PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Avvy Domains

05 May 2022

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Avvy Domains on the Avalanche network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Avvy Domains
URL	https://avvy.domains/
Network	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`Domain`	`0x797ac669a1908ca68cd9854994345f570495541a`	MATCH
`LeasingAgentV1`	`0x5c9140B835F5A74E62B49C7Ba30a7362aADbD4Ed`	MATCH
`ContractRegistryV1`	`0x5c9140B835F5A74E62B49C7Ba30a7362aADbD4Ed`	MATCH
`ConstraintsAVAXV1`	`0x121c0af084bB7FdD965dA1741687b1248e2FE465`	MATCH
`NamespaceV1`	`0x72cbb66B23dC7D98E46f0602aC2258F863297440`	MATCH
`PricingOracleV1`	`0x5A755aF3650179D02A93F37220Caf76a34D8D975`	MATCH
`RainbowTableV1`	`0x3b17bAcEDF86f4d36563d2920771ed105D8B6636`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	3	3	-	-
Medium	6	5	-	1
Low	11	11	-	-
Informational	31	31	-	-
Total	**51**	**50**	**-**	**1**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 Global Issues

ID	Severity	Summary	Status
01	MEDIUM	Governance risk: The codebase is presently centralized	ACKNOWLEDGED

1.3.2 Domain

ID	Severity	Summary	Status
02	HIGH	The register presently does not ensure that the namespace hash in the name hash in fact refers to the provided in the register function which allows users to register a domain from a different namespace to a specific namespace	✓ RESOLVED
03	MEDIUM	Users could release their domain during the recycle period	✓ RESOLVED
04	MEDIUM	Phishing risk: Usage of `tx.origin`	✓ RESOLVED
05	MEDIUM	`register` might attempt to make a transfer during the grace period while this is not permitted	✓ RESOLVED
06	LOW	`register` and `recycle` do not adhere to checks-effects-interactions	✓ RESOLVED
07	LOW	`_contractRegistry` is `private`	✓ RESOLVED
08	INFO	Lack of indexing for event parameters	✓ RESOLVED
09	INFO	Lack of events for `setBaseTokenURI` and `setContractURI`	✓ RESOLVED
10	INFO	Various functions can be made external	✓ RESOLVED
11	INFO	Typographical errors	✓ RESOLVED
12	INFO	Unused functions: `_isLeased`	✓ RESOLVED
13	INFO	`_contractRegistry` can be made immutable	✓ RESOLVED

1.3.3 LeasingAgentV1

ID	Severity	Summary	Status
14	HIGH	Registration prices are not multiplied by the lease duration	✓ RESOLVED
15	MEDIUM	commit-reveal scheme does not protect against frontrunning	✓ RESOLVED
16	LOW	Various variables are private	✓ RESOLVED
17	LOW	`register` does not adhere to checks-effects-interactions	✓ RESOLVED
18	LOW	Phishing risk: User is allowed to overpay for domains	✓ RESOLVED
19	INFO	Various functions can be made external	✓ RESOLVED
20	INFO	`MAX_YEARS` and `oneYear` can be made constant	✓ RESOLVED
21	INFO	Various functions lack events	✓ RESOLVED
22	INFO	Gas optimization: Various variables can be made callable throughout the contract	✓ RESOLVED
23	INFO	Lack of validation	✓ RESOLVED
24	INFO	`_namespaceId` and `_contractRegistry` can be made `immutable`	✓ RESOLVED

1.3.4 ContractRegistryV1

ID	Severity	Summary	Status
25	INFO	Usage of tx.origin is discouraged	✓ RESOLVED
26	INFO	Lack of events for `set`	✓ RESOLVED
27	INFO	`set` can be made external	✓ RESOLVED
28	INFO	Gas optimization: `contractName` can be made callable throughout the contract	✓ RESOLVED

1.3.5 ConstraintsAVAXV1

ID	Severity	Summary	Status
29	LOW	Admins are not able to unblock names	✓ RESOLVED
30	LOW	verifier is private	✓ RESOLVED
31	INFO	Lack of events for `blockNames` and `setVerifier`	✓ RESOLVED
32	INFO	Gas optimization: `data` and `names` can be made callable throughout the contract	✓ RESOLVED
33	INFO	Typographical errors	✓ RESOLVED

