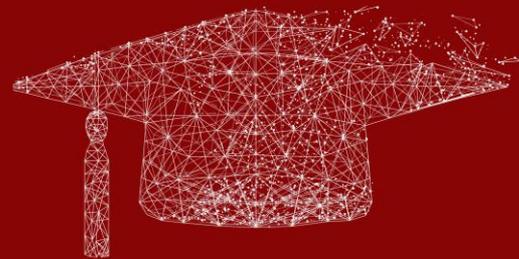




RootMe **PRO**



Webinaire CERTA

« La root est longue mais la voie est Libre »

Sommaire

1. Root-Me (PRO)
2. Offre Réseau CERTA
3. Démonstration
4. Ressources disponibles

1.

Root-Me (PRO)

Root-Me (PRO)



+ de 15 ans
d'existence

+ de 550 000
membres

+ de 700 000
visiteurs/mois

+ de 600
épreuves pour
tous niveaux

Une offre dédiée aux **professionnels** et basée sur la célèbre plateforme portée par l'association Root-Me.



Challenges et
environnements dédiés

Parcours de formation
par niveau et par métier

Ressources
pédagogiques variées

Événements (CTF) sur
mesure

Ce qui est mis à la disposition des Joueurs

Challenges et Environnements virtuels

- ▶ Près de 500 challenges (5 nouveaux/mois)
- ▶ Plus de 150 environnements virtuels (5 nouveaux/mois)
- ▶ 11 catégories de challenges
- ▶ De « Très facile » à « Très difficile »

Pour apprendre pas à pas > les catégories unitaires



Pour se challenger en condition réelle > la catégorie Réaliste



PHP - Unserialize Pop Chain

55 Points 🏆
Pop Pop Pop

Auteur: Warty, 22 octobre 2021 Niveau: [Progress bar] Validations: 253 Challenges 1% Note: ⭐⭐⭐⭐⭐ 41 votes [jaime] [je n'aime pas]

Énoncé
 Pouvez-vous contourner la sécurité mise en place par votre ami pour accéder au flag ?

