MD5 Collision Demo

Published Feb 22, 2006. Last updated Oct 11, 2011.

Collisions in the MD5 cryptographic hash function

It is now well-known that the crytographic hash function MD5 has been broken. In March 2005, Xiaoyun Wang and Hongbo Yu of Shandong University in China published an article in which they describe an algorithm that can find two different sequences of 128 bytes with the same MD5 hash. One famous such pair is the following:

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

and

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
```

Each of these blocks has MD5 hash 79054025255fb1a26e4bc422aef54eb4. Ben Laurie has a nice website that visualizes this MD5 collision. For a non-technical, though slightly outdated, introduction to hash functions, see Steve Friedle's Illustrated Guide.

Exploits

As we will explain below, the algorithm of Wang and Yu can be used to create files of arbitrary length that have identical MD5 hashes, and that differ only in 128 bytes somewhere in the middle of the file. Several people have used this technique to create pairs of interesting files with identical MD5 hashes:

- Magnus Daum and Stefan Lucks have created two PostScript files with identical MD5 hash, of which one is a letter of recommendation, and the other is a security clearance.
- Eduardo Diaz has described a scheme by which two programs could be packed into two archives with identical MD5 hash. A special "extractor" program turns one archive into a "good" program and the other into an "evil" one.
- In 2007, Marc Stevens, Arjen K. Lenstra, and Benne de Weger used an improved version of Wang and Yu's attack known as the chosen prefix collision method to produce two executable files with the same MD5 hash, but different behaviors. Unlike the old method, where the two files could only differ in a few carefully chosen bits, the chosen prefix method allows two completely arbitrary files to have the same MD5 hash, by appending a few thousand bytes at the end of each file. (Added Jul 27, 2008).
- Didier Stevens used the evilize program (below) to create two different programs with the same Authenticode digital signature. Authenticode is Microsoft's code signing mechanism, and although it uses SHA1 by default, it still supports MD5. (Added Jan 17, 2009).

An evil pair of executable programs

The following is an improvement of Diaz's example, which does not need a special extractor. Here are two pairs of executable programs (one pair runs on Windows, one pair on Linux).

- **Windows version:**
  - hello.exe. MD5 Sum: cdc47d670159eef60916ca03a9d4a007
  - erase.exe. MD5 Sum: cdc47d670159eef60916ca03a9d4a007
- **Linux version (i386):**
  - hello. MD5 Sum: da5c61e1edc0f18337e46418e48c1290
  - erase. MD5 Sum: da5c61e1edc0f18337e46418e48c1290

These programs must be run from the console. Here is what happens if you run them:

```
C:\TEMP> md5sum hello.exe
cdc47d670159eef60916ca03a9d4a007
C:\TEMP> .\hello.exe
Hello, world!

(press enter to quit)
C:\TEMP>


C:\TEMP> md5sum erase.exe
cdc47d670159eef60916ca03a9d4a007
C:\TEMP> .\erase.exe
This program is evil!!!
Erasing hard drive...1Gb...2Gb... just kidding!
Nothing was erased.

