadc4 by filtering out `maybe_header` values where the `header` is `None`.

CBST-06	Several Containers Are Exposed To LAN By Default		
Asset	`cli/src/docker_init.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: Medium	Likelihood: Medium

Description

Several Docker containers are exposed to the local network by default, even though this is likely not required. In addition to this, there is no configuration option offered to limit their exposure to *localhost* only. These containers include:

- **PBS**

```
let pbs_service = Service {
        // ...
        ports: Ports::Short(vec![format!(
            "{}:{}",
            cb_config.pbs.pbs_config.port, cb_config.pbs.pbs_config.port
        )]),
```

- **cAdvisor**

```
Some(Service {
        container_name: Some("cb_cadvisor".to_owned()),
        image: Some("gcr.io/cadvisor/cadvisor".to_owned()),
        ports: Ports::Short(vec![format!("{cadvisor_port}:8080")]),
        // ...
}),
```

- **Prometheus**

```
let prometheus_service = Service {
        // ...
        ports: Ports::Short(vec!["9090:9090".to_owned()]),
```

None of these containers are configured to use a form of authentication, as such any actor on the local network will have full access to these services. The actor could potentially read sensitive data from services such as *cAdvisor* or *Prometheus* and launch DoS attacks against the PBS service, which may result in missed blocks.

Most configurations do not require these services to be accessible over the network, as such it makes sense to limit them to the local machine network. This can be done as follows:

```
let prometheus_service = Service {
        // ...
        ports: Ports::Short(vec!["127.0.0.1:9090:9090".to_owned()]),
```

Recommendations

Consider limiting the default exposure of the containers and consider adding configuration options for users that want to expose the containers to the network.

Resolution

This issue has been addressed in commit 50ff52c by not exposing the ports for these services by default.

CBST-07	Grafana Container Uses A Trivial Default Password		
Asset	`cli/src/docker_init.rs`		
Status	**Closed:** See Resolution		
Rating	Severity: Medium	Impact: Medium	Likelihood: Medium

Description

The Grafana docker container is configured with the environment variable `GF_SECURITY_ADMIN_PASSWORD`. This defines the initial password for the Grafana web interface which a user can change later on if they wish. Currently, this is hardcoded to `admin`, which is considered trivial and insecure.

Additionally, the Grafana container is exposed to the local network. This could allow a malicious actor on the local network to access sensitive information through Grafana's web interface.

```
let grafana_service = Service {
                container_name: Some("cb_grafana".to_owned()),
                image: Some("grafana/grafana:latest".to_owned()),
                ports: Ports::Short(vec!["3000:3000".to_owned()]),
                networks: Networks::Simple(vec![METRICS_NETWORK.to_owned()]),
                depends_on: DependsOnOptions::Simple(vec!["cb_prometheus".to_owned()]),
                environment: Environment::List(vec!["GF_SECURITY_ADMIN_PASSWORD=admin".to_owned()]),
```

Recommendations

Consider generating a more secure default password that differs for every user.

Resolution

The development team has closed the issue with the following rationale:

> *The user is prompted to change the admin password on the first login, which we thought would be a better UX rather than them having to copy paste the password from the docker compose in plain text.*

Additionally, the development team added an extra warning to ensure that users change the password in commit 22fc743.

CBST-08	Missing Error Handling When Loading Keys		
Asset	`common/src/loader.rs` *(out-of-scope)*		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: Medium	Likelihood: Medium

Description

The `load_secrets_and_keys()` function ignores errors, such that keys which cannot be loaded are silently ignored.

```
if path.is_dir() {
    if let Some(maybe_pubkey) = path.file_name().and_then(|d| d.to_str()) {
        // @audit-info Error in `BlsPublicKey::from_hex()` is silently ignored
        if let Ok(pubkey) = BlsPublicKey::from_hex(maybe_pubkey) {
            let ks_path = format!("{}/{}/voting-keystore.json", keys_path, maybe_pubkey);
            let pw_path = format!("{}/{}", secrets_path, pubkey);

            // @audit Error in `load_one()` is silently ignored
            if let Ok(signer) = load_one(ks_path, pw_path) {
                signers.push(signer);
            }
        }
    };
}
```

This means that if the keystore file is malformed, or the password is incorrect, the key will not be loaded, and signature requests for this `pubkey` will fail.

