

07 Sep 2020 exploit web

During my [OSWE](#) journey I used to try to re-discover known vulnerabilities by watching exploit-db stream to narrow the scope.

Later last year [vikingfr](#) worked on rConfig 3.9.x and had found a neat path from zero to root starting with a pre-auth sql injection.

To refresh my code audit skills, this march I decided to make some practice again: let's start the timer and see what I can find.

Without reading too much information to avoid spoilers (I already know there is at least one sql injection so I'm BIASed), I installed the app and started crawling the tree to get *confident* with the structure.

The source is quite clean and separated between exposed pages, classes included out from the document root, data, and so on. Of course what I'm most interested in, at first, is a pre-auth bug. I choosed to see how authentication works before to look for IDOR.

Login process uses a class `Process` defined in file `www/lib/crud/userprocess.php`, which handles login requests by calling a `login()` function from class `Session` defined in file `classes/usersession.class.php` (line 137 circa), that invokes `checkLogin()`.

[illegible]

By looking at the file, I saw that it's not possible to invoke any of these functions by calling them from the file, so if I'll find something vulnerable I also have to find a sink.

As soon as the class is *constructed*, she runs `startSession()` that runs `checkLogin()`. The first thing `checkLogin()` does is to check for some cookies. I'll be back on this later.

She now checks if there is an LDAP server configured. Because I don't have it ready I'll skip (spoiler: there could be another vulnerability here because \$smbuser has never been sanitized before it's used at line 137, but timer said I must stop. if somebody has time to dig please let me know.) and go for the else branch at line 213.

The function `confirmUserPass()` is defined at `classes/userdatabase.class.php` and looks like safely queries the database by using a prepared statement

```

50 function confirmPassword($username, $password) {
51     // Verify that the user is logged in
52     $this->connection->query("SELECT password FROM " . $DB_PREFIX . " WHERE username = '$username'");
53     $this->connection->bind_result($password);
54     $this->connection->fetch();
55     if ($password == $password) {
56         $this->connection->close();
57         return 1; //Indicates username failure
58     }
59     // Returns password from result, strip slashes //
60     $password = stripslashes($password);
61     // Verifies that password is correct //
62     if ($password == $password) {
63         return 1; //Indicates $username and password confirmed
64     } else {
65         return 1; //Indicates password failure
66     }
67 }

```

If you look closely, you'll spot a possible **PHP type juggling vulnerability** (I didn't validated/exploited this one, ping me if you do): we cannot control the second part of the check (`$dbarray['password']`) but we can partially control the first one.

I think it's not exploitable because user controlled value is hashed as md5, and as far as I remember it will be possible to have a 32byte string or an empty \$password.

I also saw that `confirmUserPass()` receives the password as md5 string, and I saw that the comparison is made on the field retrieved from the database, therefore we know that passwords are stored as md5. This could be an issue because md5 is deprecated and not safe nowadays, but I'll notice later that `rConfig` requires strong passwords so it's not interesting now, but worth a fix in my opinion (I'm a fan of the <https://github.com/defuse/password-hashing>, that has a huge pro: dev can change algorithm/round/salt size very easily).

Of course if it was a real review I've **strongly** suggested to get rid of md5 as soon as possible, but this is another sort of exercise.

login() function checks for return and exit if user or password isn't valid.
SQL queries are good, and given that I haven't found anything useful here I'll note md5 and type juggling and move forward.

```

214         } else {
215             $result = $database->confirmPass($username, $ID,$password);
216
217             /* Check error codes */
218             if ($result == 1) {
219                 $form->setError($field, "Username not found");
220             } else if ($result == 2) {
221                 $field = "pass";
222                 $form->setError($field, "Invalid password");
223             }
224         }
225     }
226
227     /* Return if form errors exist */
228     if ($form->errors == 0) {
229         return false;
230     }
231
232     /* Username and password correct, register session variables */
233     $this->session = $database->getProfile($username);
234     $this->username = $this->session["username"];
235     $this->password = $this->session["password"];
236     $this->email = $this->session["email"];
237     $this->level = $this->session["level"];
238 }

```

As previously said, the function `login()` checks if `rememberme` flag has been checked: if it's true, she sets two cookies for later usage.

```

242 //
243 // This is the cool part: the user has requested that we remember that
244 // they logged in, so we set a cookie. We'll hold its username,
245 // and use to hold its random value secret. It expires by the time
246 // specified in constants.php. Now, next time he comes to our site, we will
247 // log him in automatically, but only if he didn't log out before he left.
248 //
249 if (isset($_COOKIE['$this->username'])) {
250     setcookie('cookie', $this->secret, time() + COOKIE_EXPIRE, COOKIE_PATH);
251 }

