MOZILLA PASSWORD-BASED ENCRYPTION

How passwords are stored/protected in Mozilla products

CC BY NC SA

v1.02
Laurent Clévy
@lorenzo2472
https://github.com/lclevy/firepwd

```
$ file key3.db
key3.db: Berkeley DB 1.85 (Hash, version 2, native byte-order)
```

KEY3.DB FILE CONTENT

```
0x000000:  00061561 00000002 000004d2 00001000
...
0x000030:  00000001 00010000 00000004 00000001
...
```

PAGE #0

```
0x001000:  0600f90f f80fea0f b60fab0f 970f850f
0x001010:  970fffff ffffffff ffffffff ffffffff
...

0x001f80:  ffffffff ffffffff ffffffff ffffffff
0x001f90:  ffffffff ffffff31 5aa1088a 0c1fba32
0x001fa0:  cf613eec e884a02a 9ac0cd67 6c6f6261
0x001fb0:  6c2d7361 6c740314 01ad6f10 16c8bf0b
0x001fc0:  eeeeeee3 add1bf05 c275da42 fd000b2a
0x001fd0:  864886f7 0d010c05 0103a989 f3015f50
0x001fe0:  87fecae9 8435ad09 fc737061 7373776f
0x001ff0:  72642d63 6865636b 03566572 73696f6e
```

PAGE #1

```
0x002000:  0200f00f 5d0f530f 5d0f0000 00000000
...

0x002f50:  00000000 00000000 00000000 00030001
0x002f60:  0030818c 3028060b 2a864886 f70d010c
0x002f70:  05010330 190414a8 db682ac5 1cfad8c0
0x002f80:  6664fe9d eb528307 3b33ee02 01010460
0x002f90:  72d56360 49d4af9e eadaf7eb 0dc1710a
0x002fa0:  62d5362f e4086dcc 0495e5ec 8e96c23c
0x002fb0:  56b72a55 2e177561 41ae8085 4d6fd03e
0x002fc0:  cdc2c8f8 3d2c02d4 c3f36e7e 2b906f2c
0x002fd0:  70a8cf57 1a06666e 53f24178 0f9e3981
0x002fe0:  5e7d840e 97e43461 4ac20ec0 9002e861
0x002ff0:  f8000000 00000000 00000000 00000001
```

```
0x003000:  fdffffff ffffffff ffffffff ffffffff
```

BERKELEY DB HEADER

```
magic     = 0x00061561 (Berkeley DB)
version   = 00000002 (1.85)
page size = 0x1000 bytes
records   = 4
```

INDEXES

key/value pairs in current page: 6 values (3 times key/value)
0xff9/0xff8, 0xfea/0xfb6, 0xfab/ 0xf97

RECORDS

```
0xfab: 676c6f62616c 2d 73616c74 == "global-salt"
0xf97: 315aa1088a0c1fba32cf613eece884a02a9ac0cd

0xfea: 73706173737 76f7264 2d 636865636b == "password-check"
0xfb6: 03 14 01  ad6f1016c8bf0beeeeeee3add1bf05c275da42fd
       00 0b 2a864886f70d010c050103  # object ID, len = 0x0b
       a989f3015f5087fecae98435ad09fc

0xff9: 56657273696f6e == "Version"
0xff8: 3
```

INDEXES

key/value pairs in current page: 2 values (1 time key/value)
0xff0, 0xf5d

RECORDS

```
0xff0: f8000000 00000000 00000000 00000001,
0xf5d: 03 00 01 00 30818c 3028060b 2a864886f70d010c050103
       3019 0414 a8db682ac51cfad8c06664fe9deb5283073b33ee
       02 0101
       0460 72d5636049d4af9eeadaf7eb0dc1710a62d5362fe4086dcc
       0495e5ec8e96c23c56b72a552e17756141ae80854d6fd03ecdc2c8f8
       3d2c02d4c3f36e7e2b906f2c70a8cf571a06666e53f241780f9e3981
       5e7d840e97e434614ac20ec09002e861
```

ASN1 decoding

```
ASN1 encoding: type, length, value

ASN1 types:
    2 = INTEGER
    4 = OCTETSTRING
    6 = OBJECTIDENTIFIER
 0x30 = SEQUENCE

Example:
020100 = INTEGER, len = 1, value = 00
```

Step #1: verify Master Password

(clearText must be "password-check\x02\x02")

