PASS THE SALT 2019

KILL MD5

DEMYSTIFYING HASH COLLISIONS

ANGE ALBERTINI

WITH THE HELP OF MARC STEVENS

This talk is about:

TL;DR

Understanding the impact of current hash collisions attacks.

Side effect: show that MD5 is **really** broken.

This talk is *not* about:

THE CURRENT SLIDE IS AN

HONEST TALK TRAILER

A CORKAMI ORIGINAL PRODUCTION

CRYPTOGRAPHY

It's not about the internals of hash collisions - only their impact.

NEW CRYPTOGRAPHIC ATTACKS

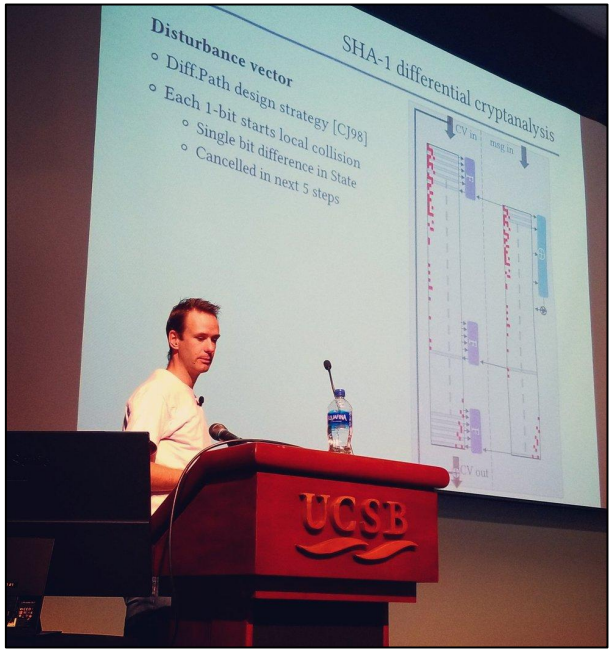
This research reuses old attacks - but some of them were never exploited.

This talk is a joint effort by:

These are our own views, Not from any of our employers.

Ange Albertini (file formats)

Google



Marc Stevens (cryptography)

CWI

1. BACKGROUND

What's exactly a hash collision?

2. KILL MD5

New results

3. HOW?

BACKGROUND

What's a hash function? MD5, SHA1...

Commonly called *checksum*.

In theory

Returns from any content a big fixed-size value, always very different.

```
␣ → d41d8cd98f00b204e9800998ecf8427e
a → 0cc175b9c0f1b6a831c399e269772661
b → 92eb5ffee6ae2fec3ad71c777531578f
A → 7fc56270e7a70fa81a5935b72eacbe29
```

Constant length
(ex: 128 bits for MD5)

Tiny content changes cause huge difference in the hash value.

One-way functions

Impossible to guess a content from its hash value.

␣ → d41d8cd98f00b204e9800998ecf8427e

? ← d41d8cd98f00b204e9800998ecf8427d

? ← d41d8cd98f00b204e9800998ecf8427f

If two contents have the same hash, they are (assumed to be) identical (if the hash is secure)

Hashes are used:

- to check passwords (compute input hash, compare with stored value)

  Confidential - do not share → `a59250af3300a8050106a67498a930f7`

  p4ssw0rd → `2a9d119df47ff993b662a8ef36f9ea20`

- to validate content integrity

- to index files (ex: your pictures in the cloud)

...unless there is a hash collision:

two different contents with the same hash result.

```
$ python
[...]
>>> crypt.crypt("5dUD&66", salt="br")
'brokenOz4KxMc'
>>> crypt.crypt("O!>',%$", salt="br")
'brokenOz4KxMc'
>>> crypt.crypt("O!>',%$", "br") == crypt.crypt("5dUD&66", "br")
True
>>>
```

This example uses the crypt(3) hash.

What are hash collisions in practice?

A computation that generates

two distinct contents with the same hash.

We can define some part of these contents.

A hash collision generates a lot of randomness!

-> the final hash is **not** known in advance.

