Hash Uzunluk Genişletme Saldırısı

written by Mert SARICA | 11 May 2015

Sızma testi uzmanları ve kripto analistler tarafından Merkle–Damgård hash fonksiyonlarının (MD5, SHA-1, SHA-256, SHA-512 vb.) uzunluk genişletme (length extension) saldırısından doğası gereği etkilendiği bilinmektedir. Bu hash fonksiyonlarının MAC (mesaj doğrulama kodu) olarak kullanıldığı durumlarda ortaya çıkan zafiyet kötüye kullanılarak gizli anahtar (secret key) bilinmeden geçerli bir MAC oluşturulabilmektedir.

Bir örnek ile kısaca açıklamak gerekirse, diyelim ki yazılımcının biri aşağıdaki gibi bir web uygulaması hazırlamış olsun. (Pi Hediyem Var #2 Hacking Oyunu)

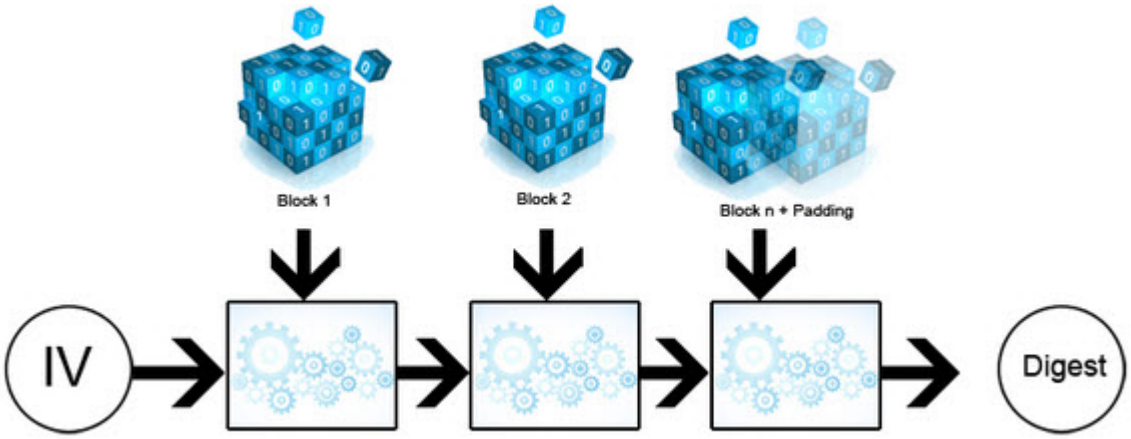
Hack 4 Career - Siber Güvenlik Blogu

```
< ?php
$username = 'misafir';
$secret = 'H4ck4C4r33r';
$pos = '';

if(isset($_GET['username'])){
    $username = $_GET['username'];
    $username = strtolower($username);
}

if(isset($_GET['hash'])){
  if (preg_match('#[0-9a-f]{32}#i', $_GET['hash'])) {
    $hash = $_GET['hash'];
```

```
    $hash = strtolower($hash);
  }
}

// Debug
//print "
" . (hash("md5",$secret.$username));
//exit;

if(isset($username) and isset($hash)){
//  print "
" . $username;
//  print "
" . $secret;
  if(hash("md5",$secret.$username) == $hash){
    $pos = strpos($username, "admin");
    if ($pos !== false) {
      print "Tebrikler $username , artık en yüksek yetkiye sahipsin :)";
      print "

Pi Hediyem Var çekilişine katılmak için bu ekran görüntüsünü ve çözüm yolunu
Mert SARICA ile paylaşabilirsin";
    } else {
      print "Merhaba $username , hala sefil kullanıcı yetkisine sahipsin :(";
      print "

Raspberry Pi 2 çekilişine katılabilmek için admin yetkisi ile giriş
yapabilmen lazım!";
      print "
Referans: https://www.mertsarica.com/pi-hediyem-var-2/";
    }
  } else {
    $username = 'misafir';
    $loc = "Location: " . "https://www.mertsarica.com/ctf/ctf2.php?" .
"username=" . $username . "&hash=" . hash("md5",$secret.$username);
    header($loc);
    exit;
  }
} else {
```

