Görevlerin türkçe çevirisi “courses.kodluyoruz.org” dan alınmıştır.

"Caesar" Görevi Tanımı | Problem Seti 2 | CS50x Ders yazılımları

10-14 dakika

Öncelikle CS50 Sandbox'ta bu sayfayı açın ve aşağıdakileri takip edin:

Et tu? (Latince, Sen de mi?)

Sözde Sezar (evet, bildiğiniz Sezar), içlerindeki her harfi bir dizi yere kaydırarak gizli mesajları “şifreliyordu” (yani geri dönüşümlü bir şekilde saklıyordu). Örneğin, Sezar A'yı B, B olarak C, C olarak D,… ve alfabetik olarak Z'yi A olarak yazabilir ve böylece birine HELLO demek için IFMMP yazabilir. Sezar'dan bu tür mesajlar aldıktan sonra, alıcıların harfleri ters yönde aynı sayıda yere kaydırarak “şifresini çözmesi” gerekecektir.

Bu “kriptosistem” in gizliliği sadece Sezar'a ve bir sırrı bilen alıcılara, Sezar'ın harfleri ne kadar değiştirdiğine dayanıyordu (örneğin 1 değiştirme). Modern standartlara göre pek de güvenli değil. Ama, hey, belki de dünyada bunu yapan ilk kişi iseniz, oldukça güvenli!

Şifrelenmemiş metin genellikle düz metin olarak adlandırılır. Şifrelenmiş metne genellikle şifre metni denir. Ve şifreli metni oluşturmak için kullanılan sırra ise anahtar denir.

Net olmak gerekirse, HELLO'yu 1 anahtarıyla şifreleyerek (1 kez kaydırarak) IFMMP'yi işte böyle elde ediyoruz:

düzyazı	H	E	L	L	O
+ anahtar	1	1	1	1	1
= şifremetni	I	F	M	M	P

Daha resmi olarak, Sezar’ın algoritması (yani şifre), her harfi k konumlarına "döndürerek" mesajları şifreler. Daha resmi olarak, p bir düz metinse (yani şifrelenmemiş bir mesajsa), p(i) p'deki i’inci karakterdir ve k gizli bir anahtardır (yani, negatif olmayan bir tamsayıdır), ardından şifreleme metnindeki her bir c(i) şu şekilde hesaplanır:

c(i) = (p(i) + k) % 26

burada % 26 ile gösterilen yer "26'ya bölündüğünde kalan" anlamına gelir. Bu formül belki de şifreyi olduğundan daha karmaşık hale getirir, ancak algoritmayı tam olarak ifade etmenin sadece kısa bir yoludur. Gerçekten de, sırf bu konuları tartışmak için A (veya a) 'yı 0, B (veya b)' yi 1,…, H (veya h) 7, I (veya i) 8,… ve Z (veya z) 25 olarak da düşünebilirsiniz. Diyelim ki Sezar bu sefer, k anahtarı olarak 3'ü kullanıp gizlice "Hi" demek istiyor. Ve böylece onun düz metni *p*, yani Hi. Bu durumda düz metnin ilk karakteri p(0), yani H (yani 7) ve düz metnin ikinci karakteri ise p(1), yani i'dir (yani 8). Şifreleme metninin ilk karakteri olan c(0), K'dir ve şifreleme metninin ikinci karakteri olan c(1), L'dir. Nedenini görebiliyor musunuz?

Ödeviniz olarak, Sezar'ın şifresini kullanarak mesajları şifrelemenizi sağlayan Sezar adlı bir program yazmanızı istiyoruz. Kullanıcı programı çalıştırırken, komut satırında argüman vererek gizli mesajda anahtarın ne olması gerektiğine karar vermelidir. Kullanıcının anahtarının bir sayı olacağını varsaymamalıyız; yine de, eğer bir sayı ise, pozitif bir tamsayı olacağını varsayabilirsiniz. İşte programın nasıl çalışabileceğine dair birkaç örnek. Örneğin, kullanıcı 1 anahtarını ve HELLO düz metnini girerse:

```
$ ./caesar 1

plaintext:  HELLO

ciphertext: IFMMP
```

Kullanıcı 13 anahtarını ve "hello, world" düz metnini sağlarsa program şu şekilde çalışabilir:

```
$ ./caesar 13

plaintext:  hello, world

ciphertext: uryyb, jbeyq
```

Ne virgül ne de boşluğun şifre tarafından "kaydırılmadığına" dikkat edin. Sadece alfabetik karakterleri kaydırın!

