CS9163 - Application Security
Grisha Kumar
0494402
grk236@students.poly.edu

Implementing a turing complete sandbox in Python

Task:
The main task of this assignment is to create a truing complete sandbox which is does not have any vulnerabilities.

Programming language:
Python 2.7 is used to implement the turing complete sandbox

Test cases:
The following modules can be run inside the sandbox:

- Count form 10 to 1 and print the output
- Compute the first 10 Fibonacci numbers and print the output

Features of the sandbox:

- Restricts the execution of arbitrary code
- Denys the access to the file system
- Restricts the import of other python modules
- Error handling
- Allocation of memory for the program to run
- Restricts the execution of eval and exec functions
- Has a whitelist of executable functions
- Extension checking

The following modules are included in the sandbox github:

- main.py
- sandbox.py
- ReadMe file
- Documentation
- MAKEFILE
- T1_count.py
- T2_fib.py
- T3_tc.py

Python has an inbuilt compiler package that creates an abstract syntax tree(AST) that will check the right list of nodes. The AST is traversed to check for the nodes which is safe, if it finds a node which is not part of the which list that we have provided then it will exit with an error.

Running the Count problem

Considering the case of the count problem where the program is running inside the sandbox.

The AST representation is given below:

```
Module(None, Stmt([For(AssName('cnt', 'OP_ASSIGN'), CallFunc(Name('range'), [Const(10),
Const(0), UnarySub(Const(1))], None, None), Stmt([Printnl([Name('cnt')], None)]), None)]))
```

Sample output:

```
Reading script...
Running Static tests...
Static tests completed successfully...
Running Dynamic tests...
Executing Program in controlled environment...
10
9
8
7
6
5
4
3
2
1
Dynamic tests completed successfully
Sandbox exiting...
```

Running the Fibonacci problem

Considering the case of the fibonacci problem where the program is running inside the sandbox.

The AST representation is given below:

```
Module(None, Stmt([Function(None, 'fib', ['x'], [], 0, None, Stmt([If([(Compare(Name('x'), [('==',
Const(0))]), Stmt([Return(Const(0))])), (Compare(Name('x'), [('==', Const(1))]),
Stmt([Return(Const(1))]))], Stmt([Return(Add((CallFunc(Name('fib'), [Sub((Name('x'), Const(1)))],
None, None), CallFunc(Name('fib'), [Sub((Name('x'), Const(2)))], None, None))))]))])),
For(AssName('foo', 'OP_ASSIGN'), CallFunc(Name('range'), [Const(10)], None, None),
Stmt([Printnl([CallFunc(Name('fib'), [Name('foo')], None, None)], None)]), None)]))
```

Sample output:

```
Reading script...
Running Static tests...
Static tests completed successfully...
Running Dynamic tests...
Executing Program in controlled environment...
0
1
1
2
3
5
8
```

```
13
21
34
Dynamic tests completed successfully
Sandbox exiting...
```

Running the turing complete problem

The following turing machine is taken to test out the sandbox – The code can be found in the link below
http://www.python-course.eu/turing_machine.php

Since the sandbox runs the turing complete problem we can conclude that the snadbox is truing complete.

Sample output:

```
Reading script...
Running Static tests...
Static tests completed successfully...
Running Dynamic tests...
Executing Program in controlled environment...
Input on Tape:
010011
Result of Turing machine calculation:
101100
Dynamic tests completed successfully
Sandbox exiting...
```