



CFG NINJA AUDITS

Security Assessment

Morty Token

June 3, 2023

Audit Status: Fail

Audit Edition: Advance














POWERED BY
BLADE POOL

Risk Analysis

Classifications of Manual Risk Results

Classification	Description
 Critical	Danger or Potential Problems.
 Major	Be Careful or Fail test.
 Minor	Pass, Not-Detected or Safe Item.
 Informational	Function Detected

Manual Code Review Risk Results

Contract Priviledge	Description
 Buy Tax	7
 Sale Tax	7
 Cannot Sale	Pass
 Cannot Sale	Pass
 Max Tax	100
 Modify Tax	Detected
 Fee Check	Fail
 Is Honeypot?	Detected
 Trading Cooldown	Not Detected
 Can Pause Trade?	Pass
 Pause Transfer?	Owner has to enable trade/launch.



Contract Priviledge	Description
🟡 Max Tx?	Fail
🟡 Is Anti Whale?	Detected
🟡 Is Anti Bot?	Detected
🔴 Is Blacklist?	Detected
🔴 Blacklist Check	Fail
🟡 is Whitelist?	Detected
🟢 Can Mint?	Pass
🟢 Is Proxy?	Not Detected
🟢 Can Take Ownership?	Not Detected
🟢 Hidden Owner?	Not Detected
🔵 Owner	0x61d6d459585f456d47e4bd2f878ddea3f6c330d3
🟢 Self Destruct?	Not Detected
🔵 External Call?	Detected
🟢 Other?	Detected
🟢 Holders	1
🔴 Auditor Confidence	Low

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.



Project Overview

Token Summary

Parameter	Result
Address	0xBB8a9de1027382A1b56E02EA7cfC2d6f0AddE215
Name	Morty
Token Tracker	Morty (Morty)
Decimals	18
Supply	1,000,000,000,000,000
Platform	Binance Smart Chain
compiler	v0.8.18+commit.87f61d96
Contract Name	Morty
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/0xbb8a9de1027382a1b56e02ea7cfc2d6f0adde215#code
Payment Tx	Corporate



Main Contract Assessed Contract Name

Name	Contract	Live
Morty	0xBB8a9de1027382A1b56E02EA7cfC2d6f0AddE215	Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
Morty	0xc4778ca95dc3478637bfafdbcf0f2794e602591	Yes

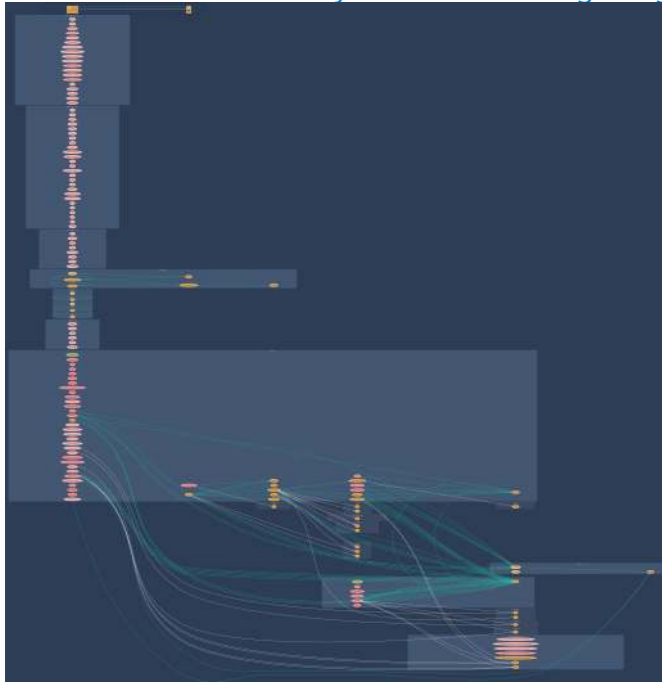
Solidity Code Provided

SolID	File Sha-1	FileName
Morty	ad39078758adbf1dcbebf2ca9a400a75c71de7f0	morty.sol
Morty		
Morty		
Morty		



Call Graph

The contract for Morty has the following call graph structure.



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	morty.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	morty.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	morty.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	morty.sol	L: 30 C: 0
SWC-104	Pass	Unchecked Call Return Value.	morty.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	morty.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	morty.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	morty.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	morty.sol	L: 421 C: 33,L: 456 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	morty.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	morty.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	morty.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	morty.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	morty.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	morty.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	morty.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	morty.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	morty.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	morty.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	morty.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randomness.	morty.sol	L: 567 C: 24,L: 738 C: 24
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	morty.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	morty.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	morty.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	morty.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	morty.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	morty.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	morty.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	morty.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	morty.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	morty.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	morty.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	morty.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	morty.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	morty.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	morty.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	morty.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-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables



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?)

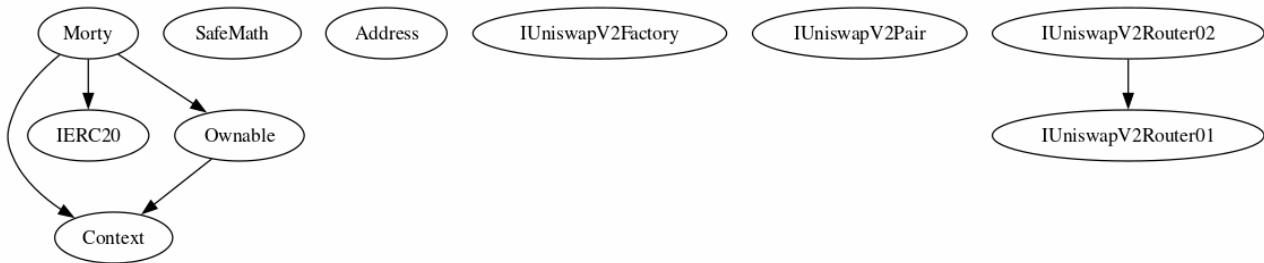
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.



