|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 武汉大学国家网络安全学院教学实验报告 | | | | | |
| 课程名称 | WEB安全 | | 实验日期 | | 2024/5/20 |
| 实验名称 | Web漏洞分析与代码审计 | | 实验周次 | | 1 |
| 姓名 | 学号 | | 专业 | | 班级 |
| 赵伯俣 | 2021302181156 | | 信安 | | 6 |
| 刘竞优 | 2021302181057 | | 信安 | | 6 |
| 张竣尧 | 2021302181004 | | 信安 | | 7 |
| 一、实验目录 [一、实验目录 1](#_Toc167092801)  [二、实验目的 2](#_Toc167092802)  [三、实验要求 2](#_Toc167092803)  [四、实验内容 3](#_Toc167092804)  [五、实验步骤 4](#_Toc167092805)  [5.1 xpath之bwaspp实列一 4](#_Toc167092806)  [5.2 XPath之bwaspp实列二 8](#_Toc167092807)  [5.3 反序列化 11](#_Toc167092808)  [5.4 SSRF 15](#_Toc167092809)  [5.4.1 172.72.23.21 - SSRF 获取本地信息 15](#_Toc167092810)  [5.4.2 172.72.23.22 - 代码注入 17](#_Toc167092811)  [5.4.3 172.72.23.23 - SQL 注入 20](#_Toc167092812)  [5.4.4 172.72.23.24 - 命令执行 24](#_Toc167092813)  [5.4.5 172.72.23.25 - XML 实体注入 28](#_Toc167092814)  [5.4.6 172.72.23.26 - CVE-2017-12615 31](#_Toc167092815)  [5.4.7 172.72.23.27 - Redis 未授权 33](#_Toc167092816)  [5.5信息收集实验 35](#_Toc167092817)  [5.5.1 Whois 查询 35](#_Toc167092818)  [5.5.2备案信息查询 37](#_Toc167092819)  [5.5.3 IP 资产收集 37](#_Toc167092820)  [5.5.4 CDN 检测 38](#_Toc167092821)  [5.5.5 企业信息收集 39](#_Toc167092822)  [5.5.6 域名信息收集 40](#_Toc167092823)  [5.5.7 Web 服务探测 42](#_Toc167092824)  [5.5.8 指纹识别 44](#_Toc167092825)  [5.5.9 邮箱收集 49](#_Toc167092826)  [5.5.10 敏感信息收集 50](#_Toc167092827)  [5.6.Linux提权实验 52](#_Toc167092828)  [六、实验过程分析 53](#_Toc167092829)  [七、实验结果总结 53](#_Toc167092830)  [八、各人实验贡献与体会 54](#_Toc167092831) | | | | | |
| 二、实验目的  * 掌握XPath在Web应用中的使用，包括使用XPath成功登陆和注入获取XML文件内容。 * 熟悉反序列化漏洞，了解在PHP中如何利用反序列化写入session，并完成字符串逃逸，以及如何在session中更改cookie输出flag。 * 理解SSRF（Server-Side Request Forgery）攻击原理，学会如何利用SSRF获取内网地址，并对内网服务器进行渗透以获取内网主机上的flag。 * 熟悉常见的Web漏洞类型，如代码注入、SQL注入、命令执行、XML实体注入、CVE-2017-12615等，并了解相应的利用方法。 * 熟练使用各种信息收集工具和技术，包括但不限于网络侦察、端口扫描、开放服务识别、漏洞扫描等，以对目标单位“武汉大学”进行全面的信息收集。 * 能够将收集到的信息进行整理和汇总，并输出为xlsx表格格式。 * 理解Linux系统权限管理机制，了解提权的原理和方法。 * 掌握针对Ubuntu系统的提权方法，能够有效地获取root权限。 | | | | | |