Bir tane örneğe daha ne dersiniz? Kullanıcı tekrar 13 anahtarını daha karmaşık bir düz metinle sağlarsa program şu şekilde çalışabilir:

```
$ ./caesar 13

plaintext:  be sure to drink your Ovaltine

ciphertext: or fher gb qevax lbhe Binygvar
```

Orijinal mesajın korunduğuna dikkat edin. Küçük harfler küçük, büyük harfler büyük kalır.

Peki ya bir kullanıcı bize yardımcı olmazsa?

$ ./caesar HELLO

Usage: ./caesar key

Cidden olmazsa?

$ ./caesar

Usage: ./caesar key

Ya da hatta…

$ ./caesar 1 2 3

Usage: ./caesar key

Siz deneyin:

Bu probleme ekibin bulduğu çözümü denemek için bu sandbox'ta aşağıdaki komutu çalıştırın (tabii ki *key* yerine geçerli bir tamsayı yazarak):

./caesar key

Peki, bu probleme nasıl başlamalıyız? Bu soruna adım adım yaklaşalım:

Sözde Kod (Pseudocode)

Öncelikle, bu programı uygulayan kodun nasıl yazıldığından emin olmasanız bile (şimdilik!), Ceaser CS50 Lab'de yer alan pseudocode.txt dosyasına bu programı uygulayan bir sözde kod yazın. Sözde kod yazmanın tek bir doğru yolu yoktur, yalnızca kısa cümleler yeterlidir. Mike Smith'i bulmak için sözde kodu nasıl yazdığımızı hatırlayın. Olasılıkla, sözde kodunuz bir veya daha fazla işlev, koşul, Bool ifadesi, döngü ve / veya değişken kullanacaktır (ya da ima edecektir!)

Spoiler:

Bunu yapmanın birden fazla yolu var, işte sadece bir tanesi!

1. Programın bir komut satırı değişkeniyle çalıştırılıp çalıştırılmadığını kontrol edin.
2. Tüm karakterlerin rakam olduğundan emin olmak için sağlanan argümanı yineleyin.
3. Bu komut satırı argümanını string'den int'e dönüştürün.
4. Kullanıcıdan düz metin isteyin.
5. Düz metnin her karakteri üzerinde yineleme yapın:

i. Büyük harfse, şifreye göre kaydırın, harf büyüklüğünü koruyun, sonra kaydırılmış karakteri yazdırın.

ii. Küçük harfse, şifreye göre kaydırın, harf büyüklüğünü koruyun, sonra kaydırılmış karakteri yazdırın.
iii. İkisi de değilse karakter neyse onu yazdırın.

6. Yeni satır yazdırın.

Bu sözde kodu gördükten sonra kendinizinkini buna bakarak düzenleyebilirsiniz, ancak bizimkini alıp direkt kopyalayıp yapıştırmayın!

Komut Satırı Argümanlarını Sayma

Sözde kodunuz ne olursa olsun, ek işlevsellik eklemeden önce yalnızca programın tek bir komut satırı bağımsız değişkeniyle çalıştırılıp çalıştırılmadığını kontrol eden C kodunu yazalım.

Özellikle, CS50 Lab'de "caesar.c" yi şu şekilde değiştirin: kullanıcı tam olarak bir komut satırı argümanı sağlarsa, "Success" yazar; kullanıcı herhangi bir komut satırı argümanı veya iki ya da daha fazlasını sağlamazsa, "Usage: ./caesar key" yazdırır ve main hemen 1 değerini (bir hata olduğunu gösterir) döndürür. Unutmayın, bu anahtar doğrudan komut satırından geldiğinden ve "get_string" yoluyla gelmediğinden, kullanıcıdan yeniden girdi isteme fırsatımız yoktur. Ortaya çıkan programın davranışı aşağıdaki gibi olmalıdır.