```

Before going back where these cookies are validated I'll check where userid value used at line 250 came from

I saw that it comes from a `generateRandID()` function, which generates a new ID of 16chars for this purpose:

```
229 /* Username and password correct, register session variables */
230 $this->userInfo = $database->getUserInfo($username);
231 $this->username = $_SESSION['username']; $this->userInfo['username'];
232 $this->userid = $_SESSION['userid']; $this->generateRandID();
233 $this->userlevel = $this->userInfo['userlevel'];
```

During the login phase, that actually does some sort of random by using a known vulnerable `mt_rand()` function (see more [here](#)) and generates an md5 out of it

```

561 /*
562  * generateHandID - Generates a string made up of randomized
563  * letters (lower and upper case) and digits and returns
564  * the md5 hash of it to be used as a uuid.
565  */
566
567 function generateHandID() {
568     return md5(this->generateHandStr(16));
569 }
570
571 /*
572  * generateHandStr - Generates a string made up of randomized
573  * letters (lower and upper case) and digits, the length
574  * is a specified parameter.
575  */
576
577 function generateHandStr(length) {
578     $randStr = "";
579     for ($i = 0; $i < length; $i++) {
580         $randStr = mt_rand(0, 61);
581         if ($randStr < 10) {
582             $randStr = chr($randStr + 48);
583         } else if ($randStr < 36) {
584             $randStr = chr($randStr + 55);
585         } else {
586             $randStr = chr($randStr + 61);
587         }
588     }
589     return $randStr;
590 }
591
592 }

```

The verification process takes place somewhere around `checkLogin()`, where `userid` taken from cookie `cookid` is checked with what's stored in the database, and if it's false passes over to standard auth.

- [Symfony JMose](#)
- [CommandScheduler RCE](#)
- [rConfig 3.9.4 multiple vulnerabilities](#)
- [Achieve Pareto Principle in secure code review, or die trying](#)
- [Long the Ripper](#)
- [eLearnSecurity eXploit Development Student](#)

- [assembly](#)
- [certifications](#)
- [courses](#)
- [exploit](#)
- [noise](#)
- [red](#)
- [tools](#)
- [web](#)

confirmUserID() function looks safe, still a type juggling one, but I think again not exploitable so I'll move on.

```
66
67
68 + confirmUserID - Checks whether or not the given
69 + username is in the database. If so it checks if the
70 + email exists in the same table in the database.
71 + For that user, if the user doesn't exist or if the
72 + email one is null or is, returns the error code
73 + (1 or 2). On success it returns 0.
74 +
75 +
76 +
77 +
78 +
79 +
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 +
```

Anyway I think it could be possible to bypass login by abusing mt_rand() weakness, but we need some value to get the seed, therefore a valid account. Plus, we don't know if somebody has logged in using rememberme function. Still a vulnerability I'd fix, but not useful to me right now.

Will add this mt_rand() and a new type juggling to my notes, just in case.

Password reset

Back at [www/lib/crud/userprocess.php](#) to review procForgotPass() function, I see that she generates a 8chars string using mt_rand() again

```
570
571 + generateRandStr - Generates a string made up of randomized
572 + letters (lower and upper case) and digits, the length
573 + is a specified parameter.
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 +
```

Then she loads user information and sends an email with the new password.

I don't see any other weakness but mt_rand() usage, that would still need some valid value as far as I know, and a weak password generation (new password length is 8 chars). An attacker could try to bruteforce the password online, but it would generate a lot of noise, and the user would receive a notification for the change as soon as she checks her email. Again vulnerabilities I would fix, but again out of scope now.

Registration

The function in charge of registering new users is procRegister(), defined in file [www/lib/crud/userprocess.php](#).

What's really interesting here is the call at line 96: function register() is called with usual value (user, password, password confirm, email) plus an interesting one: ulevelid.

```
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 +
```

If you open file [classes/userSession.class.php](#) you can see that, after doing some sanity checks for username, password, and email (the regex looks not valid to me, but we don't care about it right now) the function addNewUser() is called with the same argument.

```
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
69
```

OK, it's not a LFI nor a Mysql injection here, it's a **command** one, but it doesn't matter. What matters here is that the dev was aware of the vulnerability and tried to patch (note: this was part of what vikingfr discovered earlier, but I didn't know this because I tried to avoid as much spoil as possible). Because you should **never** trust patches, you should also review commit when you find messages like this.

