Beware the hash reset attack

devblogs.microsoft.com/oldnewthing/20040519-00

May 19, 2004

Raymond Chen

There are a variety of message digest algorithms out there, MD5 being a particularly popular one. These generate a "message digest" (essentially, a hash) so you can detect whether somebody has tampered with a file, the theory being that it's hard to tamper with a file without changing its hash.

But make sure you record the file size as well as the digest.

Not that collisions are necessarily easy to create by mistake. (I've heard a rumor that the deployment team has seen an MD5 collision, but it's just a rumor. I have no evidence. Heck, maybe what really happened was that somebody on the deployment got their MR2 into a car accident…)

Anyway, the possibility of a "reset attack" makes collisions trivial to create.

Hash generators typically operate on a stream. The hash engine maintains some state. The file to be hashed is broken up into chunks, and each chunk is combined with the engine's state variables in some complex way. When you have passed all the data through the engine, you push a button on the engine and out pops the hash value (which is typically a copy of the state variables, or possibly a subset of them).

Now suppose somebody came up with a way of "resetting" the engine; that is, returning it to the initial state. Here's how they can make any document match your digest:

First, create an alternate message and send it through the hash engine.

Next, generate the bytes necessary to "reset" the engine.

Finally, append the original message.

In other words, the fake file looks like this:

```
[alternate message][garbage][original message]
```

where "garbage" is the reset.

This fake file has the same hash as the original message, since the “garbage” resets the hash engine to the initial state, at which point the replay of the original message regenerates the hash.

Result: A file with the same hash as the original, but with different content!

In a proper attack, of course, the “alternate message” would be crafted so the garbage and original mesage would be ignored. You might end it with a marker that means “Ignore everything after this point.” (For HTML, you can just say <NOFRAMES> and everything after that point will be largely ignored by all modern browsers.) Many other file types encode the expected file length in the header, in which case you can append whatever garbage you want without having any effect.

But if you also store the file size with the hash, then the reset attack fails, because a reset attack always generates a file bigger than the original. To create a collision, they would have to create a shorter alternate message than the original, and then fiddle with the extra bytes to get the desired target hash to come out. This is significantly harder than just resetting.

(I’m not aware of anybody who has successfully been able to reset MD5, mind you. This is a protective measure: If somebody figures out how to reset MD5, a small bit of work on your side will prevent you from falling victim.)

Raymond Chen

Follow