

sqlmap

sqlmap 就是一个自动化的 sql 注入工具，他的主要功能就是扫描，发现并利用给定 url 的 sql 注入漏洞，sqlmap 很多绕过插件，同时也支持很多数据库 mysql--oracle--postgresql--microsoft sql server--microsoft access--ibm db2--sqlite--firebird--sybase--sap maxdb--

它采用了 5 种独特的 sql 注入技术

基于布尔类型的盲注入，根据他的返回页面判断条件真假的注入

sqlmap 有很多强大的功能 包括数据库指纹识别 数据库枚举 数据提取 访问目标文件系统 如果有权限时可以实行任意命令 加-r 还可以从文本文件中加载 http 请求

在确定存在注入时

-dbs 列出当前用户下所有数据库 --tables 列表 --columns 字段名

然后如果想具体查询字段中信息的数据可以 --dump 下载

在权限允许的情况下可以 --users 列出含所有用户的表

还可以使用--level 5 设定探测等级 一共有 5 个 默认 1 sqlmap 使用的 payload 可以在

xml/payloads.xml 当中 也可以自己添加 payload 5 的 payload 最多会自动破解 cookie，xff 登头部注入 但是运行速度叫慢 level 2 http cookie level 3

http ua、referer 当然也可以使用 --referer 伪造 http 中的 referer 如: --

referer http://www.baidu.com 也可以自己自定义语句 使用 --sql-shell

在数据库 mysql, postgresql, mssql, 在当前用户有权限使用特定的函数的

话可以上传一个二进制库 使用 sys_exec() sys_eval() 就可以执行系统命令

--os-shell 可以模拟一个真实的可执行环境输入自己想执行的命令 在权限允许

的情况下可以使用 into outfile 讲一个 webshell 写入一个可写目录

--tamper 参数就是对数据修改来绕过 waf 等设备。其中大部分脚本主要用正

则模块替换攻击载荷字符编码的方式来尝试绕过 waf 在日常使用中 我们会对

一些网站查看是否有安全防护如: ids ips waf 可以使用 -- identity-waf 常见

的 base64encode.py 将语句替换为 base64 编码。multiplespace.py 在 sql

语句关键字添加空格

burpsuite

burpsuite 是一款由 java 编写的集成化渗透测试工具, 因为是 java 编写的运行

时依赖 jre

需要安装 java 环境才可以运行。

其中分为免费和专业版,

burp scanner

工作空间的保存和修复

还有些拓展攻击也不能使用

burp suite 是以拦截代理的方式，可以通过 burp proxy 代理模式拦截所有通过代理的网络流量 其主要是拦截 http/https 协议的流量 通过拦截 burpsuite 以中间人的方式对客户端的请求数据服务端的返回信息进行拦截 查看 修改 最常用的一般就是 web 浏览器 可以通过设置代理信息 拦截 web 浏览器的流量，并对其处理 burpsuite 运行后默认本地代理端口 8080

burpsuite 拦截功能主要由 intercept 选项中的 forward drop interception is on/off 和 action 构成

forward：表示讲拦截的数据包或修改后的数据包发送至服务器端

drop：表示丢弃当前拦截的数据包

interception：是 burp 拦截功能的开关

action：可以讲数据包进一步发送到 spider scanner repeater intruder 等组件当中也包括改变数据包请方式及 body 的编码等功能，在拦截了客户端和服务端的交互之后可以在消息分析选项卡中查看这次请求的实体内容，消息头，请求参数等信息。