1.3.6 NamespaceV1

ID	Severity	Summary	Status
34	LOW	`_initializedNamespaces` and `_constraints` are `private`	✓ RESOLVED
35	INFO	Various functions can be made can be made external	✓ RESOLVED
36	INFO	The `checkName` function could be a `view` function	✓ RESOLVED
37	INFO	Lack of events for `setGracePeriodLength`, `setRecyclePeriodLength` and `setConstraints`	✓ RESOLVED
38	INFO	Typographical errors	✓ RESOLVED

1.3.7 PricingOracleV1

ID	Severity	Summary	Status
39	HIGH	The user can avoid paying larger amounts for shorter names through providing an unexpected minLength of 2 in the zero-knowledge proof	✓ RESOLVED
40	LOW	`priceFeed` and `verifier` are private	✓ RESOLVED
41	INFO	`priceFeed` and `verifier` could be made immutable	✓ RESOLVED
42	INFO	Chainlink price feed could return 0	✓ RESOLVED
43	INFO	Typographical errors	✓ RESOLVED

1.3.8 RainbowTableV1

ID	Severity	Summary	Status
44	MEDIUM	Empty or uneven length preimages are not hashable but permitted which could allow for abuse	✓ RESOLVED
45	LOW	Hashes are theoretically mutable	✓ RESOLVED
46	LOW	`contractRegistry` and `_getHash` are private	✓ RESOLVED
47	INFO	`lookup` and `reveal` can be made external	✓ RESOLVED
48	INFO	`contractRegistry` can be made `immutable`	✓ RESOLVED
49	INFO	Lack of indexing for event parameters	✓ RESOLVED
50	INFO	Gas optimization: preimage can be made callable throughout the contract	✓ RESOLVED
51	INFO	Typographical error	✓ RESOLVED

2 Findings

2.1 Global issues

The issues here apply to all the contracts within the scope of the audit.

2.1.1 Issues & Recommendations

Issue #01	Governance risk: The codebase is presently centralized
Severity	MEDIUM SEVERITY
Description	Presently almost all components of the contract can be upgraded which allows the Avvy team to have crucial control over the codebase. Users that use their domain name to resolve crucial parameters might want to take this into consideration.
Recommendation	Consider carefully safeguarding the `ContractRegistryV1` which allows for the upgrading of the components. We recommend eventually moving to a timelocked multisig or similar form of governance.
Resolution	ACKNOWLEDGED The client has indicated that during the initial period of the protocol, centralization is by design to be able to react and develop quickly. The client plans to address the governance issue more carefully when their launch approaches by setting up a well-known multisig with various community members and protocols to govern the protocol.

2.2 Domain

`Domain` is a single NFT contract which represents all domain names within the Avvy ecosystem. It allows for leasing contracts (eg. domain name sale contracts and auctioning contracts) to register new domain names for users. These users will then be given a transferable NFT which represents their domain name.

Domain names are identified by an ID which is a long, random-looking number. The domain name is in fact the hash of the complete domain name and its namespace. The whole Avvy system does not in fact store the complete domain name. This allows for a certain degree of privacy as users are not required to disclose the actual domain name on-chain.

To read more about the lifecycle of a domain, users can refer to the following Avvy blog post: https://avvy.domains/blog/name-lifecycle/.

2.2.1 Privileged Functions

- `suspend [ SUSPENSION_AGENT ]`
- `register [ LEASING_AGENT ]`
- `recycle [ RECYCLING_AGENT ]`
- `revoke [ REVOCATION_AGENT ]`
- `setBaseTokenURI`
- `setContractURI`
- `grantRole`
- `revokeRole`
- `renounceRole`

2.2.2 Issues & Recommendations

Issue #02	**The `register` presently does not ensure that the namespace hash in the name hash in fact refers to the `namespaceId` provided in the `register` function which allows users to register a domain from a different namespace to a specific namespace**
Severity	HIGH SEVERITY
Description	The `register` function does not in any way validate that the domain name has in fact the namespace which the user claims it has. This allows a user to lock in a specific domain name using the rules and prices of a different namespace. This issue has been rated high severity given the fact that different domain names might have highly different rules and this issue would allow a malicious party to squat on all domain names of a specific namespace by minting them in a different, less expensive, `namespaceId`.
Recommendation	Consider validating that the namespace hash matches a newly exposed public signal in the ZK-proof, similar to how the name hash is validated.
Resolution	RESOLVED The client has added a public `namespaceId` signal to the zk-proof that validates that the zk-proof provided inputs in fact reside in this `namespaceId`. This signal is then required to be equal to the `namespaceId` provided to the function. This logic was added within the `ConstraintsAVAXV1` contract.

