



CFG NINJA AUDITS

Security Assessment

Bunny AI Token

February 4, 2023



Table of Contents

1 Assessment Summary

2 Technical Findings Summary

3 Project Overview

3.1 Token Summary

3.2 Risk Analysis Summary

3.3 Main Contract Assessed

4 Smart Contract Risk Checks

4.1 Mint Check

4.2 Fees Check

4.3 Blacklist Check

4.4 MaxTx Check

4.5 Pause Trade Check

5 Contract Ownership

6 Liquidity Ownership

7 KYC Check

8 Smart Contract Vulnerability Checks

8.1 Smart Contract Vulnerability Details

8.2 Smart Contract Inheritance Details

8.3 Smart Contract Privileged Functions

9 Assessment Results and Notes(Important)

10 Social Media Check(Informational)

11 Technical Findings Details

12 Disclaimer



Assessment Summary

This report has been prepared for Bunny AI Token on the Ethereum network. CFGNINJA provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Thorough line-by-line manual review of the entire codebase by industry experts.








Technical Findings Summary

Classification of Risk

Severity	Description
 Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
 Major	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
 Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
 Minor	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
 Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
 Critical	0	0	0
 Major	2	2	0
 Medium	0	0	0
 Minor	3	2	1
 Informational	1	1	0
Total	6	6	0



Project Overview

Token Summary

Parameter	Result
Address	0xC3158937C9E3DFA27267529b8ac429240e6fE9e7
Name	Bunny AI
Token Tracker	Bunny AI (\$BUNAI)
Decimals	18
Supply	23,000,000
Platform	Ethereum
compiler	v0.8.17+commit.8df45f5f
Contract Name	BunnyAI
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/token/0xC3158937C9E3DFA27267529b8ac429240e6fE9e7#code
Payment Tx	0x7c01448176f2e9c940183803a4b648038cc99df24f0b796575b3ddf549ebe89f



Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	8%
Sale Tax	10%
Is honeypot?	Clean
Can edit tax?	Yes
Is anti whale?	Yes
Is blacklisted?	No
Is whitelisted?	Yes
Holders	1
Confidence Level	Medium

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.



Main Contract Assessed Contract Name

Name	Contract	Live
Bunny AI	0xC3158937C9E3DFA27267529b8ac429240e6fE9e7	Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
Bunny AI	0x3c0e473649633f656c66ae4ffdc5a94afba3975d	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
bunny	fec5ba77051557baeaaa8b0a3f1e9901feef6921	bunnyv2.sol



Mint Check

The project owners of Bunny AI do not have a mint function in the contract, owner cannot mint tokens after initial deploy.

The Project has a Total Supply of 23,000,000 and cannot mint any more than the Max Supply.

Mint Notes:

Auditor Notes:

Project Owner Notes:



Owner can't mint new coins



Fees Check

The project owners of Bunny AI do not have the ability to set fees higher than 25% .

The team May have fees defined; however, they can't set those fees higher than 25% or may not be able to configure the same.

Tax Fee Notes:

Auditor Notes: The contract currently has 8% buy and 10% sale taxes, and cannot be set higher than 12%.

Project Owner Notes:



Fees can be changed up to a maximum of 25%



Blacklist Check

The project owners of Bunny AI do not have a blacklist function their contract.

The Project allow owners to transfer their tokens without any restrictions.

Token owner cannot blacklist the contract: Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

Blacklist Notes:

Auditor Notes: blacklist is set for those tx origin from early buy

Project Owner Notes: undefined



MaxTx Check

The Project Owners of Bunny AI can set max tx amount.

The ability to set MaxTx can be used as bad actor, this can limit the ability of investors to sale their tokens at any given time if is set too low..

We recommend the project to set MaxTx to Total Supply or simiar to avoid swap or transfer from failures

MaxTX Notes:

Auditor Notes: `uint256 public maxBuyLimit = 230_000 * 10**18;uint256 public maxSellLimit = 230_000 * 10**18; uint256 public maxWalletLimit = 230_000 * 10**18;`

Project Owner Notes: Max TX is done to avoid bots. Customer Reply to issue.

Project Has MaxTX



Pause Trade Check

The Project Owners of Bunny AI can stop or pause trading

We recommend the Team only allow Open Trade and never use Stop Trade, as this will be catastrophic for the Project and Investors.

We recommend the Team create a new contract without the stop trade function.

Pause Trade Notes:

Auditor Notes: Open/Pause is capable on contract

Project Owner Notes: Just in case we need to migrate, however may renounce contract.

Owner can pause trading



Contract Ownership

The contract ownership of Bunny AI is not currently renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address
0x4517E6EF52Cab97EB7f732dA634BabdbFA55c4Ec
which can be viewed:
[HERE](#)

The owner wallet has the power to call the functions displayed on the privileged functions chart below, if the owner's wallet is compromised, they could exploit these privileges.