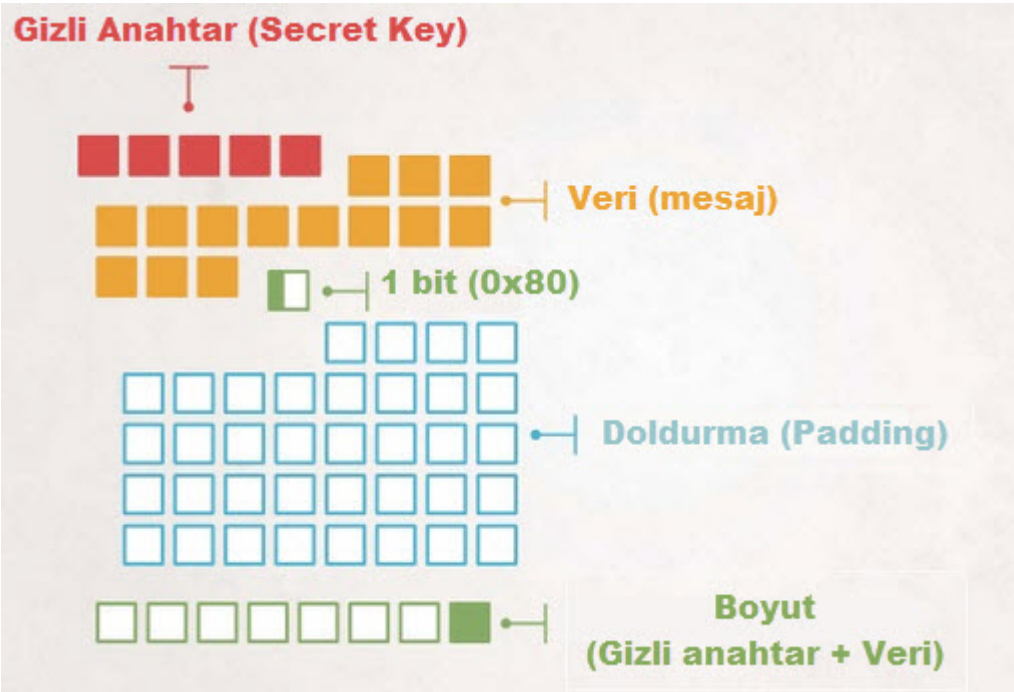
```
    $username = 'misafir';
    $loc = "Location: " . "https://www.mertsarica.com/ctf/ctf2.php?" .
"username=" . $username . "&hash=" . hash("md5",$secret.$username);
    header($loc);
    exit;
}
?>
```

Bu uygulama, doğrulama adımından geçen (geçtiyse örnek kullanıcımız mert olsun) ve geçmeyen her bir kullanıcıya (geçmediyse örnek kullanıcımız misafir olsun), o kullanıcıya özel olarak oluşturulan bir $hash değeri atıyor. Bu $hash değerini de, MD5(kullanıcı adı ($username) + sunucu tarafında oluşturulan bir anahtar ($secret)) ile yani hash("md5",$secret.$username) şeklinde oluşturuyor. Kullanıcı, doğrulama adımından geçtikten veya geçmedikten sonra uygulama üzerinde gerçekleştirdiği her işlem için sunucuya artık MAC olarak bu $hash değerini gönderiyor. Bu sayede web uygulaması, gelen her $hash değerini, anlık olarak oluşturduğu hash("md5",$secret.$username) değeri ile kıyaslayarak (hash("md5",$secret.$username) == $hash) kullanıcının yetkili (mert) veya yetkisiz bir kişi (misafir) olup olmadığını anlıyor. Eğer kullanıcı adında (username=) admin kelimesi geçiyor ve anlık oluşturulan MAC ile kullanıcıdan gelen $hash birbiri ile tutuyorsa kullanıcıyı yönetici (admin) sayfasına yönlendiriyor. Alfanümerik, 11 hane uzunluğunda olan $secret değeri de uygulama sunucusunun kaynak kodunda tanımlı olduğu için ($secret = 'H4ck4C4r33r';), art niyetli bir kişinin deneme yanılma saldırısı (brute-force) ile admin kullanıcısının $hash değerini bulması ve admin sayfasına erişmesi pratikte yıllar sürecektir diye düşünüyoruz, ancak yanılıyoruz!