| 三、实验要求  * 通过xpath成功登陆XML/XPath Injection * 通过xpath成功注入获取xml文件中所有的内容 * 通过index.php将反序列化写入session中，并完成字符串逃逸，在session.php中更改cookie成功输出flag * 通过SSRF获得内网地址，并对内网服务器进行渗透并获取内网主机上存在的flag   + 172.72.23.21 - SSRF 获取本地信息   + 172.72.23.22 - 代码注入   + 172.72.23.23 - SQL 注入   + 172.72.23.24 - 命令执行   + 172.72.23.25 - XML 实体注入   + 172.72.23.26 - CVE-2017-12615   + 172.72.23.27 - Redis 未授权 * 对目标单位 "武汉大学" 进行全面的信息收集，汇总输出xlsx表格 * 对ubuntu系统进行提权，获取root权限 | | | | | |
| 四、实验内容 （1）Web漏洞利用：  使用XPath成功登陆和注入获取XML文件内容。  通过xpath成功登陆XML/XPath Injection  通过xpath成功注入获取xml文件中所有的内容  实施反序列化漏洞攻击，将数据写入session中，并在session中修改cookie输出flag。  通过index.php将反序列化写入session中，并完成字符串逃逸，在session.php中更改cookie成功输出flag  （2）进行SSRF攻击，获取内网地址，并渗透内网服务器以获取flag。  172.72.23.21 - SSRF 获取本地信息  172.72.23.22 - 代码注入  172.72.23.23 - SQL 注入  172.72.23.24 - 命令执行  172.72.23.25 - XML 实体注入  172.72.23.26 - CVE-2017-12615  172.72.23.27 - Redis 未授权  （3）信息收集：  对目标单位“武汉大学”进行全面的信息收集，包括网络侦察、端口扫描、服务识别等，将结果汇总输出。  （4）系统提权：  对Ubuntu系统进行提权，以获取root权限。 | | | | | |
| 五、实验步骤5.1 xpath之bwaspp实列一 首先根据给出的地址进入登陆界面XML/XPath Injection (Login Form)    用户名为bee，密码为bug登陆该页面，选择XML/XPath Injection(Login Form)    进入到目标网页中    打开burp suite开启抓包功能    在网站中随便输入用户名和密码之后登陆    查看抓取的流量包如下    在用户名中添加一个单引号会产生xmli报错，说明这里存在xml注入    查看报错内容中的源码位置/app/xmli\_1.php文件如下    通过分析源码可以知道，该网站通过读取heroes.xml文件的内容, 使用xpath匹配用户名与密码来进行登陆验证  接下来我们查看源码中保存用户名和密码的文件passwords/heroes.xml,发现该文件中包含有所有用户的用户名和密码信息    通过源码我们可以知道登陆页面使用xml的查询语句如下  $result = $xml->xpath("/heroes/hero[login='" . $login . "' and password='" . $password . "']");  通过观察搜索语句构造永真密码  123' or 1=1 or ''='  利用该永真密码将语句构造成  /heroes/hero[login='123' or 1=1 or ''='' and password='" . $password . "']"  登陆发现成功登陆   5.2 XPath之bwaspp实列二 首先登陆目标网站XML/XPath Injection (Search)    修改genre变量的内容，在网址栏添加单引号，发现产生报错，由此可得该页面存在XML注入漏洞    查看/xmli\_2.php文件的源码如下可以看出xpath的搜索语句如下  //hero[contains(genre, '$genre')]/movie    在这里查看源码中调用到的xml文件    由此可以发现正确的xpath语句的作用是  遍历XML文档中所有的hero元素；  筛选出genre元素中包含特定类型（由占位符$genre指定）的hero元素；  选取这些符合条件的hero元素下的movie子元素。  