We recommend the team renounce ownership at the right time, if possible, or gradually migrate to a timelock with governing functionalities regarding transparency and safety considerations.

We recommend the team use a Multisignature Wallet if the contract is not going to be renounced; this will give the team more control over the contract.



Liquidity Ownership

Most of the liquidity is currently locked; the lock can be seen here:

Liquidity Locker Link can be viewed from:
[HERE](#)



KYC Information

The Project Owners of Bunny AI have provided KYC Documentation.

KYC Certificated can be found on the Following:
KYC Data

KYC Information Notes:

Auditor Notes: KYC Completed

Project Owner Notes:



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	bunnyv2.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	bunnyv2.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	bunnyv2.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	bunnyv2.sol	L: 7 C: 0
SWC-104	Pass	Unchecked Call Return Value.	bunnyv2.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	bunnyv2.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	bunnyv2.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	bunnyv2.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	bunnyv2.sol	L: 405 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	bunnyv2.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	bunnyv2.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	bunnyv2.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	bunnyv2.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	bunnyv2.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	bunnyv2.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	bunnyv2.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	bunnyv2.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	bunnyv2.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	bunnyv2.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	bunnyv2.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randomness.	bunnyv2.sol	L: 715 C: 24
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	bunnyv2.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	bunnyv2.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	bunnyv2.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	bunnyv2.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	bunnyv2.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	bunnyv2.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	bunnyv2.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	bunnyv2.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	bunnyv2.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	bunnyv2.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	bunnyv2.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	bunnyv2.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	bunnyv2.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	bunnyv2.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	bunnyv2.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	bunnyv2.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.



Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (<https://github.com/ethereum/solidity/releases>) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.



Smart Contract Vulnerability Details

SWC-120 – Weak Sources of Randomness from Chain Attributes

CWE-330: Use of Insufficiently Random Values

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

Remediation:

Using commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

References:

How can I securely generate a random number in my smart contract?)

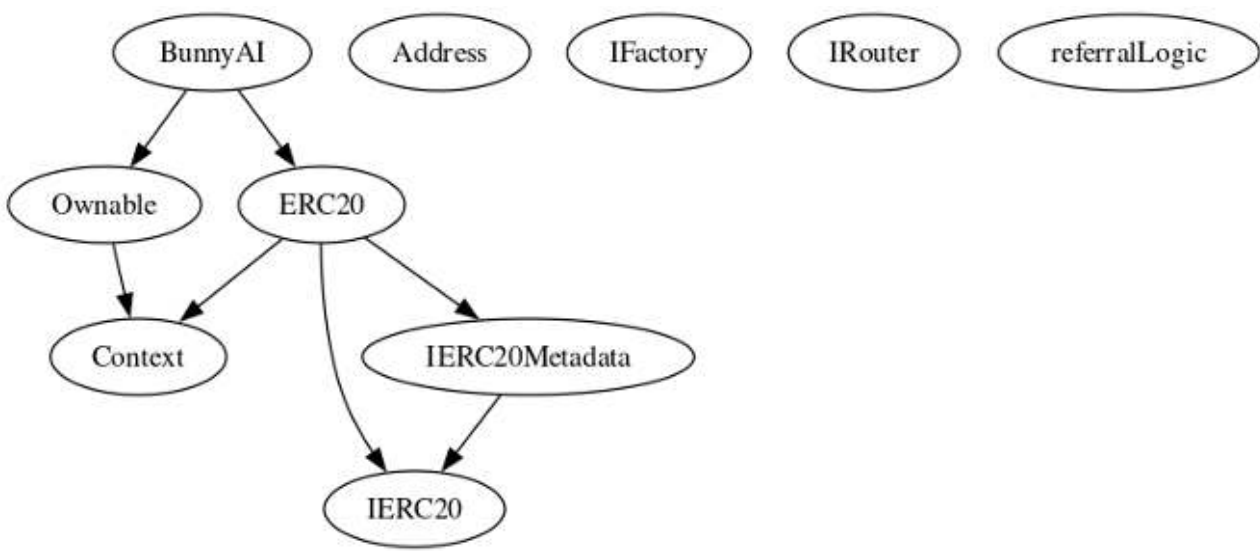
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.



Inheritance

The contract for Bunny AI has the following inheritance structure.



