Piecrust Audit Report

Prepared for Dusk by **Porter Adams**

DUSK

Piecrust Intro

Piecrust is the Dusk Virtual Machine, a vital component of the Dusk tech stack, and closely related to the broader Rusk node.

The Piecrust VM is an advanced VM, including the ability to have privacy-friendly transactions within a shared state.

Codebase: https://github.com/dusk-network/piecrust/tree/main

Commit: 603b6d8a08b0cfa123aed9db8a00fd8ee3a9efc8

Time of audit: January 24, 2024 to March 27, 2024

Scope: Everything under

- piecrust/*
- piecrust-uplink/*
- crumbles/*

Summary:

There were two high severity findings:

1. **Overflows/underflows**
2. **Denial of Service by allowing maximum gas**

Overall, the codebase was very high quality. The code is idiomatic, there is good test coverage, and good documentation both inline and in the README.

Besides fixing the findings, my only suggestion for improvement is to consider incorporating some static analysis rules into your CI/CD.

For example, there could be a semgrep rule to catch overflow bugs.

Findings:

1. Overflows/Underflows

Severity: High

The repository did not use checked math in three important places:

- piecrust/src/imports.rs#L94
- piecrust/src/call_tree.rs#L153
- piecrust/src/instance.rs#L269

For example, exploiting the code in this screenshot could have led to an OutOfBounds memory access:

```
87  pub fn check_ptr(
88      instance: &WrappedInstance,
89      offset: usize,
90      len: usize,
91  ) -> Result<(), Error> {
92      let mem_len = instance.with_memory(|mem| mem.len());
93
94      if offset + len >= mem_len {
95          return Err(Error::MemoryAccessOutOfBounds {
96              offset,
97              len,
98              mem_len,
99          });
100     }
101
102     Ok(())
103 }
```

(Potential overflow in piecrust/src/import.rs line 94)

Recommendation:

Use checked math everywhere, unless you are completely certain a value cannot overflow.

Findings:

2. Denial of Service by allowing maximum gas [Severity: High]

In the feeder_call and feeder_call_raw functions, the calls were performed with gas set to u64::MAX.

The Dusk team explained “Initially this was meant to only be called for contracts we control - such as the genesis contracts - to provide a large data feed to users.”

At the time of the audit, these could be called by users, giving users maximum gas for free.

This would have allowed users to waste large amounts of resources, ie a Denial of Service attack.

```
433     pub fn feeder_call_raw<V: Into<Vec<u8>>>(
434         &mut self,
435         contract: ContractId,
436         fn_name: &str,
437         fn_arg: V,
438         feeder: mpsc::Sender<Vec<u8>>,
439     ) -> Result<CallReceipt<Vec<u8>>, Error> {
440         self.inner.feeder = Some(feeder);
441         let r = self.call_raw(contract, fn_name, fn_arg, u64::MAX);
442         self.inner.feeder = None;
443         r
444     }
```

Recommendation:

There are potentially many ways to resolve this issue. Here are two possible resolutions:

1. These calls could be limited to “system” contracts and nothing user facing
2. Only allow users to call with gas they have paid for.

Solutions:

Intro

In a recent audit of the Piecrust - Dusk WASM virtual machine, experts discovered two potential security issues:

a high-severity numerical overflow that could allow sandbox escape and a low-severity vulnerability enabling potential DOS attacks through malicious smart contracts.

These findings were promptly addressed with effective solutions, enhancing the platform's security.

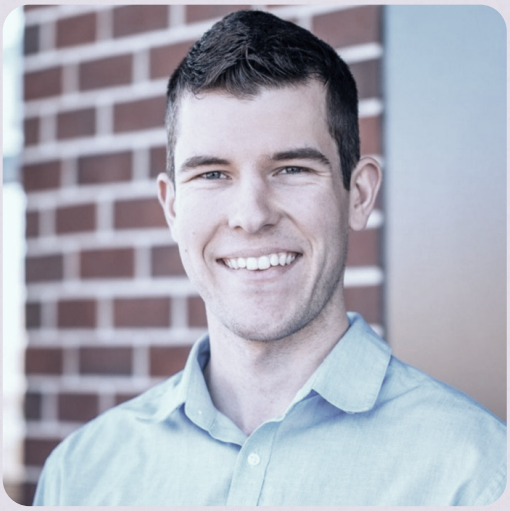
The audit's success illustrates the critical nature of regular security assessments in identifying and mitigating risks, ensuring the Piecrust - Dusk WASM virtual machine remains a secure and reliable platform for executing smart contracts and other applications.

Resolutions:

Merged Pull Request #799: Fix inconsistent gate ordering

Merged Pull Request #800: Fix leading coefficients might be zero

About the Autor



Porter Adams

Porter is a Security Blockchain Engineer at Matter Labs.

With over 10 years experience in software engineering, with a specific focus on blockchain, cryptography, and zero-knowledge, he has a wealth of expertise building, assessing, and securing protocols.

He has previously served as the Director of AI and Cryptography at Learning Economy, and is the current Director of Cryptography and Zero-Knowledge at FYEO.

Prepared for Dusk by **Porter Adams**

DUSK