$ ./caesar 20

Success

ya da

$ ./caesar

Usage: ./caesar key

ya da

$ ./caesar 1 2 3

Usage: ./caesar key

İpuçları

- Programınızı make ile derleyebileceğinizi hatırlayın.
- Printf ile yazdırabileceğinizi hatırlayın.
- Argc ve argv'nin komut satırına girilenler hakkında bilgi verdiğini hatırlayın.
- Programın adının (burada, ./caesar) argv[0]'da olduğunu hatırlayın.

Anahtar Erişimi

Artık programınız (umuyoruz ki!) öngörülen şekilde giriş kabul ettiğine göre, şimdi başka bir adım atmanın zamanı geldi.

Programımızı, teknik olarak bir argüman (anahtar) giren, ama girdiği argüman bir tamsayı olmayan kullanıcılara karşı korumamız gerektiğini hatırlayın, örneğin:

$ ./caesar xyz

Ancak anahtarı geçerlilik açısından analiz etmeye başlamadan önce, gerçekten okuyabildiğimizden emin olalım. "Caesar.c" dosyasını, kullanıcının yalnızca bir komut satırı argümanı sağlayıp sağlamadığını denetlemekle kalmayıp, bunu doğruladıktan sonra, bu tek komut satırı argümanını da yazdıracak şekilde değiştirin. Örneğin, bu davranış şöyle görünebilir:

$ ./caesar 20

Success

20

İpuçları:

- Argc ve argv'nin komut satırına girilenler hakkında bilgi verdiğini hatırlayın.
- Argv'nin bir string dizisi olduğunu hatırlayın.
- Printf ile %s'i yer tutucu olarak kullanıp bir string yazdırabileceğimizi hatırlayın.
- Bilgisayar bilimcilerin 0'dan başlayarak saymayı sevdiğini hatırlayın.
- Köşeli parantez kullanarak argv gibi bir dizinin tek tek öğelerine erişebileceğimizi hatırlayın, örneğin: argv [0].

Anahtarı Doğrulama

Artık anahtarı nasıl okuyacağınızı bildiğinize göre, onu analiz edelim. CS50 Lab üzerindeki Caesar.c dosyasını, verilen komut satırı argümanı yazdırmak yerine, o komut satırı argümanının her karakterinin ondalık bir rakam (ör. 0, 1, 2 vb.) olup olmadığını denetleyecek şekilde değiştirin ve bunlardan herhangi biri değilse, Usage: ./caesar key iletisini yazdırdıktan sonra hemen sona ersin. Ancak argüman yalnızca rakamlardan oluşuyorsa, bu string'i (argv'nin, bu string'ler sayı gibi görünse bile bir string dizisi olduğunu hatırlayın) gerçek bir tam sayıya dönüştürmelisiniz. Sonra da bu tam sayıyı priftf ile %i üzerinden yazdırın. Örneğin, bu davranış şöyle görülebilir:

$ ./caesar 20

Success

20

ya da

$ ./caesar 20x

Usage: ./caesar key

İpuçları

- Argv'nin bir string dizisi olduğunu hatırlayın.
- Bu arada bir string'in sadece bir karakter(char) dizisi olduğunu hatırlayın.
- String.h başlık dosyasının, string'lerle çalışan bir dizi yararlı fonksiyon içerdiğini hatırlayın.
- Uzunluğunu biliyorsanız, bir string'in her bir karakterini yinelemek için döngü kullanabileceğimizi hatırlayın.
- Ctype.h başlık dosyasının, karakterler hakkında bize bilgi veren bir dizi yararlı fonksiyon içerdiğini hatırlayın.
- Programımızın başarıyla tamamlanmadığını belirtmek için sıfır olmayan değerleri main öğesinden döndürebileceğimizi hatırlayın.
- Printf ile %i'yi yer tutucu olarak kullanıp bir tam sayı yazdırabileceğimizi hatırlayın.
- atoi fonksiyonunun, bir sayı gibi görülen bir string'i o sayıya dönüştürdüğünü hatırlayın.