四中消息类型：

raw 主要显示 web 请求的 raw 格式，纯文本的形式显示数据包，包含请求地址，http 协议版本，主机头 浏览器信息，accept 可接受的内容类型，字符集。编码方式，cookie 等。可以手动修改这些信息

params 主要显示客户端请求的参数信息，包括 get 或者 post 请求的参数。cookie 参数

headers 中显示的是数据包中的头信息，以名称，值的形式显示数据包

hex 对应的是 raw 中信息的二进制内容，可以通过 hex 编辑器对请求

修改在进行 00 截断时非常好用

burp spider 爬虫功能 可以帮助我们爬去了解系统的结构 其中爬去道德内容将在 target 中会显示主机和目录树 选择具体分支可以查看对应的请求响应。主要是通过 html javascript robots.txt 以及提交的表单中的超链接来映射目标应用程序

decoder 是 burp 中自带的编码解码及散列转换的工具 编码解码
decode as

解码 encode as 还有 hash 目前所支持的编码 url html base64

ascii 十六进制八进制 二进制

hash 支持 sha sha-256 sha-224 sha-256 md5 等格式的转换

burp scanner 主要自动检测或者主动扫描和被动扫描去检测 web 系统的各种漏洞 会扫描通过代理服务的请求 也会显示在 target 当中可以点击 issues report issues 到处报告

主动扫描：向应用发送新的请求并验证漏洞 这种模式会产生大量请求和应答数据会影响到服务端性能，用于非生产环境

客户端漏洞：xss http 头注入 操作重定向

服务端漏洞：sql 注入，命令注入，文件遍历

被动扫描：不发送新的请求 利用现有的请求和应答进行分析

提交明文的密码

不安全的 cookie 的属性 缺少 httponly 安全标示

ssl 保护的内容缓存

session 的不安全传输

intruder 对 web 应用程序进行自动化的攻击 通过表示服 对用户名 id 账号 模糊测试 sql 注入 跨站 就是在请求原始数据的基础上 通过就各种请求参数得到不同的请求应答 通过会携带一个或多个不同的 payload 在不同位置重放 通过获取的 status length 的值和返回包的数据 分析有什么不同之处 是否登录邓公

repeater 可以对请求的消息进行请求重放 修改请求参数 去验证逻辑 注入 越权

comparer 可以对两个数据包进行对比查看差异可用于枚举 intruder 盲住等

sequencer 用于检测数据随机性的工具 session token 是否可预测

nmap

network mapper 是一款开源的网络探测和安全审计工具 用于主机探测 发现开放的端口情况 以及操作系统与应用服务的指纹识别 waf 识别 他的图形化是 zmap 分布式框架 dnmap

特点

主机探测：去探测目标主机是否在线

端口扫描：探测目标主机所开放的端口

版本检测：探测目标主机的网络服务的名称版本

系统检测：操作系统 及硬备的特性

实战过程中可以使用 -sF -T4 查看防火墙是否关闭

nmap 常用脚本 在 /nmap/scripts 文件下 也可以 --script=auth 对目标主机进行应用弱口令检测 以及暴力破解功能 --script=brute 可对数据库 smb snmp 等进行简单的密码猜解 --script=vuln 对主机扫描是否存在常见漏洞 常见的应用服务 vnc mysql telnet 等 --script=realvnc-auth-bypass

探测局域网内更多服务开启的情况：

```
nmap -n -p 445 --script=broadcast
```

利用第三方数据库或资源查询目标地址的信息列入 whois 解析

```
nmap --script external baidu.com
```

脚本使用方法 <https://nmap.org/nsedoc/categories>

open 表示开放应用程序正在监听 该端口的连接

常见的 6 种状态：

open 开放 表示应用程序正在监听该端口的连接 外部可访问

filtered 被过滤 表示端口被防火墙或其他网络设置组织 不能访问

closed 关闭 表示目标主机未开启该端口

unfiltered 未被过滤 表示 nmap 无法确定端口所处状态

appscan

appscan 是根据起始页面爬去站下所有可见的页面 同时测试常见的管理后台
利用 sql 注入的原理进行测试是否有注入点或者跨站脚本的可能 还会对 cookie
管理会话周期常见的漏洞进行检测, 同时支持登录功能和移动端应用 app 的扫描 扫描
完成之后提供了详细的漏洞原理 修复建议 验证等一些功能 缺点 作为商业软件 价格比
较昂贵

