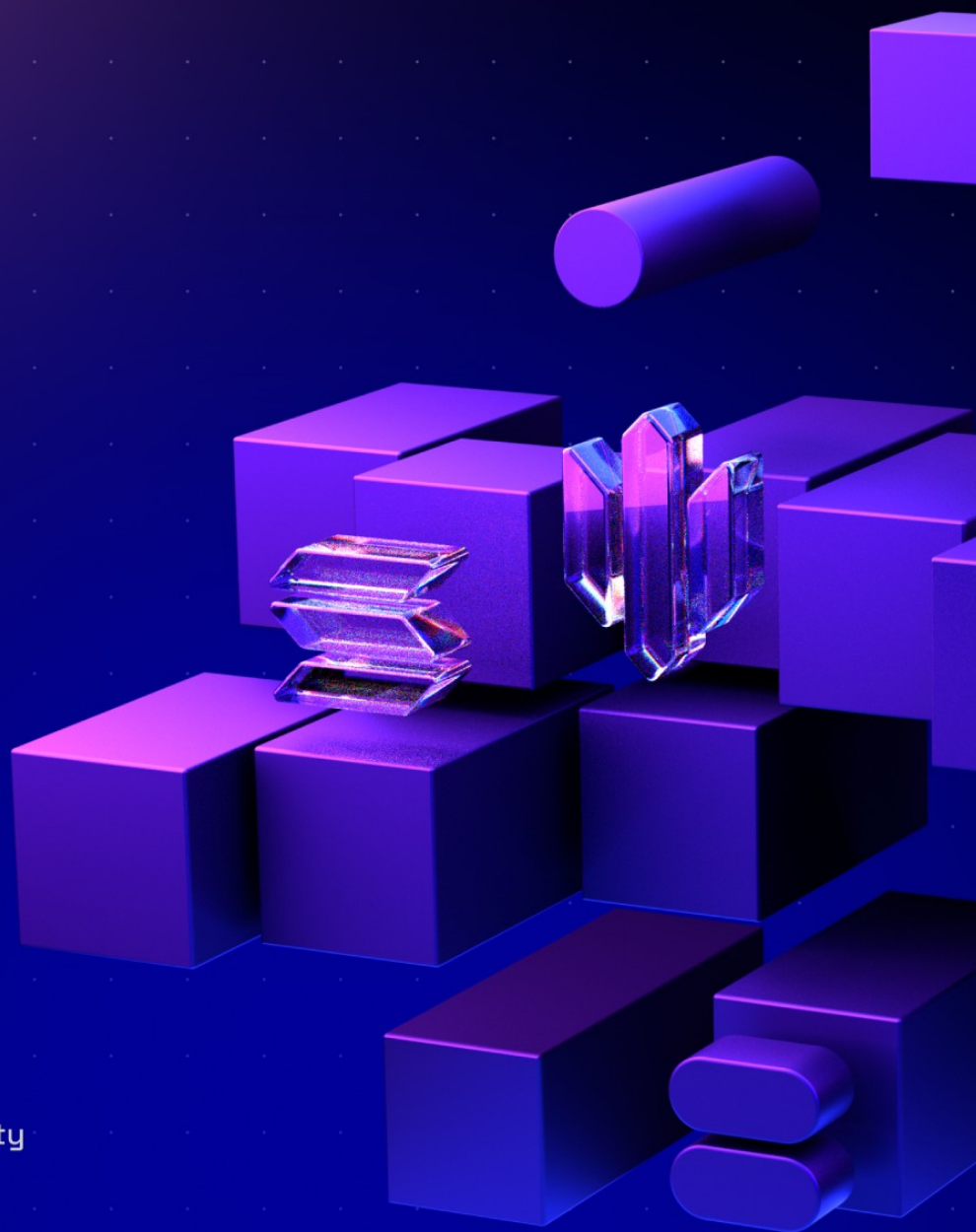
ackee
blockchain security

Unstoppable Domains

Web3 Domains

24.4.2025



Ackee Blockchain Security

Contents

1. Document Revisions

1.0-draft	Draft Report	15.04.2025
1.0	Final Report	24.04.2025
1.1	Fix Review	24.04.2025

2. Overview

This document presents our findings in reviewed contracts.