Neden yanılıyoruz çünkü yukarıdaki resimden de anlaşılacağı üzere örnek olarak bir verinin MD5 hashi üretilirken, veri 512 bitlik bloklar halinde işlendikten sonra hash değeri üretilmektedir. Bir blok en fazla 64 bayt uzunluğunda olabilir ve eğer son blok, 56 bayttan küçük ise doldurma (padding) işlemi yapılır. Son bloğun son 8 baytı, blokların boyutunu tanımlamak için kullanılır. İlk blok, IV (initialization vector) ile işleme alınır. Diğer bloklar ise bir önceki bloğun çıktısı ile işleme devam eder. Hash işleminin devam etmesi için bir önceki bloğun çıktısını işleme almak yeterlidir. (length extension)

Kabaca elimizde iki ayrı veri var diyelim, biri ABC diğeri ise AB. Her bir harfin bir blok olduğunu düşünelim. ABC için gerçekleştirilen hashleme işleminde önce A işlenir, çıktısı B'ye girdi olarak iletilir sonra B işlenir, çıktısı C'ye girdi olarak iletilir ve C son blok olduğu için eğer 56 bayttan küçük ise doldurma işlemi (padding) gerçekleştirilir. Ardından A,B ve C bloğunun büyüklüğü toplanarak bit olarak son 8 bayta little-endian olarak yazılır ve işlenerek ortaya bir hash değeri çıkar. AB verisinin hash çıktısına C verisi eklenerek ABC verisinin hash değeri üretilebilir. Bu durum da hash genişletme (length extension) zafiyetine yol açmakta ve kötüye kullanılabilmektedir.



Bu zafiyetin nasıl kötüye kullanılabileceğine kısaca göz atalım;

Web uygulamasının kaynak kodunda aşağıdaki gibi hatalı bir şekilde hash fonksiyonu ile MAC üretildiği ve ardından strpos fonksiyonu ile kullanıcı

