PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Avvy Domains
(Reverse Resolvers)

29 August 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's Reverse Resolvers contracts 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
`EVMReverseResolverV1`	`0xF4A1328B2d3BFd7aca965B6CcB688F1BE54838D0`	✓ MATCH
`ReverseResolverRegistryV1`	`0x87388F6EAAfA4bB970EEefd97D29e487949fBbBd`	✓ MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
🔴 High	2	2	-	-
🟠 Medium	0	-	-	-
🟡 Low	2	2	-	-
🟣 Informational	9	8	-	1
Total	**13**	**12**	**-**	**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 EVMReverseResolverV1

ID	Severity	Summary	Status
01	HIGH	Previous entries can reset details of the entry of the new owner when they unset their entry	✓ RESOLVED
02	HIGH	Previous entries will be maintained and cannot be unset by the new owner	✓ RESOLVED
03	LOW	The `entries` mapping does not revert when querying a suspended or expired domain	✓ RESOLVED
04	INFO	The returned domain (or subdomain) of an address might be different from the returned address of the domain name when querying the `publicResolver` with that same domain (or subdomain)	ACKNOWLEDGED
05	INFO	`set`, `clear` and get can be made `external`	✓ RESOLVED
06	INFO	Gas optimizations	✓ RESOLVED
07	INFO	Unused import: `PoseidonInterface.sol`	✓ RESOLVED

1.3.2 ReverseResolverRegistryV1

ID	Severity	Summary	Status
08	LOW	Owner should likely always be able to write on its domain	✓ RESOLVED
09	INFO	Lack of events for `setResolver`	✓ RESOLVED
10	INFO	`setResolver`, `getResolver`, `canWrite` and `setAuthenticator` can be made	✓ RESOLVED
11	INFO	Gas optimizations	✓ RESOLVED
12	INFO	Unused import: PoseidonInterface.sol	✓ RESOLVED
13	INFO	Typographical errors	✓ RESOLVED

2 Findings

2.1 EVMReverseResolverV1

`EVMReverseResolverV1` allows users to link their address to their domain or subdomain. One address can only link to one domain or subdomain and one domain or subdomain can only link to a single address. It should be noted that the address linked to a domain name might be different from the address returned by the `PublicResolver`.

The reverse resolvers could be used by users of a GameFi project to allow the game to get their domain name and pull certain information such as an avatar. It could also allow users to display their online nickname (as an Avvy domain) in the explorer, therefore allowing domain owners to link their address to their (sub)domain or link addresses whitelisted by the authenticator to their (sub)domain.

Only the owner of the domain or addresses that were whitelisted in the authenticator contract (that was previously set by the owner) can call `set` to link their address to the related domain or subdomain. If a user wants to link their address to a subdomain, they need to use the `path` parameter. If a path is provided, it will be hashed by the `RaindowTable` contract that was audited during a previous audit by Paladin. If no path is provided, the hash will be the name itself.

To reset the link to its address, the user needs to call `clear`. Only the user can clear the link to its own address, and even the owner will not be able to reset other's links.

Key	Name	Label	Description
1	X_CHAIN	Address on Avalanche X-Chain	Address on Avalanche X-Chain
2	P_CHAIN	Address on Avalanche P-Chain	Address on Avalanche P-Chain
3	EVM	C-Chain / EVM Address	Address on EVM-type network, including Avalanche C-Chain
4	VALIDATOR	Validator NodeID	Validator NodeID on the Avalanche Network
5	DNS_CNAME	DNS CNAME Record	DNS CNAME Record
6	DNS_A	DNS A Record	DNS A Record
7	AVATAR	Avatar	An image which the user wishes to use as their avatar. Value should be a URL which references the image.
8	CONTENT	Content	A downloadable file. Value should be a URL (e.g. IPFS, HTTPS, ..) which references the image.
9	PHONE	Phone number	A telephone number. Should conform to the "tel" URL scheme defined in RFC2806.

