破密分析-從古典密碼學到現代密碼學

資訊安全人才培育計畫 | Hacking Weekend | MyFirstCTF Training

HASH 攻擊

By 外星人

Security Mentors

臺灣好厲駭 讓你更厲害

資安實務導師培訓計畫

HASH

單向雜湊函數

HASH 單向雜湊函數

訊息摘要函數message digest function
密碼雜湊函數cryptographic hash function

原像pre-image

Message
訊息

單向雜湊函數

Hash Value
雜湊值

訊息摘要message digest

無法從雜湊值反算出訊息
所以叫單向

只有單向

任意長度的訊息計算後得到
固定長度的雜湊值

http://www.unixwiz.net/techtips/iguide-

著名的Hash function

[這些演算法及其不同程式的實作(python,c/c++,ruby,...) ➔上大學在學]

MD4	✓ Rivest(1990) | 雜湊值的長度為128 位元(RFC 1186 ，修改版RFC 1320) ✓ Dobbeertin 發現了MD4 雜湊值的碰撞方法，所以並不安全。
MD5	✓ Rivest (1991) |雜湊值的長度為128 位元(RFC l321) ✓ MD5 的強碰撞抵抗性已經被破解 MD ==Message Digest 訊息摘要
SHA-1	◘ NIST (National Institute of Standards and Technology) ◘ 雜湊值的長度為160 位元 ◘ 1993 年美國發表 FIPS PUB 180 稱為SHA ◘ 1995 年發表的修改版FIPS PUB 180-1 稱作SHA-1 ◘ 2005 年SHA-I 的強碰撞抵抗性被破解 SHA ==Secure Hash Algorithm
SHA-2	◆ NIST (National Institute of Standards and Technology) ◆ SHA-256 、SHA-384 、SHA-512 雜湊值的長度分別是256 位元、384 位元、512 位元。這些單向雜湊函數統稱為SHA-2 ◆ 訊息的長度有限制(SHA-256 是不超過264 位元， SHA-384 與SHA-512 是不超過2128 位元) ◆ 這些SHA-2單向雜湊函數與SHA-1 公開為FIPS PUB 180-2(2002年) https://zh.wikipedia.org/wiki/SHA-2
RIPEMD-160	European Union PIPE 計畫設計出的RIPEMD RIPEMD-160是RIPEMD修訂版,雜湊值長度為160 位元 Hans Dobbertin 、Antoon Bosselaers 、Bart Preneel(1996) 還有RIPEMD-128 、RIPEMD-256 、RIPEMD-320 等版本 RIPE MD 的強碰撞抵抗性在2004 年被破解，但是RIPEMD-160 還未被破解 比特幣使用RIPEMD-160
SHA-3	NIST (National Institute of Standards and Technology) 2007|2012(KECCAK 演算法) SHA-3 在2015年8月5日由 NIST 通過 FIPS 202 正式發表 https://zh.wikipedia.org/wiki/SHA-3

https://www.openfoundry.org/en/tech-column/8608-opensource-security-algorithm-tool-openssl1-basic-function-and-hash-algorithms-

Hashing with openssl

OpenSSL 支援常見的雜湊演算法 (Hash algorithms)，如 MD5, SHA1, SHA256 等。

1.列出所有支援的雜湊演算法➔openssl dgst -help

```
-md5            to use the md5 message digest algorithm (default)
-md4            to use the md4 message digest algorithm
-md2            to use the md2 message digest algorithm
-sha1           to use the sha1 message digest algorithm
-sha            to use the sha message digest algorithm
-sha224         to use the sha224 message digest algorithm
-sha256         to use the sha256 message digest algorithm
-sha384         to use the sha384 message digest algorithm
-sha512         to use the sha512 message digest algorithm
-mdc2           to use the mdc2 message digest algorithm
-ripemd160      to use the ripemd160 message digest algorithm
```

https://www.openfoundry.org/en/tech-column/8608-opensource-security-algorithm-tool-openssl1-basic-function-and-hash-algorithms-

Hashing with openssl

2.計算與驗證檔案的md5 hash

echo 'HappyHackingDay' > test.txt

openssl dgst -md5 -c test.txt　　計算檔案的md5 hash
➔MD5(test.txt)= 79:50:86:fd:74:d8:48:fe:7b:de:d8:77:10:a1:20:b4

md5sum test.txt　　驗證檔案的md5 hash
➔795086fd74d848fe7bded87710a120b4 test.txt

https://www.openfoundry.org/en/tech-column/8608-opensource-security-algorithm-tool-openssl1-basic-function-and-hash-algorithms-

Hashing with openssl

3.計算與驗證檔案的SHA1 hash

openssl dgst -sha1 -c test.txt　　計算檔案的SHA1 hash

➔SHA1(test.txt)= 64:56:39:fe:1d:f6:40:25:c9:1e:69:43:3b:d5:aa:b3:b2:87:2b:82

shasum test.txt　　驗證檔案的SHA1 hash

➔ 645639fe1df64025c91e69433bd5aab3b2872b82 test.txt

Hashing with python

每一個程式語言(Python/ruby)幾乎都有Hash運算的相關模組

使用Hashlib模組 https://docs.python.org/2/library/hashlib.html

```
import hashlib

a = 'HappyCTFP{It is FUN hashing with python hashlib}'

print hashlib.md5(a).hexdigest()
print hashlib.sha1(a).hexdigest()
print hashlib.sha224(a).hexdigest()
print hashlib.sha256(a).hexdigest()
print hashlib.sha384(a).hexdigest()
print hashlib.sha512(a).hexdigest()
```

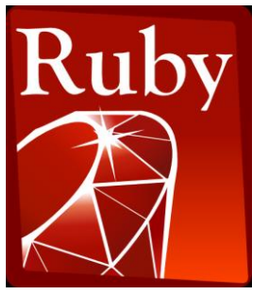
2df8404191835aef870ee32ae6870317

fb2f51e7f2973bfed59218a1c446b9571ff3ac9d

cf526656fe98363230e603a8f975c623a288e183839103041392c9e1

bc395528e192147f03915733e47b89f8afe632404a559132a2236502b1afba71

Hashing with ruby



gedit ruby_hash.rb

chmod +x ruby_hash.rb

./ruby_hash.rb

```
#!/usr/bin/ruby -w

require 'digest'

puts Digest::SHA256.hexdigest "Hello World"
puts Digest::SHA256.hexdigest "Hello MWorld"
puts Digest::MD5.hexdigest "Hello World"
puts Digest::MD5.hexdigest "Hello MWorld"
```

只差一個字母,hash後完全不同

```
a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b57b277d9ad9f146e
dad0dd87b309c3d8b364541f2a69bdb347832ccec172ac8ebfa0f02945d9172d
b10a8db164e0754105b7a99be72e3fe5
2ad6f329c9f3a968941a38affa0af2fe
```