适用于企业的内外网和面向客户 厂商 的 web 网站

sql-注入

就是利用现有应用程序 将不受信任的数据或者是恶意的 sql 语句插入到 web
表单提交或输入域名或者页面的请求的查询字符串, 在代码逻辑不严谨或参数可控的
情况下, 将恶意的 sql 命令注入到后台数据使愿意产生奇异 最终达到欺骗服务器执行
恶意的 sql 命令 从而间接的操控一个存在漏洞的网站数据库

构造恶意语句最大程度的原因是因为字面量的原因, 字面量中使用单引号必须连续使
用两个单引号来表示 称为单引号转义 由于没有进行单引号转义造成了拼接恶意的 sql
语句

所产生的危害:

数据库内的个人信息被窃取

数据库中的内容被恶意篡改

登录认证绕过

类型:

sql 注入, Nosql 注入, os 注入, ldap 注入`

sql 注入方式:

报错型注入。盲注 (布尔, 时间) ,

请求方式: get, post

注入位置: u a , cookie

手工注入: <https://www.2cto.com/article/200604/8334.html>

sql 注入防御:

数据代码分离原则

预编译语句, 绑定变量。使用预编译的 sql 语句, sql 的语意不会变化, 攻击者无法改变 sql 的结构, 即使攻击者插入了 'or' 1 '='1 的字符串, 也只是会讲此字符串作为 username 查询。

从存储的过程来防御: 先将 sql 语句定义在数据库中, 存储过程中可能存在注入问题。应该尽量避免在存储过程中使用动态 sql 语句

数据库类型角度: 限制数据库类型, 并同意数据格式,

从数据库管理这角度: 以最小权限原则, 避免 root, sa, dbowner 等高权限用户直接连接数据库

sql 注入命令

- **SELECT** - 从数据库中提取数据

- **UPDATE** - 更新数据库中的数据
- **DELETE** - 从数据库中删除数据
- **INSERT INTO** - 向数据库中插入新数据
- **CREATE DATABASE** - 创建新数据库
- **ALTER DATABASE** - 修改数据库
- **CREATE TABLE** - 创建新表
- **ALTER TABLE** - 变更（改变）数据库表
- **DROP TABLE** - 删除表
- **CREATE INDEX** - 创建索引（搜索键）
- **DROP INDEX** - 删除索引

文件上传

详解: <https://www.jianshu.com/p/5ebba0482980>

在 web 应用程序中 有很多常见的上传功能如上传 图片 头像 文件 视频 如果 web 应用存在上传漏洞 或者说 服务端在未对客户端上传的文件进行严格的验证和过滤 就容易造成可以上传任意文件 包括上传一些 (asp.aspx.php.jsp) 脚本格式的文件 到服务器中, 获得网站的权限 甚至控制整个服务器, 恶意脚本称为 webshell 》

webshell 就是以 asp php jsp 等网页文件格式存在的一种命令行执行环境, 通常攻击者在入侵一个网站后会将 asp php 后门文件上传到网站 web 根目录下 然后用浏览器去访问这个后门 就可以得到一个命令行的执行环境

原理：

在 WEB 中进行文件上传的原理是通过将表单设为 multipart/form-data，同时加入文件域，而后通过 HTTP 协议将文件内容发送到服务器，服务器端读取这个分段 (multipart) 的数据信息，并将其中的文件内容提取出来并保存的。通常，在进行文件保存的时候，服务器端会读取文件的原始文件名，并从这个原始文件名中得出文件的扩展名，而后随机为文件起一个文件名（为了防止重复），并且加上原始文件的扩展名来保存到服务器上。