2.1.1 Privileged Functions

- `set [ owner or authentified address ]`
- `clear [only the msg.sender that previously set this entry]`

2.1.2 Issues & Recommendations

Issue #01	Previous entries can reset details of the entry of the new owner when they unset their entry
Severity	● HIGH SEVERITY

Location	L34 - 37
	`Entry memory currEntry = reverseLookups[msg.sender];`
	`if (currEntry.name != 0 && currEntry.hash != 0) {`
	`  entries[currEntry.name][currEntry.hash] = address(0);`
	`}`

Description	Unlike `entries`, the `reverseLookups` value is not reset when `set` is called. This allows the previous address that points to the (sub)domain to reset the new entry by calling `set` on another domain, which causes it to reset the entry of the new owner. A proof of concept to conduct this exploit is provided: 1. Alice owns `domain1.avax` and `domain2.avax` 2. Alice reverse resolves `domain1.avax` to her address, updates entries 3. Alice sells `domain1.avax` to Bob 4. Bob reverse resolves `domain1.avax` to his address, overwrites and updates entries 5. Alice reverse resolves `domain2.avax` to her address 6. Bob is angry because his entries mapping just got unset!

Recommendation	Consider resetting the previous value of the `reverseLookups` when a new one is set.
	
	L35
	`if (currEntry.name != 0 && currEntry.hash != 0) {`
	`reverseLookups[entries[currEntry.name][currEntry.hash]] = Entry(0, 0);`
	`  entries[currEntry.name][currEntry.hash] = address(0);`
	`}`

Resolution	✓ RESOLVED

Issue #02	**Previous entries will be maintained and cannot be unset by the new owner**
Severity	🔴 HIGH SEVERITY
Description	Previous entries that were set prior to the transfer of ownership will be maintained and cannot be unset by the new owner directly. Even if the owner changes the authenticator, this issue still remains. However, if the first issue is fixed, the owner could use `set` to set all the different domains and subdomains that were used to himself, and reset them one by one.
Recommendation	Consider allowing the owner to clear the different addresses directly if this is not desired behavior. Consider allowing `clear()` to also reset the previous `reverseLookups` entry.
Resolution	✓ RESOLVED

Issue #03	**The `entries` mapping does not revert when querying a suspended or expired domain**
Severity	🟡 LOW SEVERITY
Description	Unlike the `get` function that reverts if the domain name is suspended or expired, the `entries` mapping does not. Additionally, it should be noted that it does not revert if the entry was not set either — in this case, it returns `address(0)`.
Recommendation	Consider making the `entries` mapping `private` and adding a getter function that behaves the same way the `get` function does if this behavior is not expected.
Resolution	✓ RESOLVED

Issue #04	**The returned domain (or subdomain) of an address might be different from the returned address of the domain name when querying the `publicResolver` with that same domain (or subdomain)**
Severity	● INFORMATIONAL
Description	The returned domain of an address when using `get` on this contract and the returned address of the same domain using the `publicResolver` contract might be different. It might be desired that the reverse resolver and the public resolver directly map 1:1 to each other in some way. This is presently not enforced.
Recommendation	Consider fixing this behavior if it is not desired.
Resolution	● ACKNOWLEDGED The client indicated that this is desired behavior.

Issue #05	**`set`, `clear` and `get` 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 lower gas usage in certain cases.
Recommendation	Consider marking the functions mentioned above as `external`.
Resolution	✓ RESOLVED

Issue #06	Gas optimizations
Severity	INFORMATIONAL
Location	L12 `ContractRegistryInterface contractRegistry;`
Description	Consider marking `contractRegistry` as immutable to save some gas as it will hardcode the value in the bytecode during deployment. It should also be made `public` to improve variable readability for users. `calldata` can also be used throughout the contract to save on gas.
Recommendation	Consider implementing the gas optimizations mentioned above.
Resolution	✓ RESOLVED

Issue #07	Unused import: `PoseidonInterface.sol`
Severity	INFORMATIONAL
Location	L6 `import "../PoseidonInterface.sol";`
Description	Files imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the import to keep the contract short and simple.
Resolution	✓ RESOLVED

2.2 ReverseResolverRegistryV1

`EVMReverseResolverV1` allows users to set an authenticator to allow other addresses to set entries on the different `reverseResolvers`. This authenticator is a custom contract and can choose what would be accepted or not. For example, other users may be unable to set the main domain and only subdomain or even specific domain names. Additionally, it allows users to get the resolver addresses depending on the `standardKey` provided.

