Logotipo

Descrição gerada automaticamente com confiança baixa

Relatório de CTF

Very Smooth - picoCTF

|  |  |
| --- | --- |
| **Informações do documento** | |
| **Referência** | Very Smooth – Guilherme Gonsales de Sá |
| **N° Revisão** | 1 |
| **Data de publicação** | 29/09/2025 |
| **Link** | https://play.picoctf.org/practice/challenge/315?page=1&search=very%20smo |

|  |  |  |
| --- | --- | --- |
| **Redação** | Guilherme Gonsales de Sá | Estudante |
| **Revisão** |  |  |
| **Aprovação** |  |  |

|  |  |  |
| --- | --- | --- |
| **Histórico de revisões** | | |
| **N°** | **Entregas** | **Descrição** |
| **0** | 29/09/2025 | Entrega |
| **1** |  | Revisão |
| **2** |  | Aprovação |

|  |  |
| --- | --- |
| **Informações do CTF** | |
| **Nível de Dificuldade** | Difícil |
| **Tipo de acesso** | Gratuito |
| **Conceitos envolvidos** | Criptografia, algoritmos de fatoração, RSA, aritmética modular. |
| **Plataforma** | picoCTF |
| **Área** | Red |

**Sumário**

[Contextualização 3](#_heading=h.gjdgxs)

[Desenvolvimento 3](#_heading=h.1fob9te)

Flag3

[Conclusão](#_heading=h.1t3h5sf) 5

[Referências](#_heading=h.4d34og8) 5

Scripts 6

**Contextualização**

O CTF ‘Very Smooth’ é um ctf de criptografia pura que foca em um método não tão convencional de decriptação de cifras. Nele, somos levados a estudar tópicos matemáticos mais complexos quando comparados a uma cifra RSA convencional, testando com profundidade o entendimento de aritmética modular e cifras assimétricas.

**Desenvolvimento**

**F****lag**

Ao abrir o ctf, observamos dois arquivos: ‘gen.py’, que é a função que criptografa nossa mensagem, e ‘output.txt’, onde podemos encontrar ‘n’ e ‘c’.



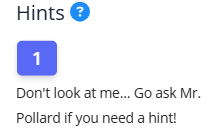
A priori, suspeita-se que estamos lidando com uma criptografia RSA (ignorando que está no tema do CTF). Tal suspeita é reafirmada na análise do código, onde encontramos um valo de ‘e’:



Sabendo que estamos trabalhando com números hexadecimais (confirmado pelos imports do algoritmo de cifragem), ao converter o número de hexadecimal para decimal, descobrimos que o expoente público ‘e’ é 65537, número usual em CTFs para o expoente.

Ao pesquisar sobre o módulo ‘n’ fornecido, sabe-se que o mesmo já foi fatorado, o que indica que, provavelmente, o CTF pode ser resolvido de forma ‘padrão’: Sabendo que a mensagem original começa com ‘picoCTF{...}’, poderíamos realizar um ataque de força bruta no expoente privado. No entanto, não parece ser o caminho proposto para solução do exercício.

Ao observar a dica, tem-se:



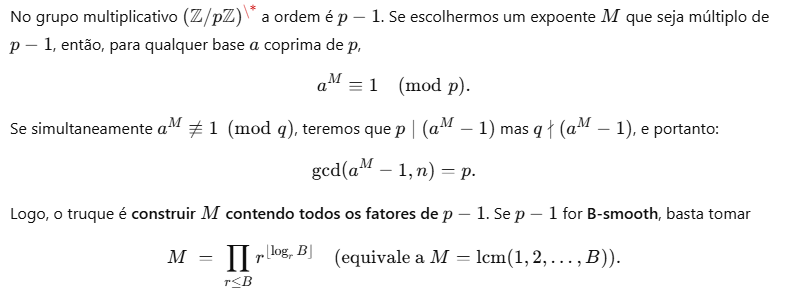
Com uma pesquisa rápida, encontra-se o nome ‘John Pollard’, autor do algoritmo denominado ‘Pollard p-1’.

Devemos entrar brevemente na definição de um número ‘smooth’. Um número chamado *‘n-smooth’* é um número tal que todos os seus fatores primos sejam menores ou iguais a *n*. Como exemplos, sabe-se que:

E

Ambos os números podem ser denominados *7-smooth*, pois seu maior fator primo é 7 e todos os outros fatores são menores que *n.* Dessa forma, o número 11 não pode ser considerado *7-smooth,* pois seu maior fator primo é 11 (o mesmo poderia ser considerado um número *11-smooth*, por exemplo).

Pode-se entender o fluxo do algoritmo como:



Muitas das derivações são feitas a partir do *pequeno teorema de Fermat (Fermat´s little theorem)*, que dita que um número inteiro qualquer *a* elevado a um expoente inteiro *m*, onde *m*  é múltiplo de *p-1* será congruente com 1 *modulo p*.

Definamos um valor arbitrário *B* (no caso do CTF, utilizemos *2^17)*, depois escolhamos uma base *a* coprima de n (na prática, utiliza-se um número primo baixo, como 2,3,5,7). O algoritmo deverá fazer o cálculo de para todos os primos no intervalo de *B*, para todo primo .

Em seguida, deve-se calcular o maior divisor comum (gcd) de a-1 e n. Deve-se fazer a seguinte análise:





Com *p* e *q* em mãos, calculemos a função de *Carmichael* (λ(n)), o inverso modular de e (que nos retorna nosso expoente privado) e por fim, decriptamos a mensagem. De forma mais visual, tem-se:







Ao fazer um script que cumpra essas condições, obtemos a flag:



Concluindo o desafio.

**Conclusão**

Como visto, o CTF requer uma solução não trivial de uma cifra RSA, demonstrando novamente a complexidade da mesma. Somos forçados a reconhecer algumas fraquezas da cifra e relembrados da importância de escolher primos ‘fortes’ e que, mesmo com toda a complexidade matemática e computacional, o RSA ainda possui vulnerabilidades críticas que podem ser exploradas.

**Referências**

<https://chatgpt.com/>

<https://en.wikipedia.org/wiki/Pollard%27s_p_%E2%88%92_1_algorithm>

**Scripts**

Pollard p-1 algorithm



