The Flaw

The Diffie-Hellman prime (*p*) generated is a "DSA prime". These primes are of form $p = Rq + 1$. Here, *q* should be prime, but in this implementation *q* (in our scenario "*phi*") is not prime but has "small" factors relative to the size of the other parameters. This means we can force the shared secret to be in a small subgroup of phi & it is computationally feasible to find the shared secret with an algorithm. I used the Pollard Rho Algorithm (original source code: https://github.com/ashutosh1206/Crypton/blob/master/Discrete-Logarithm-Problem/Algo-Pollard-Rho/pollardrho.py) to compute/guess the shared secret.

A great resource I found, upon which I built most of the "Small Subgroup Confinement Attack"-Implementation, is https://people.scs.carleton.ca/~paulv/toolsjewels/ch4-long.pdf (especially Subsection 4.8)

The Mitigation

The Attack can be mitigated by using safe primes ($p = 2q + 1$) where q **is** prime, as the checks for 1 (mod p), -1 (mod p) & p – 1 (mod p) have already been implemented.

The Solution:

1. I wrote a python script which automates the Small Subgroup Confinement Attack & generates the shared key, as well as iterating through all possible UUID characters to get the flag.
   Run the script & enter the FQDN prefix to get the flag.
   An error might occur if the script fails to find a factor for phi before reaching the (hard-coded) limit of 2 ^ 26. If this error occurs, run the script again. Most of the times phi should have factor(s) smaller than 2 ^ 26.