|  |  |  |  |
| --- | --- | --- | --- |
| **用例编号** | **测试用例** | **测试内容** | **测试方法** |
| 7.1.4 智能合约安全形式化验证（go语言智能合约） | | | |
| 4-1 | bare-return1.go | 检测bare-return1.go智能合约，验证不通过  （bare-return要求智能合约函数的实现需要严格按照其函数定义，f280\_4函数没有返回规定类型的数据，验证不通过） | 1. 打开网页  <https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81>  2. 选择bare-return1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-2 | bare-return2.go | 检测bare-return2.go智能合约，验证通过  （bare-return要求智能合约函数的实现需要严格按照其函数定义，f280\_4函数按要求返回了规定类型的数据，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择bare-return2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-3 | context-as-argument1.go | 检测context-as-argument1.go智能合约，验证不通过  （context-as-argument要求context类型必须作为函数的第一个参数，y函数的第一个参数是string类型，不符合要求，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择context-as-argument1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-4 | context-as-argument2.go | 检测context-as-argument2.go智能合约，验证通过  （context-as-argument要求context类型必须作为函数的第一个参数，y函数的第一个参数是context类型，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择context-as-argument2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-5 | defer1.go | 检测defer1.go智能合约，验证不通过  （recover()功能必须在被延迟的函数中调用，在deferrer函数中，defer关键词直接用于recover()功能，并不能起到延迟的效果，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择defer1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-6 | defer2.go | 检测defer2.go智能合约，验证通过  （recover()功能必须在被延迟的函数中调用，在deferrer函数中，defer关键词用于一个被函数包裹的recover()语句，符合recover()的调用要求，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择defer2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-7 | modifies-param1.go | 检测modifies-param1.go智能合约，验证不通过  （modifies-parameter要求函数的值参数不应被修改，two函数中对参数b进行了修改，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modifies-param1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-8 | modifies-param2.go | 检测modifies-param2.go智能合约，验证通过  （modifies-parameter要求函数的值参数不应被修改，two函数中没有对参数的修改操作，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modifies-param2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-9 | modifies-value-receiver1.go | 检测modifies-value-receiver1.go智能合约，验证不通过  （modifies-value-receiver要求在数据接收器里边不应该对接收的数值进行更改，vmethod接收器对接收到的this.num进行了修改，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modifies-value-receiver1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-10 | modifies-value-receiver2.go | 检测modifies-value-receiver2.go智能合约，验证通过  （modifies-value-receiver要求在数据接收器里边不应该对接收的数值进行更改，vmethod接收器没有对对接收到的值数据进行了修改，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modifies-value-receiver2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-11 | range-val-in-closure1.go | 检测range-val-in-closure1.go智能合约，验证不通过  （range-val-in-closure要求在循环中不应对循环变量进行多线程操作，foo函数中对循环变量index, value进行多线程的输出，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择range-val-in-closure1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-12 | range-val-in-closure2.go | 检测range-val-in-closure2.go智能合约，验证通过  （range-val-in-closure要求在循环中不应对循环变量进行多线程操作，foo函数中未对循环变量index, value进行多线程操作，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择range-val-in-closure2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-13 | string-of-int1.go | 检测string-of-int1.go智能合约，验证不通过  （string-of-int要求不应直接将int类型的数据强制类型转换成string类型，StringTest函数将int类型的变量i强制转换成string类型，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择string-of-int1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-14 | string-of-int2.go | 检测string-of-int2.go智能合约，验证通过  （string-of-int要求不应直接将int类型的数据强制类型转换成string类型，StringTest函数将int类型的变量i通过类型转换函数strconv.Itoa进行安全操作，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择string-of-int2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-15 | time-equal1.go | 检测time-equal1.go智能合约，验证不通过  （time-equal要求不能用==和!=比较时间变量，t函数中直接用==和!=比较时间变量u和t，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择time-equal1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-16 | time-equal2.go | 检测time-equal2.go智能合约，验证通过  （time-equal要求不能用==和!=比较时间变量，t函数中使用时间比较函数t.Equal比较时间变量，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择time-equal2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-17 | unconditional-recursion1.go | 检测unconditional-recursion1.go智能合约，验证不通过  （unconditional-recursion要求函数中不能出现没有退出条件的递归，ur2tris函数中使用了无条件的递归，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unconditional-recursion1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-18 | unconditional-recursion2.go | 检测unconditional-recursion2.go智能合约，验证通过  （unconditional-recursion要求函数中不能出现没有退出条件的递归，ur2tris函数中在进入递归前增加了限制条件，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unconditional-recursion2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-19 | unexported-naming1.go | 检测unexported-naming1.go智能合约，验证不通过  （unexported-naming要求函数的导出变量应当对其他智能合约是可见的，unexportednaming函数中导出变量S，Result和NotExportableConstant的可见性仅局限于函数内部，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unexported-naming1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-20 | unexported-naming2.go | 检测unexported-naming2.go智能合约，验证通过  （unexported-naming要求函数的导出变量应当对其他智能合约是可见的，导出变量S，Result和NotExportableConstant的可见性为全局可见，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unexported-naming2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-21 | unhandled-error1.go | 检测unhandled-error1.go智能合约，验证不通过  （unhandled-error要求任何函数或API错误类型的返回值都应该被检查，unhandledError2函数中未检查unhandledError1、fmt.Fprintf和os.Chdir的错误返回值，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unhandled-error1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-22 | unhandled-error2.go | 检测unhandled-error2.go智能合约，验证通过  （unhandled-error要求任何函数或API错误类型的返回值都应该被检查，unhandledError2函数中检查了unhandledError1、fmt.Fprintf和os.Chdir的错误返回值，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unhandled-error2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-23 | unreachable-code1.go | 检测unreachable-code1.go智能合约，验证不通过  （unreachable-code要求智能合约中不应存在不可到达的代码，return、os.Exit、t.Fatal语句后边的代码都不会被执行，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unreachable-code1.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |
| 4-24 | unreachable-code2.go | 检测unreachable-code2.go智能合约，验证通过  （unreachable-code要求智能合约中不应存在不可到达的代码，return、os.Exit、t.Fatal语句后边无其他代码，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/golang%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择unreachable-code2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，形式化验证  4. 点击Analyze Now |