Issue #03	Users could release their domain during the recycle period
Severity	MEDIUM SEVERITY
Description	Currently, the recycle period is not checked during `register`. This means that the user can just increase their lease during the recycle period. The purpose of the recycle period is to give the domain to the winner of the auction. The current owner of the auctioned domain can thus cancel it and regain access to its domain when it shouldn't be possible.
Recommendation	Consider accounting for the recycle period inside the `register` function.
Resolution	RESOLVED The codebase now prevents domain extension during the recycle period. The client explains this requirement as follows: During the grace period, the current registrant should be able to renew the registration. Their domain has expired, and is somewhat "disabled", but they have a chance to renew it. However, during the recycling period, they should not be able to renew.

Issue #04	Phishing risk: Usage of `tx.origin`
Severity	MEDIUM SEVERITY

Location	Line 228
	`require(owner == tx.origin, "Only owner can lease during grace period");`
	
	Line 238
	`require(owner == tx.origin, "Only owner can modify registration if leased");`

Description	In various locations of the codebase, `tx.origin` is used to validate the authorization of the registrant. This is a risk as malicious contracts could phish the user in calling a function on them, and then use their `tx.origin` authority to execute actions in their name.
Recommendation	Consider using a different form of authentication compared to `tx.origin`. For example, the agents could take care of the authorization instead, or a `requester` parameter could be provided to `register`.
Resolution	RESOLVED The `registrant` parameter is now used instead of `tx.origin`, thus `LeasingAgent` should be very careful and validate `registrant`.

Issue #05	**`register` might attempt to make a transfer during the grace period while this is not permitted**
Severity	MEDIUM SEVERITY

Location	Line 230
	`_safeTransfer(owner, to, name, '');`
	
	Line 299
	`require(!_isGracePeriod(name, namespaceId, namespace), "Cannot transfer expired domain");`

Description	The `register` function will allow the owner to transfer the domain name to a different address during the grace period, however, this is not permitted by the NFT itself in a later line of code. The logic at line 230 therefore seems useless.
Recommendation	Consider whether users should be able to transfer during register calls in the grace period. If not, consider simply not having the transfer logic in the `register` function.
Resolution	RESOLVED The client has indicated that the user should still be allowed to extend throughout the grace period. What is not permitted is to extend during the recycle period. As this was previously not validated, the client has now instated validation for this requirement. The client explains this requirement as follows: During the grace period, the current registrant should be able to renew the registration. Their domain has expired, and is somewhat "disabled", but they have a chance to renew it. However, during the recycling period, they should not be able to renew.

Issue #06	**`register` and `recycle` do not adhere to checks-effects-interactions**
Severity	LOW SEVERITY
Location	Line 242 `_leaseExpiresAt[name] = block.timestamp + leaseLength;` Line 269 `_leaseExpiresAt[name] = block.timestamp + leaseLength;`
Description	Presently, the `register` and `recycle` functions are not written in the checks-effects-interactions pattern as the `_safeTransfer` and `_safeMint` hooks create an external call to the recipient and line 242 (respectively 269) still does an effect. This has the side-effect that during the external call, the user will already have the NFT but can execute actions on a wrongly-set `_leaseExpiresAt` time. At the time of this audit Paladin did not find serious ways to exploit this hence this issue was rated as low severity. We still highly recommend resolving it.
Recommendation	Consider moving line 242 to above the `if` statement where the `_safeMint` occurs (above line 210).
Resolution	RESOLVED Both functions now adhere to checks-effects-interactions.