Kaputun Altına Bir Göz Atmak

İnsanlar olarak, "H + 1 = I" diyebildiğimiz gibi, yukarıda açıklanan formülü sezgisel olarak anlamak bizim için kolaydır. Fakat bir bilgisayar aynı mantığı anlayabilir mi? Hadi öğrenelim.Şimdilik, kullanıcının sağladığı anahtarı geçici olarak göz ardı edeceğiz ve bunun yerine kullanıcıdan gizli bir mesaj isteyecek ve tüm karakterlerini sadece 1 kaydırmaya çalışacağız.

Caesar.c fonksiyonunu anahtarı doğruladıktan sonra kullanıcıdan bir string ("açık metin" istediğimizi belirterek) ister ve sonra tüm karakterlerini 1 kaydırarak "şifre metin" çıktısını verir, sonra da yeni satıra geçeriz. Programınız daha sonra main'den 0 döndürerek çıkmalıdır. Ayrıca bu noktada, önceden "Success" yazdırdığımız kod satırını da silebiliriz. Bütün bunlarla programın davranışı şöyle olur:

$ ./caesar 1

plaintext: hello

ciphertext: ifmmp

İpuçları

- Düz metindeki her karakteri yinelemeye çalışın ve kelimenin tam anlamıyla 1 ekleyin, ardından yazdırın.
- Eğer c, C dilinde char türünde bir değişkense, printf ("% c", c + 1) çağrıldığında ne olur?

Şimdi Sıra Sizde

Şimdi her şeyi birbirine bağlama zamanı! Karakterleri 1 kaydırmak yerine, caesar.c dosyasını değiştirerek gerçek anahtar değerine kaydırın. Ve büyük-küçük harf farkını koruduğunuzdan emin olun! Büyük harfler büyük, küçük harfler küçük olmalı ve alfabetik olmayan karakterler değişmemelidir.

İpuçları

- Modulo (yani, geri kalan) operatörünü (%), Z'den A'ya kadar sarmalamak için kullanmak en iyisidir! Peki nasıl?
- Önceki bölümdeki tekniği kullanarak Z veya z'yi 1'e sarmaya çalışırsak işler garipleşir.
- Noktalama işaretlerini bu tekniği kullanarak sarmaya çalışırsak yine işler garipleşir.
- ASCII'nin yazdırılabilir tüm karakterleri rakamlarla eşlediğini hatırlayın.
- A'nın ASCII değerinin 65 iken a'nın ASCII değerinin 97 olduğunu hatırlayın.
- Printf'yi çağırdığınızda hiç çıktı görmüyorsanız, bunun nedeni muhtemelen geçerli ASCII aralığının dışındaki karakterleri yani 0 ile 127 arasındakileri yazdırmamanızdır. Hangi değerleri yazdırdığınızı görmek için önce karakterleri sayı olarak yazdırmayı (%c yerine %i kullanarak) deneyin ve yalnızca geçerli karakterleri yazdırmaya çalıştığınızdan emin olun!

"Vigenere" Görevi Tanımı | Problem Seti 2 | CS50x Ders yazılımları

8-11 dakika

Öncelikle CS50 Sandbox'ta bu sayfayı açın ve aşağıdakileri takip edin:

Ooh, la la!

Vigenère'in şifresi, mesajları bir dizi anahtarla şifreleyerek (veya başka bir deyişle, bir anahtar kelimeyi kullanarak) Sezar'ın şifresini geliştirir.

Başka bir deyişle, p bir düz metinse ve k bir anahtar kelimeyse (yani alfabetik bir string ise, burada A (veya a) 0, B (veya b) 1, C (veya c) 2,… ve Z (veya z) 25'i temsil eder), o zaman şifre metin, c, içinde bulunan her bir harf, c(i), şöyle hesaplanır:

c(i) = (p(i) + k(j)) % 26

Bu şifrenin sadece k yerine k(j) kullanımına dikkat edin. Ve k, p'den daha kısaysa, k içindeki harfler p'yi şifrelemek için döngüsel olarak tekrar tekrar kullanılmalıdır.