It is noted that this file is out of scope, however the issue is included in the report for completeness.

Recommendations

Handle the two errors in the code snippet above to make sure the program panics if any keys cannot be loaded.

Also, consider warning the user if no keys are loaded.

Resolution

This issue has been addressed in commit b5122ca by logging a warning if a key fails to load.

CBST-09	`get_header_response.version` Is Unchecked		
Asset	`pbs/src/mev_boost/get_header.rs`		
Status	**Closed:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

The `get_header_response.version` field, which indicates what version the received payload is (e.g. `Deneb`), is not checked in `send_one_get_header()`.

This means that the relay can send a payload with an unexpected or invalid version and Commit-Boost will consider it valid. As a result, Commit-Boost may send this invalid header to the consensus client. The consensus client will consider the header invalid and be forced to use a locally built block, causing the user to potentially lose out on some MEV rewards.

```
let get_header_response: GetHeaderResponse = serde_json::from_slice(&response_bytes)?;

// ...

validate_header(
    &get_header_response.data, // `get_header_response.version` is unchecked
    chain,
    relay.pubkey(),
    params.parent_hash,
    skip_sigverify,
    min_bid_wei,
)?;
```

Recommendations

Consider verifying that `get_header_response.version` is correct.

Resolution

The development team has closed the issue since `get_header_response.version` is checked during deserialiation.

CBST-10	Invalid `PbsConfig` Can Result In Skipped Proposals		
Asset	`common/src/config/pbs.rs, pbs/src/mev_boost/get_header.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

The `PbsConfig.validate()` method is a no-op, and does not verify that the configuration is valid.

This can lead to unexpected behaviour, such as falling back to local block production if `PbsConfig.timeout_get_header_ms` is set to zero, as `get_header()` does not request any headers from relays if the `max_timeout_ms` is zero:

```
let max_timeout_ms = state
    .pbs_config()
    .timeout_get_header_ms
    .min(state.pbs_config().late_in_slot_time_ms.saturating_sub(ms_into_slot));

if max_timeout_ms == 0 {
    warn!(
        ms_into_slot,
        threshold = state.pbs_config().late_in_slot_time_ms,
        "late in slot, skipping relay requests"
    );

    return Ok(None);
}
```

Recommendations

Make sure that the `PbsConfig.validate()` method verifies that the `PbsConfig` is valid by making sure that:

- Timeout values are positive; and
- Timeout values are less than the `late_in_slot_time_ms` value.

Resolution

This issue has been addressed in commit 901cd5e by validating `PbsConfig`.

CBST-11	CLI Does Not Support Windows		
Asset	`cli/src/docker_cmd.rs`		
Status	**Closed:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

Users on Windows cannot start or stop the Commit-Boost services, as the `command` command is not available in Windows.

The `run_docker_compose!` macro determines which docker compose command to use by calling the function `is_command_available()`. This function uses `command -v` to check if the command is available, which is not available on Windows.

```rust
fn is_command_available(command: &str) -> bool {
    Command::new("sh")
        .arg("-c")
        // @audit `command` is not available in Windows
        .arg(format!("command -v {}", command))
        .output()
        .map_or(false, |output| output.status.success())
}
```

Since errors are mapped to `false`, this function will always return `false` on Windows, and so the CLI will not be able to start or stop the Commit-Boost services on Windows even if `docker compose` is installed.

Recommendations

The `Get-Command` PowerShell cmdlet can be used to check if a command is available on Windows. However, this does not support subcommands, so you cannot check if `docker compose` is available with this method.