Password Based Decryption

```
HP = SHA1( global-salt | MasterPassword )
CHP = SHA1( HP | ES )
k1 = HMAC-SHA1( key=CHP, msg= (PES | ES) )
tk = HMAC-SHA1( CHP, PES )
k2 = HMAC-SHA1( CHP, tk | ES )
k = k1 | k2
key = k[:24] # first 24 bytes (EDE keying, also called 3TDEA)
iv = k[-8:]  # last 8 bytes
clearText = DES3.new( key, DES3.MODE_CBC, iv ).decrypt( cipherText )

 notes:
  "|" means concatenation, HP=HashedPassword, ES=EntrySalt,
  CHP=Combined HP, PES=Padded Entry Salt (to 20 bytes with 0)
 reference: http://www.drh-consultancy.demon.co.uk/key3.html
```

global-salt

entry-salt
length=0x14 (20)

cipherText

from user:
MasterPassword = "MISC*"

global-salt

Step #2: decrypt Master Key

```
SEQUENCE { # length = 143 (encoded 0x818c)
  SEQUENCE { # length = 0x28 (40)
    OBJECTIDENTIFIER 1.2.840.113549.1.12.5.1.3   # PKCS#5 pbeWithSha1AndTripleDES-CBC
    SEQUENCE { # like MacData from PKCS#12v1. length = 0x19 (25)
      OCTETSTRING a8db682ac51cfad8c06664fe9deb5283073b33ee # used as entry-salt
      INTEGER 1 # iteration count
    }
  }
  OCTETSTRING 72d5636049d4af9eeadaf7eb0dc1710a62d5362fe4086dcc0495e5ec8e96c23c56b72a55
  2e17756141ae80854d6fd03ecdc2c8f83d2c02d4c3f36e7e2b906f2c70a8cf571a06666e53f241780f9e
  39815e7d840e97e434614ac20ec09002e861
}
```

entry-salt

cipherText

Password Based Decryption

```
key = aefa8fb2d3c9377d
      2c7e43af195c30b7
      d87750265cca7d21
IV  = 0342a42c8eef355c
```

ASN1 decoding

clearText

```
SEQUENCE {
  INTEGER 00
  SEQUENCE {
    OBJECTIDENTIFIER 1.2.840.113549.1.1.1 # PKCS#8
    NULL 0
  }
  OCTETSTRING 3042020100021100f80000000000000000000000000000010201000
    021813c1e53d51a1e60bc79419f7d59107ef97976d075832a45b020100020100
    020100020100020115
}
```

ASN1 decoding

```
SEQUENCE { # length= 0x42
   INTEGER 00
   INTEGER 00f80000000000000000000000000001 # Key name
   INTEGER 00
   INTEGER 13c1e53d51a1e60bc79419f7d59107ef97976d075832a45b
   INTEGER 00
   INTEGER 00
   INTEGER 00
   INTEGER 00
   INTEGER 15
 }
```

3DES EDE Master Key

```
$ file signons.sqlite
signons.sqlite: SQLite 3.x database, user version 5
```

SIGNONS.SQLITE DATABASE CONTENT

```
select hostname, encryptedUsername, encryptedPassword from moz_logins
```

```
hostname=
 http://challenge01.root-me.org
```

```
encryptedUsername=
 MDIEEPgAAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECMRMLjokk1SaBAiS/0sPBIYdEQ==
```

```
encryptedPassword=
 MDoEEPgAAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECGt3JoezKgoUBBB4Jwij9kC+NmHDW3zfcnnA
```

base64 + ASN1 decoding

base64 + ASN1 decoding

Step #3: decrypt login and password

```
SEQUENCE {
  OCTETSTRING f80000000000000000000000000001 # key name
  SEQUENCE {
    OBJECTIDENTIFIER 1.2.840.113549.3.7 # DES-EDE3-CBC
    OCTETSTRING c44c2e3a2493549a # IV
  }
  OCTETSTRING 92ff4b0f04861d11 # cipherText
}
```

IV

cipherText

3DES CBC decryption

```
'login\x03\x03\x03'
```

PKCS#7 padding to remove

```
SEQUENCE {
  OCTETSTRING f80000000000000000000000000001
  SEQUENCE {
    OBJECTIDENTIFIER 1.2.840.113549.3.7
    OCTETSTRING 6b772687b32a0a14
  }
  OCTETSTRING 782708a3f640be3661c35b7cdf7279c0
}
```

IV

cipherText

3DES CBC decryption

```
'password\x08\x08\x08\x08\x08\x08\x08\x08'
```