Inheritance

The contract for Morty has the following inheritance structure.



Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
writebcList		Public
setKillBlock		External
changeRouterVersion		External
setSwapAndLiquifyByLimitOnly		External
setSwapAndLiquifyEnabled		External
setTeamWalletAddress		External
setMarketingWalletAddress		External
setNumTokensBeforeSwap		External
setWalletLimit		External



Function Name	Parameters	Visibility
setIsWalletLimitExempt		External
enableDisableWalletLimit		External
setMaxTxAmount		External
setDistributionSettings		External
setSellTaxes		External
setBuyTaxes		External
setIsExcludedFromFee		External
setIsTxLimitExempt		External
setMarketPairStatus		External
Launch		External



Smart Contract Advance Checks



ID	Severity	Name	Result	Status
Morty-01	Minor	Potential Sandwich Attacks.	Pass	Not-Found
Morty-02	Minor	Function Visibility Optimization	Fail	Detected
Morty-03	Minor	Lack of Input Validation.	Fail	Pending
Morty-04	Major	Centralized Risk In addLiquidity.	Fail	Detected
Morty-05	Minor	Missing Event Emission.	Fail	Pending
Morty-06	Minor	Conformance with Solidity Naming Conventions.	Fail	Detected
Morty-07	Minor	State Variables could be Declared Constant.	Pass	Not-Found
Morty-08	Minor	Dead Code Elimination.	Pass	Not-Found
Morty-09	Major	Third Party Dependencies.	Fail	Detected
Morty-10	Major	Initial Token Distribution.	Pass	Not-Found
Morty-11	Minor	AntiBot is present on the transfer.	Fail	Detected
Morty-12	Major	Centralization Risks In The X Role	Pass	Not-Found
Morty-13	Informational	Extra Gas Cost For User..	Fail	Detected
Morty-14	Medium	Unnecessary Use Of SafeMath	Fail	Detected
Morty-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not-Found



ID	Severity	Name	Result	Status
Morty-16	Medium	Invalid collection of Taxes during Transfer.	Pass	Not-Found
Morty-17	Informational	Conformance to numeric notation best practice.	Pass	Not-Found
Morty-18	Medium	Stop Transactions by using Enable Trade.	Fail	Detected



Morty-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	 Minor	morty.sol: L: 421 C: 33	 Detected

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
inSwapAndLiquify		internal
_balances		internal
writebcList		public
setKillBlock		public
Launch		public
setIsExcludedFromFee		public

The functions that are never called internally within the contract should have external visibility

Remediation



We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

References:

external vs public best practices.



Morty-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Minor	morty.sol: 474,14	 Pending

Description

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

The given input is missing the check for the all onlyOwner are 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. all onlyOwner are missing required function.



Morty-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	● Major	morty.sol: 817,13	 Detected

Description

`uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this), tokenAmount, 0, 0, owner(), block.timestamp);`

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the Morty-WBNB pool.

As a result, over time the `_owner` address will accumulate a significant portion of LP tokens. If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Remediation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

1. Indicatively, here are some feasible solutions that would also mitigate the potential risk:
2. Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
3. Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;



Introduction of a DAO / governance / voting module to increase transparency and user involvement

Project Action

liquidity is set to `teamWallet`.



Morty-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Minor	morty.sol: 474, 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.



Morty-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	 Minor	morty.sol: 693,13	 Detected

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.

```
writebcList  
isbcList
```



Remediation

Follow the Solidity naming convention.

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



Morty-09 | Third Party Dependencies.

Category	Severity	Location	Status
Volatile Code	 Major	morty.sol: 109,9	 Detected

Description

The contract is serving as the underlying entity to interact with third party `0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470` protocols. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Remediation



We understand that the business logic of Morty requires interaction with `0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Project Action

Pending Customer Response



Morty-11 | AntiBot is present on the transfer..

Category	Severity	Location	Status
Optimization	 Minor	morty.sol: 739,14	 Detected

Description

During the transfer it sends the transaction to an external contract 'IGemAntiBot(gemAntiBot).onPreTransferCheck(from, to, amount)'



Remediation

Ensure the IGemAntiBot library is audited and the process is clean during the transfer.

Project Action



Morty-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	 Informational	morty.sol: 726, 13	 Detected

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

is declared public



Morty-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	 Medium	morty.sol: 56,9	 Detected

Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations

will automatically revert in case of integer overflow or underflow.

library SafeMath {

An implementation of SafeMath library is found.

using SafeMath for uint256;

SafeMath library is used for uint256 type in contract.

Remediation



We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the

Solidity programming language

Project Action



Morty-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	 Medium	morty.sol: 565, 13	 Detected

Description

Enable Trade is present on the following contract and when combined with Exclude from fees it can be considered a whitelist process, this will allow anyone to trade before others and can represent an issue for the holders.

Remediation






We recommend the project owner to carefully review this function and avoid problems when performing both actions.

Project Action








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	1	0	0
 Major	2	0	0
 Medium	0	0	0
 Minor	5	0	0
 Informational	2	0	0
Total	10	0	0



Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/MortySmith_BSC	Pass
Other		Fail
Website		Fail
Telegram	https://t.me/Morty_Smith_Bsc	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	42/100
Auditor Score	80/100
Review by Section	Score
Manual Scan Score	23/53
SWC Scan Score	34 /37
Advance Check Score	-15 /19

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 Fail



Assessment Results

Important Notes:

- The contract has a launch mode, this act enables trade.
- the contract has a lot of logic that can go wrong and the contract may fail.

Auditor Score =80
Audit Fail



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.