因为我们的目的是查看所有<genre>为Action的用户的所有信息，  构造payload如下，我们首先使用|符号将前后分开，在后面添加xx[contains(a,'与查找move的语句进行闭合；在前面的话我们构造Action')]/\*来查找所有<genre>为Action的用户，/\*来回显他们的所有信息。最终构造的xpath如下  action')]/\* | xx[contains(a,'  从而将xpath语句构造为  //hero[contains(genre, 'Action')]/\* | xx[contains(a,'')]/movie  构造的xpath语句的作用是  遍历XML文档中所有的hero元素；  筛选出genre元素中包含"Action"字符串的hero元素；  选取这些符合条件的hero元素下的所有子元素；  同时，查找XML文档中任何元素（xx），其中包含一个名为'a'的属性，但由于在路径中没有这样的元素，因此此部分将不会产生结果；  最后，选取所有符合条件的元素的电影（movie）子元素。  将构造好的payload放置到对应的位置来验证，发现能够成功获取到每一个用户的所有信息   5.3 反序列化 任务地址：<http://58.240.236.231:21003/index.php>、<http://58.240.236.231:21003/session.php>  **题目解析：**  首先我们来到第一个界面    分析代码，上面是一个message类，下面的qwer是我们需要传递的四个参数，当四个参数都设置之后，进入到下面的if语句块中  进入之后由one变量根据我们的输入参数生成一个实例。下一步str\_replace将这个实例序列化之后，将这个序列化字符串中“bear”替换为“beiji”，传递给uone变量  最后一个语句将我们上面替换之后的序列化字符串赋值给“SESSION[‘one‘]”，此时的“SESSION[‘one‘]”属于“user”用户的  接下来到第二个界面    在这个页面中需要我们发送的数据包的cookie-user字段=编码后的“beijixiong”=YmVpaml4aW9uZw==，才能够进入到里面的if循环语句中  进来之后将变量one赋值为前面第一个页面的“SESSION[‘one‘]”（注意是“beijixiong”的sessionid），并先后经过base64解码和反序列化。此时变量one将会被视为message结构体，它的token变量（应该是算偏移位置）=“beijixiong”的话就会成功输出flag  那我们的大致思路就是在第一个页面中通过qwer四个参数，传递我们的字符串，这个字符串需要满足以下条件  + 一个message结构体，令其token=beijixiong（因为只传递了4个参数，对此参数的修改还要依据下面一条）  + 将上述结构体先进行序列化，之后将里面的“bear”替换为  “beiji”之后满足上述条件  满足上述条件之后，提示“beijixiong cheng gong”此时用户“beijixiong”的SESSION[‘one’]被赋值    在第二个页面中发送数据包，因为该页面中的setcookie()使得我们发送的数据包的cookie-user=damahou，而非if要求的“beijixiong”因此需要抓包后修改该字段为相应值，如此一来我们在第二个界面数据包发送之后，就能够正确进入到if语句块中    下来根据第一个页面赋给的“SESSION[‘one‘]”，满足条件，因此可以正确输出flag    简单来说就是：**通过第一个页面给“beijixiong”的SESSION[‘one’]赋值，接下来通过第二个界面修改cookie-user为“beijixiong”，此时cookie-user和sessionid能够对应上，输出flag**  思路如上，重点在于上面的字符串替换，替换之后发生了长度变化，从4个字符到5个字符，会使得我们后面第二个页面在进行反序列化时进行截断，因此我们需要合理构造传入的序列化结构体，这里参考如下网址  [PHP反序列化字符逃逸详解\_php filter字符串溢出-CSDN博客](https://blog.csdn.net/qq_45521281/article/details/107135706)  每个“bear”—>“beiji”会增加一个字符，也就是我们可以由此多一个字符逃逸，在第一个页面中我们需要做的就是将token赋值为“beijixiong”，对应的序列化字符串就是“";s:5:"token";s:10:"beijixiong";}”一共33个字符串，因此我们需要令参数r=33\*“bear”，这样一来就会替换token=user——>token=beijixiong，来让第二个页面进一步利用  