文件上传后导致的常见安全问题一般有：

- 上传文件是 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；
- 上传文件是 Flash 的策略文件 crossdomain.xml，黑客用以控制 Flash 在该域下的行为（其他通过类似方式控制策略文件的情况类似）；
- 上传文件是病毒、木马文件，黑客用以诱骗用户或者管理员下载执行；
- 上传文件是钓鱼图片或为包含了脚本的图片，在某些版本的浏览器中会被作为脚本执行，被用于钓鱼和欺诈。

除此之外，还有一些不常见的利用方法，比如将上传文件作为一个入口，溢出服务器的后台处理程序，如图片解析模块；或者上传一个合法的文本文件，其内容包含了 PHP 脚本，再通过“本地文件包含漏洞（Local File Include）”执行此脚本；等等。此类问题不在此细述。

常见的上传漏洞：

``前端：

js 检测绕过的攻击形式 上传一个脚本格式文件直接会弹窗告诉你格式不允许并且也抓不到数据包说明数据包没有发到服务端 他的验证代码存放在本地客户端 用 js 对数据包检测
绕过：删除检测函数，上传相符格式文件抓包更改成脚本文件。

函数：selectfile()函数 将其转为小写 substr 获得 点号后面的后缀是否符合

文件后缀 “

服务端代码限制了某些后缀的文件不允许上传，但是有些 apache 允许解析其他文件后缀 如可以解析 php phtml 可以上传此文件类型的 webshell 因为 apache 解析顺序是从有往左解析文件后缀 如果最右侧扩展名不可能识别继续解析到可以识别的文件后缀，如上传

1.php.xxxx

函数 phpinfo() 获取文件后缀 将其转为小写 判断是否是 php 通过 phtml 绕过

文件类型

比如说：上传一个 php 格式文件 抓包可以看到

content-type:application/octet.stream

上传一个正常 jpg 格式文件时 值是：image/jpeg

服务端通过 content-type 的值来判断文件类型 以为是通过客户端传递的可以抓包进行修改 就可以绕过服务端

服务端代码判断：\$_FILES["file"]["type"] 是不是 image/jpeg 格式

而\$_FILES["file"]["type"] 是客户端的 content-type 则通过修改值绕过

getmagessize ()

在 php 中 `getimagesize ()` 函数可以获取图片的宽 高信息，
如果上传 的不是图片文件 那么此函数就获取不到信息则不允许
上传

绕过：制作图片码

文件截断绕过

00 代表结束符 00 后面的所有字符删除

截断条件 php 版本小于 5.3.4 php 的 magic——quotes_gpc off 状态

服务端 get 参数截断后的内容作为上传的第一部分然后按时间生成的图片

文件名作为上传后文件名的第二部分，多数截断是用%00 HEX 形式进行

的 filename=1.php%001.jpg 00 之后的内容被截断 所以

`$_files['file']['name']`得到的后缀是 php 而不是 php%00jpg

通过 `substr` 获取后缀 判断是否符合正规格式 但是保存路径中有

`$_request['jieduan']` 可以利用%00 截断

竞争条件

一些上传文件的逻辑是先允许上传文件 在检测是否是 webshell 如果是就

删除该文件 在上传成功和删除文件之间有一个短的时间差。攻击者可以

利用这个短的时间差完成竞争条件的攻击

如上传一个 10.php 其内容是生成一个 shell.php 上传成功后客户端就会

访问 10.php 则会在服务器当前目录生成 shell.php 然后删除此文件在删

除之前访问极可能生成 webshell

原因：

对于上传后的文件名的后缀没有做严格的校验

对于上传的文件 mimetype（描述文件类型的方法）没有做检查

没有对上传的文件目录设置不可执行权限

web server 对于上传文件或者指定目录的行为没有做限制

防御：

通过白名单方式截断后缀是否合法

对上传后的文件进行重命名 rand(10,99).data("ymdhis").".jpg"

简易的 webshell <?php echo shell_exec(\$_GET['cmd']);?>

XXE - 外部实体注入

XML: (XML External Entity)