A possible workaround is the following *one-liner* in PowerShell:

```
try { & docker compose version; exit 0 } catch { exit 1 }
```

This *one-liner* attempts to run `docker compose version`. If it succeeds, the command exits with code 0, indicating success. Otherwise, it exits with code 1.

Resolution

The development team has decided to close the issue since they only offer partial Windows support. A warning for Windows users was added in commit 46ee802.

CBST-12	Insufficient `MAX_SIZE` For `submit_block()` Responses		
Asset	`common/src/pbs/relay.rs, pbs/src/mev_boost/submit_block.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

Responses from relays have a maximum size limit of 10 MiB, which can be too small for some payloads.

The `MAX_SIZE` constant defined as 10 MiB is used to reject responses that are too large to prevent memory-related DoS conditions.

However, it is possible for payloads returned by the `/eth/v1/builder/blinded_blocks` endpoint to be larger than 10 MiB.

This is because it is possible to build a block with transactions that are filled with zeroed out bytes of `calldata`, which use up less gas compared to non-zero `calldata`. This allows for a block to reach up to ~14 MiB in size.

Accounting for the `BlobsBundle`, the response can be as large as ~15.5 MiB at the current `MAX_BLOB_GAS_PER_BLOCK` in EIP-4844, which is equivalent to 6 blobs per block. Keep in mind that the maximum number of blobs per block will increase in future Ethereum upgrades.

This issue has a low impact rating as the relay broadcasts the block once it has verified the validator has signed the blinded block header, so the validator is still able to propose a block for their assigned slot. However, Commit-Boost's `PbsService` will treat the payload as missed and publish an event for the relays that provided the block.

Recommendations

Increase the `MAX_SIZE` constant to 20 MiB to account for the larger payloads. This should also be able to accommodate increases to the maximum number of blobs per block in the near future.

Keep in mind that the `/eth/v1/builder/header` endpoint should use a smaller size limit of 10 KiB as blinded block headers are significantly smaller than full blocks, and the `PbsService` sends multiple requests to all relays in parallel which will increase the load on the validator.

Resolution

This issue has been addressed in commit a293bdb by setting `MAX_SIZE_SUBMIT_BLOCK` to 20 MiB.

CBST-13	Precision Loss In Wei To ETH Conversion		
Asset	`common/src/utils.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

The `wei_to_eth()` function converts a `U256` to a string and then parses it as an `f64`. This can lead to precision loss due to `f64`'s limited precision compared to `U256`. However, this issue is actually moot because the function is not used in practice - it is only used in the serialization code path for `PbsConfig` when loading from a TOML file, while the actual code path only uses the `as_eth_str` module for deserialization.

Recommendations

Replace the current implementation with `alloy-rs`'s utilities in the `as_eth_str` module.

- For Wei to ETH conversion (serialization), use `format_ether()` for direct decimal `String` formatting.
- For ETH to Wei conversion (deserialization), use `parse_units()` to convert with built-in functions.

Resolution

This issue has been addressed in commit 49effd5 by using `alloy-rs`'s `format_ether()` and `parse_ether()` functions.

CBST-14	CLI Does Not Use Non-Zero Exit Code For Errors		
Asset	`cli/src/docker_cmd.rs`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

Errors such as `docker compose` or `docker-compose` command failures only print to `stdout` without properly terminating the program.

```
match cmd_info {
    Some((mut command, _version)) => {
        command.env("COMPOSE_FILE", $compose_path);
        match command.args(&[$($arg),*]).output() {
            Ok(output) => {
                if !output.status.success() {
                    let stderr = str::from_utf8(&output.stderr).unwrap_or("");
                    if stderr.contains("permission denied") {
                        println!("Warning: Permission denied. Try running with sudo.");
                    } else {
                        println!("Command failed with error: {}", stderr);
                    }
                }
            }
            Err(e) => {
                println!("Failed to execute command: {}", e);
            }
        }
    }
    None => {
        println!("Neither `docker compose` nor `docker-compose` were found on your operating system.");
    }
}
```

Recommendations

Replace `println!` error messages with proper error handling that terminates the program.

