Labreport 06

Julian Deinert, Tronje Krabbe

30. Juni 2016

Inhaltsverzeichnis

1. TODO

1.2

TODO

2. TODO

2.1

TODO

3. Unsichere selbstenwtickelte Verschlüsselungsalgorithmen

3.1 BaziCrypt

Die letzten 10 Bytes des Ciphertexts sind exakt der Key. Oder, genauergesagt, der Key XOR Null, was den Key unverändert lässt. Jetzt ist es sehr einfach, die Dateien zu entschlüsseln. Siehe unser Skript im Appendix. Die drei Plaintexte sind:

> [n01.txt.enc] Hallo Peter. Endlich koennen wir geheim kommunizieren! Bis bald, Max
>
> [n02.txt.enc] Hi Max! Super, Sicherheitsbewusstsein ist ja extrem wichtig! Schoene Gruesse, Peter.
>
> [n03.txt.enc] Hi Peter, hast du einen Geheimtipp fuer ein gutes Buch fuer mich? Gruss, Max

3.2 AdvaziCrypt - Denksport

Wenn wir davon ausgehen, dass der Key vollständig (wenn auch 'verschlüsselt') in der Datei vorhanden ist, dann sind auf jeden Fall die letzten 10 Bytes des Ciphertexts die verschlüsselten Key-Bytes. Jetzt kann man auf die Anzahl der Padding-Bytes schließen, indem man darauf achtet, wann die hinteren Bytes aufhören, Teil des vermeintlichen, verschlüsselten Keys zu sein. Man zählt also mit und schaut, wann ein Byte nicht mehr passt. Für die genaue Implementation, siehe unser Skript im Appendix.

3.3 AdvaziCrypt

Die Plaintexte lauten:

> [n04.txt.enc] Hi Max, natuerlich: Kryptologie von A. Beutelspacher ist super. Gruss Peter
>
> [n05.txt.enc] Hi Peter, worum geht es in dem Buch? Ciao, Max.
>
> [n06.txt.enc] Hi Max, das ist ein super Buch, das viele Krypto-Themen abdeckt. Gruss Peter

Appendix

3.4 3. Skripte

Bazi Crack

```
#!/usr/bin/env python
def crack(filename, bsize):
    """Decrypt a bazi-encrypted file given the file's
    name and the blocksize (usually 10)
    """

    result = ''
    counter = -1
    with open(filename, "rb") as f:
        f.seek(-bsize, 2) # seek last n bytes
        key = [int(b) for b in f.read(bsize)]
        f.seek(0, 0) # seek beginning of file again
        while True:
            counter += 1
            counter = counter % bsize
            b = f.read(1)
            if b == b'':
                break
            num = int.from_bytes(b, 'little')
            result += chr(num ^ key[counter])
    return result


if __name__ == "__main__":
    import sys
    print(crack(sys.argv[1], 10))
```

Advazi Crack

```
#!/usr/bin/env python
def crack(filename, bsize):
    """Decrypt an advazi-encrypted file given the file's
    name and the blocksize (usually 10)
    """

    result = ''
    with open(filename, "rb") as f:
        f.seek(-bsize, 2) # seek last n bytes

        # initialize key as last n bytes
        key = [int(b) for b in f.read(bsize)]

        f.seek(0, 0) # seek beginning of file again
        whole_file = f.read() # read in the entire file

        # compute padding
        padding = 0
        counter = bsize - 1
        for i in range(len(whole_file) - 1, 0, -1):
            if whole_file[i] != key[counter]:
                break
            else:
                padding += 1
            counter -= 1
            counter %= 10

        # decrypt key with padding value
        for i in range(len(key)):
            key[i] ^= padding

        counter = -1
        f.seek(0, 0) # seek beginning of file again

        # decrypt plaintext
        while True:
            counter += 1
            counter %= 10
            b = f.read(1)
            if b == b'':
                break
            num = int.from_bytes(b, 'little')
            result += chr(num ^ key[counter])
    return result[:len(result) - padding]

if __name__ == "__main__":
    import sys
    print(crack(sys.argv[1], 10))
```