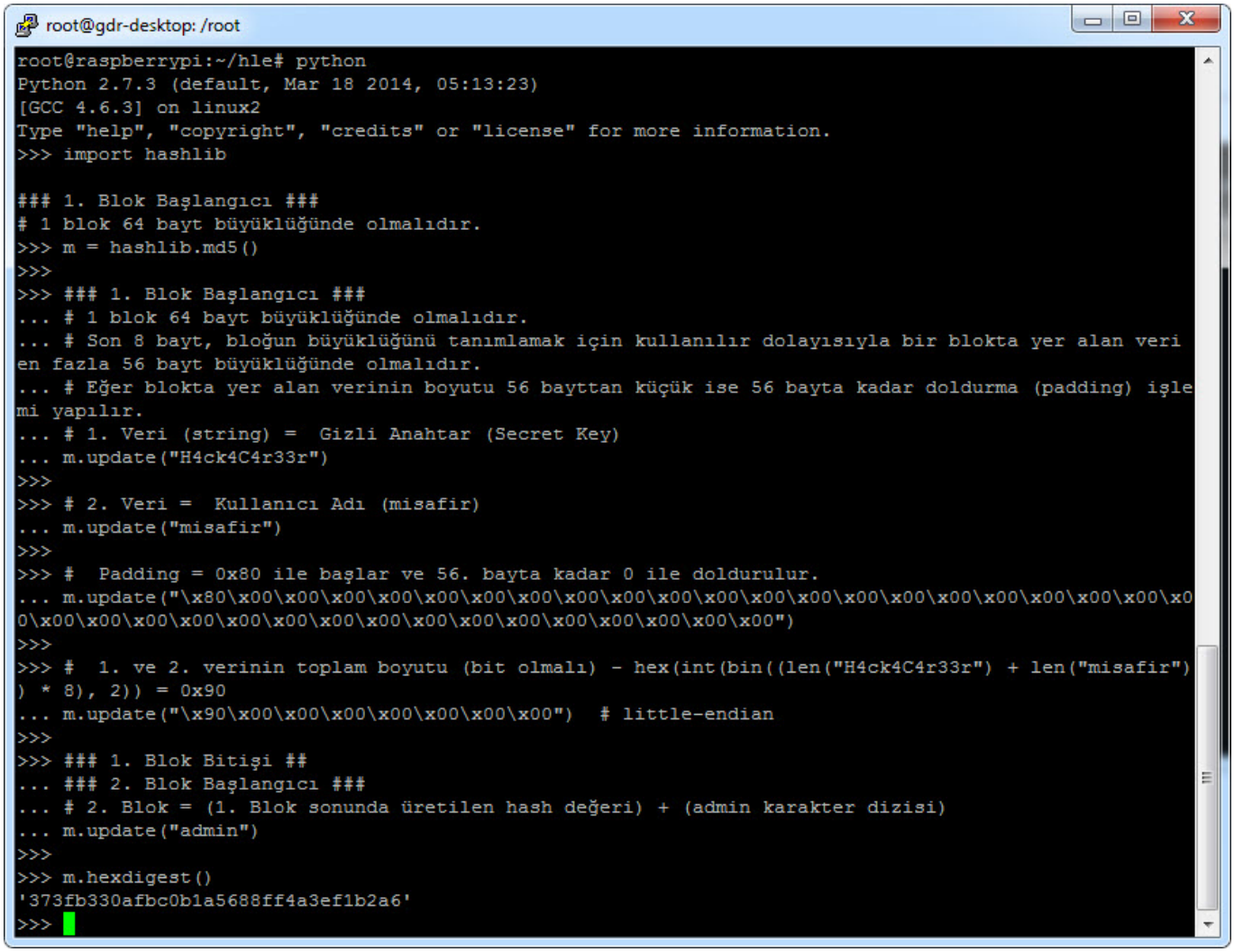
adında admin karakter dizisi geçiyor ise bu kullanıcıya yönetici yetkisi verildiği görülmektedir.

```
  ...
  if(hash("md5",$secret.$username) == $hash){
    $pos = strpos($username, "admin");
    if ($pos !== false) {
      print "Tebrikler $username , artık en yüksek yetkiye sahipsin :)";
  ...
```