可扩展性标记语言，用于标记电子文件使其具有结构性的标记语言，可允许用户自己的标记语言进行定义的源语言 xml 主要是把数据从 html 中分离，xml 是独立于软硬件的一种传输工具

ENTITY（实体），XML 中的实体类型，一般有下面几种：字符实体、命名实体（或内部实体）、外部普通实体、外部参数实体。除外部参数实体外，其它实体都以字符（&）开始，以字符（;）结束

xml 用来传递和存储数据

html 用来显示数据

为什么要引用 DTD 文档? : 引用 DTD 文件和 xml 一起使用, 目的是对 xml 格式的约束

漏洞产生原因: 主要是利用 DTD 引用外部实体导致的漏洞

1、 xml 文档类型:xml 声明>DTA 文档定义>文档元素

DTD 文档定义作用: 用来定义 xml 文档的合法构建模块

DTD 的构建分为两种模式:

内部实体声明: `<!DOCTYPE 根元素 [元素声明]>`

外部实体声明: `<!ENTITY 实体名称 SYSTEM "URI/URL">`

`<!ENTITY entity-nmae PUBLIC "public_ID" "URL">`

entity-name

外部实体引用资源分为两种类型关键字 system 本地计算机 Public 公共计算机

根据不同的语言选择支持的协议:http php file 去引用本地或公共的资源 就会造成任意文件读取, 命令 (代码) 执行, SQL 注入, 内外网扫描端口

xml 引用实体格式如下:

`<!ENTITY 实体引用名 "引用内容" >`

实体分两类:

1.一般实体 (格式: &实体引用名)

2.参数实体 (格式: %实体引用名)

一般实体, 可以在 XML 文档中的任何位置出现的实体称为一般实体。实体可以声内部实体还是外部实体。外部实体分 SYSTEM 和 PUBLIC 两种。

SYSTEM 引用本地计算机，PUBLIC 引用公共计算机，外部实体格式如下：

<!ENTITY 引用名 SYSTEM(PUBLIC) "URL 地址">

在 XML 中，实体必须在 DOCTYPE 声明中声明。<!DOCTYPE [...]> 声明在 XML 声明的后面

如：<?xml version="1.0" encoding="UTF-8">

<!DOCTYPE 根元素名 [

<!ENTITY 实体引用名 "引用内容" >

<!ENTITY 实体引用名 SYSTEM "引用内容">

]>

。

xxe 两种显示场景：

有回显，无回显，有回显的情况可以直接在页面中看到 payload 的执行结果或现象

无回显情况又称盲注 blind.exe 无回显 XXE 不同于有回显的 XXE 漏洞，可以直接利用，我们需要通过构建一条带外信道提取数据。

这里使用的是内部实体嵌套，先由 %remote 引入外部 DTD，然后引入 %all，此时检测到其内部嵌套的实体 %send，引用执行，达到带外信道提取数据的目的

漏洞检测：

检测请求中 content-type 为 xml 的相关类型的数据包 以及发送相关的 xml 请求 可以很大概率的判断 存在 xml 的解析逻辑

业务上传 xml 文件数据的请求接口，可通过上传含外部实体加载的 POC 去验证是否存在漏洞

漏洞防御：

禁止 DTD 在不能禁用 DTD 使用一下资源

防止外部普通及参数实体 poc 的攻击，禁止加载外部实体

XSS

在生成 html 过程中 html 中含有特殊意义的元字符 因为没有被正确的处理而导致注入了 hrml 或者 javascript，使原先的 html 结构产生变化。可以通过转义的方式讲将特殊意义的字符转换为普通字符。如 html 中显示<时 可以使用< 如果忽略这一步会将<解释为标签的开始从而就会招致恶意利用此漏洞进行 xss 攻击。