字符串逃逸类似于sql注入的语句闭合，借助引号的闭合作用，由于传入时i:132指定了解析长度为132，而对应s:””实际上为空，就导致了后续字段的解析，从而造成溢出、解析、赋值    根据上面的步骤我们编写脚本如下：    代码重点就两步：第一个页面的sessionid赋值使得“token=beijixiong”；第二个页面的“cookie-user=base64(‘beijixiong’)” 5.4 SSRF 任务地址： <http://58.240.236.231:30080/> 5.4.1 172.72.23.21 - SSRF 获取本地信息 此时的本地信息获取的是该爬虫服务器的信息，可以看到对应的还有一个172.72.23.21的网卡    之后对该网段下存活主机端口进行爆破，设置两个变量，并加入对应的字典（比如，对第一个变量设置数字范围，第二个设置字典为常见的端口号）    将攻击载荷设置为从21-31的数值类型    爆破的结果如下，可以看出30和31的响应时间过长，所以30和31不存活；实验中给出的目标网段均处于存活状态    端口爆破原理同上，第二个攻击载荷即网站端口设置为常用的端口    结果如下：   5.4.2 172.72.23.22 - 代码注入 首先访问该网站发现只是一个简单的提示页面    查看phpinfo.php文件    查看shell.php文件发现是一个简单的一句话木马    首先构造一个请求包如下所示    将流量包发送到intuder中将payload设置为攻击载荷    之后在[cpkkcb/fuzzDicts: 渗透测试路径字典，爆破字典。内容来自互联网和实战积累。 (github.com)](https://github.com/cpkkcb/fuzzDicts)该网址可以下载到字典文件  将字典文件导入到攻击载荷的参考文件中    对其进行攻击得到的结果如下所示，可以发现确实存在有shell.php和phpinfo.php文件    因为在网站中有shell.php存在，所以我们可以直接访问如下网址来进行漏洞利用  172.72.23.22/shell.php?cmd=cat%20/flag  可以直接获取到flag的值   5.4.3 172.72.23.23 - SQL 注入 首先在搜索框中输入http://172.72.23.23/?id=1’得到的结果如下    将末尾的单引号换成双引号后发现没有报错，成功查找出对应的字段    由此可以判断SQL注入的类型为单引号字符型注入。  接下来需要使用order by语句判断列数，该语句中的%20代表SQL语句中的空格，其中--%20会被识别成注释符号  http://172.72.23.23/?id=1’%20order%20by%201--%20    接下来修改order by 的参数，发现当参数值为5时数据库才会报错，由此可见数据库中存在有4个字段。    接着在搜索框中访问如下网址来判断回显位但发现与直接访问效果一样  172.72.23.23/?id=1'%20union%20select%201,2,3,4--%20    接下来查询172.72.23.23/?id=1'%20and%201=2%20union%20select%201,2,3,4--%20，得到的结果如下，可以得到正常的回显位    接下来先在网站的搜索框中输入给出的SQL注入指令如下    burp suite的抓包结果如下    将抓取的流量包发送到重放器中之后将id之后的内容进行对所有字节进行URL编码，之后将修改后的流量包进行发送可以得到flag为ggctf{63cafe619721609713aedbddfdf4bba，SQL注入攻击成功   5.4.4 172.72.23.24 - 命令执行 首先在网站中访问http://172.72.23.24/访问该网站的内容，发现该网站是一个测试网络端口的页面    在网站页面访问http://172.72.23.24/ip=127.0.0.1;id    抓取到的数据包为    将数据包发送给重放器之后将数据包修改为如下形式的攻击包    点击发送按钮确定攻击包的长度    之后将攻击包全体进行两次URL编码    然后在网页中随便访问一个地址之后进行抓包    将编码后的攻击包放置在抓取的流量包之后添加url=gopher://172.72.23.24:80/\_,将编码两次的攻击包放置在之后，最后构造成的流量包如下    发送流量包之后获得结果flag的值   5.4.5 172.72.23.25 - XML 实体注入 首先登陆之后进行抓包    抓包结果如下    将流量包发送到Repeter之后将下面的利用代码替换<user>部分  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE user [  <!ENTITY xxe SYSTEM "file:///flag" >]>  <user><username>&xxe;</username><password>2</password></user>  构造好的流量包如下    将攻击代码进行两次URL编码如下图所示    在网站中访问一个地址之后进行抓包，抓包结果如下    将编码两次过后的攻击脚本放置到url之后，构造请求如下，发送后可以获取到flag结果   5.4.6 172.72.23.26 - CVE-2017-12615 当 Tomcat 运行在 Windows 主机上启用了 HTTP PUT 请求方法，可以通过构造请求向服务器上传包含任意代码的 JSP 文件，导致 JSP 文件中的代码能够被服务器执行。要上传的jsp数据包，这里相当于一个反向代理。可以看到此时的host=172.72.23.26:8080，也就是这个包是tomcat反弹给这个爬虫网站的shell    下面是需要从爬虫中间网站发送的数据包，将前面的数据包作为包的主体，设置之后进行两次URL编码，因为这里相当于从本地——》爬虫网站——》tomcat服务，中间经历了两次转换    成功上传之后，上面两个数据包的流程就相当于是，爬虫网站给tomcat发送一个包，来让tomcat返回给爬虫站一个shell（即反弹shell），如此一来我们就可以通过这个爬虫中间站进行远程代码执行。  我们就可以在中间跳板网站上执行“cat /flag”了（命令经过了一次url编码）   5.4.7 172.72.23.27 - Redis 未授权 这里我们使用redis中的info命令，来判断是否存在未授权    看来是有了，接下来清空key    设置定时任务目录，/etc/crontab或者/etc/cron.d/也是可以存放定时任务    设置文件名    向其中写入我们的反弹shell（没有设置反弹地址，是因为这里我们没有公网ip，无法监听2333端口返回的shell）    保存上述操作   5.5信息收集实验5.5.1 Whois 查询 使用 Whois 工具查询武汉大学(whu.edu.cn)的注册信息，查看注册者、注册商等信息  % IANA WHOIS server  % for more information on IANA, visit http://www.iana.org  % This query returned 1 object  refer: whois.cnnic.cn  domain: CN  organisation: China Internet Network Information Center (CNNIC)  address: No. 4, South 4th Street  address: Zhong Guan Cun  address: Beijing 100190  address: China  contact: administrative  name: Yu Zeng  organisation: China Internet Network Information Center (CNNIC)  address: No. 4, South 4th Street  address: Zhong Guan Cun  address: Beijing 100190  address: China  phone: +8610-58813686  fax-no: +8610-58813632  e-mail: ceo@cnnic.cn  contact: technical  name: Anlei Hu  organisation: China Internet Network Information Center (CNNIC)  address: No. 4, South 4th Street  address: Zhong Guan Cun  address: Beijing 100190  address: China  phone: +8610-58813202  fax-no: +8610-58812666  e-mail: tech@cnnic.cn  nserver: A.DNS.CN 2001:dc7:0:0:0:0:0:1 203.119.25.1  