Issue #07	**`_contractRegistry` is `private`**
Severity	LOW SEVERITY
Description	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
Recommendation	Consider marking the variable as public.
Resolution	RESOLVED

Issue #08	Lack of indexing for event parameters
Severity	INFORMATIONAL
Description	Essential identifying parameters within events should be marked as indexed. This allows for off-chain components to filter events only including these values.
Recommendation	Add indices to the key variables within the events you might want to filter on.
Resolution	RESOLVED

Issue #09	Lack of events for `setBaseTokenURI` and `setContractURI`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above functions. The client should also consider adding functions to permanently lock the `baseTokenURL` and `contractURI`, as users appreciate the fact that NFT URIs can be locked in the spirit of decentralization.
Resolution	RESOLVED

Issue #10	Various functions can be made external
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the external keyword. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases. The following functions can be marked as external: - `exists` - `setBaseTokenURI` - `setContractURI` - `contractURI` - `getDomainExpiry` - `suspend` - `isSuspended` - `register` - `recycle` - `revoke` - `getNamespaceId`
Recommendation	Consider marking the functions mentioned above as external.
Resolution	RESOLVED

Issue #11	Typographical errors
Severity	INFORMATIONAL

Description	The contract contains a number of typographical errors which we have consolidated below in a single issue in an effort to keep the report size reasonable.
	
	Line 199
	`require(_domainToNamespace[name] == namespaceId, "Namespace mismatch");`
	
	`_checkNameMatchesNamespace` should be used here for more consistency, as it is used throughout the other functions.
	
	Line 309
	`return super.supportsInterface(interfaceId);`
	
	The `supportsInterface` should support all three parent interfaces, not just one of them. It should return the union of the three. Note that OpenZeppelin has a library for EIP165 which can be considered where you can register different interfaces to support in a mapping.
	
	Line 312
	`constructor(string memory name, string memory symbol, address contractRegistryAddress) ERC721(name, symbol) {`
	
	The `contractRegistryAddress` parameter can be made of type `ContractRegistryInterface`.

Recommendation	Consider fixing the typographical errors.
Resolution	RESOLVED

Issue #12	Unused functions: `_isLeased`
Severity	INFORMATIONAL
Description	Functions which are defined in a contract but remain unused could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the function to keep the contract short and simple.
Resolution	RESOLVED The function has been removed.

Issue #13	`_contractRegistry` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variable explicitly immutable.
Resolution	RESOLVED

2.3 LeasingAgentV1

The LeasingAgentV1 contract allows for users to register new domain names. Users can call the `register` and `registerWithPreimage` methods to purchase a domain name with AVAX.

To allow for a fair distribution of initial domain names, a basic dutch auction mechanism is included where over the initial distribution period, domain names gradually go down in price over a set period of time. This will allow for a fairer outcome as people who desire to purchase a specific domain are given the opportunity to do so at an increased price.

2.3.1 Privileged Functions

- `enable [ MANAGER ]`
- `setRegistrationPremiumDetails [ MANAGER ]`
- `grantRole`
- `revokeRole`
- `renounceRole`

2.3.2 Issues & Recommendations

Issue #14	Registration prices are not multiplied by the lease duration
Severity	HIGH SEVERITY
Description	Users can lease a domain for up to 5 years. Presently however, the price is the same as one year as the codebase does not multiply the required price by the amount of years leased.
Recommendation	Consider multiplying the price by the amount of years leased.
Resolution	RESOLVED The prices are now multiplied by the lease "quantity".

Issue #15	commit-reveal scheme does not protect against frontrunning
Severity	MEDIUM SEVERITY
Description	The contract contains a commit-reveal scheme where users first create a transaction with an intent to purchase a domain name and later create the actual purchase. The client has explained this two step flow is desired as this way the purchase cannot be frontrun if bots see the purchase in the mempool. However, a smart bot can just quickly create two transactions in a single transaction through a smart contract, causing the current design of the commit-reveal scheme to be futile.
Recommendation	Consider whether frontrunning protection is desired, if so consider one of the design methodologies like a commit-reveal-wait-claim scheme or redesigning the leasing agent to always auction in dutch auction.
Resolution	RESOLVED The commit-reveal scheme has been removed in favor of relying on the domain name zero-knowledge proof generation to be a large enough friction for frontrunning to not be a large concern. The client has indicated they will include countermeasures if frontrunning ever does become a concern.

Issue #16	Various variables are private
Severity	LOW SEVERITY
Description	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts. The following variables are private: - `_namespaceId` - `_enabled` - `_commits` - `_premiumStartTime` - `_premiumEndTime` - `_premiumStartPrice`
Recommendation	Consider marking the variables as public.
Resolution	RESOLVED

Issue #17	`register` does not adhere to checks-effects-interactions
Severity	LOW SEVERITY
Description	The `domain.register` call directly allows the user to execute arbitrary code as it does a `_safeMint` to the user. This poses a risk as the user might be able to abuse the system given that there is still code to be executed after the `_registerName` calls are made. This issue is rated as low severity as Paladin was unable to abuse this reentrancy vector.
Recommendation	Consider rewriting the `register` function to do the `domain.register` calls at the bottom of the function.
Resolution	RESOLVED The function now adheres to checks-effects-interactions.