Configuration de l'environnement
 SRC: Accès au code source ✓

```

1. <?php
2.
3. $getflag = false;
4.
5. class GetMessage {
6.     function __construct($receive) {
7.         if ($receive === "Helloooooooy") {
8.             die("[FRIEND]: Aahah you get fooled by my security my friend-br");
9.         } else {
10.            $this->receive = $receive;
11.        }
12.    }
13.
14.    function __toString() {
15.        return $this->receive;
16.    }
17.
18.    function __destruct() {
19.        global $getflag;
20.        if ($this->receive === "Helloooooooy") {
21.            $getflag = true;
22.        }
23.    }
24.
25.    function __wakeup() {
26.        global $getflag;
27.        if ($this->receive === "Helloooooooy") {
28.            $getflag = true;
29.        }
30.    }
31.
32.    function __sleep() {
33.        return array("receive");
34.    }
35.
36.    function __unset($var) {
37.        return true;
38.    }
39.
40.    function __isset($var) {
41.        return true;
42.    }
43.
44.    function __call($method, $args) {
45.        return true;
46.    }
47.
48.    function __callStatic($method, $args) {
49.        return true;
50.    }
51.
52.    function __clone() {
53.        return true;
54.    }
55.
56.    function __destruct() {
57.        return true;
58.    }
59.
60.    function __wakeup() {
61.        return true;
62.    }
63.
64.    function __isset($var) {
65.        return true;
66.    }
67.
68.    function __call($method, $args) {
69.        return true;
70.    }
71.
72.    function __callStatic($method, $args) {
73.        return true;
74.    }
75.
76.    function __clone() {
77.        return true;
78.    }
79.
80.    function __destruct() {
81.        return true;
82.    }
83.
84.    function __wakeup() {
85.        return true;
86.    }
87.
88.    function __isset($var) {
89.        return true;
90.    }
91.
92.    function __call($method, $args) {
93.        return true;
94.    }
95.
96.    function __callStatic($method, $args) {
97.        return true;
98.    }
99.
100.   function __clone() {
101.       return true;
102.   }
103.
104.   function __destruct() {
105.       return true;
106.   }
107.
108.   function __wakeup() {
109.       return true;
110.   }
111.
112.   function __isset($var) {
113.       return true;
114.   }
115.
116.   function __call($method, $args) {
117.       return true;
118.   }
119.
120.   function __callStatic($method, $args) {
121.       return true;
122.   }
123.
124.   function __clone() {
125.       return true;
126.   }
127.
128.   function __destruct() {
129.       return true;
130.   }
131.
132.   function __wakeup() {
133.       return true;
134.   }
135.
136.   function __isset($var) {
137.       return true;
138.   }
139.
140.   function __call($method, $args) {
141.       return true;
142.   }
143.
144.   function __callStatic($method, $args) {
145.       return true;
146.   }
147.
148.   function __clone() {
149.       return true;
150.   }
151.
152.   function __destruct() {
153.       return true;
154.   }
155.
156.   function __wakeup() {
157.       return true;
158.   }
159.
160.   function __isset($var) {
161.       return true;
162.   }
163.
164.   function __call($method, $args) {
165.       return true;
166.   }
167.
168.   function __callStatic($method, $args) {
169.       return true;
170.   }
171.
172.   function __clone() {
173.       return true;
174.   }
175.
176.   function __destruct() {
177.       return true;
178.   }
179.
180.   function __wakeup() {
181.       return true;
182.   }
183.
184.   function __isset($var) {
185.       return true;
186.   }
187.
188.   function __call($method, $args) {
189.       return true;
190.   }
191.
192.   function __callStatic($method, $args) {
193.       return true;
194.   }
195.
196.   function __clone() {
197.       return true;
198.   }
199.
200.   function __destruct() {
201.       return true;
202.   }
203.
204.   function __wakeup() {
205.       return true;
206.   }
207.
208.   function __isset($var) {
209.       return true;
210.   }
211.
212.   function __call($method, $args) {
213.       return true;
214.   }
215.
216.   function __callStatic($method, $args) {
217.       return true;
218.   }
219.
220.   function __clone() {
221.       return true;
222.   }
223.
224.   function __destruct() {
225.       return true;
226.   }
227.
228.   function __wakeup() {
229.       return true;
230.   }
231.
232.   function __isset($var) {
233.       return true;
234.   }
235.
236.   function __call($method, $args) {
237.       return true;
238.   }
239.
240.   function __callStatic($method, $args) {
241.       return true;
242.   }
243.
244.   function __clone() {
245.       return true;
246.   }
247.
248.   function __destruct() {
249.       return true;
250.   }
251.
252.   function __wakeup() {
253.       return true;
254.   }
255.
256.   function __isset($var) {
257.       return true;
258.   }
259.
260.   function __call($method, $args) {
261.       return true;
262.   }
263.