nserver: B.DNS.CN 2001:dc7:1:0:0:0:0:1 203.119.26.1  nserver: C.DNS.CN 2001:dc7:2:0:0:0:0:1 203.119.27.1  nserver: D.DNS.CN 2001:dc7:1000:0:0:0:0:1 203.119.28.1  nserver: E.DNS.CN 2001:dc7:3:0:0:0:0:1 203.119.29.1  nserver: NS.CERNET.NET 202.112.0.44  ds-rdata: 57724 8 2 5d0423633eb24a499be78aa22d1c0c9ba36218ff49fd95a4cdf1a4ad97c67044  whois: whois.cnnic.cn  status: ACTIVE  remarks: Registration information: http://www.cnnic.cn/  created: 1990-11-28  changed: 2023-09-21  source: IANA  Out of this registry.   5.5.2备案信息查询 在备案信息查询网站上查询武汉大学的备案信息，了解网站备案主体信息   5.5.3 IP 资产收集 进行 IP 范围扫描，识别武汉大学的主机和开放端口情况，使用工具如 Nmap     5.5.4 CDN 检测 使用工具检测武汉大学是否使用 CDN（内容分发网络）服务。  使用[https://www.whu.edu.cn/ CDN测试结果 网站速度测试 17CE](https://www.17ce.com/)工具对武汉大学官网[www.whu.edu.cn进行ping](http://www.whu.edu.cn进行ping)测试，发现所有检测点解析的ip仅有一个，说明武汉大学没有使用CDN服务。   5.5.5 企业信息收集 使用搜索引擎和工具查找与武汉大学相关的企业信息。     5.5.6 域名信息收集 **子域名查询**：利用工具如 oneforall、Amass、Sublist3r 查询武汉大学的所有子域名  **枚举**：使用子域名爆破工具对武汉大学的子域名进行枚举  **网络空间搜索引擎**：利用 Google Dork、FOFA 等搜索引擎搜索武汉大学的相关信息  **（1）子域名查询**：利用工具如 oneforall、Amass、Sublist3r 查询武汉大学的所有子域名      **（2）枚举**：使用子域名爆破工具对武汉大学的子域名进行枚举  在网站中对武汉大学下的所有子域名进行查找结果如下    使用JSFinder工具爆破武汉大学的所有子域名结果如下    **（3）网络空间搜索引擎**：利用 Google Dork、FOFA 等搜索引擎搜索武汉大学的相关信息  使用FOFA搜索武汉大学根域名信息，domain=”www.whu.edu.cn”   5.5.7 Web 服务探测 使用工具如 WhatWeb、Finger 对武汉大学网站的 Web 服务进行探测和漏洞扫描  （1）whatweb  执行如下命令扫描武汉大学网站的web服务，可以获取到网站中web服务的各种信息  whatweb www.whu.edu.cn    加入参数-v之后可以看到web服务中的各种详细信息          （2）finger  在<https://github.com/EASY233/Finger>下载Finger工具并配置完成相关环境之后运行如下指令探测web服务的信息。  python3 Finger.py -u www.whu.edu.cn    运行的结果保存到了对应的xlsx文件中如下图所示     5.5.8 指纹识别 使用工具如 Wappalyzer、BuiltWith 对武汉大学网站运行的框架、技术等进行指纹识别  （1）wappalyzer  wappalyzer工具是一个浏览器插件，在浏览器中下载该插件之后访问武汉大学网站如下图所示，可以获取到目标网站的运行框架和技术等信息    将插件获取到的数据导出后可以得到如下结果      （2）BuiltWith  登陆BuiltWith网站之后在搜索界面直接输入www.whu.edu.cn可以获取到该网站的所有信息。查看目标网站的所属公司信息如下图所示    查看目标网站的所有入站重定向信息如下    查看目标网站的IP地址历史记录和关系如下图所示      查看目标网站的元概要文件如下图所示    查看目标网站的详细技术简介如下图所示         5.5.9 邮箱收集 通过搜索引擎或EmailAll等工具查找武汉大学网站中的相关邮箱地址。  