Nat McHugh

An important site on a minor internet.

Home About PGP Key

Monday, September 7, 2015

MD5 collisions in ssh keys

You can insert MD5 collision blocks in many data formats and if you do it right the result will be 2 objects that differ but which when passed through MD5 have the same hash value.

MD5 hashes were used as a unique fingerprints in openssh right up until version 6.8 released 2015-03-16. The idea behind public key fingerprints is that as a client you should be able to uniquely identify the public key of a server you are connecting to. Your client will snow you the fingerprint of the public key and hopefully you will recognise it. To aid this you can set your ssh client to draw you a pretty piece of random art based on the hash giving you a visual clue if something has changed.

```
Host key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48
+--[ RSA 2048]----+
|                 |
|       . .       |
|      + .        |
|     . B .       |
|    o * +        |
|   . X * S       |
|  + B O . .      |
|   .   E . o     |
|         . o     |
|         . .     |
+-----------------+
```

That is an example of connecting to github. You can check that fingerprint here.

MD5 is no longer considered collision resistant and in fact it is relatively simple to insert collisions in any binary format. This means you can create two ssh public keys that will have the same fingerprint in MD5 and same random art image. All we need to do is take a pre generated ssh public key cut out 128 bytes out of the modulus and replace it with a pair of collision blocks

Here are a couple of 2048 bit public key files to which this technique has been applied:

```
ssh-rsa AAAAB3NzaC1yc2EAAAAOAQABAAABAQD74lCtUoaXkRwcQW5ibotLrFygSCiSk3C7RR6cfFnp
```

```
ssh-rsa AAAAB3NzaC1yc2EAAAAOAQABAAABAQD74lCtUoaXkRwcQW5ibotLrFygSCiSk3C7RR6cfFnp
```

Despite looking pretty much identical they do in fact have different moduli. To check the fingerprint you can use ssh-keygen.

```
$ ssh-keygen -l -f MD5 -v -F 1.pub
2048 MD5:9e:ab:13:86:e0:40:06:59:70:84:b3:7e:57:1a:5d:1d hostname (RSA)
+---[RSA 2048]----+
|+O+       .E.    |
|*.        . .    |
|oo    . .        |
|+. . . o         |
|.... o = S       |
| . o = o .       |
|  . + .  o       |
|   .             |
|                 |
+------[MD5]------+
```

```
$ ssh-keygen -l -f MD5 -v -F 2.pub
2048 MD5:9e:ab:13:86:e0:40:06:59:70:84:b3:7e:57:1a:5d:1d hostname (RSA)
+---[RSA 2048]----+
|+O+       .E.    |
|*.        . .    |
|oo    . .        |
|+. . . o         |
|.. . o = S       |
| . o = o .       |
|  . + .  o       |
|   .             |
|                 |
+------[MD5]------+
```

To prove they are not the same try replacing MD5 in the above command with anther hash algorithm like SHA1 or SHA256.

The problem is after messing with the modul they almost certainly don't have two prime factors of length ~ root n and even if they did I have no idea what it might be without trying to factor them which is hard. So effectively they now have no private key which makes them useless. They do in fact have trivial factors which makes them doubly useless.

Is there a way to insert a collision into a key and still maintain basic RSA functionality?

Well it turns out this at least partially quite simple too. You see following our collision blocks we can put what ever additional value of the modulus we like as long as it's the same in both keys. So if we imagine the modulus being made up of three parts concatenated together A||B||C. A and B are essentially fixed but we can adjust C until we find an n which meets the RSA requirement that

n = A||B||C = p.q

Where p and q are prime and of length ~ half that of n. The best way I found to do this is to generate a random C, generate a random prime p of the correct length, divide our n made up of A||B||C by p to get the nearest integer q. Then test q to see is prime. If is we can move on however it is very unlikely that A||B|| will divide exactly by p and q without remainder. But if we multiply our p and q together they will quite likely give us a new n which can be written in the form A||B||D where D is a new number. This n still has the collision blocks B in it and we have the two prime factors p and q. The whole process takes a few seconds to run on a normal laptop. It helps to have a larger public key and so a longer C to mess around with so I have used 4096 bit public keys.

With p and q we can reconstruct the ssh private key to match public key we just created.