264.   function __callStatic($method, $args) {
265.       return true;
266.   }
267.
268.   function __clone() {
269.       return true;
270.   }
271.
272.   function __destruct() {
273.       return true;
274.   }
275.
276.   function __wakeup() {
277.       return true;
278.   }
279.
280.   function __isset($var) {
281.       return true;
282.   }
283.
284.   function __call($method, $args) {
285.       return true;
286.   }
287.
288.   function __callStatic($method, $args) {
289.       return true;
290.   }
291.
292.   function __clone() {
293.       return true;
294.   }
295.
296.   function __destruct() {
297.       return true;
298.   }
299.
300.   function __wakeup() {
301.       return true;
302.   }
303.
304.   function __isset($var) {
305.       return true;
306.   }
307.
308.   function __call($method, $args) {
309.       return true;
310.   }
311.
312.   function __callStatic($method, $args) {
313.       return true;
314.   }
315.
316.   function __clone() {
317.       return true;
318.   }
319.
320.   function __destruct() {
321.       return true;
322.   }
323.
324.   function __wakeup() {
325.       return true;
326.   }
327.
328.   function __isset($var) {
329.       return true;
330.   }
331.
332.   function __call($method, $args) {
333.       return true;
334.   }
335.
336.   function __callStatic($method, $args) {
337.       return true;
338.   }
339.
340.   function __clone() {
341.       return true;
342.   }
343.
344.   function __destruct() {
345.       return true;
346.   }
347.
348.   function __wakeup() {
349.       return true;
350.   }
351.
352.   function __isset($var) {
353.       return true;
354.   }
355.
356.   function __call($method, $args) {
357.       return true;
358.   }
359.
360.   function __callStatic($method, $args) {
361.       return true;
362.   }
363.
364.   function __clone() {
365.       return true;
366.   }
367.
368.   function __destruct() {
369.       return true;
370.   }
371.
372.   function __wakeup() {
373.       return true;
374.   }
375.
376.   function __isset($var) {
377.       return true;
378.   }
379.
380.   function __call($method, $args) {
381.       return true;
382.   }
383.
384.   function __callStatic($method, $args) {
385.       return true;
386.   }
387.
388.   function __clone() {
389.       return true;
390.   }
391.
392.   function __destruct() {
393.       return true;
394.   }
395.
396.   function __wakeup() {
397.       return true;
398.   }
399.
400.   function __isset($var) {
401.       return true;
402.   }
403.
404.   function __call($method, $args) {
405.       return true;
406.   }
407.
408.   function __callStatic($method, $args) {
409.       return true;
410.   }
411.
412.   function __clone() {
413.       return true;
414.   }
415.
416.   function __destruct() {
417.       return true;
418.   }
419.
420.   function __wakeup() {
421.       return true;
422.   }
423.
424.   function __isset($var) {
425.       return true;
426.   }
427.
428.   function __call($method, $args) {
429.       return true;
430.   }
431.
432.   function __callStatic($method, $args) {
433.       return true;
434.   }
435.
436.   function __clone() {
437.       return true;
438.   }
439.
440.   function __destruct() {
441.       return true;
442.   }
443.
444.   function __wakeup() {
445.       return true;
446.   }
447.
448.   function __isset($var) {
449.       return true;
450.   }
451.
452.   function __call($method, $args) {
453.       return true;
454.   }
455.
456.   function __callStatic($method, $args) {
457.       return true;
458.   }
459.
460.   function __clone() {
461.       return true;
462.   }
463.
464.   function __destruct() {
465.       return true;
466.   }
467.
468.   function __wakeup() {
469.       return true;
470.   }
471.
472.   function __isset($var) {
473.       return true;
474.   }
475.
476.   function __call($method, $args) {
477.       return true;
478.   }
479.
480.   function __callStatic($method, $args) {
481.       return true;
482.   }
483.
484.   function __clone() {
485.       return true;
486.   }
487.
488.   function __destruct() {
489.       return true;
490.   }
491.
492.   function __wakeup() {
493.       return true;
494.   }
495.
496.   function __isset($var) {
497.       return true;
498.   }
499.
500.   function __call($method, $args) {
501.       return true;
502.   }
503.
504.   function __callStatic($method, $args) {
505.       return true;
506.   }
507.
508.   function __clone() {
509.       return true;
510.   }
511.
512.   function __destruct() {
513.       return true;
514.   }
515.
516.   function __wakeup() {
517.       return true;
518.   }
519.
520.   function __isset($var) {
521.       return true;
522.   }
523.
524.   function __call($method, $args) {
525.       return true;
526.   }
527.
528.   function __callStatic($method, $args) {
529.       return true;
530.   }
531.
532.   function __clone() {
533.       return true;
534.   }
535.
536.   function __destruct() {
537.       return true;
538.   }