(press enter to quit)
C:\TEMP>
```

How it works

The above files were generated by exploiting two facts: the block structure of the MD5 function, and the fact that Wang and Yu's technique works for an arbitrary initialization vector. To understand what this means, it is useful to have a general idea of how the MD5 function processes its input. This is done by an iteration method known as the Merkle-Damgard method. A given input file is first padded so that its length will be a multiple of 64 bytes. It is then divided into individual 64-byte blocks $M_0, M_1, ..., M_{n-1}$. The MD5 hash is computed by computing a sequence of 16-byte **states** $s_0, ..., s_n$, according to the rule: $s_{i+1} = f(s_i, M_i)$, where f is a certain fixed (and complicated) function. Here, the initial state s_0 is fixed, and is called the **initialization vector**. The final state s_n is the computed MD5 hash.

The method of Wang and Yu makes it possible, for a given initialization vector s, to find two pairs of blocks M,M' and N,N', such that $f(f(s, M), M') = f(f(s, N), N')$. It is important that this works for any initialization vector s, and not just for the standard initialization vector s_0.

Combining these observations, it is possible to find pairs of files of arbitrary length, which are identical except for 128 bytes somewhere in the middle of the file, and which have identical MD5 hash. Indeed, let us write the two files as sequences of 64-byte blocks:

$M_0, M_1, ..., M_{i-1}, M_i, M_{i+1}, M_{i+2}, ..., M_n,$

$M_0, M_1, ..., M_{i-1}, N_i, N_{i+1}, M_{i+2}, ..., M_n.$

The blocks at the beginning of the files, $M_0, ..., M_{i-1}$, can be chosen arbitrarily. Suppose that the internal state of the MD5 hash function after processing these blocks is s_i. Now we can apply Wang and Yu's method to the initialization vector s_i, to find two pairs of blocks M_i, M_{i+1} and N_i, N_{i+1}, such that

$s_{i+2} = f(f(s_i, M_i), M_{i+1}) = f(f(s_i, N_i), N_{i+1}).$

This guarantees that the internal state s_{i+2} after the i+2st block will be the same for the two files. Finally, the remaining blocks M_{i+2}, ..., M_n can again be chosen arbitrarily.

So how can we use this technique to produce a pair of programs (or postscript files) that have identical MD5 hash, yet behave in arbitrary different ways? This is simple. All we have to do is write the two programs like this:

```
Program 1: if (data1 == data1) then { good_program } else { evil_program }
Program 2: if (data2 == data1) then { good_program } else { evil_program }
```

and arrange things so that "data1" = M_i, M_{i+1} and "data2" = N_i, N_{i+1} in the above scheme. This can even be done in a compiled program, by first compiling it with dummy values for data1 and data2, and later replacing them with the properly computed values.

Do it yourself: the "evilize" library

Here, you can download the software that I used to create MD5-colliding executable files.

- Download: evilize-0.2.tar.gz.

This software is based on Patrick Stach's implementation of Wang and Yu's algorithm. You can find his original implementation here.

Quick usage instructions:

Note for Windows users: the below instructions are for Unix/Linux. On Windows, you may have to append ".exe" to the names of executable files. Also, to use "make", you must have the GNU tools installed and working.

1. Unpack the archive and build the library and tools:

   ```
   tar zxf evilize-0.2.tar.gz
   cd evilize-0.2
   make
   ```

   This creates the programs "evilize", "md5coll", and the object file "goodevil.o".
2. Create a C program with multiple behaviors. Instead of the usual top-level function main(), write two separate top-level functions main_good() and main_evil(). See the file hello-erase.c for a simple example.
3. Compile your program and link against goodevil.o. For example:

   ```
   gcc hello-erase.c goodevil.o -o hello-erase
   ```

4. Run the following command to create an initialization vector:

   ```
   ./evilize hello-erase -i
   ```

5. Create an MD5 collision by running the following command (but replace the vector on the command line with the one you found in step 4):

   ```
   ./md5coll 0x23d3e487 0x3e3ea619 0xc7bdd6fa 0x2d0271e7 > init.txt
   ```

   Note: this step can take several hours.
6. Create a pair of good and evil programs by running:

   ```
   ./evilize hello-erase -c init.txt -g good -e evil
   ```

   Here "good" and "evil" are the names of the two programs generated, and "hello-erase" is the name of the program you created in step 3.

   NOTE: steps 4-6 can also be done in a single step, as follows:

```
        ./evilize hello-erase -g good -e evil
```

However, I prefer to do the steps separately, since step 5 takes so long.

7. Check the MD5 checksums of the files "good" and "evil"; they should be the same.
8. Run the programs "good" and "evil" - they should exhibit the two different behaviors that you programmed in step 2.

Back to Peter Selinger's Homepage:

Peter Selinger / Department of Mathematics and Statistics / Dalhousie University
selinger@mathstat.dal.ca / PGP key