And this is the case of a incomplete fix: we still have two unescaped variable taken from the GET and passed to the exec call, both \$searchTerm and \$grepNumLineStr are used with no sanitization.

```
08 // search - find all instances of the search term under the specified sub-dir
09 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
10 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
11 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
12 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
13 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
14 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
15 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
16 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
17 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
18 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
19 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
20 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
21 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
22 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
23 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
24 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
25 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
26 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
27 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
28 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
29 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
30 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
31 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
32 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
33 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
34 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
35 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
36 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
37 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
38 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
39 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
40 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
41 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
42 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
43 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
44 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
45 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
46 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
47 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
48 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
49 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
50 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
51 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
52 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
53 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
54 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
55 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
56 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
57 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
58 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
59 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
60 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
61 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
62 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
63 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
64 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
65 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
66 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
67 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
68 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
69 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
70 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
71 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
72 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
73 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
74 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
75 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
76 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
77 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
78 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
79 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
80 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
81 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
82 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
83 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
84 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
85 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
86 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
87 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
88 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
89 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
90 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
91 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
92 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
93 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
94 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
95 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
96 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
97 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
98 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
99 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
100 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
```

This of course leads to an easy RCE: the query can be done only by authorized users, doesn't matter by the level, but we already achieved auth bypass so we now are zero->admin->RCE.

NOTE: this vulnerability has been fixed in 3.9.6, please read [Update](#)

Because of the architecture of the app itself, it will be very easy to escalate to root, but I won't disclose any details here.

Sql Injection

I started this journey knowing I'm looking for a sql injection, but the time I gave myself for this exercise is almost over.

Rushing, I'll grep for SELECT/UPDATE/INSERT with a match on \$ to look for queries done with a variable. Of course it could be escaped before, but it's a good starting point (note: I have my own script that greps with a context, so I'm sure I won't miss multiline queries).

Suddenly four interesting files came out:

- www/compliancepolicies.inc.php
- www/compliancepolicyelements.inc.php
- www/devices.inc.php
- www/snippets.inc.php

Because the vulnerability is almost the same, I will discuss just the first one.

Vulnerable code looks like

```
10 // GET: /api/v1/policies/1
11 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
12 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
13 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
14 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
15 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
16 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
17 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
18 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
19 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
20 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
21 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
22 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
23 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
24 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
25 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
26 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
27 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
28 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
29 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
30 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
31 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
32 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
33 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
34 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
35 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
36 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
37 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
38 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
39 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
40 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
41 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
42 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
43 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
44 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
45 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
46 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
47 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
48 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
49 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
50 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
51 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
52 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
53 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
54 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
55 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
56 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
57 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
58 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
59 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
60 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
61 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
62 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
63 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
64 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
65 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
66 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
67 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
68 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
69 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
70 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
71 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
72 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
73 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
74 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
75 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
76 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
77 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
78 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
79 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
80 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
81 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
82 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
83 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
84 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
85 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
86 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
87 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
88 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
89 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
90 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
91 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
92 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
93 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
94 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
95 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
96 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
97 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
98 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
99 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
100 // $searchTerm is the search term, $grepNumLineStr is the number of lines to grep
```

As you can see, \$searchColumn is not escaped nor used as param in a prepared statement nor the query uses parameters. This will be an easy sql injection.

Looking at the file itself, we know that it's reachable from the webserver. Reading it from the beginning also shows that it can be used without authentication, leading to at least four pre-auth sql injection.

During a real engagement I would have it exploited to download IP/user/password of controlled devices (see rConfig website to see what she does and how a dump of the database could be useful to an attacker).

This could have huge impact on the network, because devices' data are encrypted with an hardcoded key.

I'd suggest the dev to encrypt IP/user/password of managed devices with a unique key, generated during rConfig setup, splitted between filesystem and database like Filippo Valsorda explain in his [blog](#) for hashing. This would make more difficult for an attacker to read both the part of the key and decrypt data.

This vulnerability has been assigned four CVEs, one foreach vulnerable endpoint: CVE-2020-10546 CVE-2020-10547 CVE-2020-10548 CVE-2020-10549.

NOTE: this vulnerability has been fixed in 3.9.7, please read [Update](#)

This attack is not over yet: if you notices that \$db2 handler, it pointed me to also review how that object is built and if it's abusable, and I think this will lead to another blog post about a stacked sql injection.

Conclusion

This could've been a good playground for OSWE preparation, not so complex and with some pathes from zero to root by chaining multiple vulnerabilities.

I've not fully tested two RCE, and did not look for more "public" pages. There is still room for analysis, please ping me if you do some so we can share knowledge.

This journey also reminded me of a very very important thing: **never** trust a patch, always review because it's partial/incomplete or maybe introduced more bugs.

Updates

Stephen Stack, lead dev of rConfig, was kind enough to follow up with a fix for some of the reported vulnerabilities with version 3.9.6 and hoply in 3.9.7 later.