539.
540.   function __wakeup() {
541.       return true;
542.   }
543.
544.   function __isset($var) {
545.       return true;
546.   }
547.
548.   function __call($method, $args) {
549.       return true;
550.   }
551.
552.   function __callStatic($method, $args) {
553.       return true;
554.   }
555.
556.   function __clone() {
557.       return true;
558.   }
559.
560.   function __destruct() {
561.       return true;
562.   }
563.
564.   function __wakeup() {
565.       return true;
566.   }
567.
568.   function __isset($var) {
569.       return true;
569.   }
570.
571.   function __call($method, $args) {
572.       return true;
573.   }
574.
575.   function __callStatic($method, $args) {
576.       return true;
577.   }
578.
579.   function __clone() {
580.       return true;
580.   }
581.
582.   function __destruct() {
583.       return true;
583.   }
584.
585.   function __wakeup() {
586.       return true;
586.   }
587.
588.   function __isset($var) {
589.       return true;
589.   }
590.
591.   function __call($method, $args) {
592.       return true;
592.   }
593.
594.   function __callStatic($method, $args) {
595.       return true;
595.   }
596.
597.   function __clone() {
598.       return true;
598.   }
599.
600.   function __destruct() {
601.       return true;
601.   }
602.
603.   function __wakeup() {
604.       return true;
604.   }
605.
606.   function __isset($var) {
607.       return true;
607.   }
608.
609.   function __call($method, $args) {
609.       return true;
609.   }
610.
611.   function __callStatic($method, $args) {
612.       return true;
612.   }
613.
614.   function __clone() {
615.       return true;
615.   }
616.
617.   function __destruct() {
617.       return true;
617.   }
618.
619.   function __wakeup() {
619.       return true;
619.   }
620.
621.   function __isset($var) {
621.       return true;
621.   }
622.
623.   function __call($method, $args) {
623.       return true;
623.   }
624.
625.   function __callStatic($method, $args) {
625.       return true;
625.   }
626.
627.   function __clone() {
627.       return true;
627.   }
628.
629.   function __destruct() {
629.       return true;
629.   }
630.
631.   function __wakeup() {
631.       return true;
631.   }
632.
633.   function __isset($var) {
633.       return true;
633.   }
634.
635.   function __call($method, $args) {
635.       return true;
635.   }
636.
637.   function __callStatic($method, $args) {
637.       return true;
637.   }
638.
639.   function __clone() {
639.       return true;
639.   }
640.
641.   function __destruct() {
641.       return true;
641.   }
642.
643.   function __wakeup() {
643.       return true;
643.   }
644.
645.   function __isset($var) {
645.       return true;
645.   }
646.
647.   function __call($method, $args) {
647.       return true;
647.   }
648.
649.   function __callStatic($method, $args) {
649.       return true;
649.   }
650.
651.   function __clone() {
651.       return true;
651.   }
652.
653.   function __destruct() {
653.       return true;
653.   }
654.
655.   function __wakeup() {
655.       return true;
655.   }
656.
657.   function __isset($var) {
657.       return true;
657.   }
658.
659.   function __call($method, $args) {
659.       return true;
659.   }
660.
661.   function __callStatic($method, $args) {
661.       return true;
661.   }
662.
663.   function __clone() {
663.       return true;
663.   }
664.
665.   function __destruct() {
665.       return true;
665.   }
666.
667.   function __wakeup() {
667.       return true;
667.   }
668.
669.   function __isset($var) {
669.       return true;
669.   }
670.
671.   function __call($method, $args) {
671.       return true;
671.   }
672.
673.   function __callStatic($method, $args) {
673.       return true;
673.   }
674.
675.   function __clone() {
675.       return true;
675.   }
676.
677.   function __destruct() {
677.       return true;
677.   }
678.
679.   function __wakeup() {
679.       return true;
679.   }
680.
681.   function __isset($var) {
681.       return true;
681.   }
682.
683.   function __call($method, $args) {
683.       return true;
683.   }
684.
685.   function __callStatic($method, $args) {
685.       return true;
685.   }
686.
687.   function __clone() {
687.       return true;
687.   }
688.
689.   function __destruct() {
689.       return true;
689.   }
690.
691.   function __wakeup() {
691.       return true;
691.   }
692.
693.   function __isset($var) {
693.       return true;
693.   }
694.
695.   function __call($method, $args) {
695.       return true;
695.   }
696.
697.   function __callStatic($method, $args) {
697.       return true;
697.   }
698.
699.   function __clone() {
699.       return true;
699.   }
700.
701.   function __destruct() {
701.       return true;
701.   }
702.
703.   function __wakeup() {
703.       return true;
703.   }
704.
705.   function __isset($var) {
705.       return true;
705.   }
706.
707.   function __call($method, $args) {
707.       return true;