Başka bir deyişle eğer Vigenère birine gizlice HELLO demek isterse ve ABC anahtar kelimesiyle, bunu şu şekilde yapabilir: H'yi 0 anahtarıyla (yani A), E'yi 1 anahtarıyla(yani B), ilk L'yi 2 anahtarıyla (yani C) şifreler, bu noktadan sonra elindeki anahtar kelimenin karakterleri bittiği için tekrardan anahtar kelimenin başına dönecek ve ikinci L harfini 0 anahtarıyla (yani A) ve O'yu da 1 anahtarıyla (yani B) şifreleyecek. Ve böylece HELLO'yu HFNLP olarak şöyle yazacaktır:

düzmetin	H	E	L	L	O
+ anahtar	A	B	C	A	B
(kayma değeri)	0	1	2	0	1
= şifremetin	H	F	N	L	P

Şimdi Vigenère'in şifresini kullanarak mesajları şifrelemenizi sağlayan vigenere adlı bir program yazalım. Kullanıcı programı yürütürken, bir komut satırı argümanı sağlayarak, programın döndüreceği gizli mesaj için anahtar kelimenin ne olması gerektiğine karar vermelidir.

İşte programın nasıl çalışabileceğine dair birkaç örnek:

$ ./vigenere bacon

plaintext: Meet me at the park at eleven am

ciphertext: Negh zf av huf pcfx bt gzrwep oz

veya kullanıcı tam olarak alfabetik olmayan bir anahtar kelime sağladıysa:

$ ./vigenere 13

Usage: ./vigenere keyword

veya hiç anahtar kelime sağlamadıysa:

$ ./vigenere

Usage: ./vigenere keyword

veya çok fazla anahtar kelime sağladıysa:

$ ./vigenere bacon and eggs

Usage: ./vigenere keyword

Deneyin

Ekibimizin bu problem için yazdığı çözümü denemek için, bu sandbox içinde anahtar kelime (keyword) yerine geçerli bir alfabetik dizge kullanın.

./vigenere keyword

Nasıl başlayalım? Tanıdık bir şeyle!

Déjà vu

Daha önce de kavradığınız üzere, bu şifrenin temel mantığı Sezar'ın şifresinin altında yatan mantığa çarpıcı bir şekilde benzemektedir. Bu nedenle, Sezar kodumuz başlamak için iyi bir yer gibi görünüyor, bu yüzden "vigenere.c" nin tüm içeriğini "caesar.c" çözümünüzle değiştirerek başlamaktan çekinmeyin.

Sezar ve Vigenère şifreleri arasındaki bir fark, Vigenère'in anahtarının sayıdan ziyade bir dizi harf olmasıdır. Yani kullanıcının bize bir anahtar kelime verdiğinden emin olalım! Anahtar kelimenin her karakterinin bir rakam yerine alfabetik olduğundan emin olmak için Sezar'da uyguladığınız kontrolü değiştirin. Bunlardan herhangi biri değilse, Usage: ./vigenere keyword yazdırın ve daha önce yaptığımız gibi sıfırdan farklı bir değer döndürün. Hepsi alfabetik ise, kontrol ettikten sonra Success ve, return 0; yazdırmalısınız (şimdilik), zira şifreleme kodumuz henüz tam olarak çalışmaya hazır olmadığından çalıştırmayacağız.

Örnek davranış:

$ ./vigenere alpha

Success

ya da

$ ./vigenere 123

Usage: ./vigenere keyword

İpuçları

- String.h başlık dosyasının, string'lerle çalışan bir dizi yararlı fonksiyon içerdiğini unutmayın. Bunlardan bazıları için CS50 Referans menüsüne bakın!
- Uzunluğunu biliyorsanız, bir string'in her bir karakterini yinelemek için döngü kullanabileceğimizi hatırlayın.
- Ctype.h başlık dosyasının, karakterler hakkında bize bilgi veren bir dizi yararlı fonksiyon içerdiğini hatırlayın. Bunlardan bazıları için CS50 Referans menüsüne bakın!