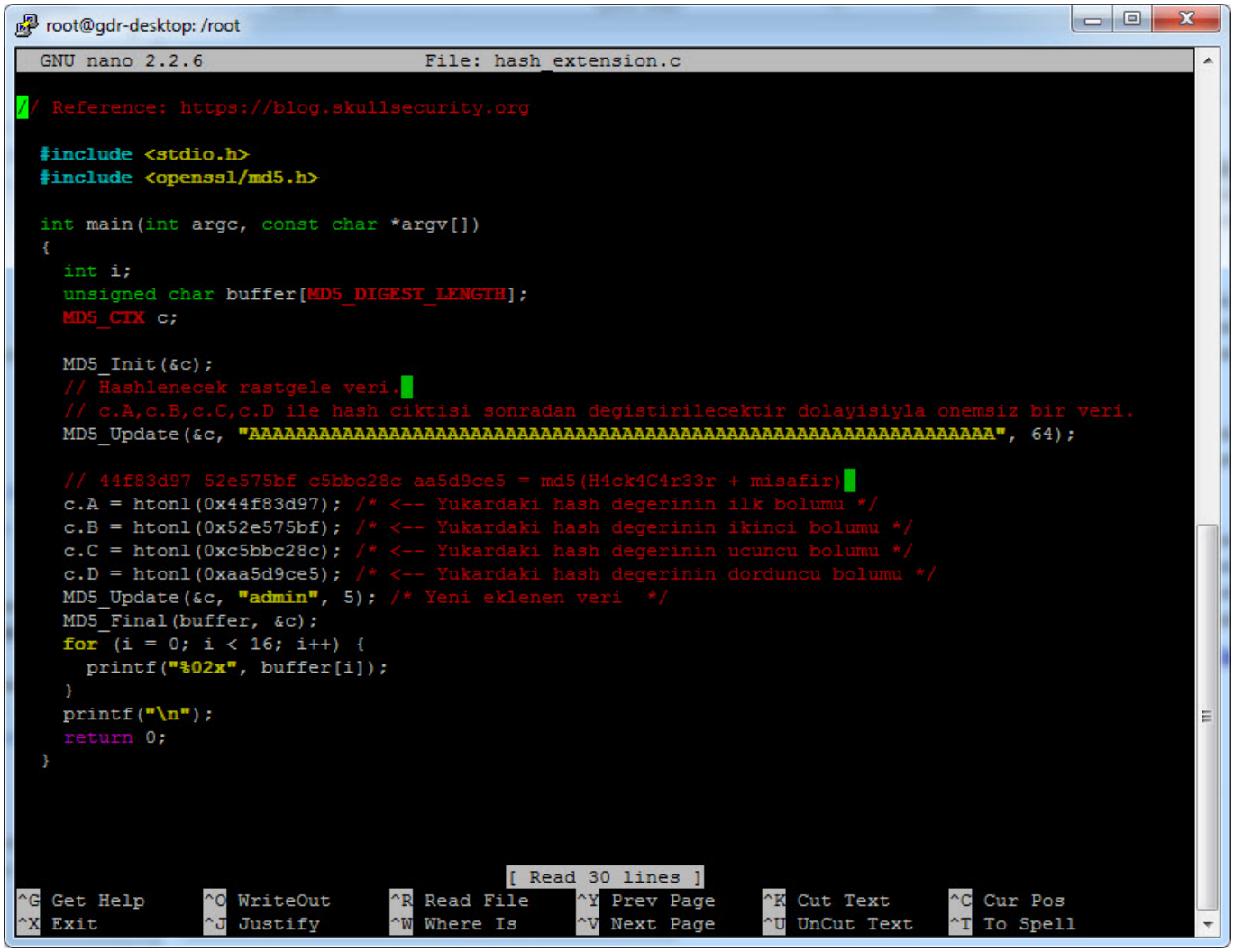
Bu web uygulaması sayfayı ziyaret eden herhangi bir kullanıcıyı, sayfayı ziyaret ettiği zaman otomatik olarak misafir yetkisine (kullanici=misafir) atamakta ve bu kullanıcıya ait $hash değerini kullanıcıya göndermektedir. (hash=44f83d9752e575bfc5bbc28caa5d9ce5)

