ONTOLOGY FORMAL VERIFICATION TOOL DETECTION FUNCTION LIST

1. Static Detection Rule Table

As shown in Table 1-1 below, it is the rule library of static detection which included by the formal verification tool. There are three main types of vulnerabilities, the first is the type of vulnerability specified by the language itself, the second is the type of vulnerability specified by the platform for the language, and the third is the type of vulnerability specified in the platform rule library. More details please refer to the <FORMAL VERIFICATION TOOL FUNCTION TEST CASE INSTRUCTIONS>.

Table 1-1 Ontology Formal Verification Tool Static Rule

No.	Detection item title	type	description	example
1	Nonexistent-Operator	Error	Use of the non-existent {} operator	Use operator unsupported in Python like “++” or “--”.
2	Unreachable	Warning	Unreachable code	The unreachable code behind “break” or “return” in code block.
3	Duplicate-key	Warning	Duplicate key {} in dictionary	More details refer to test case: Duplicate-key
4	Using-Constant-Test	Warning	Using a conditional statement with a constant value	More details refer to test case: Using-Constant-Test
5	Unnecessary-Pass	Warning	Unnecessary pass statement	More details refer to test case: Unnecessary-Pass
6	Invalid-Slice-Index	Error	Slice index is not an int, None	Using string, id, None and other method as slice index.
7	Reimported	Warning	Reimport {} (imported line {})	More details refer to test case: Reimported
8	Multiple-Imports	Warning	Multiple imports on one line ({})	More details refer to test case: Multiple-Imports
9	Lack-Of-Event	Error	Should emit NEP5 event relate to method {}	Lack of triggering event transfer，transferfrom，approve
10	Lack-Of-Witness	Error	The "from" address should be verified by CheckWitness	transfer，transferfrom，approve whether uses checkwitness to verify contract address and

				“from” address.
11	Unsupported-Operator	Error	Unsupported operator	eg: using “+”, “*”,“+=” and other operators on string and list.
12	Unsupport-String-Formatted	Error	Unsupported string Format: {}	More details refer to test case: Unsupport-String-Formatted
13	Find-Negative-Index-In-String-Or-Bytearray	Error	Avoid negative index in string or bytearray	eg: the slice index of string or bytearray passes negative numbers by calculating formulas.
14	Comparison-with-itself	Error	Comparison with itself	eg: if a is a
15	Assignment-from-no-return	Error	Assigning to function call which doesn't return	eg: the simple contract transfer function, can not judge transfer result because that function TransferOntOng has no return value.
16	Invalid-Sequence-Index	Error	Sequence index is not an int, slice	eg: using id or none for array index.
17	Assignment-from-None	Error	Assigning to function call which only returns None	eg: returning none exist on TransferOntOng function.
18	Unhashable-Dict-Key	Error	Dict key is unhashable	More details refer to test case: Unhashable-Dict-Key
19	No-Name-In-Module	Error	No name {} in module {}	eg: imported a non-existent module or imported functions that do not exist in the module.
20	Unused-Variable	Warning	Unused variable {}	More details refer to test case: Unused-Variable
21	Unused-Argument	Warning	Unused argument {}	More details refer to test case: Unused-Argument
22	Redefined-Outer-Name	Error	Redefining name {} from outer scope (line {})	More details refer to test case: Redefined-Outer-Name
23	Not-support-slice-	Error	Not support slice access	eg: Intercept 5 players,

	access			slice the array
24	Random-number-attack	warning	Random number can be predicted	More details refer to test case: Random-number-attack
25	Check-invoke-result	Warning	Invoke function return without check	More details refer to test case: Check-invoke-result
26	No-except-handler	Warning	ONT do not support except handler	More details refer to test case: no except handler
27	No-return	Error	Return function which dosen't return	More details refer to test case: no-return
28	Only-slice-access	Error	Only slice access	More details refer to test case: only-slice-access
29	Possibly-unused-variable	Warning	Unused variable	More details refer to test case: Possibly-unused-variable
30	Redefined-builtin	Error	Redefining built-in {}	More details refer to test case: Redefined-builtin
31	Wrong-import-position	Error	Import "{}" should be placed at the top of the contract	More details refer to test case: Wrong-import-position
32	Unused-wildcard-import	Warning	Unused import {} from wildcard Import	More details refer to test case: Unused-wildcard-import
33	Unused-Import	Warning	Unused import{}	More details refer to test case: Unused-Import
34	Slice-elem-out-of-bounds	Error	Slice elem out of bounds	eg: string end indexes as function, the function returns as out of bounds.

2. Dynamic Detection Rule library

As shown in Table 1-2 below, it is the rule library of static detection which included by the formal verification tool. It is mainly the vulnerabilities will occur in dynamic execution process can cover the content which static detection cannot cover. It complements the static detection of shortcomings, to optimize detection. More

details please refer to the <ONTOLOGY FORMAL VERIFICATION TOOL FUNCTION TEST CASE INSTRUCTIONS>.

Table 1-2 Ontology Formal Verification Tool Dynamic Detection Rule

No.	title	type	description	example
1	Div-zero-occurred	Error	Div zero	More details refer to test case: Div-zero-occurred
2	Index-out-of-range	Error	Index out of range	More details refer to test case: index out of range
3	Assert-fail	Error	Assert conditions are not always satisfied	eg:assert (a>10) requires 'a' must meets 'a>10' but this assert condition is not always satisfied when running.
4	Require-fail	Error	Require Conditions are not satisfied	eg:require (a>10) requires 'a' must meets 'a>10' but this condition is not always satisfied when running.

5	Integer-overflow-occurred	Error	Integer may overflow	eg:int_8 a = 126;
				int_8 b = 10;
				a = a+b; If run it without using SafeMath, the integer overflow will occur here.

6	Integer-underflow-occurred	Error	Integer may underflow	eg:int_8 a = -127; Int_8 b = 10; a = a - b; If run it without using SafeMath, the integer underflow will occur here.
7	Data-injection-attack	Error	Data injection attack	More details refer to test case: Data injection attack