Issue #18	Phishing risk: User is allowed to overpay for domains
Severity	LOW SEVERITY
Location	Line 117 `require(msg.value >= total, "insufficient payment");`
Description	The smart contract presently solely checks that the user provided sufficient AVAX to pay for a domain name lease, it does not validate that they provided the exact amount. A malicious frontend could therefore send too much AVAX.
Recommendation	Consider implementing a refund mechanism — this will prove especially useful for the Dutch auction.
Resolution	RESOLVED `refund` logic has been instated.

Issue #19	Various functions can be made external
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the external keyword. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases. The following functions can be made external: - `enable` - `commit` - `setRegistrationPremiumDetails` - `registerWithPreimage`
Recommendation	Consider marking the functions above as external.
Resolution	RESOLVED

Issue #20	`MAX_YEARS` and `oneYear` can be made constant
Severity	INFORMATIONAL
Description	Variables that are never modified can be indicated as such with the constant keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variables explicitly constant.
Resolution	RESOLVED

Issue #21	Various functions lack events
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications. The following functions lack an event: - `enable` - `commit` - `setRegistrationPremiumDetails` - `register`
Recommendation	Add events for the above functions.
Resolution	RESOLVED

Issue #22	Gas optimization: Various variables can be made callable throughout the contract
Severity	INFORMATIONAL
Description	The `register` and `registerWithPreimage` functions contain various variables that can be made `calldata` to save on gas.
Recommendation	Consider marking all the variables as `calldata` to save on gas.
Resolution	RESOLVED

Issue #23	Lack of validation
Severity	INFORMATIONAL

Description	The contract contains functions with parameters which are not properly validated. Having unvalidated parameters could allow the governance or users to provide variable values which are unexpected and incorrect. This could cause side-effects or worse exploits in other parts of the codebase.
	
	Consider validating the following function parameters:
	
	Lines 37-40
	`function setRegistrationPremiumDetails(uint256 premiumStartTime, uint256 premiumEndTime, uint256 premiumStartPrice)`
	
	The premiumEndTime should be after the premiumStartTime.
	
	Line 91
	`uint256[] memory quantities,`
	
	The code contains various length checks but the quantities length is not validated.
	
	Line 133
	`require(preimages.length / names.length == 4, "LeasingAgent: incorrect preimage length");`
	
	A modulo check should be done to ensure that preimages.length is divisible by four.

Recommendation	Consider validating the function parameters mentioned above.
Resolution	RESOLVED

Issue #24	`_namespaceId` and `_contractRegistry` can be made immutable
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variables explicitly immutable.
Resolution	RESOLVED

2.4 ContractRegistryV1

`ContractRegistryV1` acts as a central database that keeps track of all contracts within the Avvy system. The managers of the contract (the governance can promote multiple wallets to manager) are allowed to link a contract name to a specific contract address.

`ContractRegistryV1` is used by almost all contracts as most contracts do not store any addresses except for the registry. This means that whenever contract A wants to communicate with contract B, it will call `ContractRegistryV1.get("B")` to figure out the address of B.

`ContractRegistryV1` therefore kind of acts like a nameservice within the Avvy system.

2.4.1 Privileged Functions

- `set [ MANAGER ]`
- `grantRole`
- `revokeRole`
- `renounceRole`

2.4.2 Issues & Recommendations

Issue #25	Usage of `tx.origin` is discouraged
Severity	INFORMATIONAL

Location	Lines 25-27
	`constructor() {`
	`  _setupRole(DEFAULT_ADMIN_ROLE, tx.origin);`
	`}`

Description	The `admin` role is granted to the wallet that instantiates the deployment transaction of the `ContractRegistryV1`. This is generally discouraged as usage of `tx.origin` can be more difficult to reason about.
Recommendation	Consider adding an address `admin` parameter. This issue can also be resolved on the note that the client is comfortable with using `tx.origin` here.
Resolution	RESOLVED The recommendation has been implemented.