2.1. Ackee Blockchain Security

Ackee Blockchain Security is an in-house team of security researchers performing security audits focusing on manual code reviews with extensive fuzz testing for Ethereum and Solana. Ackee is trusted by top-tier organizations in web3, securing protocols including Lido, Safe, and Axelar.

We develop open-source security and developer tooling Wake for Ethereum and Trident for Solana, supported by grants from Coinbase and the Solana Foundation. Wake and Trident help auditors in the manual review process to discover hardly recognizable edge-case vulnerabilities.

Our team teaches about blockchain security at the Czech Technical University in Prague, led by our co-founder and CEO, Josef Gattermayer, Ph.D. As the official educational partners of the Solana Foundation, we run the School of Solana and the Solana Auditors Bootcamp.

Ackee's mission is to build a stronger blockchain community by sharing our knowledge.

Ackee Blockchain a.s.
Rohanske nabrezi 717/4
186 00 Prague, Czech Republic
https://ackee.xyz
hello@ackee.xyz

2.2. Audit Methodology

The Ackee Blockchain Security auditing process follows a routine series of steps:

1. **Code review**
    a. High-level review of the specifications, sources, and instructions provided to us to make sure we understand the project's size, scope, and functionality.
    b. Detailed manual code review, which is the process of reading the source code line-by-line to identify potential vulnerabilities. We focus mainly on common classes of Solana program vulnerabilities, such as:

       missing ownership checks, missing signer authorization, signed CPI of unverified programs, cosplay of Solana accounts, missing rent exemption assertion, bump seed canonicalization, incorrect accounts closing, casting truncation, numerical precision errors, arithmetic overflows or underflows.

    c. Comparison of the code and given specifications, ensuring that the program logic correctly implements everything intended.
    d. Review of best practices to improve efficiency, clarity, and maintainability.
2. **Testing and automated analysis**
    a. Run client's tests to ensure that the system works as expected, potentially write missing unit or fuzzy tests using our testing framework Trident.
3. **Local deployment + hacking**
    a. The programs are deployed locally, and we try to attack the system and break it. There is no specific strategy here, and each project's attack attempts are unique to its implementation.

2.3. Finding Classification

A *Severity* rating of each finding is determined as a synthesis of two sub-ratings: *Impact* and *Likelihood*. It ranges from *Informational* to *Critical*.

If we have found a scenario in which an issue is exploitable, it will be assigned an impact rating of *High*, *Medium*, or *Low*, based on the direness of the consequences it has on the system. If we haven't found a way, or the issue is only exploitable given a change in *configuration* (system settings or parameters, such as deployment scripts, compiler configurations, using multi-signature wallets for owners, etc.) or given a change in the codebase, then it will be assigned an impact rating of *Warning* or *Info*.

Low to *High* impact issues also have a *Likelihood*, which measures the probability of exploitability during runtime.

The full definitions are as follows:

Severity

		Likelihood			
		High	**Medium**	**Low**	**N/A**
Impact	**High**	Critical	High	Medium	-
	Medium	High	Medium	Low	-
	Low	Medium	Low	Low	-
	Warning	-	-	-	Warning
	Info	-	-	-	Info

Table 1. Severity of findings

Impact

- **High** - Code that activates the issue will lead to undefined or catastrophic consequences for the system.
- **Medium** - Code that activates the issue will result in consequences of serious substance.
- **Low** - Code that activates the issue will have outcomes on the system that are either recoverable or don't jeopardize its regular functioning.
- **Warning** - The issue cannot be exploited given the current code and/or *configuration*, but could be a security vulnerability if these were to change slightly. If we haven't found a way to exploit the issue given the time constraints, it might be marked as a "Warning" or higher, based on our best estimate of whether it is currently exploitable.
- **Info** - The issue is on the borderline between code quality and security. Examples include insufficient logging for critical operations. Another example is that the issue would be security-related if code or *configuration* was to change.

Likelihood

- **High** - The issue is exploitable by virtually anyone under virtually any circumstance.
- **Medium** - Exploiting the issue currently requires non-trivial preconditions.
- **Low** - Exploiting the issue requires strict preconditions.

2.4. Review Team

The following table lists all contributors to this report. For authors of the specific revision, see the “Revision team” section in the respective “Report revision” chapter.