Kaydırma değerini alma

Şimdilik kullanıcının tek karakterlik anahtar kelimeler sağladığını varsayalım. Bu karakteri doğru kaydırma değerine dönüştürebilir miyiz? Bunu bir fonksiyon yazarak yapalım.

Dosyanızın üst kısmına yakın, #include satırlarının altında, amacı tam olarak bunu yapmak olan yeni bir fonksiyon için prototip tanımlayalım. Girdi olarak tek bir karakter alacak ve bu karakter için kaydırma değerini verecektir.

```
int shift(char c);
```

Şimdi, giriş olarak tek bir karakter(c) alan ve bir tamsayı çıkaran shift adlı bir fonksiyon tanımladık.

Şimdi, main bloğunun altında, kendimize bu yeni fonksiyonu tanımlamak (yani uygulamak) için bir yer verelim.

```
int shift(char c)
{
  // TODO
}
```

TODO yazan yerde karakteri konumsal tam sayı değerine dönüştürme işini yapacağız (yani, yine A veya a 0, B veya b 1, Z veya z 25, vb. olacaktır)

Bunu test etmek için, "Success" yazdığınız satırı silin (ancak şimdilik "return 0"ı bırakın) ve silinen satır yerine, kodunuzun çalışıp çalışmadığını test etmek için aşağıdaki satırları ekleyin.

int key = shift(argv[1][0]);

printf("%i\n", key);

Programınız A veya a anahtar sözcüğüyle çalıştırılırsa 0 yazmalıdır. Anahtar kelime olarak programı diğer büyük ve küçük harflerle çalıştırmayı deneyin. Davranış beklediğiniz gibi mi?

İpuçları

- Fonksiyonların girdileri ve çıktıları vardır.
- Bir fonksiyon tanımladığımızda, her birinin de bir tipi olan, dönüş tipini, adını ve argüman listesinide sağlamamız gerekiyor.
- Bir fonksiyonu kullandığımızda veya çağırdığımızda, yalnızca argüman listesine uygun değerleri ekleriz ve fonksiyonun çıktısını fonksiyonun dönüş tipine karşılık gelen bir değişkene atarız.
- Argv [1] bir string ise, argv [1] [0] bu string'in sadece ilk karakteridir.
- Ctype.h başlık dosyasının, karakterler hakkında bize bilgi veren bir dizi yararlı fonksiyon içerdiğini hatırlayın.
- A'nın ASCII değeri 65'tir. a'nın ASCII değeri 97'dir.
- B'nin ASCII değeri 66'dır. b'nin ASCII değeri 98'dir. Ortaya çıkan bir potansiyel örüntüyü görüyor musunuz?

Tek karakterlik anahtar kelimeler

Daha önce yazdığınız şifreleme kodunu kullanmaya başlama zamanı! K anahtar kelimeniz tam olarak bir harf içeriyorsa (örneğin, H veya h), Vigenère'in şifresinin etkili bir şekilde bir Sezar şifresi haline geldiğini fark etmiş olabilirsiniz (bu 7. örnekte). Şimdilik gerçekten kullanıcının anahtar kelimesinin tek bir harf olacağını varsayalım. Sağlanan harfin kaydırma değerini hesaplamak için yeni yazdığınız "shift" fonksiyonunu kullanın, bu fonksiyonun dönüş değerini tam sayı değişkeni "anahtara"a atayın ve "anahtarı"ı tam olarak Sezar'ın şifresinde yaptığınız gibi kullanın! Aslında şimdi yeni eklediğiniz printf ve return 0; satırlarını silmeniz, programın daha önce yazdığınız Sezar şifreleme kodu gibi çalışması için yeterli olacaktır!