Issue #26	Lack of events for set
Severity	INFORMATIONAL
Description	Function that affects the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the function.
Resolution	RESOLVED

Issue #27	**`set` can be made external**
Severity	INFORMATIONAL
Description	Function that is not used within the contract but only externally can be marked as such with the external keyword. Apart from being a best practice when the function is not used within the contract, this can lead to lower gas usage in certain cases.
Recommendation	Consider marking the functions above as external.
Resolution	RESOLVED

Issue #28	**Gas optimization: `contractName` can be made callable throughout the contract**
Severity	INFORMATIONAL
Description	Once the `set` function has been changed to external, the `contractName` parameters can be marked as `calldata` to save gas.
Recommendation	Consider marking the `contractName` parameters as `calldata`.
Resolution	RESOLVED

2.5 ConstraintsAVAXV1

The ConstraintsAVAXV1 contract allows for the Avvy nameservice to assess that the names adhere to certain requirements.

The following requirements will be enforced on launch:

- name must not be blocked via `blockNames` method
- domain name must end in NUL characters (these signal the end of the name)
- domain name must have a max length of 62 letters
- domain name must have a min length of 3 letters
- letters must all be a-z, 0-9, or "-" (and NUL).
- domain name must not start with xn-- as Avvy does not plan to initially support IDN notation
- the hash is the hash of the label these assertions were made on plus the namespace

As domain names never need to be submitted on-chain for privacy reasons, the constraints are validated through a user-submitted zero knowledge proof.

2.5.1 Privileged Functions

- `blockNames [ MANAGER ]`
- `setVerifier [ MANAGER ]`
- `grantRole`
- `revokeRole`
- `renounceRole`

2.5.2 Issues & Recommendations

Issue #29	Admins are not able to unblock names
Severity	LOW SEVERITY
Description	The `blockNames` function currently only allows to block names, but there is no way to unblock them. If an admin blocks a name by mistake, that name will be forever locked.
Recommendation	Consider adding a way to unblock names if blocking them forever (even if it was a mistake) is an unwanted feature.
Resolution	RESOLVED The `unblockNames` function has been introduced.

Issue #30	`verifier` is `private`
Severity	LOW SEVERITY
Description	Important variables that third parties might want to inspect should be marked as public so that these third parties can easily inspect them through the explorer, web3 and derivative contracts.
Recommendation	Consider marking the variable as public.
Resolution	RESOLVED

Issue #31	Lack of events for `blockNames` and `setVerifier`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Consider adding an event for the above functions. Additionally, the verifier event should also be emitted within the constructor as it sets the verifier as well.
Resolution	RESOLVED

Issue #32	Gas optimization: `data` and `names` can be made callable throughout the contract
Severity	INFORMATIONAL
Description	The data and names parameters can be marked as `calldata` to save gas.
Recommendation	Consider marking the data parameters as `calldata`.
Resolution	RESOLVED

Issue #33	Typographical errors
Severity	INFORMATIONAL

Description	The contract contains a number of typographic mistakes which we have consolidated below in a single issue in an effort to keep the report size reasonable.
	
	L23
	`require(_blockedNames[name] == false, "name blocked");`
	
	Consider using `!_blockedNames[name]` as it's easier to read and shorter.
	
	L40, L47
	`address verifierAddress`
	
	Consider casting the `verifierAddress` directly to `VerifierInterface`.

Recommendation	Consider fixing the typographical errors.
Resolution	RESOLVED

2.6 NamespaceV1

The NamespaceV1 contract contains governance defined metadata about a specific namespace. Each namespace is identified by an id and contains a gracePeriod, a recyclePeriod and a constraint circuit (which allows to execute a zero-knowledge proof check with certain requirements/constraints) on names within the namespace.