Member’s Name	Position
Andrej Lukačovič	Lead Auditor
Matej Hyčko	Auditor
Max Kupchenko	Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

2.5. Disclaimer

We’ve put our best effort to find all vulnerabilities in the system, however our findings shouldn’t be considered as a complete list of all existing issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

3. Executive Summary

Unstoppable Domains is a protocol which allows managing domains on Solana.

Revision 1.0

Unstoppable Domains engaged Ackee Blockchain Security to perform a security review of Unstoppable Domains Web3 Domains with a total time donation of 13 engineering days in a period between April 1 and April 10, 2025, with Andrej Lukačovič as the lead auditor.

The audit was performed on the commit `ab4cecd`[1] and the scope was the following:

- Unstoppable Domains Solana Contract, excluding external dependencies.

We began our review by understanding the protocol's design and architecture. During this initial phase, we gathered all available information, including documentation, web page functionality, and project intentions. Then we moved to the second phase.

In the second phase, we performed manual review and wrote fuzz tests in parallel. This process helped us better understand the project's source code while implementing the fuzz tests. During the manual review, we dove deeper into the functionality of the code, simultaneously writing proof of concept tests to support our thoughts and test the correctness of instructions.

During this phase we paid special attention to:

- ensuring the program logic is implemented as intended;
- ensuring all Program Derived Addresses are correctly derived;
- ensuring there are no possible access violations;
- ensuring the protocol behaves fairly;

- ensuring the Cross-Program Invocation is implemented correctly;
- ensuring the Token-2022 Transfer Hook follows the standard;
- ensuring the architecture fits together; and
- ensuring there are no spots where the protocol could be misused.

The final stage consisted of writing the invariant checks, writing the report, and finalizing our thoughts.

For fuzz testing, we used Trident fuzzing framework. The framework is designed for fuzz testing Solana programs written using the Anchor framework. During fuzzing, we identified one issue L1, where the refund recipient in some of the instructions lacks writable privileges, resulting in situations where instruction execution becomes problematic.

Our review resulted in 9 findings, ranging from Info to Low severity. The most severe finding L1 reveals the possibility of instruction failure due to improper refund recipient writable privileges.

Ackee Blockchain Security recommends Unstoppable Domains:

- resolve all identified issues;
- improve validation of the Top-Level Domain; and
- reconsider the architecture behind the Second-Level Domain expiration.

See Report Revision 1.0 for the system overview and trust model.

Revision 1.1

Unstoppable Domains engaged Ackee Blockchain Security to perform a fix review of the findings from the previous revision.

The review was performed between April 24 and April 24, 2025 on the commit 844296e[2].

From the reported 9 findings:

- The L1 issue was fixed in line with the recommendations;
- The W1 issue was acknowledged by the client; however, we recommend fixing the issue;
- The W3 issue was acknowledged by the client, as it is similar to the W1; this means it is recommended to be fixed, as records should not be modifiable after the Second Level Domain is expired, potentially only if the user wants to remove the record;
- The W4 issue was fixed by decreasing allocated account space in case new value is shorter than the old one;
- The W2 issue was acknowledged by the client, as it is in their responsibility to correctly transfer the authority role;
- The W5 issue was acknowledged;
- The I1 issue was fixed in line with the recommendations;
- The I2 issue was fixed in line with the recommendations; and
- The I3 issue was acknowledged by the client.

No new findings were reported.

[1] full commit hash: `ab4cecdae7d2f4e249884810a04a3092992d2c44`, link to commit

[2] full commit hash: `844296e8e8d803b7e5cb7836eaf5e71aa3831212`, link to commit

4. Findings Summary

The following section summarizes findings we identified during our review. Unless overridden for purposes of readability, each finding contains:

- *Description*
- *Exploit scenario* (if severity is low or higher)
- *Recommendation*
- *Fix* (if applicable).