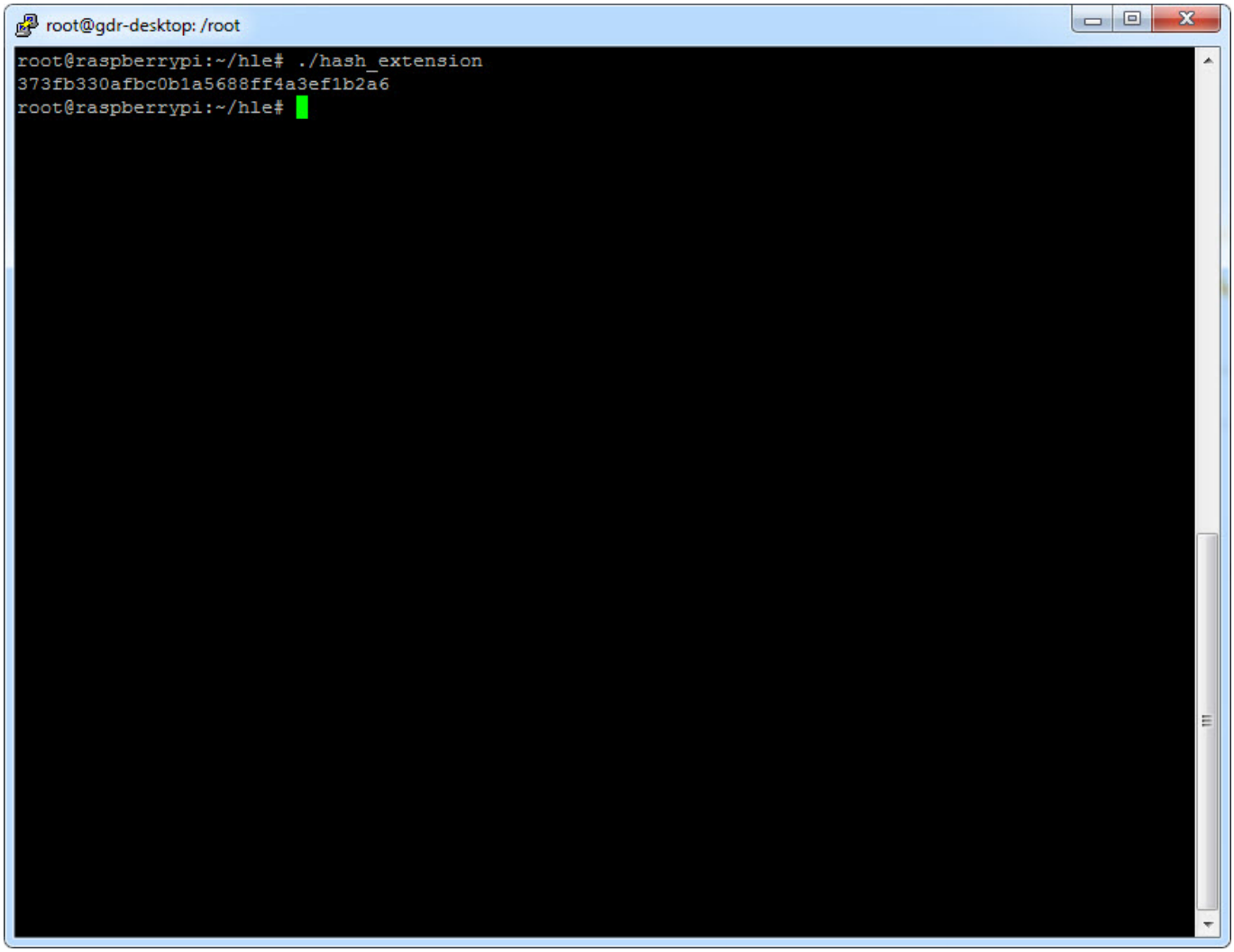
$username değeri bizim kontrolümüzde olduğuna göre hash kontrolünü atlatmak için $secret değerini bilmeden yukarıda anlatmış olduklarıma göre şunu yapabiliriz. $secret değeri ile $username değeri birbirine eklendiğine (concatenate) göre bunun toplamını 64 bayta tamamlayarak üzerine admin karakter dizisini ekleriz ve ardından bizden beklenen $hash değerini de web uygulamasına göndererek bu kontrolü başarıyla atlatabiliriz.

username olarak kullanılacak olan misafir + padding + boyut + admin değerinin yeni $hash değerini öğrenmek için misafir kullanıcısının $hash değeri olan 44f83d9752e575bfc5bbc28caa5d9ce5 değerinden faydalanabiliriz. Yukarıda da belirttiğim üzere ilk blokta kullanılmak üzere bu hash değerini başlangıç değeri olarak (internal state) kullanarak bu değerin üzerine padding + (misafir + admin)‘in uzunluğunu ekleyerek birinci bloğun işlemini tamamlayabiliriz. Ardından ikinci bloğun başlangıç değeri olarak admin karakter dizisini kullanarak yeni $hash değerini (373fb330afbc0b1a5688ff4a3ef1b2a6) öğrenebiliriz. Normalde $secret değerini biliyor olsaydık bunu aşağıdaki gibi Python ile kolaylıkla öğrenebilirdik.

