Can COViD steal Bob's idea?

My Little Trojan

December 9, 2020

Cryptography challenge from **STACK the Flags 2020**.

1 The Challenge

Bob wants Alice to help him design the stream cipher's keystream generator base on his rough idea. Can COViD steal Bob's "protected" idea?

File provided: 1x PCAPNG file that stores a dump of packets captured over a network

Expected Flag: Numeric String

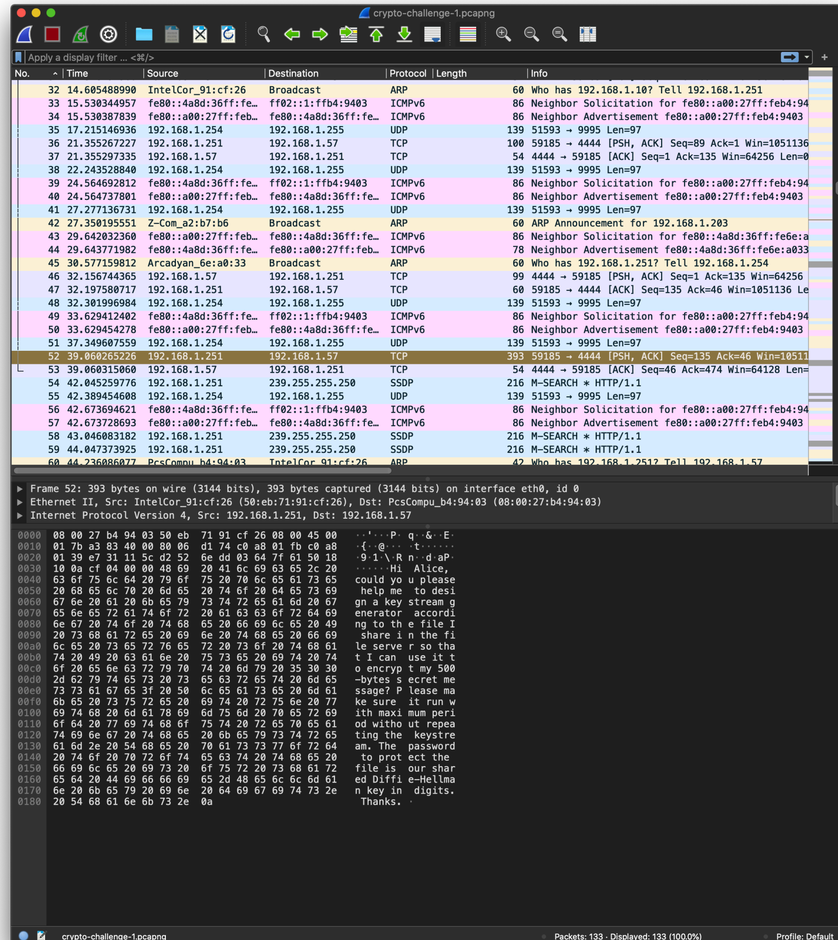
2 Tools Used

1. Wireshark: Good and free pcapng viewer.
2. Discrete Logarithm Calculator: `https://www.alpertron.com.ar/DILOG.HTM`
3. Python Console: For doing math on large numbers

3 The Hack

3.1 The Wireshark Interface

After launching Wireshark, you will get a list view of all the packets. After scrolling about, you should notice a message in packet 52.



The message in the rightmost column reads:
`Hi Alice, could you please help me to design a keystream generator according to the file I share in the file server so that I can use it to encrypt my 500-bytes secret message? Please make sure it run with maximum period without repeating the keystream. The password to protect the file is our shared Diffie-Hellman key in digits. Thanks.`

3.2 Diffie-Hellman key exchange

This is a cryptographic method that involves large prime numbers and modular arithmetic that was published in 1976.

1. Alice and Bob publicly agree to use a modulus p and generator g (that generates the cyclic group $\mathbb{Z}/p\mathbb{Z}$)

2. Alice chooses a secret integer a and sends Bob $A = g^a \ mod \ p$
3. Bob chooses a secret integer b and sends Alice $B = g^b \ mod \ p$
4. Alice and Bob would be able to arrive at a shared secret s like so:

$$s = A^b \ mod \ p = B^a \ mod \ p = g^{ab} \ mod \ p$$

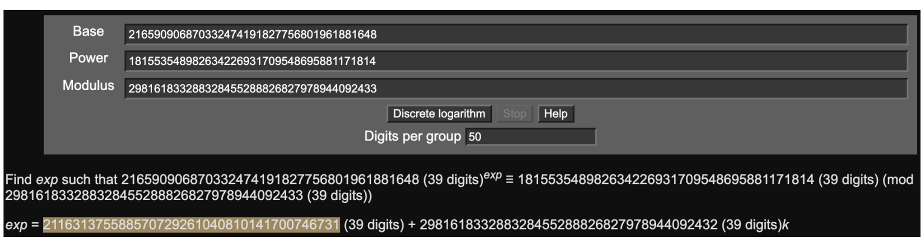
We are to look out for the publicly available p, g, g^a, g^b. Once we have these information, we can use the Discrete Logarithm Calculator to work out a or b to calculate $g^{ab} \ mod \ p$.