产生元原因：

html 没有对< "进行转义

修复：

使用字符实体转义 元素内容转义 < &

属性 使用双引号转义 属性内容转义 < " &

php 使用 htmlspecialchars 函数进行 html 的转义 此函数最多接收 4 个参数

输入校验

cookie 中添加 httponly 属性

在 perl 中转义 html 使用 cgi.pm 中的 escapehtml 方法

SSRF 服务器请求伪造

有攻击者构造请求，有服务器发起请求的漏洞，一般情况下 SSRF 攻击的目标是外网无法访问的内部系统 因为请求是有服务端发起的，所以服务端能请求到与自身相连而与外网隔离的内部系统，ssrf 形成的原有多都是由于服务器提供了从其他服务器应用获取数据的功能且没有对目标地制作过滤与限制，比如 攻击者操作服务端从指定 url 地址获取网页文本内容，加载指定地址的图片，ssrf 利用存在缺陷的 web 应用作为代理攻击远程和本地的服务器

ssrf 与 csrf 的区别

csrf 是服务端没有对用户提交的数据进行随机值校验，并且对 http 请求包内的 referer 字段校验不严格 导致攻击者可以利用用户的 cookie 信息伪造请求发送至服务器。

ssrf 是服务端对于用户提供的可控 url 过于自信，没有对用户提交的 url 进行限制和严格的检测

导致攻击者可以一次为跳板攻击内网中的 其他服务器

SSRF 类型

显示对攻击者响应：在服务器获得攻击者请求的 url 的内容之后 直接把数据发送给攻击者

不显示响应：在服务器获得攻击者请求的 url 后 不发送相应内容给攻击者，需要通过抓包查看服务器中日志去判断是否存在服务器请求伪造

利用 ssrf 所造成的攻击：

对外网，服务器所在的内网，本地进行端口扫描 获取一些服务的 banner 信息（软件开发商，软件名称）以及收集内网的 web 应用的指纹识别（开放的端，中间件版本）

可以攻击处在内网的系统和应用，获取内网系统的弱口令 进行内网漫游，对内 web 应用实施攻击获取 webshell。

利用 file 协议读取本地文件等，，

如果存在 ssrf 漏洞所实现的功能是获取 GET 参数 url 将 url 的内容返回网页上，如

果请求的网页是 <http://www.baidu.com> 返回的内容就是 <http://www.baidu.com>

但如果吧参数 url 设置为内网地址时 就会泄露内网的信息比如

127.0.0.1/ssrf.php?url=http://192.168.0.2:3306

127.0.0.1/ssrf.php?url=file:///C:/Windows/win.ini 即可读取本地文件

ssrf 漏洞修复建议：

限制请求的端口为 web 端口，只允许访问 http 和 https 的请求

限制不能访问内网的 ip，以防止对内网进行攻击

屏蔽返回的详细信息

weblogic ——ssrf

Weblogic 的 UDDI 功能可以对 web service 进行操作，利用该功能发起对本地端口探测等操作。我们可以通过 CRLF 注入来构造攻击 payload，CRLF 是“回车 + 换行”（\r\n）的简称，在 HTTP 协议中，HTTP Header 与 HTTP Body 是用两个 CRLF 分隔的，浏览器就是根据这两个 CRLF 来取出 HTTP 内容并显示出来。所以，一旦我们能够控制 HTTP 消息头中的字符，注入一些恶意的换行，这样我们就能注入一些会话 Cookie 或者 HTML 代码，所以 CRLF Injection 又叫 HTTP Response Splitting，简称 HRS。

php 反序列化

序列化：

当在 php 中创建了一个对象后，为了方便保存以及之后的传递和使用可以通过

序列化 `serialize()`[连载](#) 把这个对象转变成一个字符串。

本质上 `serialize()`和 `unserialize()` 在 php 内部实现上是没有漏洞的。漏洞主要产生是由于应用程序在处理对象，魔术函数以及序列化相关问题的时候导致的