Summary of findings:

Critical	High	Medium	Low	Warning	Info	Total
0	0	0	1	5	3	9

Table 2. Findings Count by Severity

Findings in detail:

Finding title	Severity	Reported	Status
L1: Insufficient mutability for refund recipient	Low	1.0	Fixed
W1: Second-level domain can be blocked forever	Warning	1.0	Acknowledged
W2: Possibility for losing the `ProgramAuthority` access	Warning	1.0	Acknowledged
W3: Expiration does not sufficiently limit the Second-level domain updates	Warning	1.0	Acknowledged

Finding title	Severity	Reported	Status
W4: Record values are not fully overwritten	Warning	1.0	Fixed
W5: Insufficient Top-Level Domain Validation	Warning	1.0	Acknowledged
I1: Unnecessary space allocation for the `Tld` account	Info	1.0	Fixed
I2: Unnecessary source code	Info	1.0	Fixed
I3: InitSpace macro can be used instead of literal values	Info	1.0	Acknowledged

Table 3. Table of Findings

Report Revision 1.0

Revision Team

Member's Name	Position
Andrej Lukačovič	Lead Auditor
Matej Hyčko	Auditor
Max Kupchenko	Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

System Overview

The Unstoppable Domains is a protocol which allows management of domains on the Solana blockchain. The protocol allows for creation of Top-Level domains. Top-Level domains can additionally have Second-Level domains. Second-level domains are in form of Non-Fungible Tokens (NFTs) minted to the users. It is ensured that there will be only one domain in the circulation for a given domain name.

Second-level domains can additionally have records tied to them. Records are in form of key-value pairs. There are no specific recommendations for the records. Potential usage can be to tie a twitter account, Solana Wallet or some other records to the domain.

Top-level domains can have turned on expiration. This means that the Second-level domains can be optionally set with expiration. After the expiration, the Second-level domain cannot be transferred (sold) to another user.

The protocol utilizes Token2022 with the Transfer Hook token extension enabled. The Transfer Hooks ensures that the domain cannot be transfered after the expiration.

Trust Model

The protocol implements to some extent a Role-based Access Control (RBAC) mechanism. The roles are:

- `program authority` - apart from the smart contract upgrade authority, this is a role with the highest privileges (e.g. appointing new minters);
- `minter` - a role with the ability to mint new Second-level domains, update domain metadata, modify the domain expiration, add and remove record before minting a domain.

User must trust:

- `program authority` to appoint responsible minters.

Fuzzing

During the audit, manually-guided fuzz tests were developed to assess the protocol's correctness, security, and robustness. Fuzz test templates are generated from the IDL created by Anchor and then implemented based on user needs. Notably, Trident allows the specification of flows or invariant checks.

Flows are important for helping the fuzzer better cover valid instruction sequences. On the other hand, invariant checks allow for the detection of undesired changes made during instruction execution. When an instruction is successfully invoked, the user can specify multiple invariant checks to ensure that the contents of the accounts were updated as expected during the execution.

Over the period of fuzzing, two common types of failures can occur, a panic during instruction execution or a failure of the specified invariant check. The former can happen, for instance, when an unchecked arithmetic overflow is detected, while the latter is triggered by behavior that is defined as

undesired (e.g. undesired balance change).

The fuzz tests implemented multiple flows, detailed in Appendix B. These flows tested various protocol operations, such as:

- minting a domain;
- updating domain metadata; and
- operations related to records.

This process revealed one issue L1, where the `refund_recipient` lacked the required mutability property.

Findings

The following section presents the list of findings discovered in this revision. For the complete list of all findings, Go back to Findings Summary

L1: Insufficient mutability for refund recipient

Low severity issue

Impact:	Low	Likelihood:	Medium
Target:	instructions/minter.rs, instructions/tld.rs	Type:	Logic error

Description

The `RemoveMinter` and `RemoveTld` functions allow for the removal of minters and top-level domains, respectively. However, the refund recipient account is not marked as mutable in these functions. The instruction will fail to execute when the refund recipient is neither:

- the transaction fee payer (which is mutable by default); nor
- an account marked as mutable in the corresponding context.

This issue requires updating the generated IDL to mark the account as mutable.

Exploit scenario

Alice, a program authority, has the ability to invoke `RemoveMinter` and `RemoveTld` instructions.

1. Alice adds a new minter;
2. Alice adds a new top-level domain;
3. Alice discovers errors in both additions;
4. Alice invokes `RemoveMinter` and `RemoveTld` instructions to remove the minter and TLD; and
5. the instructions fail because the refund recipient is not mutable.

Recommendation

Mark the refund recipient account as mutable in both the `RemoveMinter` and `RemoveTld` contexts.

