|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **用例编号** | | **测试用例** | **测试内容** | | **测试方法** |
| 7.1.2 区块链合约层漏洞检测（go语言智能合约） | | | | | |
| 2-1 | global-variable.go | | | 检测global-variable.go智能合约，结果中应当包含chaincode-global-variable漏洞  （如果Go语言智能合约中用到了全局变量，会导致不同节点全局变量不同步，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择global-variable.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-global-variable漏洞 |
| 2-2 | blacklist-imports.go | | | 检测blacklist-imports.go智能合约，结果中应当包含black-listed-import(math/rand)漏洞  （如果Go语言智能合约中用到了随机数库，会导致不同节点生成的随机数不一致，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择blacklist-imports.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含black-listed-import(math/rand)漏洞 |
| 2-3 | blacklist-imports.go | | | 检测blacklist-imports.go智能合约，结果中应当包含black-listed-import(time)漏洞  （如果Go语言智能合约中用到了时间戳库，会导致不同节点获取的时间戳不一致，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择blacklist-imports.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含black-listed-import(time)漏洞 |
| 2-4 | blacklist-imports.go | | | 检测blacklist-imports.go智能合约，结果中应当包含black-listed-import(net)漏洞  （如果Go语言智能合约中用到了net库获取网络数据，会导致不同节点获取的网络数据不一致，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择blacklist-imports.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含black-listed-import(net)漏洞 |
| 2-5 | blacklist-imports.go | | | 检测blacklist-imports.go智能合约，结果中应当包含black-listed-import(os)漏洞  （如果Go语言智能合约中用到了os库获取本地操作系统数据，由于不同节点物理环境不一致，获取的操作系统数据不一致，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择blacklist-imports.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含black-listed-import(os)漏洞 |
| 2-6 | blacklist-imports.go | | | 检测blacklist-imports.go智能合约，结果中应当包含black-listed-import(io)漏洞  （如果Go语言智能合约中用到了io库获取本地文件数据，由于不同节点文件系统不一致，获取的文件数据数据不一致，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择blacklist-imports.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含black-listed-import(io)漏洞 |
| 2-7 | range-over-map.go | | | 检测range-over-map.go智能合约，结果中应当包含chaincode-range-over-map漏洞  （如果Go语言智能合约中，使用range来遍历map数据结构的内容进行计算，由于range每次随机从map中抽取数据，不同节点获取数据的顺序不一致，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择range-over-map.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-range-over-map漏洞 |
| 2.8 | range-over-map.go | | | 检测range-over-map.go智能合约，结果中应当包含chaincode-math-overflow漏洞  （如果Go语言智能合约中，对整数类型数据进行算术操作，计算结果超出类型的范围，导致结果出错） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择range-over-map.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-math-overflow漏洞 |
| 2-9 | goroutine.go | | | 检测range-over-map.go智能合约，结果中应当包含chaincode-go-routine漏洞  （如果Go语言智能合约中，使用多线程操作，由于线程之间数据不同步，导致计算出错） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择goroutine.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-go-routine漏洞 |
| 2-10 | address.go | | | 检测address.go智能合约，结果中应当包含chaincode-pointer漏洞  （如果Go语言智能合约中用到了指针获取变量的地址，由于不同节点运行环境不一致，获取的地址数据不一致，交易失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择address.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-pointer漏洞 |
| 2-11 | channel.go | | | 检测address.go智能合约，结果中应当包含chaincode-invoke-chaincode漏洞  （如果Go语言智能合约中用到了invokechaincode API并向被调用智能合约中写入数据，由于权限限制写入数据会失败） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择channel.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-invoke-chaincode漏洞 |
| 2-12 | phrantom.go | | | 检测address.go智能合约，结果中应当包含chaincode-phandom-read漏洞  （如果Go语言智能合约中用到了GetQueryResult，GetHistoryForKey或GetPrivateDataQueryResult等API查询并修改数据，则会出现幻影读，导致结果出错） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择phrantom.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-phandom-read漏洞 |
| 2-13 | read-write.go | | | 检测address.go智能合约，结果中应当包含chaincode-read-after-write漏洞  （如果Go语言智能合约中对同一个数据进行了先写后读的操作，则读到的是过时的数据，结果出错） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择read-write.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-read-after-write漏洞 |
| 2-14 | overflow2.go | | | 检测overflow2.g智能合约，结果中应当包含chaincode-conversion-overflow漏洞  （如果Go语言智能合约中对整数类型数据进行从大范围到小范围的类型转换，导致超出类型的范围，结果出错） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/go%E5%90%88%E7%BA%A6%E6%BC%8F%E6%B4%9E  2. 选择overflow2.go将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Go，漏洞检测  4. 点击Analyze Now  5. 检测结果中包含chaincode-conversion-overflow漏洞 |