What about the other private key?Here we face a pretty much insurmountable problem. The value at the end of the modulus following the collision blocks needs to be the same in both public keys. However, in order to be able to build a private key for the modulus we would need to know two prime factors the likelihood of finding two is extremely small.

So this attack is impractical?
Yes as far as I can tell collisions in ssh keys of this sort don't throw up any security concerns. Hopefully you are all running newer versions of openssh and now comparing keys with sha256 anyway.

Posted by Nathaniel McHugh at 5:08 PM

G+1 +6 Recommend this on Google

No comments:

New comments are not allowed.

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Twitter

Tweets by @natmchugh

natmchugh
@natmchugh
@katie_lenn ok 🔥🔥🔥
17 May

natmchugh
@natmchugh
@katie_lenn I have one if you need to borrow one.
17 May

natmchugh
@natmchugh
Ah yes short Weierstrass form versus Edwards form. very amusing.
12 May

natmchugh
@natmchugh
@katie_lenn Sadly not available beta.companieshouse.gov.uk/search/companie...+

Embed View on Twitter

Popular Posts

Create your own MD5 collisions
A while ago a lot of people visited my site (~ 90,000) with a post about how easy it is to make two images with same MD5 by using a chos...

How I created two images with the same MD5 hash
I posted the following images the other day which although looking totally different have exactly the same MD5 hash (e0672b4f67a3c39553e...

How I made two PHP files with the same MD5 hash
I recently posted a link on twitter to two PHP scripts which have different behaviours but the same MD5 hash. To verify this download the f...

How to make two binaries with the same MD5 hash
One question I was asked when I demoed creacing two PHP files with the same hash is: does it work on compiled binaries? Well the answ...

Three way MD5 collision
Previously I explained how I created two images one of James Brown the other of Barry White with the same MD5 hash. At the end of the post I...

Hash Collisions Reading List
Lately in an effort to code up and properly understand the Wang attack on the MD4 family of hash functions I've been reading a lot of pa...

Two Magic Words are Squeamish Ossifrage - Factoring RSA-129 using CABOCHON
This thing got long and can basically be summarised as: I factored smaller RSA numbers on free cloud computing using excellent open sou...

Images with colliding MD5 hash
These images of James Brown and Barry White have the same MD5 hash a9d723c4f65a0a3f950c778bf376638. Don't believe me. You can....

Bug Bounty - Remote Code Execution in Magento 1.x
Magento is a popular ecommerce solution written in PHP. It is widely used for web shops both large and small. The most current product is Ma...

MD5 collisions in ssh keys
You can insert MD5 collision blocks in many data formats and if you do it right the result will be 2 objects that differ but which when pass...

Labels
- Cloud
- Hash functions
- Maths
- MD5
- PHP
- RSA

Blog Archive
- ► 2017 (1)
- ▼ 2015 (8)
  - ▼ September (1)
    - MD5 collisions in ssh keys
  - ► July (2)
  - ► June (1)
  - ► May (1)
  - ► March (1)
  - ► February (1)
  - ► January (1)
- ► 2014 (5)

Links
- Invigo
- Sheffield PHP User Group
- Fishcrap

Simple theme. Powered by Blogger.


```
pqkqSM6o/BYWO1kTKLHgz2zs5LwmERZnPsrK79o9KbrXCytCrj3wFCcpyBRo+TJrJPvXgqFZLanezyYLt2rNx/Fbly44wNzsJLIGTIhs26UrfnH7Wl76FlMZZp48xi3rSHJ3INFTOzwFg3PFgc2YFtzUow1TQSxaamyPHD/UOXBOj2pfCp75vJTWDwwqJqllBVGpxUhe/ouBu1iaRETSVTN44FNzlQZoxY4WWnCnCJxjBPL71lM9VzouJZggRNXct8uPorl hostname
```

```
pqCqSM6o/BYWO1kTKLHgz2zs5LwmERZnPsrK3Jo+KbrXCytCrj3wFCcz5BRe+TJrJPvXgqFZLanezyYLt2rNx/FbFy44wNzsJLIGTIhsz6UrfnH7Wl76FlMZZp68xK3rSHJ3INFTOzWFg/PFgc2YFtzUow1TQSxaamyPHD/UOXBOj2pfCp75vJTWDwwqJqllBVGpxUhe/ouBu1iaRETSVTN44FNzlQZoxY4WWnCnCJxjBPL71lM9VzouJZggRNXct8uPorl hostname
```