Fix 1.1

The issue was fixed by adding the `mut` keyword to the refund recipient account in both the `RemoveMinter` and `RemoveTld` instructions.

Go back to Findings Summary

W1: Second-level domain can be blocked forever

Impact:	Warning	Likelihood:	N/A
Target:	-	Type:	Logic error

Description

Top-Level domains can be marked as expirable, which enables expiration settings for their Second-Level domains. When a Second-Level domain has expiration enabled, it cannot be transferred to other users. While one might expect that an expired Second-Level domain would become available for purchase at its original price, this is not implemented. Instead, the domain remains locked to its previous owner indefinitely, even if they choose not to extend the domain's expiration.

The client has acknowledged this behavior. Below is their response:

> *We don't plan to have expirable domains now, it is just sufficient piece of the logic for the future. When this functionality is needed we will add a needed logic to be able to return and resell expired domains. It is not yet decided how it will work. We added it initially because it would be impossible to extend mint accounts with transfer hook later.*
>
> — Unstoppable Domains Team

Recommendation

Implement logic to:

1. burn expired domains; and
2. close their associated mint accounts

This will enable the re-creation of mint accounts and re-minting of expired

domains.

Acknowledgment 1.1

The issue was acknowledged by the client.

Go back to Findings Summary

W2: Possibility for losing the `ProgramAuthority` access

Impact:	Warning	Likelihood:	N/A
Target:	instructions/program_authority.rs	Type:	Code quality

Description

The authority transfer process for the `ProgramAuthority` uses a single-step mechanism. If the new authority address is specified incorrectly during the transfer, the protocol will permanently lose access to authority-gated functions, as there is no way to recover from an incorrect transfer.

Recommendation

Implement a two-step authority transfer process:

1. Current authority initiates the transfer by designating a new authority address; and
2. New authority must explicitly accept the transfer by calling an acceptance function.

This ensures the new authority address is valid and able to control the contract before the transfer is finalized.

Acknowledgment 1.1

The issue was acknowledged by the client.

Go back to Findings Summary

W3: Expiration does not sufficiently limit the Second-level domain updates

Impact:	Warning	Likelihood:	N/A
Target:	instructions/sld.rs, instructions/records.rs	Type:	Logic error

Description

Top-Level domains can be marked as expirable, enabling expiration settings for their Second-Level domains. When a Second-Level domain has expiration enabled, it cannot be transferred to other users. While expired domains should be locked from modifications, the current implementation allows expired Second-Level domains to be modified by:

- adding, updating, and removing records; and
- modifying the metadata URL.

Recommendation

Restrict all domain modification functions to be executable only when the Second-Level domain is not expired. This includes:

1. record management (add, update, remove); and
2. metadata URL updates.

Acknowledgment 1.1

The issue was acknowledged by the client.

Go back to Findings Summary

W4: Record values are not fully overwritten

Impact:	Warning	Likelihood:	N/A
Target:	instructions/records.rs	Type:	Logic error

Description

The `update_record` function allows updating the most recent active record. When the new record value is longer than the currently allocated space, the account is reallocated. However, when the new value is shorter, it simply overwrites the beginning of the old value, leaving the remaining data unchanged on-chain.

For example, consider these record values:

- Original: ”this-is-a-longer-original-value”
- Update: ”shorter”

After updating the original value with ”shorter”, the on-chain data will still contain: ”shorter-a-longer-original-value”. While the deserialization works correctly due to the 4-byte string length prefix, the residual data stored on-chain can be misleading.

Recommendation

When updating a record with a shorter value:

1. Zero out the entire previously allocated space; and
2. Write the new value.

This ensures no residual data remains on-chain.

Fix 1.1

The issue was fixed by reallocating the account for exact length of the new

value.

Go back to Findings Summary

W5: Insufficient Top-Level Domain Validation

Impact:	Warning	Likelihood:	N/A
Target:	helpers.rs	Type:	Data validation

Description

The protocol allows the creation of Top-Level Domains (TLDs) that are not listed in the IANA List of Top-Level Domains. For example, the protocol accepts invalid TLD strings such as `m..---…m`, which violates IANA standards.

Recommendation

Implement strict TLD validation by:

- maintaining an up-to-date list of valid TLDs from IANA;
- validating TLD strings against IANA format requirements; and
- rejecting any TLD registration attempts that do not meet these criteria.