Privileged Functions (onlyOwner)

Function Name	Parameters	Visibility
renounceOwnership		public
transferOwnership	newOwner (address)	public
rescueERC20	address _token, uint256 _amount	external
rescueBNB	uint256 _amount	external
updateMaxTxLimit	uint256 maxBuy, uint256 maxSell, uint256 maxWallet	external
bulkExemptFee	address[] memory accounts, bool state	external
updateExemptFee	address _address, bool state	external
updateMarketingWallet	address newWallet	external
_toggleTrading	bool status	external
_openTrading		external



Function Name	Parameters	Visibility
toggleReferral	bool status	external
updateReferralContract	address _newReferralContract	external
updateReferralContract	address newRouter, address newPair	external
SetTransferTaxes	uint256 _marketing,uint256 _liquidity,uint256 _burn	external
SetSellTaxes	uint256 _marketing,uint256 _liquidity,uint256 _burn	external
SetBuyTaxes	uint256 _marketing,uint256 _liquidity,uint256 _burn	external
updateLiquidityThreshold	uint256 new_amount	external
updateLiquidityProvide	bool state	external



Smart Contract Advance Checks



ID	Severity	Name	Result	Status
\$BUNAI-01	Minor	Potential Sandwich Attacks.	Pass	Resolved
\$BUNAI-02	Minor	Function Visibility Optimization	Pass	Not-Found
\$BUNAI-03	Minor	Lack of Input Validation.	Fail	Pending
\$BUNAI-04	Major	Centralized Risk In addLiquidity.	Pass	Not-Found
\$BUNAI-05	Major	Missing Event Emission.	Fail	Pending
\$BUNAI-06	Minor	Conformance with Solidity Naming Conventions.	Fail	Pending
\$BUNAI-07	Minor	State Variables could be Declared Constant.	Pass	Not-Found
\$BUNAI-08	Major	Dead Code Elimination.	Pass	Not-Found
\$BUNAI-09	Major	Third Party Dependencies.	Pass	Not Found
\$BUNAI-10	Major	Initial Token Distribution.	Fail	Pending
\$BUNAI-11	Critical	distributeTokensBetween Holders is a multisender of tokens from contract.	Pass	Not-Found
\$BUNAI-12	Major	Centralization Risks In The X Role	Pass	Not Found
\$BUNAI-13	Informational	Extra Gas Cost For User..	Fail	Pending
\$BUNAI-14	Medium	Unnecessary Use Of SafeMath	Pass	Not-Found



ID	Severity	Name	Result	Status
\$BUNAI-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not-Found
\$BUNAI-16	Medium	Invalid collection of Taxes during Transfer.	Pass	Not-Found



\$BUNAI-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Minor	bunnyv2.sol: 659,14	 Pending

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the updateLiquidityProvide, updateRouterAndPair, toggleReferral, _toggleTrading, updateExemptFee, bulkExemptFee and rescueBNB is missing required function.

Remediation



We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
require(receiver != address(0), "Receiver is the zero address");
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. updateLiquidityProvide, updateRouterAndPair, toggleReferral, _toggleTrading, updateExemptFee, bulkExemptFee and rescueBNB is missing required function.



\$BUNAI-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Major	bunnyv2.sol: 659, 14	 Pending

Description



Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.



\$BUNAI-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	 Minor	bunnyv2.sol: 294,14	 Pending

Description

Solidity defines a naming convention that should be followed. Rule exceptions: Allow constant variable name/symbol/decimals to be lowercase. Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
updateLiquidityTreshhold  
SetBuyTaxes  
SetSellTaxes  
SetTransferTaxes
```



Remediation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-convention>



\$BUNAI-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	 Major	bunnyv2.sol: 53,6	 Pending

Description

All of the Bunny AI tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Remediation



We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Project Action

Token Distribution goes to the `_tokengeneration(msg.sender, 23_000_000 * 10**decimals());`



\$BUNAI-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	 Informational	bunnyv2.sol: 236, 8	 Pending

Description

The user may trigger a tax distribution during the transfer process, which will cost a lot of gas and it is unfair to let a single user bear it.

Remediation

We advise the client to make the owner responsible for the gas costs of the tax distribution.

Project Action



Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/bunnyai_eth	Pass
Other	https://instagram.com/bunnyai_eth	Pass
Website	https://bunnyai.app	Pass
Telegram	https://t.me/bunny_ai	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:



Assessment Results

Score Results

Review	Score
Overall Score	82/100
Auditor Score	80/100
Review by Section	Score
Manual Scan Score	29/35
SWC Scan Score	35 /37
Advance Check Score	18 /28

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed



Assessment Results

Important Notes:

- No issues or vulnerabilities were found.
- Project seems to be related to AI.
- Project Owner have experience building and developing projects.
- This is an Ethereum based project, always DYOR, from an Audit perspective it has some limitations and controls.

Auditor Score =80
Audit Passed



Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different requirements on the input variables than a setter function.

Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria and is readable by all the developers.



Disclaimer

CFGNINJA has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocacy for the Project, and users relying on this 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 the Project in question may pursue, and 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 entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

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. Information is provided 'as is, and CFGNINJA is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will CFGNINJA 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 or actions with regards to the information provided in this audit report.

The assessment services provided by CFGNINJA are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.