2.6.1 Privileged Functions

- `initNamespace [ MANAGER ]`
- `setGracePeriodLength [ MANAGER ]`
- `setRecyclePeriodLength [ MANAGER ]`
- `setConstraints [ MANAGER ]`
- `grantRole`
- `revokeRole`
- `renounceRole`

2.6.2 Issues & Recommendations

Issue #34	`_initializedNamespaces` and `_constraints` are `private`
Severity	LOW SEVERITY
Description	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
Recommendation	Consider marking the variables as `public`.
Resolution	RESOLVED

Issue #35	Various functions can be made can be made external
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the external keyword. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases. The following functions can be made external: - `initNamespace` - `setGracePeriodLength` - `setRecyclePeriodLength` - `setConstraints` - `checkName`
Recommendation	Consider marking the functions above as external.
Resolution	RESOLVED

Issue #36	The `checkName` function could be a `view` function
Severity	INFORMATIONAL
Description	The `checkName` function is currently not a `view` function. As for now, the `checkName` function within `ConstraintsAVAXV1` is a `view` function, one could argue that `checkName` could be made as a `view` function.
Recommendation	Consider whether there is any reason why the interface should not be a `view` function. If there is no such argument, consider adjusting both the interface and the `NameSpaceV1` contract to have these functions marked as `view`.
Resolution	RESOLVED

Issue #37	Lack of events for `setGracePeriodLength`, `setRecyclePeriodLength` and `setConstraints`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above functions.
Resolution	RESOLVED

Issue #38	Typographical errors
Severity	INFORMATIONAL

Description	The contract contains a number of typographic mistakes which we have consolidated below in a single issue in an effort to keep the report size reasonable.
	
	L18
	`function initNamespace(uint256 id, address constraintsAddress) public onlyRole(MANAGER_ROLE) {`
	
	L50
	`function setConstraints(uint256 id, address constraintsAddress) public onlyRole(MANAGER_ROLE) {`
	
	`constraintsAddress` can be made type `ConstraintsInterface` to avoid casting it later on.

Recommendation	Consider fixing the typographical errors.
Resolution	RESOLVED

2.7 PricingOracleV1

`PricingOracleV1` is a utility contract that fetches the price of a domain name. It contains the logic that prices each domain name based on its length.

The following prices are set in USD and must be paid in AVAX. The ChainLink oracle is used to convert the USD to AVAX.

- 3 letters: $500
- 4 letters: $160
- 5 letters or more: $5

`PricingOracleV1` can only be used in combination with a ChainLink AVAX/USD price feed of 8 decimals (`0x0A77230d17318075983913bC2145DB16C7366156`) which it in fact will use as its encoded in the contract.

To validate that a domain name has at least a user-provided number of digits, a zero-knowledge proof is employed. This means that the user could voluntarily decide to pay more for their domain name, but not less.

2.7.1 Issues & Recommendations

Issue #39	**The user can avoid paying larger amounts for shorter names through providing an unexpected minLength of 2 in the zero-knowledge proof**
Severity	HIGH SEVERITY

Description	A user that wants to pay less for a 3 or 4 letters domain name could do so by providing a cleverly crafted zero-knowledge proof about the length of their domain name.
	
	Essentially the way the zero-knowledge proof of the domain name length works is that the user must provide their "encoded" domain name (we use encoded as terminology here) and some minLength which the user chooses themselves. The proof will pass if the name is longer or equal than the user provided minLength and fail if it is shorter.
	
	`uint256 namePrice = 500;`
	`if (minLength == 3) {`
	`  namePrice = 64000;`
	`} else if (minLength == 4) {`
	`  namePrice = 16000;`
	`}`
	
	However, as mentioned in the code snippet above, the codebase currently assumes that the user will provide a `minLength` greater or equal to three. It does not account for the fact that the user is not constrained at all by the `minLength` variable they provide, it might as well be 0.
	
	If the user for example provides a `minLength` of 2 and registers a domain name of four letters, they would simply need to pay $5 as the zero-knowledge proof correctly validates that four is larger than two while the if statement is not caught as it only checks 3 and 4.

Recommendation	Consider validating `minLength` with a requirement. It is generally best practice to always validate user input to the fullest. We would therefore also recommend validating the `pubSignals` length even though we presently have no exploit vector to abuse it.
Resolution	RESOLVED An explicit requirement for the `minLength` to be at least 3 has been introduced.

Issue #40	`priceFeed` and `verifier` are `private`
Severity	LOW SEVERITY
Description	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
Recommendation	Consider marking the variables as `public`.
Resolution	RESOLVED

Issue #41	`priceFeed` and `verifier` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variables explicitly `immutable`.
Resolution	RESOLVED

Issue #42	Chainlink price feed could return 0
Severity	INFORMATIONAL
Description	The `feedPrice` returned by the Chainlink oracle could be 0. The call would still revert as a division by zero would occur at a later point in time. This requirement should be made explicit in light of not having implicit/accidental security requirements.
Recommendation	Consider requiring that the `priceFeed` is greater than 0.
Resolution	RESOLVED The recommendation has been implemented.