EmailAll是一款邮箱收集工具，在[Taonn/EmailAll: EmailAll is a powerful Email Collect tool — 一款强大的邮箱收集工具 (github.com)](https://github.com/Taonn/EmailAll)下载该工具的项目文件之后配置环境之后执行如下指令对目标网站的邮箱地址进行查找  python3 emailall.py --domain www.whu.edu.cn run  最终结果会保存到www.whu.edu.cn\_All.json文件中如下所示    终端中输出的信息如下所示   5.5.10 敏感信息收集 搜索可能泄露的敏感信息，如密码、配置文件等。  尝试使用BBScan.py工具进行扫描，结果如下    搜索可能泄露的敏感信息，如密码、配置文件等。    Skipfish       5.6.Linux提权实验 首先查看主机的内核版本    将下载到的dirty文件进行编译  gcc -pthread dirty.c -o dirty -lcrypt  之后执行获取的可执行文件  ./dirty 1    在创建用户之后执行如下命令,然后查看/etc/passwd文件的内容如下图所示  mv /tmp/passwd.bak /etc/passwd    尝试使用新账户登陆可以发现成功实现提权过程 | | | | | |
| 六、实验过程分析 1.在SSRF命令执行任务中，需要注意的是在构造完成攻击包之后，攻击包二次URL编码之前需要讲攻击包发送一次来校对攻击包长度。并且在攻击包注入命令时需要注意命令格式，注意指令中的空格。如果没有按照正常的操作顺序则会在发送数据包之后不能够获取到目标flag或者无法访问网页。  2.Linux提权实验中首先尝试Ubuntu14.04版本的操作系统，但是在运行脏牛漏洞利用程序之后查看用户信息之后并不会成功提权，在经过多次尝试之后使用Ubuntu 10运行脏牛程序后发现可以成功复现实验结果，最后分析原因发现需要Linux的内核版本尽可能与2.6.22相近才能够成功，推测原因可能是在之后的Linux内核打入补丁避免了脏牛漏洞的执行。  3.SQL注入实验中使用burp suite抓包之后使用重放器讲构造好的流量包进行重发之后发现没有成功返回数据。最后通过将url中部分内容进行url编码之后再将流量包进行发送才能够成功获取信息。 | | | | | |
| 七、实验结果总结 在本次实验中成功完成了课程任务中的所有任务，在第一天的反序列化任务中时，在配置本地服务器环境的过程中遇到了很大的困难，最后采取phpstudy来运行本地环境，在解决的过程中不仅尝试了手动方式还尝试使用了php自动化脚本来完成实验内容。  在第二天的SSRF实验过程中构造攻击流量包的过程中尝试了各种构造方法以及注入指令之后成功构造能产生效果的攻击流量包。  在Linux提权实验中尝试了Ubuntu的各种版本的系统以及能够获取的各种版本的脏牛程序最终下载到了内核版本与实验要求最为接近的Ubuntu版本成功实现了脏牛提权实验。 | | | | | |
| 八、各人实验贡献与体会 赵伯俣  贡献：xpath之bwaspp实例一；xpath之bwaspp实例二；SSRF 代码注入；SQL注入；命令执行；XML实体注入；Web服务探测；指纹识别；邮箱收集；实验报告汇总  体会：  在本次实验过程中，我掌握了XPath在Web应用中的应用，包括成功登陆和注入获取XML文件内容，通过反序列化漏洞攻击和SSRF攻击，学习到如何利用Web漏洞进行系统渗透和内网攻击；在信息收集方面，通过对 武汉大学的全面信息收集，学习到使用各种信息收集工具和技术，并能够将收集到的信息进行整理和汇总；在系统提权方面，我掌握了针对Ubuntu系统的提权方法。  总的来说，虽然在实验开始前并没有对web安全有比较深层次的研究，但是在实验操作的过程中以及小组共同解决问题时让我对网络安全中的web安全领域有了更深入的理解，还提高了我的实践能力和问题解决能力。 | | | | | |
| 教师评语 | | | | | |
|  | | | | | |
| **教师评分（请填写好姓名、学号）** | | | | | |
| 姓名 | | 学号 | | 分数 | |
| 赵伯俣 | | 2021302181156 | |  | |
| 刘竞优 | | 2021302181057 | |  | |
| 张竣尧 | | 2021302181004 | |  | |
| 教师签名：  2024年 5月 日 | | | | | |