Resolution

This issue has been addressed in commit 91712c9 by terminating the program with code 1 when these errors are encountered.

CBST-15	Loss Of ERC-55 Checksum In Beacon Block `fee_recipient` Serialization Test
Asset	`common/src/utils.rs`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The `test_encode_decode()` function fails when testing beacon blocks because the serialization process does not preserve ERC-55 checksum encoding of addresses.

```
pub fn test_encode_decode(d: &str) -> T {
    let decoded = serde_json::from_str::(d).expect("deserialize");

    let encoded = serde_json::to_string(&decoded).unwrap();
    let original_v: Value = serde_json::from_str(d).unwrap();
    let encoded_v: Value = serde_json::from_str(&encoded).unwrap();

    assert_eq!(original_v, encoded_v, "encode mismatch");
    decoded
}
```

The test fails specifically when processing beacon blocks' `fee_recipient` field, which contains an ERC-55 checksummed address. During serialization, the address's ERC-55 checksum encoding is lost.

Recommendations

Either preserve the original ERC-55 checksum during serialization or modify the test to compare addresses using their canonical ERC-55 format rather than direct string comparison.

Resolution

This issue has been addressed in commit f38d309 by modifying the test data.

CBST-16	Duplicate & Incorrect JWT Tokens In `envs`
Asset	`cli/src/docker_init.rs`
Status	**Closed:** See Resolution
Rating	Informational

Description

The mapping `envs` contains all JWT tokens used by commit-type modules. It is written to disk in `.cb.env` for later use by the containers. However, a JWT token is added to `envs` twice.

First on line [**131**]:

```
envs.insert(jwt_name.clone(), jwt.clone());
jwts.insert(module.id.clone(), jwt);
```

And later on line [**294**]:

```
// write jwts to env
envs.insert(JWTS_ENV.into(), format_comma_separated(&jwts));
```

Additionally the formatting for the latter is incorrect. An example of a resulting `.cb.env` file would consist of the following:

```
CB_JWT_DA_COMMIT=<jwt_token>
CB_JWTS=DA_COMMIT=<jwt_token>
```

Recommendations

Ensure the above comments are understood and consider appropriate changes.

Resolution

The development team has decided to close the issue with the following comment:

> *This is the expected behaviour since eg:*
>
> - `CB_JWT` *would be loaded by the commit module*
> - `CB_JWTS` *would be loaded by the signer module, so it will have all other JWTs*

CBST-17	Docker Images Are Not Version Pinned
Asset	`cli/src/docker_init.rs`
Status	**Resolved:** See Resolution
Rating	Informational

Description

Several of the docker images used do not have a pinned version. As a result, the used image will change if a new version is available. This may result in breaking changes.

```
let prometheus_service = Service {
            container_name: Some("cb_prometheus".to_owned()),
            image: Some("prom/prometheus:latest".to_owned()), //@audit non-pinned version
            // ...
```

Recommendations

Ensure the above comments are understood and consider changes if desired.

Resolution

This issue has been addressed in commit a7a4d7d by pinning the versions of the docker images.

CBST-18	Running CLI With `sudo` Suggestion
Asset	`cli/src/docker_cmd.rs`
Status	**Resolved:** See Resolution
Rating	Informational

Description

In `cli/src/docker_cmd.rs` on line [22], when encountering permission errors during Docker command execution, the code suggests using `sudo` as a solution:

```
if !output.status.success() {
    let stderr = str::from_utf8(&output.stderr).unwrap_or("");
    if stderr.contains("permission denied") {
        println!("Warning: Permission denied. Try running with sudo.");
    }
}
```

Docker commands should be executed with restricted user permissions through group membership rather than elevated privileges.

Recommendations

Remove the suggestion to use `sudo`. Instead, suggest adding the user to the docker group by following Docker's official post-installation steps.