$ ./vigenere A

plaintext: hello

ciphertext: hello

ya da

$ ./vigenere b

plaintext: HELLO

ciphertext: IFMMP

ya da

$ ./vigenere C

plaintext: HeLlO

ciphertext: JgNnQ

İpuçları

Sezar çözümünüzdeki bazı değişkenleriniz bu laboratuvarda şu ana kadar adlandırılanlarla eşleşmiyorsa, eşleşmeleri için adlarını düzenleyin!

Son Adımlar

Şimdi "vigenere.c" de kalan işlevselliği yazarak işleri bitiş çizgisine götürme sırası sizde. Kullanıcının anahtar kelimesinin muhtemelen birden çok harften oluşacağını unutmayın, bu nedenle düz metnin her harfi için yeni bir kaydırma değeri hesaplamanız gerekebilir; bunu yaptıktan sonra "shift" fonksiyonunuzu döngünüzde kullanmak isteyebilirsiniz.

Ayrıca, bir karakteri her şifrelediğinizde, anahtar kelime olan k'deki bir sonraki harfe, (ve tüm karakterlerini tüketirseniz anahtar kelimenin başına sarılır) ihtiyacınız olduğunu unutmayın. Ancak bir karakteri (ör. Boşluk veya noktalama işareti) şifrelemiyorsanız, bir sonraki k karakterine ilerlemeyin!

Ve daha önce olduğu gibi, büyük/küçük harfleri koruduğunuzdan emin olun, ama bunu sadece orijinal mesajın durumuna göre yapın. Anahtar kelimedeki bir harfin büyük harfle yazılıp yazılmadığının, şifreli metindeki bir harfin değişip değişmemesi üzerinde hiçbir etkisi olmamalıdır!

İpuçları

- Muhtemelen iki sayaca ihtiyacınız olacak, bir tane düz metin üzerine yineleme yapmak için i; ve bir tane de , anahtar kelime üzerinde yineleme yapmak için j.
- Düz metin üzerinde yineleme yapmak için kullandığınız for döngüsüne güvenmek yerine, anahtar kelime sayacını kendiniz kontrol etmenin kolay bir yolunu bulacaksınız!
- Anahtar kelimenin uzunluğu 4 karakter ise, o anahtar kelimenin son karakteri keyword[3]'te bulunabilir. Ardından, şifrelediğiniz bir sonraki karakter için keyword[0]'ı kullanmak isteyeceksiniz.

"Crack" Görevi Tanımı | Problem Seti 2 | CS50x Ders yazılımları

5-6 dakika

"Crack" Görevi Tanımı

Video

HATA: Oynatılabilir video kaynağı bulunamadı!

Web tarayıcınız bu video biçimini desteklemiyor. Lütfen farklı bir tarayıcı ile deneyin.

Bu videoyu sessize almak veya sesini açmak için bu düğmeye tıklayın, ses seviyesini artırmak ya da azaltmak için YUKARI veya AŞAĞI düğmelerine basın.

Sessizde Ses Şiddeti.

Öncelikle CS50 Sandbox'ta bu sayfayı açın ve aşağıdakileri takip edin:

Shadow Dosyası Bilir

Bugünlerde Linux çalıştıran sistemlerin çoğunda kullanıcı adları ve parolaları içeren /etc/shadow adlı bir dosya bulunmaktadır. Neyse ki, buradaki parolalar "açıkta" saklanmaz, bunun yerine "tek yönlü hash fonksiyonu" kullanılarak şifrelenir. Bir kullanıcı, kullanıcı adı ve parola yazarak bu sistemlere giriş yapmayı denediğinde, yazdığı parola aynı hash fonksiyonu ile şifrelenir ve sonuç, kullanıcı adının /etc/shadow içerisindeki girdiyle karşılaştırılır. Eğer iki hash eşleşirse, kullanıcının içeri girmesine izin verilir. Parolanızı daha önce unuttuysanız, teknik destek parolanızı bulamayacağını, ancak sizin için değiştirebileceğini söylemiş olabilir. Muhtemelen bunun nedeni, teknik desteğin parolanızın kendisini değil, yalnızca parolanızın bir hash'ini görebilmesidir. Ama sizin için yeni bir hash yaratabilirler.