3.3 Computation

After scouring the packets, you should have found the 4 public numbers:

- p = 298161833288328455288826827978944092433
- g = 216590906870332474191827756801961881648
- g^a = 181553548982634226931709548695881171814
- g^b = 64889049934231151703132324484506000958

We now compute a.



$$a = 211631375588570729261040810141700746731$$

Lastly, using Python Console we easily compute $(g^b)^a \ mod \ p$ using the pow function, i.e. `pow(`g^b`, `a`, `p`)`.
We arrive at the flag, **246544130863363089867058587807471986686**.

Appendix: Pohlig-Hellman Algorithm

This is the algorithm used by the Discrete Logarithm Calculator. Here we provide a worked example with small numbers to show how it is done.

$$\beta = \alpha^x \ mod\ p,\ 0 \leq x \leq p-1$$

Let $p = 41,\ \alpha = 7,\ \beta = 12$, i.e. $12 = 7^x \ mod\ 41$. Solve for x.

1. Find the prime factors of Euler totient function, $\varphi(p)$. We know p is prime, so
$$\varphi(p) = p - 1 = 2^3 \cdot 5$$
$$q = \{2, 5\}$$

2. Find a congruence for prime factor, q.

3. For $q = 2$, $x = 2^0 \cdot x_0 + 2^1 \cdot x_1 + 2^2 \cdot x_2$, x has 3 terms as 2 has power 3.

   - Solving for x_0,
$$\beta^{\frac{p-1}{q}} = \alpha^{\frac{p-1}{q}x_0} \quad (1)$$
$$12^{20} = 7^{20x_0} \quad (2)$$
$$-1\ (mod\ 41) = (-1)^{x_0}\ (mod\ 41) \quad (3)$$
$$\text{Test } x_0 = 0, 1, 2, ... \implies x_0 = 1 \quad (4)$$

   - Solving for x_1,
$$\beta_1 = \beta\alpha^{-x_0} = 12 \cdot 7^{-1} = 31\ (mod\ 41) \quad (5)$$
$$\beta_1^{\frac{p-1}{q_1}} = \alpha^{\frac{p-1}{q}x_1},\ q_1 = 2^2 \quad (6)$$
$$31^{\frac{40}{4}} = 7^{\frac{40}{2}x_1} \quad (7)$$
$$31^{10} = 7^{20x_1} \quad (8)$$
$$31^{10}\ (mod\ 41) = 1\ (mod\ 41) \implies x_1 = 0 \quad (9)$$

   - Solving for x_2,
$$\beta_2 = \beta_1\alpha^{-x_1} = 31 \cdot 7^{-0} = 31\ (mod\ 41) \quad (10)$$
$$\beta_2^{\frac{p-1}{q_2}} = \alpha^{\frac{p-1}{q}x_2},\ q_2 = 2^3 \quad (11)$$
$$31^{\frac{40}{8}} = 7^{\frac{40}{2}x_1} \quad (12)$$
$$31^5 = 7^{20x_1} \quad (13)$$
$$-1\ (mod\ 41) = (-1)^{x_2}\ (mod\ 41) \implies x_2 = 1 \quad (14)$$

   - Hence, $x = 2^0 \cdot 1 + 2^1 \cdot 0 + 2^2 \cdot 1 = 5$
   - $x = 5\ (mod\ 2^3) = 5\ (mod\ 8)$

4. For $q = 5$, $x = 5^0 \cdot x_0$, x has 1 term as 5 has power 1.

   - Solving for x_0,

$$\beta^{\frac{p-1}{q}} = \alpha^{\frac{p-1}{q} x_0} \tag{15}$$

$$12^{\frac{40}{5}} = 7^{\frac{40}{5} x_0} \tag{16}$$

$$12^8 = 7^{8x_0} \tag{17}$$

$$18 \equiv 37^{x_0} \ (mod\ 41) \tag{18}$$

$$\text{Test } x_0 = 2, 3, 4, ... \implies x_0 = 3 \tag{19}$$

   - $x = 5^0 \cdot x_0 = 1 \cdot 3 = 3 \ (mod\ 5)$

5. We now solve the simultaneous congruence equation by Chinese Remainder Theorem

$$x \equiv 5 \ (mod\ 8) \tag{20}$$

$$x \equiv 3 \ (mod\ 5) \tag{21}$$

   - $gcd(8, 5) = 1 = 8(2) + 5(-3)$ by Euclidean Algorithm

$$5(-3) \equiv 1 \ (mod\ 8) \tag{22}$$

$$8(2) \equiv 1 \ (mod\ 5) \tag{23}$$

   - Consider $x = 3 \cdot 8(2) + 5 \cdot 5(-3) = -27 \ (mod\ 8 \cdot 5) = 13 \ (mod\ 40)$.

6. Thus, we arrive at $12 = 7^{13} \ (mod\ 41)$.

The time complexity of this algoritm is $\mathcal{O}(\sqrt{\beta})$ when $\varphi(p)$ is large. The online calculator has further optimisations that enable the discrete logarithm problem to be solved much quicker than a direct implementation.