Acknowledgment 1.1

The issue was acknowledged by the client.

Go back to Findings Summary

I1: Unnecessary space allocation for the `Tld` account

Impact:	Info	Likelihood:	N/A
Target:	instructions/tld.rs	Type:	Code quality

Description

The `CreateTld` instruction allocates unnecessary space for the `Tld` account. The account contains a primitive data type `bool`, which occupies 1 byte of space. However, the initialization allocates 2 bytes.

Recommendation

Change the 2-byte allocation to 1 byte.

Fix 1.1

The issue was fixed by changing the allocation to 1 byte.

Go back to Findings Summary

I2: Unnecessary source code

Impact:	Info	Likelihood:	N/A
Target:	instructions/sld.rs	Type:	Unused code

Description

The `MintSld` instruction contains an unnecessary `Rent` account in its input parameters.

The `#[interface(spl_transfer_hook_interface::initialize_extra_account_meta_list)]` attribute macro is unnecessary. This source code marks the corresponding instruction to match the Instruction's discriminator with `InitializeExtraAccountMetaListInstruction` for automatic fallback. However, the `ExtraAccountMetaList` is explicitly initialized in the `MintSld` instruction, which makes the usage of this attribute macro unnecessary.

Recommendation

- remove the `Rent` account from the instruction input; and
- remove the `#[interface(spl_transfer_hook_interface::initialize_extra_account_meta_list)]` attribute macro.

Fix 1.1

The issue was fixed by removing the `Rent` account from the instruction input and the `#[interface(spl_transfer_hook_interface::initialize_extra_account_meta_list)]` attribute macro.

Go back to Findings Summary

I3: InitSpace macro can be used instead of literal values

Impact:	Info	Likelihood:	N/A
Target:	instructions/sld.rs, instructions/tld.rs	Type:	Code quality

Description

The codebase uses numerical literals for space allocation (e.g., `space = 8 + 2`) instead of the `#[derive(InitSpace)]` macro. Using manual space calculations can lead to errors, while the macro automatically evaluates and allocates the correct space.

Recommendation

Replace manual space calculations with the `#[derive(InitSpace)]` macro to ensure correct space allocation.

Acknowledgment 1.1

The issue was acknowledged by the client.

Go back to Findings Summary

Appendix A: How to cite

Please cite this document as:

Ackee Blockchain Security, Unstoppable Domains: Web3 Domains, 24.4.2025.

Appendix B: Trident Findings

This section lists the outputs from the Trident framework used for fuzz testing during the audit.

B.1. Fuzzing

The following table lists all implemented execution flows in the Trident fuzzing framework.

ID	Flow	Added	Status
F1	Test program authority update and initialization	1.0	Success
F2	Test immutable refund recipients	1.0	**Fail** (L1)
F3	Test minter addition and removal operations	1.0	Success
F4	Test pre-mint record operations	1.0	Success
F5	Test post-mint record operations	1.0	Success
F6	Test domain expiration setting	1.0	Success
F7	Test TLD creation and removal operations	1.0	Success
F8	Test domain metadata update	1.0	Success

Table 4. Trident fuzzing flows

The following table lists all implemented invariant checks in the Trident fuzzing framework.

ID	Invariant	Added	Status
IV1	Program authority account must match the authority specified in `initialize` instruction	1.0	Success
IV2	Program authority must be correctly updated to match the `new_authority` pubkey	1.0	Success

ID	Invariant	Added	Status
IV3	When closed, account must be drained and lamports transferred to `refund_receiver`	1.0	Success
IV4	TLD expirability flag must match the value provided during creation	1.0	Success
IV5	Domain expiration must follow TLD rules: only set when TLD is expirable and future dated, always zero when not expirable	1.0	Success
IV6	Domain expiration updates must be valid: cannot set on non-expirable domains and must be future dated	1.0	Success
IV7	Record properties must be valid: value matches input, key is not empty, and value length is within limits	1.0	Success
IV8	Record updates must follow rent rules: increase lamports for larger values, maintain lamports for smaller values, respect size limits	1.0	Success

Table 5. Trident fuzzing invariants

ackee
blockchain security

Thank You

Ackee Blockchain a.s.

Rohanske nabrezi 717/4
186 00 Prague
Czech Republic

hello@ackee.xyz