707.   }
708.
709.   function __callStatic($method, $args) {
709.       return true;
709.   }
710.
711.   function __clone() {
711.       return true;
711.   }
712.
713.   function __destruct() {
713.       return true;
713.   }
714.
715.   function __wakeup() {
715.       return true;
715.   }
716.
717.   function __isset($var) {
717.       return true;
717.   }
718.
719.   function __call($method, $args) {
719.       return true;
719.   }
720.
721.   function __callStatic($method, $args) {
721.       return true;
721.   }
722.
723.   function __clone() {
723.       return true;
723.   }
724.
725.   function __destruct() {
725.       return true;
725.   }
726.
727.   function __wakeup() {
727.       return true;
727.   }
728.
729.   function __isset($var) {
729.       return true;
729.   }
730.
731.   function __call($method, $args) {
731.       return true;
731.   }
732.
733.   function __callStatic($method, $args) {
733.       return true;
733.   }
734.
735.   function __clone() {
735.       return true;
735.   }
736.
737.   function __destruct() {
737.       return true;
737.   }
738.
739.   function __wakeup() {
739.       return true;
739.   }
740.
741.   function __isset($var) {
741.       return true;
741.   }
742.
743.   function __call($method, $args) {
743.       return true;
743.   }
744.
745.   function __callStatic($method, $args) {
745.       return true;
745.   }
746.
747.   function __clone() {
747.       return true;
747.   }
748.
749.   function __destruct() {
749.       return true;
749.   }
750.
751.   function __wakeup() {
751.       return true;
751.   }
752.
753.   function __isset($var) {
753.       return true;
753.   }
754.
755.   function __call($method, $args) {
755.       return true;
755.   }
756.
757.   function __callStatic($method, $args) {
757.       return true;
757.   }
758.
759.   function __clone() {
759.       return true;
759.   }
760.
761.   function __destruct() {
761.       return true;
761.   }
762.
763.   function __wakeup() {
763.       return true;
763.   }
764.
765.   function __isset($var) {
765.       return true;
765.   }
766.
767.   function __call($method, $args) {
767.       return true;
767.   }
768.
769.   function __callStatic($method, $args) {
769.       return true;
769.   }
770.
771.   function __clone() {
771.       return true;
771.   }
772.
773.   function __destruct() {
773.       return true;
773.   }
774.
775.   function __wakeup() {
775.       return true;
775.   }
776.
777.   function __isset($var) {
777.       return true;
777.   }
778.
779.   function __call($method, $args) {
779.       return true;
779.   }
780.
781.   function __callStatic($method, $args) {
781.       return true;
781.   }
782.
783.   function __clone() {
783.       return true;
783.   }
784.
785.   function __destruct() {
785.       return true;
785.   }
786.
787.   function __wakeup() {
787.       return true;
787.   }
788.
789.   function __isset($var) {
789.       return true;
789.   }
790.
791.   function __call($method, $args) {
791.       return true;
791.   }
792.
793.   function __callStatic($method, $args) {
793.       return true;
793.   }
794.
795.   function __clone() {
795.       return true;
795.   }
796.
797.   function __destruct() {
797.       return true;
797.   }
798.
799.   function __wakeup() {
799.       return true;
799.   }
800.
801.   function __isset($var) {
801.       return true;
801.   }
802.
803.   function __call($method, $args) {
803.       return true;
803.   }
804.
805.   function __callStatic($method, $args) {
805.       return true;
805.   }
806.
807.   function __clone() {
807.       return true;
807.   }
808.
809.   function __destruct() {
809.       return true;
809.   }
810.
811.   function __wakeup() {
811.       return true;
811.   }
812.
813.   function __isset($var) {
813.       return true;
813.   }
814.
815.   function __call($method, $args) {
815.       return true;
815.   }
816.
817.   function __callStatic($method, $args) {
817.       return true;
817.   }
818.
819.   function __clone() {
819.       return true;
819.   }
820.
821.   function __destruct() {
821.       return true;
821.   }
822.
823.   function __wakeup() {
823.       return true;
823.   }
824.
825.   function __isset($var) {
825.       return true;
825.   }
826.
827.   function __call($method, $args) {
827.       return true;
827.   }
828.
829.   function __callStatic($method, $args) {
829.       return true;
829.   }
830.
831.   function __clone() {
831.       return true;
831.   }
832.
833.   function __destruct() {
833.       return true;
833.   }
834.
835.   function __wakeup() {
835.       return true;
835.   }
836.
837.   function __isset($var) {
837.       return true;
837.   }
838.
839.   function __call($method, $args) {
839.       return true;
839.   }
840.
841.   function __callStatic($method, $args) {
841.       return true;
841.   }
842.
843.   function __clone() {
843.       return true;
843.   }
844.
845.   function __destruct() {