Issue #43	Typographical error
Severity	INFORMATIONAL
Location	L68 `constructor(address verifierAddress) {`
Description	The `verifierAddress` parameter could be cast directly to `VerifierInterface`.
Recommendation	Consider fixing the typographical error.
Resolution	RESOLVED

2.8 RainbowTableV1

The Avvy system is interesting because it does not require users to publish the actual domain name on-chain. Instead, if the user desires privacy, they can simply publish the hash of the name.

For users to understand why this could be valuable*: you could use your real name as a domain name and let someone send funds to you using your real name, however, all that would be published on-chain is someone sending funds to some non-decodable number.

However, many users (eg. websites...) have no need for such privacy and this is where the `RainbowTableV1` contract comes into play. It allows users to publish what name is actually linked to their hashed (non-decodable number) domain name.

*Disclaimer: The example provided above is not really a good use-case for Avvy domain names and its privacy features. The reason for this is that a malicious party can easily hash all names in existence and figure out that the "private" hash that is in fact shared is the one linked to your real name. Ironically, this is what a rainbow table tends to be used for in the real world. Users should therefore be careful with the privacy preserving assumptions they make when they decide not to register their domain name on-chain.

2.8.1 Issues & Recommendations

Issue #44	**Empty or uneven length preimages are not hashable but permitted which could allow for abuse**
Severity	MEDIUM SEVERITY
Description	`for (uint256 i = 0; i < preimage.length; i += 1) { if (i % 2 == 0) { hash = pos.poseidon([hash, preimage[i], preimage[i+1]]); } }` Within the current contract design, it is impossible to hash an empty or uneven length `preimage`. This issue has been marked as medium as an honest reveal with a `preimage` length N (with N divisible by two) could be griefed by submitting a `preimage` of length N + 1. This new submission would still pass as the reveal function does not enforce even length but the last element is completely ignored which means it overrides the hash.
Recommendation	Consider adjusting the business logic to allow for these cases in case they are going to be present in the implementation. Consider enforcing an even length. One could also consider optimizing the for loop mentioned above to increment by 2 which would allow for removing the modulus operation.
Resolution	RESOLVED The `preimage` must now be divisible by two and greater than zero.

Issue #45	Hashes are theoretically mutable
Severity	LOW SEVERITY
Description	Presently, the `_getHash` method is not pure. It also calls the contracts registry to find the hashing contract. As a user can presently call reveal multiple times, this could cause a hash to be linked to a different `preimage`.
Recommendation	Consider whether this is desired. If not, consider only allowing reveal to be called if the current `preimage` has a length of zero.
Resolution	RESOLVED A zero length check is now made before a reveal can happen preventing them from being set twice.

Issue #46	`contractRegistry` and `_getHash` are private
Severity	LOW SEVERITY
Description	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
Recommendation	Consider marking the variables as public.
Resolution	RESOLVED

Issue #47	`lookup` and `reveal` can be made `external`
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only externally can be marked as such with the `external` keyword. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases.
Recommendation	Consider marking the functions above as `external`.
Resolution	RESOLVED

Issue #48	`contractRegistry` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variable explicitly `immutable`.
Resolution	RESOLVED

Issue #49	Lack of indexing for event parameters
Severity	INFORMATIONAL
Description	Essential identifying parameters within events should be marked as indexed. This allows for off-chain components to filter events only including these values. `event Revealed(uint256 hash);`
Recommendation	Add indices to the key variables within the events you might want to filter on.
Resolution	RESOLVED

Issue #50	Gas optimization: `preimage` can be made callable throughout the contract
Severity	INFORMATIONAL
Description	Once the functions have been changed to external, the preimage parameters can be marked as `calldata` to save gas.
Recommendation	Consider marking the preimage parameters as calldata in both reveal and _getHash (this might require _getHash to be made private).
Resolution	RESOLVED

Issue #51	Typographical error
Severity	INFORMATIONAL
Description	L25 `// indicies in the preimage[] array.` This line should say indices.
Recommendation	Consider fixing the typographical error.
Resolution	RESOLVED


PALADIN
BLOCKCHAIN SECURITY