B03102082

陳孝思

NA

1. Cryptolocker
    1. Flag: NASA{Ev3n_s3cure_PRF_1s_bre4k4ble_1f_u5ed_unc0rr3ctly}
    2. 方法：透過觀察，會發現每一次加密前都會 pad，pad 會把要加密的字串的長度補到 32 的倍數，且如果後面缺 5 個，就補 5 個 ascii 為 5 的東西，在觀察到每次加密回傳的結果是前面補上長度為 16 的字串加上後面加密過後長度為 32 倍數的字串，假設 input 的長度補成 32*n（n 為任意正整數），每次加密完拿到的結果的長度是 16+32*n
    透過這樣的結果，會發現，經過第一次加密之後，字串長度變成 16+32*n，第二次加密前，程式會把字串補上 16 個「\x16」，這樣待加密的字串長度才會是 32 的倍數，經過第二次加密後，長度又變成 16+32*k(k 為任意正整數)，後面幾次同理。
    而且他把密碼拆成 4 個 256bit，一層只用一個加密，這樣我們每層只要處理 2 個字(128*128 種可能)
    所以我們要做的是，一層一層暴力找出解答，解外面三層加密時，每次試一組 key 時，檢查 decrypt 後的字串後面是否為 16 個「\x16」(其實只要後兩位是\x16，那該組為正確的機率就蠻大的)，如果是的話就可能是我們要的，把它存起來給裡面那層暴力解。
    執行到最裡面那層時，剩下的可能也不多，就全部解出來直接在裡面找答案（python 1.py flag.encrypted | grep 'NASA'）

    Ref: b05902002 李栢淵

2. 2AES
    1. Flag: NASA{Tw0_3qual5_to_0ne_Whyyyyyy}
    2. 因為我們有一組沒加密與加密過的字串了，先把沒加密的字串暴力加密（1-2^23）建一個表格（hash table），key 是加密過後的字串，value 是我們加密時用的 key(1-2^23)，建完表格之後，再暴力解密已加密的字串(1-2^23)，但是這次解密的同時，會用解密得到的字串去查詢剛剛建立的表格，如果解密得到的字串是表格的 key 之一，表示這次解密所用的 key 是第二次加密的 key，而查詢表格所得到的 value 是第一次加密所使用的 key，這樣兩把 key 都拿到了，就可以直接去解密了，時間複雜度大約是 O(2^24)不是 O(2^46)
    ref: http://sconce.ics.uci.edu/134-S11/LEC5.pdf

b05902002 李栢淵

3. Man in the Middle 2

1. Flag: NASA{Ahhhh...I_g0t_Mitm_4g41n...}
2. 方法：

核心概念：因為最後 linux13.csie.org:7122 給的 flag 是原本的 flag 與密碼產生的 g 等東西進行多次 XOR 的結果，且對正整數 XOR 兩次會得到一樣的數字，我們可以透過 XOR 兩次來猜測密碼。

實際作法：設 k 為 2，開兩個連線，假設為 a 和 b，每次鎖定某一輪的密碼，是該輪的話就開始暴力猜測密碼是多少(1-20)，傳猜測的數字的 k 次方給 a 和 b，這樣 a 會用 g^(ka)對 key 做 XOR，b 會用 g^(kb)對 key 做 XOR，該輪之外就傳 a 給我的 g^a 給 b，b 會用 g^(ba)對 key 做 XOR，傳 b 給我的 g^b 給 a，a 會用 g^(ba)對 key 做 XOR。

最後 a 跟 b 會用得到的 key 對 flag 做 XOR，這樣 a 給我的 flag 實際上是原本的 flag 跟其他九輪的 g^(ab)和我們鎖定的那一輪的 g^(ka)做 OXR 所得到的，b 給我的 flag 也是原本的 flag 跟其他九輪的 g^(ab)和我們鎖定的那一輪的 g^(kb)做 OXR 所得到的，因為 XOR 有交換律且 XOR 運算的反元素是同樣的元素，所以如果我們猜對了密碼，給 a 和 b 都是正確的 g 的話，那針對 a 給的 flag 跟 g^(ka)做 XOR 得到的值，應該跟對 b 給的 flag 和 g^(kb)做 XOR 得到的值一樣，透過這樣的方法，每一個密碼只要檢查 20 個可能，所以最多只需檢查 10*20 次。

而拿到密碼之後，就可以透過這組密碼重現每一輪正確的 g，把 g^k 給 linux13.csie.org:7122，然後對最後拿到的 flag 跟每一輪拿到的 ga 乘上 k 次方，做 XOR 運算，就會得到原本的 flag。

ref: b05902002 李栢淵