当传递 `unserialize()`的参数可控时。那么用户就可以注入精心构造的 `payload` 。

当进行反序列化的时候就有可能会出发对象中的一些魔术方法，造成意想不到的危害。

魔术方法：

php 中有一类特殊的方法叫 “Magic function” ， ↓↓↓↓↓↓

`__construct()` 当一个对象创建时被调用，但在 `unserialize()` 时是不会自动调用的。

`__destruct()` 当一个对象销毁时被调用

`__toString()` 当一个对象被当作一个字符串使用

`__sleep()` 在对象在被序列化之前运行

`__wakeup` 将在序列化之后立即被调用

漏洞产生原因：、

当 `unserialize()` 参数可控使 可以使用 `destruct()` 函数将任意文件删除 或者其实

问题也可能存在于 `__wakeup`、`__sleep`、`__toString` 等其他 magic 函数，一切

都是取决于程序逻辑

应用场景：

1. 某用户类定义了一个 `__toString` 为了让应用程序能够将类作为一个

字符串输出 (`echo $obj`) , 而且其他类也可能定义了一个类允许

`__toString` 读取某个文件。

2.利用 file_get_contents()获取 php 文件的源码

Unserialize 漏洞依赖几个条件

- 1) unserialize 函数的参数可控
- 2) 脚本中存在一个构造函数、析构函数、__wakeup()函数中有向 php 文件中写数据的操作的类
- 3) 所写的内容需要有对象中的成员变量的值

防范方法有：

- 1) 要严格控制 unserialize 函数的参数，坚持用户所输入的信息都是不可靠的原则
- 2) 要对于 unserialize 后的变量内容进行检查，以确定内容没有被污染

java 反序列化

该漏洞在最新版的 WebLogic、WebSphere、JBoss、Jenkins、OpenNMS 中的应用，实现远程代码执行。绝大多数编程语言会提供内建的方法允许用户

把自身应用所产生的数据存入到硬盘或者输出到其他地方 **这种数据转化格式的过程称为 序列化**，接收到数据之后去读取的过程称之为反序列化。

漏洞产生的原因：

漏洞产生的主要原因是因为 java 编写的 web 应用和 web 服务器 java 通常会发送大量的序列化的对象 如以下场景：HTTP 请求 中的参数：cookie 以及 perematers

RMI 协议使用另其两个协议为基础：java 序列化对象和 http，对象序列化协议用于编组调用和返回数据，http 协议用投寄 远程方法调用 在情况允许时获得返回数据

序列化的过程：会使用 objectoutputstream 类的 writeobject()方法，在接收到数据后会使用 objectinputstrean 类的 readobject() 方法进行反序列化数据读取

问题的主要产生点：、

java 应用对用户的输入 即可不可信的数据进行了反序列化处理。

那么用户可以构造恶意的 payload 让反序列化产生非预期的对象，非预期的对象可能就会带来任意代码执行 根源在于 objectiutputstream 类在反序列话的过程 没有对产生的对象做类型的限制 如果反序列化的对象的类型可以做成白名单的形式 会减少问题的发生

PHP 命令注入

命令注入攻击 (Command Injection) , 是指黑客通过利用 HTML 代码输入机制缺陷(例如缺乏有效验证限制的表格域)来改变网页的动态生成的内容。从而可以使用系统命令操作, 实现使用远程数据来构造要执行的命令的操作。

PHP 中可以使用下列四个函数来执行外部的应用程序或函数: system、exec、passthru、shell_exec, 四个函数的原型如下:

- string system(string command, int &return_var)
- command 要执行的命令; return_var 存放执行命令的执行后的状态值。
- string exec (string command, array &output, int &return_var)
- command 要执行的命令, output 获得执行命令输出的每一行字符串, return_var 存放执行命令后的状态值。
- void passthru (string command, int &return_var)
- command 要执行的命令, return_var 存放执行命令后的状态值。
- string shell_exec (string command)
- command 要执行的命令, 如下例所示, 表示通过提交