845.       return true;
845.   }
846.
847.   function __wakeup() {
847.       return true;
847.   }
848.
849.   function __isset($var) {
849.       return true;
849.   }
850.
851.   function __call($method, $args) {
851.       return true;
851.   }
852.
853.   function __callStatic($method, $args) {
853.       return true;
853.   }
854.
855.   function __clone() {
855.       return true;
855.   }
856.
857.   function __destruct() {
857.       return true;
857.   }
858.
859.   function __wakeup() {
859.       return true;
859.   }
860.
861.   function __isset($var) {
861.       return true;
861.   }
862.
863.   function __call($method, $args) {
863.       return true;
863.   }
864.
865.   function __callStatic($method, $args) {
865.       return true;
865.   }
866.
867.   function __clone() {
867.       return true;
867.   }
868.
869.   function __destruct() {
869.       return true;
869.   }
870.
871.   function __wakeup() {
871.       return true;
871.   }
872.
873.   function __isset($var) {
873.       return true;
873.   }
874.
875.   function __call($method, $args) {
875.       return true;
875.   }
876.
877.   function __callStatic($method, $args) {
877.       return true;
877.   }
878.
879.   function __clone() {
879.       return true;
879.   }
880.
881.   function __destruct() {
881.       return true;
881.   }
882.
883.   function __wakeup() {
883.       return true;
883.   }
884.
885.   function __isset($var) {
885.       return true;
885.   }
886.
887.   function __call($method, $args) {
887.       return true;
887.   }
888.
889.   function __callStatic($method, $args) {
889.       return true;
889.   }
890.
891.   function __clone() {
891.       return true;
891.   }
892.
893.   function __destruct() {
893.       return true;
893.   }
894.
895.   function __wakeup() {
895.       return true;
895.   }
896.
897.   function __isset($var) {
897.       return true;
897.   }
898.
899.   function __call($method, $args) {
899.       return true;
899.   }
900.
901.   function __callStatic($method, $args) {
901.       return true;
901.   }
902.
903.   function __clone() {
903.       return true;
903.   }
904.
905.   function __destruct() {
905.       return true;
905.   }
906.
907.   function __wakeup() {
907.       return true;
907.   }
908.
909.   function __isset($var) {
909.       return true;
909.   }
910.
911.   function __call($method, $args) {
911.       return true;
911.   }
912.
913.   function __callStatic($method, $args) {
913.       return true;
913.   }
914.
915.   function __clone() {
915.       return true;
915.   }
916.
917.   function __destruct() {
917.       return true;
917.   }
918.
919.   function __wakeup() {
919.       return true;
919.   }
920.
921.   function __isset($var) {
921.       return true;
921.   }
922.
923.   function __call($method, $args) {
923.       return true;
923.   }
924.
925.   function __callStatic($method, $args) {
925.       return true;
925.   }
926.
927.   function __clone() {
927.       return true;
927.   }
928.
929.   function __destruct() {
929.       return true;
929.   }
930.
931.   function __wakeup() {
931.       return true;
931.   }
932.
933.   function __isset($var) {
933.       return true;
933.   }
934.
935.   function __call($method, $args) {
935.       return true;
935.   }
936.
937.   function __callStatic($method, $args) {
937.       return true;
937.   }
938.
939.   function __clone() {
939.       return true;
939.   }
940.
941.   function __destruct() {
941.       return true;
941.   }
942.
943.   function __wakeup() {
943.       return true;
943.   }
944.
945.   function __isset($var) {
945.       return true;
945.   }
946.
947.   function __call($method, $args) {
947.       return true;
947.   }
948.
949.   function __callStatic($method, $args) {
949.       return true;
949.   }
950.
951.   function __clone() {
951.       return true;
951.   }
952.
953.   function __destruct() {
953.       return true;
953.   }
954.
955.   function __wakeup() {
955.       return true;
955.   }
956.
957.   function __isset($var) {
957.       return true;
957.   }
958.
959.   function __call($method, $args) {
959.       return true;
959.   }
960.
961.   function __callStatic($method, $args) {
961.       return true;
961.   }
962.
963.   function __clone() {
963.       return true;
963.   }
964.
965.   function __destruct() {
965.       return true;
965.   }
966.
967.   function __wakeup() {
967.       return true;
967.   }
968.
969.   function __isset($var) {
969.       return true;
969.   }
970.
971.   function __call($method, $args) {
971.       return true;
971.   }
972.
973.   function __callStatic($method, $args) {
973.       return true;
973.   }
974.
975.   function __clone() {
975.       return true;
975.   }
976.
977.   function __destruct() {
977.       return true;
977.   }
978.
979.   function __wakeup() {
979.       return true;
979.   }
980.
981.   function __isset($var) {
981.       return true;
981.   }
982.
983.   function __call($method, $args
```

