nullhat 2025: real_secure_algo

@elesquina

The script generates an RSA modulus $N = pq$ and encrypts the flag as $c \equiv m^e \pmod{N}$. The exponent is not random; it is computed by sorting a small list and then combining elements as $e = \texttt{state}[1] + \texttt{state}[2] + \texttt{state}[0]\texttt{state}[3]\texttt{state}[4]$, which evaluates to 65537. The output file provides N and c in binary, and it also leaks the first 586 bits of p as a prefix.

Write $p = p_0 \cdot 2^k + x$, where p_0 is the known high-bit prefix, k is the number of unknown low bits, and x is an unknown integer with $0 \leq x < 2^k$. Since $N = pq$, the correct p is a root of the polynomial $f(X) = p_0 \cdot 2^k + X$ in the sense that $f(x) \mid N$. This is a standard "partial prime bits" RSA break: with enough high bits of one factor, lattice methods (Coppersmith-style small root techniques) recover x, then p follows, and finally $q = N/p$.

After factoring, we compute $\varphi(N) = (p-1)(q-1)$ and the private exponent $d \equiv e^{-1} \pmod{\varphi(N)}$. Decryption is $m \equiv c^d \pmod{N}$, and converting m back to bytes reveals the flag: `ELITESEC{r34l1111k_s3cur1s_4lg0r1thm0s_n0t_s3cur3_4ft3r_4ll}`.