

Reversing the Last 7 Sub-Rounds of MD4 (because I'm too lazy to brute-force NTLM)

Andrew Zonenberg
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York U.S.A. 12180
zonena@cs.rpi.edu

September 28, 2009

1 Introduction

MD4[1] is a cryptographic hash algorithm in the “MD*” family, developed by Dr. Ron Rivest of MIT / RSA Data Security. While nowhere near as widely deployed as MD5[2], it remains used on Windows computers for password hashing in the form of the NTLM hash, which is nothing more than (unsalted) MD4 with an ASCII-to-Unicode conversion applied to the input[3]. Although MD4 is capable of being used for messages of any length (processing them in blocks of up to 448 bits), we limit our discussion here to the special case typically seen in password hashing: one block with a length that is an integer multiple of 8 bits. For a thorough discussion of the general MD4 algorithm, the reader is encouraged to read the RFC[1].

2 Algorithm Overview

MD4's state consists of four 32-bit words, termed A, B, C, and D, and a 16-word buffer containing up to 14 words of input data. The last two words contain the message length as a 64-bit little-endian integer. MD4's padding scheme is similar to that of many other popular hash functions, including MD5 and SHA-1. Specifically, the input buffer is padded with a single 0x80, followed by 0s until the password length is exactly 14 32-bit words; the remaining two elements contain length as described previ-

ously.

MD4 consists of three rounds, each with 16 “sub-rounds”. (The term “sub-round” is not used in the original RFC but it is convenient to have a name to refer to the operations by.) Each sub-round sets one of the four state words to a function of

- All four state words
- A round-specific constant
- One of the 16 words in the message buffer

More specifically, this function is defined as:

```
MD4_Round(w, x, y, z, n, s, q) =  
    (w + F(x, y, z) + buf[n] + q) <<< s
```

where w,x,y,z are a permutation of A,B,C,D, n is the index into the message buffer, and s is an integer shift amount. F and q are a function and constant whose values depend on the round. The <<< operator is defined as left circular shift.

At the end of all three rounds, constant values are added to A, B, C, and D. This has no effect on security for single-block messages but helps to add diffusion between multiple blocks in long messages.

3 The Attack

For the last round of MD4, the round function defined as $x \oplus y \oplus z$. The full sub-round structure is defined in [1] as:

```
[ABCD 0 3]      [DABC 8 9]
[CDAB 4 11]     [BCDA 12 15]
[ABCD 2 3]      [DABC 10 9]
[CDAB 6 11]     [BCDA 14 15]
[ABCD 1 3]      [DABC 9 9]
[CDAB 5 11]     [BCDA 13 15]
[ABCD 3 3]      [DABC 11 9]
[CDAB 7 11]     [BCDA 15 15]
```

where [ABCD n s] is defined as

```
A = MD4_Round(A, B, C, D, n, s, 0x6ED9EBA1)
```

Two very interesting problems become apparent after examination of the final round. First, unlike rounds 1 and 2, no irreversible operations (such as AND and OR) are used - only addition, XOR, and circular shift. Second, the final 7 sub-rounds use only words 3, 5, 7, 9, 11, 13 and 15 of the state array: the first three are not used at all!

For short to moderate-length passwords (those of under 12 bytes length), only the first three words in the state are unknown. Since the length of the password is under 2^{32} bits, the length fits completely in word 14 of the state array - also not used in the final seven sub-rounds. This permits the final seven sub-rounds to be completely reversed, as shown in the code sample below:

```
#define H(X,Y,Z) ((X) ^ (Y) ^ (Z))
#define ROTR(a,shamt) (((a) >> (shamt))
| ((a) << (32 - (shamt))))
#define rv_core(a,b,c,d,k,s,f,q, x) \
a = ROTR(a, s); a -= (f(b,c,d) + x + q);
#define rv_h(a,b,c,d,k,s, x) \
rv_core(a,b,c,d,k,s,H,0x6ED9EBA1, x)

//Subtract final values
a -= 0x67452301;
b -= 0xefcdab89;
c -= 0x98badcfe;
```

```
d -= 0x10325476;
```

```
//Reverse rounds
rv_h(b,c,d,a,15,15, 0);
rv_h(c,d,a,b,7,11, 0);
rv_h(d,a,b,c,11,9, 0);
rv_h(a,b,c,d,3,3, 0);
rv_h(b,c,d,a,13,15, 0);
rv_h(c,d,a,b,5,11, 0);
rv_h(d,a,b,c,9,9, 0);
```

4 Conclusions

When used for hashing of passwords under 13 characters, the final 7 sub-rounds of MD4 can be completely reversed. This allows an attacker to precompute the state at the end of sub-round 41 and brute-force up to that point, performing only 85% of the computations required for a full brute-force. Longer passwords (up to 28 characters) are vulnerable in the final three sub-rounds, reducing cracking effort to 94% of brute force.

NNTLM, due to its use of Unicode passwords, is less seriously affected by this vulnerability. The 7 sub-round attack is only possible for passwords of up to 6 characters, and even the 3 sub-round attack works only to 14 characters. When used in a large-scale brute-force attack, however, a detectable speedup is still likely.

References

- [1] R. Rivest, "The MD4 Message-Digest Algorithm" (RFC 1320), 1992
- [2] R. Rivest, "The MD5 Message-Digest Algorithm" (RFC 1321), 1992
- [3] J. Blair, "Samba's Encrypted Password Support", Linux Journal, 1998. Available HTTP: <http://www.linuxjournal.com/article/2717>