Ce qui est mis à la disposition des Formateurs

En amont des formations

Faciliter la préparation

1 à 3 indices par challenge

Plusieurs fiches pratiques par parcours

+ de 5 000 solutions

Durant les formations

Faciliter les TP

Sélections personnalisées et
Parcours de formation exclusifs

Démarrage des challenges et
environnements à la demande

Suivi en temps réel de la
progression des joueurs

En aval des formations

Faciliter l'évaluation

Tableaux de bord et rapports

Conservation des statistiques et
des challenges réalisés

Attestations de réussite

Des avantages pour chaque acteur

Étudiant



- Utiliser un outil connu et reconnu par les professionnels
- S'entraîner et tester ses compétences sur des environnements réalistes
- Mettre en valeur ses compétences

Formateur



- Simplifier la mise en œuvre pratique des concepts théoriques
- Simplifier la préparation des formations
- Outiller l'évaluation des compétences en cybersécurité

Responsable de formation



- Simplifier la gestion des cours, des étudiants et des formateurs
- Suivre la progression pédagogique des joueurs
- Tester les nouveaux candidats et identifier de nouveaux talents

Directeur de centre



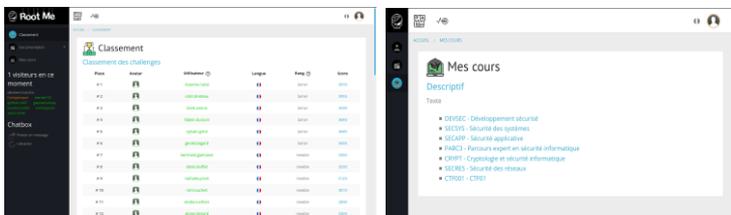
- Améliorer la qualité de ses formations et apporter une plus-value technique
- Optimiser ses coûts de formation
- Attirer de nouveaux profils et faire connaître ses formations

2.

Offre Réseau CERTA

L'offre « Train my Team » pour les établissements

1 environnement de jeu



- Pour les Etudiants et les Formateurs
- Classement par établissement / par promotion
- **NOUVEAU** Accès web à une machine Kali
- **NOUVEAU** 2 sélections de challenges personnalisées :

Niveau 1

Niveau 2

▶ Tronc commun
SLAM/SISR

▶ Sélection SLAM ou
SISR



1 environnement de pilotage



- Pour les Formateurs et Responsables des formations
- Supervision de la progression des promotions et des étudiants
- Démarrage en quelques secondes des sélections de challenges prédéfinies
- **NOUVEAU** Accès aux ressources pédagogiques associées à chaque sélection de challenges

Fiches pratiques

Indices

Solutions

▶ Synthèse illustrée
d'un concept
▶ Pointeur vers
ressources externes

▶ 1 à 3 indices /
challenge
▶ Orienter les étudiants
durant les cours

▶ 1 à plusieurs
solutions / challenge
▶ Pointeur vers
ressources externes

Les attestations de réussite Root-Me PRO



Attestations pour les parcours exclusifs aux BTS SIO

Pour la filière BTS SIO, les équipes de Root-Me PRO ont conçu 3 parcours de formation. Chaque parcours comprend plusieurs challenges visant à mettre en application les compétences acquises tout au long du cursus de formation.

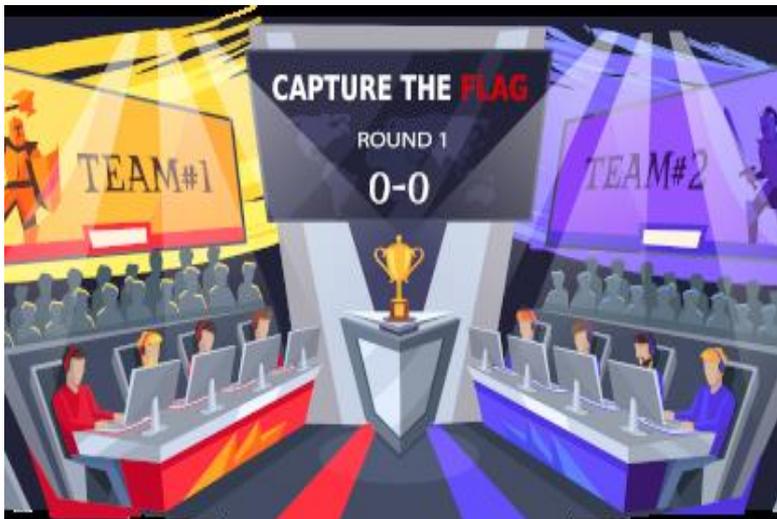
Les parcours proposés sont les suivants :

- Niveau 1 – Tronc commun SLAM/SISR
- Niveau 2 – Sélection SLAM ou SISR

Les parcours « Niveau 2 » seront sélectionnés en fonction de l'option choisie par les étudiants.

Des attestations de compétences seront délivrées par Root-Me PRO aux étudiants ayant réalisés les parcours de formation qui leur sont proposés.

CTF inter-établissement



Variables définies dans le cadre du partenariat

Durée de l'événement
1 journée

Type de contenu
Public, Privé

Bonus
Récompenses

1 équipe de 5 joueurs par
établissement

Une compétition nationale

2 1 3

Pour les étudiants – Un environnement de jeu dédié

Le CTF est accessible **uniquement** aux étudiants des promotions BTS 2^{ème} année. Dans le cadre de l'événement, chaque équipe dispose d'un accès à une sélection sur mesure de challenges parmi les nombreuses catégories disponibles : app, web, réseau, stégano, crypto, etc.



