Date: 11/28/2023

Tester: Daryl Mokolo

CWE-327 HIGH: Use of a Broken or Risky Cryptographic Algorithm

Threat overview

After analyzing the system, I found that symmetric encryption file ("symmetric.enc") exhibits a vulnerability associated with the use of a weak encryption algorithm. The specific weakness identified is the utilization of an insecure encryption method, as revealed through frequency analysis.

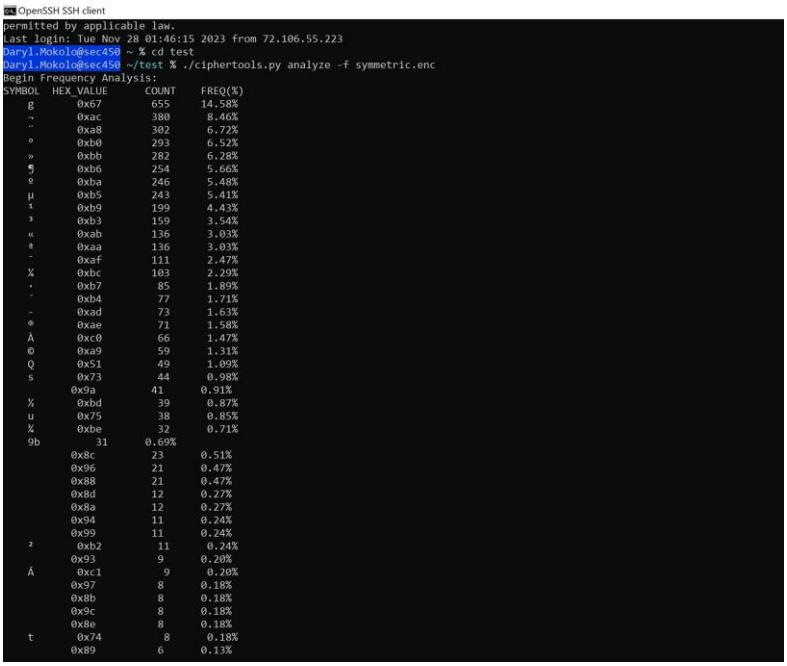
Affected:

Users data and sensitive information

Steps to recreate the vulnerability:

Step 1:

Conduct the a frequency analysis using the command: **./ciphertools.py analyze -f symmetric.enc** that use a python frequency analysis tool to determine potential keys.



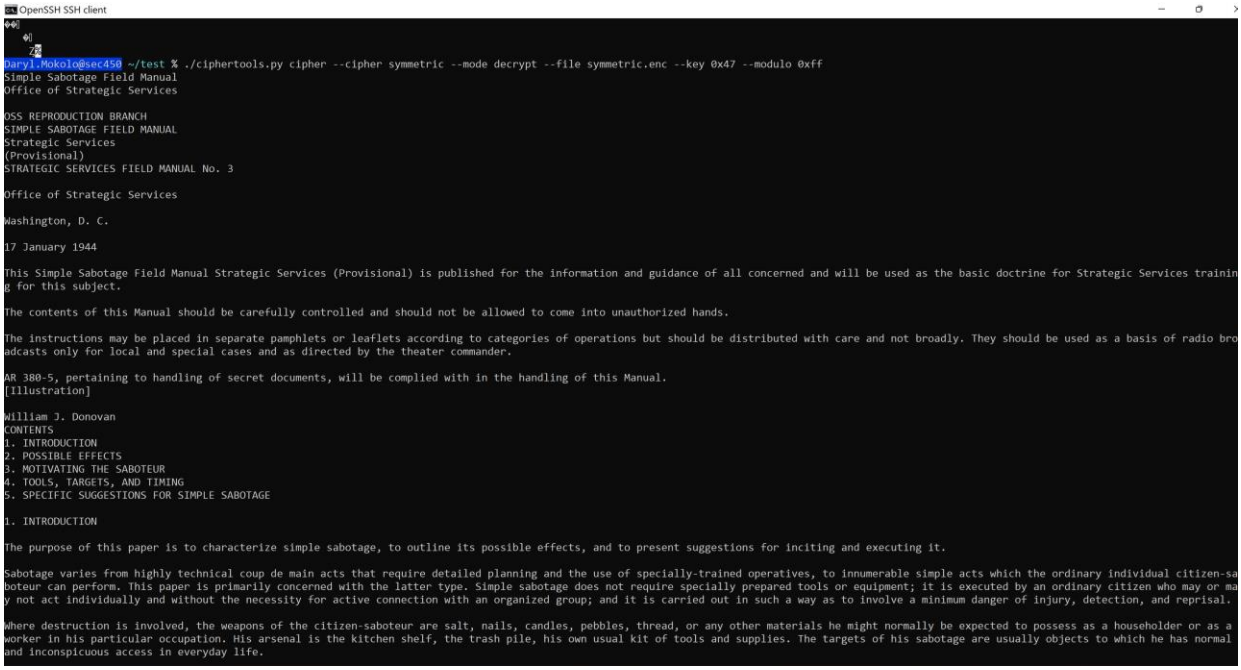
Step 2:

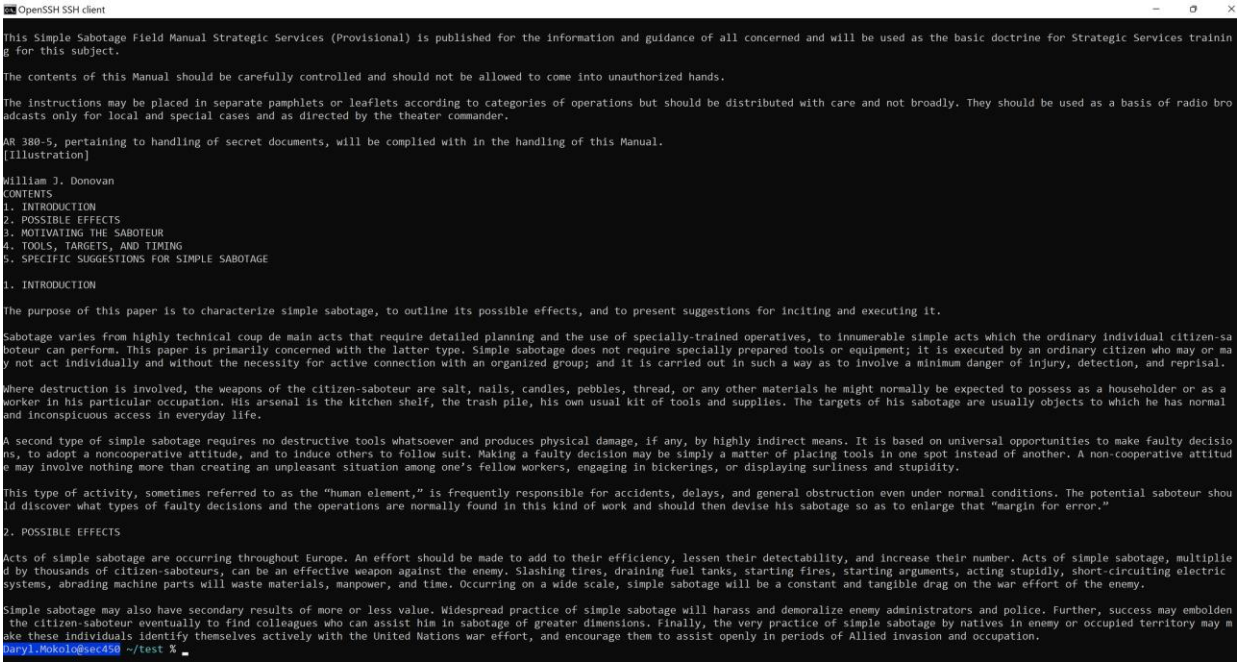
After Analyzing frequency which revealed weaknesses, allowing me to determine that 'g' corresponds to a space character and '¬' corresponds to the letter 'e.' Then calculating the hex value which will

determine the main key by subtracting the value of g which is 103 and space = 32 from the ASCII table and got the value 71 which is 0x47 in hex value. Hence using this key in the command:

./ciphertools.py cipher --cipher symmetric --mode decrypt --file symmetric.enc --key 0x47 --modulo 0xff

The symmetric.enc file could be decrypted since the correct key was entered.





Remediation:

Upgrading the encryption algorithm to a more secure symmetric encryption algorithm, such as AES-256, to mitigate the identified vulnerabilities will greatly improve the file security. In addition, implementing a robust key management system by including regular key rotation and secure storage will Strengthen overall cryptographic controls to prevent future vulnerabilities.

CWE-326 HIGH: Inadequate Encryption Strength

Threat overview

The asymmetric encryption file ("asymmetric.enc") reveals a vulnerability linked to weak key management and a potential flaw in the chosen asymmetric encryption algorithm.

Affected:

Users data and sensitive information

Steps to recreate the vulnerability:

Step 1:

Similarly to the symmetrical encryption that was decrypted earlier, the "asymmetric.enc" file exhibited certain patterns after the cipher frequency analysis was done. Hence, specific identification of characters and their probable plaintext equivalents was found. The character 's' in the ciphertext consistently mapped to a plaintext space character, revealing a potential weakness. By entering the commands:

```
./ciphertools.py analyze -f asymmetric.enc
```

```
OpenSSH SSH client
 the citizen-saboteur eventually to find colleagues who can assist him in sabotage of greater dimensions. Finally, the very practice of simple sabotage by natives in enemy
ake these individuals identify themselves actively with the United Nations war effort, and encourage them to assist openly in periods of Allied invasion and occupation.
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py analyze -f assymmetric.enc
Traceback (most recent call last):
  File "/home/Daryl.Mokolo/test/./ciphertools.py", line 325, in <module>
    do_args(mode, parsed)
  File "/home/Daryl.Mokolo/test/./ciphertools.py", line 311, in do_args
    message = get_message(infile)
              ^^^^^^^^^^^^^^^^^^
  File "/home/Daryl.Mokolo/test/./ciphertools.py", line 94, in get_message
    with open(path, "rb") as f:
         ^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory: 'assymmetric.enc'
Daryl.Mokolo@sec450 ~/test % ls
asymmetric.enc  ciphertools.py  group2.zip  symmetric.enc
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py analyze -f asymmetric.enc
Begin Frequency Analysis:
SYMBOL  HEX_VALUE      COUNT    FREQ(%)
    s       0x73        432     17.31%

       0x85         273     10.94%
    ²       0xb2        171      6.85%
    Î       0xce        159      6.37%
    <       0x3c        145      5.81%
            0xa0        140      5.61%
    N       0x4e        136      5.45%
    ▯       0x05        134      5.37%
    W       0x57        131      5.25%
           0x8e         100      4.01%
    ×       0xd7          85     3.41%
    E       0x45          68     2.72%
    *       0x2a          62     2.48%
    ò       0xf2          51     2.04%
    3       0x33          51     2.04%
    ▯       0x17          49     1.96%
    `       0x60          48     1.92%
    à       0xe0          36     1.44%
    |       0x7c          36     1.44%
          0x0e           33     1.32%
    û       0xfb          30     1.20%
    »       0xbb          29     1.16%
    Ù       0xd9          26     1.04%
    5       0x35          13     0.52%
           0x86           12     0.48%
           0x97            9     0.36%
    é       0xe9           4     0.16%
    ü       0xfc           4     0.16%
    Ä       0xc4           3     0.12%
Daryl.Mokolo@sec450 ~/test %
```

./ciphertools.py crack --cipher-byte 0xa3 --plaintext-guess 0x20 --modulo 0xff

Hence, The subsequent application of a brute-force attack using the command : **./ciphertools.py crack** identified a limited set of potential keys that eventually led lead to the plain text decrypted message.

```
OpenSSH SSH client
Daryl.Mokolo@sec450 ~/test %
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0xa3 --plaintext-guess 0x20 --modulo 0xff
Examined 30 potential keys
Given cipher byte 0xa3 and plaintext guess 0x20 the encyption key would be 0x2c
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x20 --modulo 0xff
Examined 98 potential keys
Given cipher byte 0x2c and plaintext guess 0x20 the encyption key would be 0xa3
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0xa3 --plaintext-guess 0x2c --modulo 0xff
Examined 138 potential keys
Given cipher byte 0xa3 and plaintext guess 0x2c the encyption key would be 0xbc
Daryl.Mokolo@sec450 ~/test %
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0xa3 --plaintext-guess 0x33 --modulo 0xff
Examined 255 potential keys      0.04%
[illegible]
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x33 --plaintext-guess 0x2c --modulo 0xff
Examined 255 potential keys9466866392
[illegible]
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0xa3 --modulo 0xff
Examined 189 potential keys
Given cipher byte 0x2c and plaintext guess 0xa3 the encyption key would be 0x89
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x89 --modulo 0xff
Examined 113 potential keys
Given cipher byte 0x2c and plaintext guess 0x89 the encyption key would be 0x94
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x94 --modulo 0xff
Examined 84 potential keys
Given cipher byte 0x2c and plaintext guess 0x94 the encyption key would be 0xd4
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0xd4 --modulo 0xff
Examined 83 potential keys
Given cipher byte 0x2c and plaintext guess 0xd4 the encyption key would be 0x1c
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x1c --modulo 0xff
Examined 39 potential keys
Given cipher byte 0x2c and plaintext guess 0x1c the encyption key would be 0x2f
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2f --plaintext-guess 0x1c --modulo 0xff
Examined 30 potential keys
Given cipher byte 0x2f and plaintext guess 0x1c the encyption key would be 0x2c
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x2c --modulo 0xff
Examined 2 potential keys
Given cipher byte 0x2c and plaintext guess 0x2c the encyption key would be 0x1
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x1 --modulo 0xff
Examined 45 potential keys
Given cipher byte 0x2c and plaintext guess 0x1 the encyption key would be 0x1d
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x1d --modulo 0xff
Examined 152 potential keys
Given cipher byte 0x2c and plaintext guess 0x1d the encyption key would be 0x4c
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0x4c --modulo 0xff
Examined 15 potential keys
Given cipher byte 0x2c and plaintext guess 0x4c the encyption key would be 0xa4
Daryl.Mokolo@sec450 ~/test % ./ciphertools.py crack --cipher-byte 0x2c --plaintext-guess 0xa4 --modulo 0xff
Examined 107 potential keys
```

Remediation:

The best remediation to improve these files security is to transition to a stronger asymmetric encryption algorithm, such as RSA with a larger key size, to enhance encryption strength. In addition, Ensuring the alignment with industry best practices and cryptographic standards.

Reference:

CWE - CWE-347: Improper Verification of Cryptographic Signature (4.13) (mitre.org)

CWE - CWE-327: Use of a Broken or Risky Cryptographic Algorithm (4.13) (mitre.org)