Update the error message to reflect this change in suggestion.

Resolution

This issue has been addressed in commit d2a882d by modifying the error message.

CBST-19	Incorrect Published Event
Asset	`pbs/src/routes/submit_block.rs`
Status	**Resolved:** See Resolution
Rating	Informational

Description

Commit-Boost publishes a `Builder::EventMissedPayload` event with faulted relays when the relays may not be at fault, such as when the validator's BLS signature is missing or invalid as seen below in the `handle_submit_block()` function.

```
Err(err) => {
    if let Some(fault_pubkeys) = state.get_relays_by_block_hash(slot, block_hash) {
        let fault_relays = state
            .relays()
            .iter()
            .filter(|relay| fault_pubkeys.contains(&relay.pubkey()))
            .map(|relay| &**relay.id)
            .collect::<Vec<_>>()
            .join(",");

        error!(%err, %block_hash, fault_relays, "CRITICAL: no payload received from relays");
        state.publish_event(BuilderEvent::MissedPayload {
            block_hash,
            relays: fault_relays,
        });
    }
    // ...
}
```

Depending on how the event is handled, this could lead to incorrect conclusions about the cause of the error, such as a relay being blocklisted when it is not at fault.

Recommendations

Check the type of `PbsError` and also the HTTP status code if it is a `PbsError::RelayResponse` to make sure that `fault_relays` are only included if relays are indeed at fault.

Resolution

This issue has been addressed in commit aa080ba by modifying the warning.

CBST-20	Miscellaneous General Comments
Asset	/*
Status	**Resolved:** See Resolution
Rating	Informational

Description

This section details miscellaneous findings discovered by the testing team that do not have direct security implications:

1. **Misleading Variable Name In Slot Timestamp Calculation**

   Related Asset(s): common/src/utils.rs

   The function `timestamp_of_slot_start_millis()` uses a misleading variable name that does not accurately represent its purpose:

   ```
   pub fn timestamp_of_slot_start_millis(slot: u64, chain: Chain) -> u64 {
       let seconds_since_genesis = chain.genesis_time_sec() + slot * chain.slot_time_sec();
       seconds_since_genesis * MILLIS_PER_SECOND
   }
   ```

   The variable `seconds_since_genesis` is incorrectly named because it represents the absolute UNIX timestamp in seconds for the start of a slot, not the number of seconds elapsed since genesis.

   Rename the variable `slot_start_seconds` to better reflect its actual meaning.

2. **Typo in Field Name**

   Related Asset(s): common/src/config/pbs.rs

   The field `event_publisher` in the `PbsModuleConfig` struct is misspelled as `event_publiher`.

   Change the field name to `event_publisher`.

3. **Incorrect HTTP Status Code**

   Related Asset(s): pbs/src/routes/register_validator.rs

   The `handle_register_validator()` function records a `503` status code to the `BEACON_NODE_STATUS` metric when it is unable to register the validator with any of the relays.

   This is inconsistent with the original MEV-Boost implementation, which returns a `502` status code.

   A `502` status code is more appropriate because the error is due to missing or invalid responses from the relays.

   Change the `BEACON_NODE_STATUS` metric to `502` instead of `503` in line [**48**].

4. **Minimum Bid Check Is Off By One**

   Related Asset(s): pbs/src/meb_boost/get_header.rs

   The check for the minimum bid in `validate_header()` line [**316**] includes the minimum bid value:

   ```
   if value <= minimum_bid_wei {
       return Err(ValidationError::BidTooLow { min: minimum_bid_wei, got: value });
   }
   ```

   Change the check to `value < minimum_bid_wei` to exclude the minimum bid value.

Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

Resolution

The above issues have been addressed in commit 55f4b7d.

Appendix A Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

Impact			
High	Medium	High	Critical
Medium	Low	Medium	High
Low	Low	Low	Medium
	Low	Medium	High
		Likelihood	

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

σ'