<http://www.sectop.com/ex1.php?dir=>| cat /etc/passwd 操作, 执行命令变成

了 system("ls -al | cat /etc/passwd"), 输出/etc/passwd 文件的具体内容。

```
//ex1.php
```

```
<?php
```

```
$dir = $_GET["dir"];
```

```

if (isset($dir))
{
    echo "";

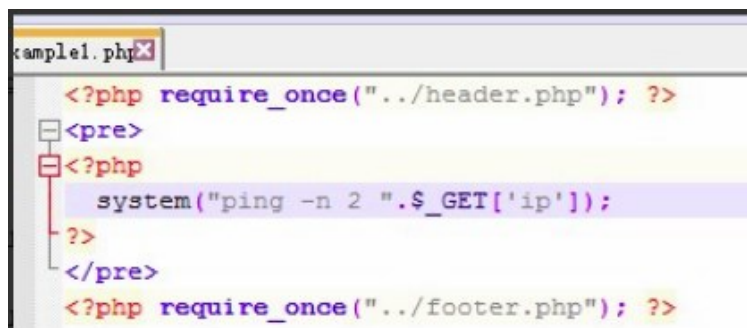
    system("ls -al ".$dir);

    echo "";
}

?>

```

关键代码：



```

example1.php
<?php require_once("../header.php"); ?>
<pre>
<?php
    system("ping -n 2 ".$_GET['ip']);
?>
</pre>
<?php require_once("../footer.php"); ?>

```



```

<?php
if (!(preg_match('/^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$/', $_GET['ip']))) {
    header("Location: example3.php?ip=127.0.0.1");
}
system("ping -c 2 ".$_GET['ip']);
?>

```

preg_match() 函数用于进行正则表达式匹配，成功返回 1，否则返回 0。

preg_match() 匹配成功一次后就会停止匹配，如果要想实现全部结果的匹配，则需使用 preg_match_all() 函数。header()函数的作用是：发送一个原始 HTTP 标头[Http Header]到客户端。标头 (header) 是服务器以 HTTP 协议传 HTML 资料到浏览器前所送出的字串，在标头与 HTML 文件之间尚需空一行分隔。

总结：应用有时会调用系统命令中的执行函数 如：

system,exec,shell_exec,passthru 当用户能够控制这些函数的参数能将一些恶意

的命令拼接到正常的命令当中 从而造成命令注入的一个条件,漏洞 应用没有对用户的输入的代码进行过滤或者过滤不验证 通过这些命令可以查看一些敏感信息 文件或者说打开服务器的远程服务 反弹 `s h e l l` 使用 `&` 符号拼接恶意命令时会被转义成 `% 2 6` 如果输入 `% 2 6` 即可继续执行 可以输入 `||` 符号当前面命令执行失败即执行后面的命令 `preg_match` 通过正则表达式匹配 `ip` 地址匹配到就执行 `ping` 命令所以要输入一个正常的 `i p`

常见的危险函数：

sql 注入：

`select update delete insert`

xss:

`echo print print_r printf sprintf vsprintf var_dump` 代码审计当中追踪

这些关键字可以反向追踪这些函数的变量的是是否可控

文件包含：

`include include_once require require_once`

命令注入：

`system exec passthru shell_exec popen proc_popen pcntl_exec`

文件管理：

```
fgetc  
fgets  
fgetss  
file_get_contents  
file  
fopen  
fread  
parse_ini_file  
readfile  
rmdir  
unlink  
delete  
fwrite  
file_put_contents  
fputcsv  
fputs  
move_uploaded_file
```

代码注入：

```
eval  
preg_replace+/  
assert  
call_user_func  
call_user_func_array  
create_function
```

在 PHP 中可由用户输入的变量列表如下：

```
$_GET  
$_POST  
$_COOKIE  
$_REQUEST  
$_FILES  
$_ENV  
$_HTTP_COOKIE_VARS  
$_HTTP_ENV_VARS  
$_HTTP_GET_VARS  
$_HTTP_POST_FILES  
$_HTTP_POST_VARS  
$_HTTP_SERVER_VARS
```