Only addresses that were granted the `MANAGER` role can set the `reverseResolver` address.

To reset the authenticator, the user needs to use `setAuthenticator` with their domain name and `address(0)` as the `contractAddress`.

At the time of this audit, there is only one resolver, the `EVMReverseResolverV1`, which is included within the scope of this audit.

2.2.1 Privileged Functions

- `setResolver [ only MANAGER role ]`
- `grantRole`
- `revokeRole`
- `renounceRole`

2.2.2 Issues & Recommendations

Issue #08 — **Owner should likely always be able to write on its domain**

Severity

● LOW SEVERITY

Location

L44 - 49

```
if (authenticators[name] != address(0)) {
  ReverseResolverAuthenticatorInterface authenticator =
ReverseResolverAuthenticatorInterface(authenticators[name]);
  return authenticator.canWrite(name, path, sender);
} else {
  return owner == sender;
}
```

Description

Currently, a badly implemented authenticator could return `false` on the owner and prevent them from doing actions that require `canWrite`.

This issue is only rated as low as the owner can always set the authenticator to `address(0)` to circumvent this issue. It should be noted that explicitly allowing the owner is therefore also not a privilege escalation.

Recommendation

Consider checking if the sender is the owner first, then if an authenticator has been set.

```
if (owner == sender) {
  return true;
}
if (authenticators[name] != address(0)) {
  ReverseResolverAuthenticatorInterface authenticator =
ReverseResolverAuthenticatorInterface(authenticators[name]);
 return authenticator.canWrite(name, path, sender);
}
return false;
```

Resolution

✓ RESOLVED

Issue #09	**Lack of events for `setResolver`**
Severity	● INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Consider adding events for the function.
Resolution	✓ RESOLVED

Issue #10	**`setResolver`, `getResolver`, `canWrite` and `setAuthenticator` 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 lower gas usage in certain cases.
Recommendation	Consider marking the functions mentioned above as external.
Resolution	✓ RESOLVED

Issue #11	Gas optimizations
Severity	● INFORMATIONAL

Description	L17
	`ContractRegistryInterface public contractRegistry;`
	
	Consider marking `contractRegistry` as `immutable` to save some gas as it will hardcode the value in the bytecode during deployment.
	
	L30-31
	`require(reverseResolvers[standardKey] != address(0), 'ReverseResolverRegistryV1: address not set');`
	`return reverseResolvers[standardKey];`
	
	The return `reverseResolvers[standardKey]` value should be cached in order to save gas.
	
	L44-45
	`if (authenticators[name] != address(0)) {`
	`    ReverseResolverAuthenticatorInterface authenticator = ReverseResolverAuthenticatorInterface(authenticators[name]);`
	
	The `authenticators[name]` value should be cached in order to save gas.
	
	`calldata` can also be used throughout the contract to save on gas.

Recommendation	Consider implementing the gas optimizations mentioned above.
Resolution	✓ RESOLVED

Issue #12	Unused import: PoseidonInterface.sol
Severity	● INFORMATIONAL
Location	L5 `import "./PoseidonInterface.sol";`
Description	Files imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the import to keep the contract short and simple.
Resolution	✓ RESOLVED

Issue #13	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.
	
	L20
	`mapping(uint256 => address) public reverseResolvers;`
	
	The `reverseResolvers` should be set as private as there is a getter that reverts on `address(0)`. Having two different getters might be misleading.
	
	L23
	`mapping(uint256 => address) public authenticators;`
	
	The resolvers should be cast directly to the right type, `ReverseResolverAuthenticatorInterface`, instead of as an `address` and casting it after.
	
	L54-57
	`require(!domain.isSuspended(name), "ReverseResolverRegistry: domain suspended");`
	`require(expiresAt > block.timestamp, "ReverseResolverRegistry: domain expired");`
	`require(domain.ownerOf(name) == msg.sender, "ReverseResolverRegistry: not domain owner");`
	
	The comment should mention `ReverseResolverRegistryV1` instead of `ReverseResolverRegistry` to be consistent with the name of the contract and the line 30.

Recommendation	Consider fixing the typographical errors.
Resolution	✓ RESOLVED

PALADIN

BLOCKCHAIN SECURITY