Viggo Haimanas Bygdén, Noah Marklund

Planering: Alphabetical Authentication

Uppgiftsbeskrivning

Uppgiften handlar om att dekryptera en lång fil, ’password.txt’ men detta måste göras inom en tidsram som vi har satt till 3 minuter *preliminärt*. Efter tidsbegränsningen är nåd så raderas filen och *.exe* filen måste startas om på nytt. Planen är att filen krypterar med en relativt enkel krypteringsmetod och utmaningen ligger mer i tidsbegränsningen än i själva krypteringen. Planen är att kryptera ett lösenord som är genererat dynamiskt och slumpmässigt inuti *.exe* filen, denna krypteras sedan med hjälp av vår krypteringsmetod ”Alphabetical-exclusion” det vill säga att alla bokstäver förutom det som ska krypteras ingår i meddelandet *till exempel:*

Algoritm: Kryptera meddelande (en iteration)

1 $Meddelandet: \overset{1}{\underset{\smile}{H}}\ \overset{2}{\underset{\smile}{E}}\ \overset{3}{\underset{\smile}{J}}$ // Varje bokstav är unik

2 $1 \rightarrow \text{ABCDEFG}\ \underset{\smile}{\cancel{H}}\ \text{IJKLMNOPQRSTUVWXYZ} \rightarrow \text{ABCDEFGIJKLMNOPQRSTUVWXYZ}$

3 $2 \rightarrow \text{ABCD}\ \underset{\smile}{\cancel{E}}\ \text{FGHIJKLMNOPQRSTUVWXYZ} \rightarrow \text{ABCDFGHIJKLMNOPQRSTUVWXYZ}$

4 $3 \rightarrow \text{ABCDEFGHI}\ \underset{\smile}{\cancel{J}}\ \text{KLMNOPQRSTUVWXYZ} \rightarrow \text{ABCDEFGHIKLMNOPQRSTUVWXYZ}$

5

// Resultaten paras ihop till en slutlig sträng

$Resultatet$:

$$\overbrace{\text{ABCDEFGIJKLMN}\overset{1}{\text{O}}\text{PQRSTUVWXYZ}}\ \overbrace{\text{ABCDFGHIJKLMN}\overset{2}{\text{O}}\text{PQRSTUVWXYZ}}$$

$$\overbrace{\text{ABCDEFGHIKLMN}\overset{3}{\text{O}}\text{PQRSTUVWXYZ}}$$

Denna metod av kryptering gör att ett kort lösenord blir mycket stort och speciellt om vi gör detta rekursiv det vill säga krypterar flertalet gånger. Tanken är att den som ska få flaggan behöver dekryptera meddelandet innan tiden tagit slut och då inuti .exe filens terminal skriva lösenordet och då returneras flaggan. En annan metod är dock att användaren tar sig in i .exe filen och på så sätt läser ASCII strängen och då får flaggan utan att tänka på tidsbegränsningen men hur svår och tidskrävande denna metod är, kan variera och är inte den menade metoden.

Användaren ska få 1 fil: ’alpha_auth.exe’ instruktionerna anger att ’alpha_auto.exe’ ska läggas i en fil för att underlätta letandet efter ’password.txt’ som då genereras på samma

nivå som filen. Användaren ska nu läsa igenom filen och hitta mönstret, men om de öppnar ’alpha_auth.exe’ på nytt eller om tiden tar slut så rensas filen av sitt innehåll.

Grov lösningsskiss

Om användaren väljer att dekryptera meddelandet så har vi skrivit lite pseudo kod som kan vara ett exempel på en lösning. Fast då att message laddas in och att koden översätts till riktig kod, vad användaren väljer att använda är helt frivilligt huvudsaken är bara att de får ut flaggan inom tidsramen.

Algoritm: Dekryptera meddelande

```
1  function decrypt(message, iterations)
2      alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
3
4      for i < iterations do
5          chunks = 25 // chunks är antalet bokstäver i alfabetet
6          decrypted = ""
7
8          for i < message.length with steps of 25 do
9
10             chunk = message[from i to i + chunks]
11             missing_letter = alphabet − chunk // tar bort allt förutom den
               borttagna bokstaven
12             decrypted += missing_letter
13
14         messsage = decrypted // uppdatera meddelandet
15
16     return message
```

Algoritm: Dekryptera meddelande

```
17 function decrypt(message, iterations)
18     alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
19
20     for i < iterations do
21         chunks = 25 // chunks är antalet bokstäver i alfabetet
22         decrypted = ""
23
24         for i < message.length with steps of 25 do
25
26             chunk = message[from i to i + chunks]
```

```
27            missing_letter = alphabet − chunk // tar bort allt förutom den
              borttagna bokstaven
28            decrypted += missing_letter
29
30        messsage = decrypted // uppdatera meddelandet
31
32    return message
```

Svårighetsnivå: 0.4 (40% av hela klassen)

Vi bedömer att 0.4, 40% av hela klassen, kommer att klara vår uppgift då den tar ett tag att dekryptera då man behöver lista ut antalet iterationer i vår algoritm samt hur långt det ursprungliga lösenordet är.