Pour les formateurs – Une interface de suivi

Chaque responsable dispose d'un accès privilégié sur notre plateforme lui permettant de suivre la progression de son équipe lors de l'évènement.

8 Novembre 2023

3.

A vous de jouer !

Dans la peau d'un joueur



**KEEP
CALM**

AND

LETS GO

BACK TO SCHOOL

KeepCalmAndPosters.com

Déroulement

Durée : 30-45 minutes

Niveau : débutant

Pour participer : 1 PC + 1 connexion Internet + 1 compte sur Root-Me

www.root-me.org

1

Présentation de la sélection de challenges

2

Zoom sur quelques challenges

3

Zoom sur les ressources associées

4.

Ressources disponibles

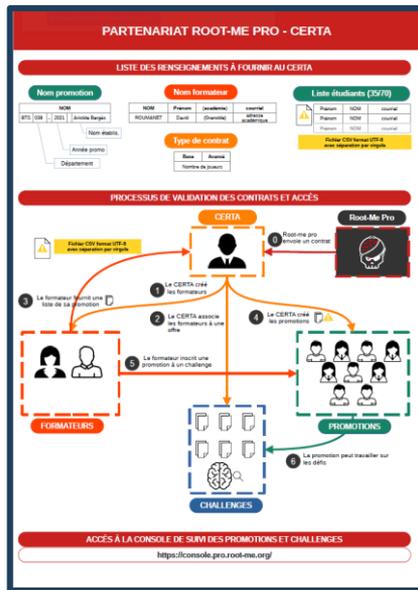
Documents de référence

Présentation offre CERTA

Partenariat Réseau CERTA
 Root-Me PRO, une référence en matière d'apprentissage de la cybersécurité

Novembre 2022 (Version 1)

Processus d'inscription



Contrat d'abonnement

BON DE COMMANDE N° RMP-CERTA-2023

1. Description des produits
 Pour le présent bon de commande, le Client commande la prestation suivante:
 Offre CERTA - RootMe PRO La 3ème édition

2. Modalités de paiement
 Le présent bon de commande est payable au capital de 4 000 euros, imputable au Prorata de l'abonnement de 12 mois de la somme de 333,33€ TTC, soit le fixe mensuel de 27,78€ TTC, payable de la date de la commande.

3. Modalités de livraison
 Le présent bon de commande est payable au capital de 4 000 euros, imputable au Prorata de l'abonnement de 12 mois de la somme de 333,33€ TTC, soit le fixe mensuel de 27,78€ TTC, payable de la date de la commande.

4. Modalités de résiliation
 Le présent bon de commande est payable au capital de 4 000 euros, imputable au Prorata de l'abonnement de 12 mois de la somme de 333,33€ TTC, soit le fixe mensuel de 27,78€ TTC, payable de la date de la commande.

Accord de confidentialité

ACCORD DE CONFIDENTIALITE

Le présent accord de confidentialité est conclu entre :

ROOT-ME PRO, SAS, au capital de 4 000 euros, dont le siège social est situé à 29 bis chemin de Graves, 69450 Saint-Cyr-au-Mont-d'Or, immatriculée au registre du commerce et des sociétés de LYON sous le numéro 870 403 906 enregistrée par son Président, Monsieur Pierre LACROIX, ci-après habilitée à cet effet.

Ci-après la "Partie Divulgeuse"

Et

_____ dont le siège social est situé à _____ immatriculée au registre du commerce et des sociétés de _____ sous le numéro _____ ci-après représentée par _____ ci-après habilitée à cet effet.

Ci-après la "Partie Bénéficiaire"

Dénommées ensemble ci-après les "Parties".

Préambule

Attendu que la Partie Bénéficiaire souhaite, dans le cadre de son abonnement à l'offre « CERTA » de Root-Me PRO, accéder à des informations (publiques et privées) en lien avec les contenus proposés sur les environnements mis à sa disposition (des challenges) et qui ne sont pas définis disponibles sur les utilisateurs ayant établi ces contenus.

Les Parties proposent par les présentes dispositions que la transmission de telles informations soit faite dans les conditions définies et reprises ("l'accord").

Webinaire 11/22

Webinaire CERTA

« La root est longue mais la voie est libre »

Novembre 2022

contact@pro.root-me.org