etc/shadow içindeki parolalar hash'li olsa da, hash fonksiyonu her zaman güvenli olamayabilir. Çoğu zaman kötü niyetli kullanıcılar, bu dosyayı bir şekilde elde ettikten sonra, kullanıcıların parolalarını tahmin edebilir (ve kontrol edebilir) veya kaba kuvvet kullanarak (yani, tüm olası parolaları deneyerek) kırabilirler. Aşağıda, basitleştirilmiş olsa da, /etc/shadow dosyasının nasıl görünebileceği verilmiştir, burada her satır kullanıcıadı:hash olarak biçimlendirilmiştir.

brian:51.xJagtPnb6s

bjbrown:50GApilQSG3E2

emc:502sDZxA/ybHs

greg:50C6B0oz0HWzo

jana:50WUNAFdX/yjA

lloyd:50n0AAUD.pL8g

malan:50CcfIk1QrPr6

natmelo:50JIIyhDORqMU

rob:51v3Nh6ZWGHOQ

veronica:61v1CDwwP95bY

walker:508ny6Rw0aRio

zamyla:50cI2vYkF0YU2

Şifrekıran

Göreviniz parolaları kıran bir program, crack, tasarlamak ve uygulamaktır. Bu konuda çok fazla ipucu vermeyeceğiz, ancak başlamak için bu laboratuvar ortamında olduğu gibi Unix/Linux sistemlerinde *crypt* işlevinin nasıl çalıştığını okumak isteyebilirsiniz. Bunu yapmak için yukarıda linkini verdiğimiz Crack CS50 Lab terminalinde şunu yazın:

man crypt

Bu programın “salt”dan bahsetmesine özellikle dikkat edin.

Çözümünüzde kullanmak üzere "crypt" fonksiyonunu kullanabilmek için,

#include <crypt.h>

dosyanızın üst kısmına yazın. Programınızı nasıl organize etmeniz gerektiği konusunda fikir edinmek için pseudocode.txt dosyasını not defteri olarak kullanın!

Tanımlamalar

- Programınız yalnızca bir komut satırı argümanı kabul etmelidir: hashlenmiş parola.
- Programınız herhangi bir komut satırı argümanı olmadan veya birden fazla komut satırı argümanı ile yürütüldüyse, programınız bir hata (sizin seçtiğiniz) yazdırmalı ve hemen main geriye 1 döndürmelidir. (böylece bir hata belirtmiş oluruz).

- Aksi takdirde, programınız verilen parolayı kırmaya devam etmeli, ideal olarak mümkün olan en kısa sürede bunu yapmalı, sonra parolayı net olarak sonunda \n olacak şekilde yazdırmalı ve ardından da main 0 döndürmelidir.
- Her bir parolanın C'nin DES tabanlı (MD5 tabanlı değil) crypt fonksiyonuyla hashlendiğini varsayın.
- Her bir parolanın beş (5) karakterden uzun olmadığını varsayalım. Pfff!
- Her parolanın tamamen alfabetik karakterlerden (büyük ve / veya küçük) oluştuğunu varsayalım.

Aşağıda bazı örnek davranışlar verilmiştir.

$ ./crack

Usage: ./crack hash

$ ./crack 50cI2vYkF0YU2

LOL

İpuçları

- Argc ve argv'nin komut satırına girilenler hakkında bilgi verdiğini hatırlayın.
- Bir string'in sadece bir karakter dizisi (char) olduğunu hatırlayın.
- Köşeli parantez ([]) kullanarak bir dizinin öğelerine ayrı ayrı erişebileceğimizi hatırlayın.
- Salt'ın, hash'in ilk iki karakteri olduğunu hatırlayın.
- Bazen insanların gerçek kelimeler olan parolalar kullandıklarını hatırlayın. Belki de yapılabilecek bir optimizasyon vardır?
- Kaba kuvvet algoritmaları, algoritmaların en hızlısı değildir ve bu bir sorun değil! Kısa parolaların kırılmasının genellikle daha uzun parolalardan daha kolay olduğunu unutmayın.