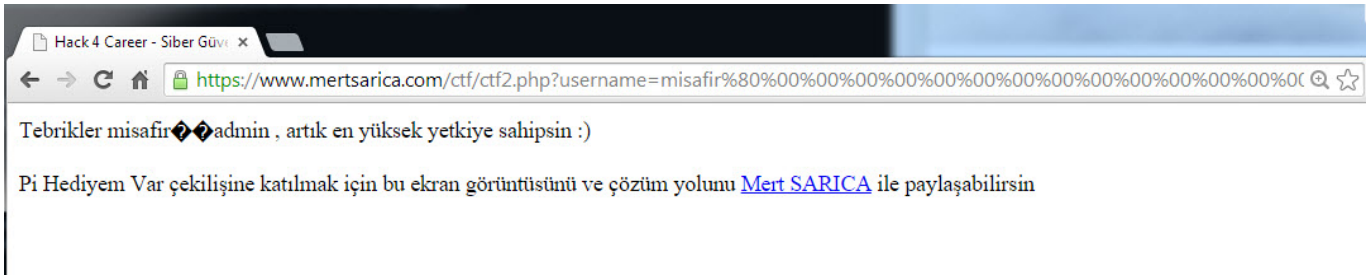


Aslında $secret değerini bilmemize gerek yok çünkü bildiğimiz bir verinin hash değerine, yeni bir veri ekleyerek, yeni oluşan $hash değerini (373fb330afbc0b1a5688ff4a3ef1b2a6) rahatlıkla öğrenebiliriz.


root@gdr-desktop: /root
GNU nano 2.2.6 File: hash_extension.c
// Reference: https://blog.skullsecurity.org
#include <stdio.h>
#include <openssl/md5.h>
int main(int argc, const char *argv[])
{
int i;
unsigned char buffer[MD5_DIGEST_LENGTH];
MD5_CTX c;
MD5_Init(&c);
// Hashlenecek rastgele veri.
// c.A,c.B,c.C,c.D ile hash ciktisi sonradan degistirilecektir dolayisiyla onemsiz bir veri.
MD5_Update(&c, "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA", 64);
// 44f83d97 52e575bf c5bbc28c aa5d9ce5 = md5(H4ck4C4r33r + misafir)
c.A = htonl(0x44f83d97); /* <-- Yukardaki hash degerinin ilk bolumu */
c.B = htonl(0x52e575bf); /* <-- Yukardaki hash degerinin ikinci bolumu */
c.C = htonl(0xc5bbc28c); /* <-- Yukardaki hash degerinin ucuncu bolumu */
c.D = htonl(0xaa5d9ce5); /* <-- Yukardaki hash degerinin dorduncu bolumu */
MD5_Update(&c, "admin", 5); /* Yeni eklenen veri */
MD5_Final(buffer, &c);
for (i = 0; i < 16; i++) {
printf("%02x", buffer[i]);
}
printf("\n");
return 0;
}
[Read 30 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell



Görüldüğü üzere misafir + admin için yeni $hash değerini $secret değerini bilmeden öğrenebildik. Şimdi sıra $username parametresinde 373fb330afbc0b1a5688ff4a3ef1b2a6 hash değerini üretecek parametreleri oluşmaya geldiğinde şunu yapacağız. $secret değerini bilmiyoruz ancak ilgili $hash değerini üretmek için bilmemize gerek olmadığını artık biliyoruz. Burada sadece iki değerin toplam boyutunu tahmin etmemiz gerekiyor. Bunun için de hash_extender aracından faydalanabiliriz. Bu araca sunucunun bize ilk atadağı/gönderdiği $hash değerini -s parametresi ile, (44f83d9752e575bfc5bbc28caa5d9ce5), hash değerine ait veriyi (yani misafir) ise -d parametresi ile, eklenecek veriyi ise (yani admin) -a ile ve toplum boyutu deneme yanılma ile tespit edebilme adına min ve max parametrelerini vererek kullanabiliyoruz. Program çıktısındaki her bir değeri (bu arada yeni üretilen hash değerinin (373fb330afbc0b1a5688ff4a3ef1b2a6) de bizimki ile aynı olduğunu görebiliyoruz) teker teker uygulama üzerinde denediğimiz zaman $secret değerinin 11 hane uzunluğunda olduğunu ve MAC olarak kullanılan $hash değerini başarıyla tespit edip, kontrolü atlatabildiğimizi görüyoruz.



Yazılım geliştiricisiyseniz, hash uzunluk genişletme (hash length extension attack) zafiyetine karşı kendi MAC kontrolünüzü (custom) oluşturmak yerine HMAC kullanabilirsiniz.

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.