An MD5 collision of YES and NO: 576 bytes of random-looking data

```
0000:  .y .e .s 00-00 00 00 00-00 00 00 00-00 00 00 00
0010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0030:  00 00 00 00-00 00 00 00-B7 46 38 09-8A 46 F1 7B
0040:  F3 45 26 13-66 60 C8 01-B9 2A 75 25-5A 67 23 A6
0050:  92 3D EB 8D-B0 B7 57 F1-45 9F 22 95-BE C0 43 75
0060:  91 98 A2 D3-E0 FD 59 ED-D1 C5 FA 0B-79 65 97 4D
0070:  B3 B3 E4 0C-11 0C 90 32-DE 4B A1 4B-B8 1B 5E C8
0080:  25 D3 8F 19-CD 10 43 07-D9 BB FF 8C-B7 5A 23 F9
0090:  4D D8 13 14-58 A3 35 97-C5 D1 D4 A9-9A E2 FD 1F
00A0:  BA 78 40 00-C3 7E 93 B2-31 A3 6E 2D-34 6A 4A C9
00B0:  53 4E C0 45-36 1E C8 6A-56 98 E6 F0-57 1D 61 98
00C0:  13 FC FF CD-4D 83 A2 D2-BB B8 DC 04-2B E2 B8 83
00D0:  DB 53 80 D7-3D E9 97 D3-23 5A 27 F9-98 9A E7 56
00E0:  7D 86 E4 35-1E B8 33 EE-EA 15 D1 81-BA 96 62 EC
00F0:  75 31 FB DA-4F AE 24 6F-67 D6 AF 10-96 29 FB C7
0100:  A3 32 BB A9-EA D5 E4 AE-1F C2 FB 23-41 22 B2 E0
0110:  69 1E 29 20-6F 5B 20 1E-5E 3D 11 2F-3E 4D 9F 39
0120:  8B C9 5C 93-A5 EF A4 22-7D 9A 66 51-6E ED AD 70
0130:  32 90 D4 BD-67 92 38 9B-DC 15 0D BF-DC 71 72 27
0140:  E0 5B 43 FA-44 59 E8 60-F7 63 7F F0-73 0A D4 BE
0150:  33 28 AA 99-2C 90 2D D0-01 58 E3 8F-58 50 30 99
0160:  E8 60 DB 91-00 13 C9 1D-7A 61 9B 9A-5D 5E BD 71
0170:  23 1A D2 BD-A6 E0 38 66-0B 8C F5 99-56 79 63 D6
0180:  6E 5E D7 7E-C3 4E 9D 5F-65 23 C0 38-C9 55 5A A1
0190:  E2 3C CA 78-58 4D B5 3B-04 45 C3 B4-44 C8 87 26
01A0:  02 60 F6 62-91 34 70 FE-C3 34 54 6D-76 07 7F 1A
01B0:  73 53 E6 0B-08 FB 82 80-AD 5F 22 15-18 69 B5 6E
01C0:  BB 06 C3 A7-FF 39 15 52-BE FE D4 5C-D2 55 5A 71
01D0:  EC E9 BC 1A-B7 BB 08 61-C5 3E E7 89-7C 93 03 FC
01E0:  1F 8A 9A D8-42 BF 6C 01-6A 39 26 84-74 58 E2 E4
01F0:  00 D4 67 7B-27 BD 93 6D-DF F0 10 4A-2B 00 7E 68
0200:  1D DE D5 8A-67 89 EA 52-0C 32 BD 30-A2 8C BE D0
0210:  A7 35 BA C6-BB 7D 07 80-49 22 EF E5-10 B2 83 6D
0220:  E6 18 6E E3-F0 52 E4 35-83 61 42 35-72 97 C5 8D
0230:  4F F7 93 68-5A 70 5F 5A-04 3A D5 42-C1 FA 0F E2
0240:  AE 57 DB AF-F1 51 B8 B7-38 18 EF 2E-B8 A6 A9 2C
0250:  81 87 FA FE-B2 C4 DC 45-A3 64 91 6D-B8 6E F5 D1
0260:  4F 9C FA 62-3D 42 46 59-67 32 EC 99-DA 89 7A 88
0270:  E7 AD E3 21-ED 3C 4B C0-4D 9F 83 3C-DC 7F B7 0A

≠

0000:  .n .o 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0030:  00 00 00 00-00 00 00 00-19 71 E7 F7-09 72 FB 06
0040:  F3 45 26 13-66 60 C8 01-B9 2A 75 25-5A 67 23 A6
0050:  92 3D EB 8D-B0 B7 57 F1-45 9F 22 95-BE C0 43 75
0060:  91 98 A2 D3-E0 FD 59 ED-D1 C5 FA 0B-79 65 97 51
0070:  B3 B3 E4 0C-11 0C 90 32-DE 4B A1 4B-B8 1B 5E C8
0080:  25 D3 8F 19-CD 10 43 07-D9 BB FF 8C-B7 5A 23 F9
0090:  4D D8 13 14-58 A3 35 97-C5 D1 D4 A9-9A E2 FD 1F
00A0:  BA 78 40 00-C3 7E 93 B2-31 A3 6E 2D-34 72 4A C9
00B0:  53 4E C0 45-36 1E C8 6A-56 98 E6 F0-57 1D 61 98
00C0:  13 FC FF CD-4D 83 A2 D2-BB B8 DC 04-2B E2 B8 83
00D0:  DB 53 80 D7-3D E9 97 D3-23 5A 27 F9-98 9A E7 56
00E0:  7D 86 E4 35-1E B8 33 EE-EA 15 D1 81-FA 96 62 EC
00F0:  75 31 FB DA-4F AE 24 6F-67 D6 AF 10-96 29 FB C7
0100:  A3 32 BB A9-EA D5 E4 AE-1F C2 FB 23-41 22 B2 E0
0110:  69 1E 29 20-6F 5B 20 1E-5E 3D 11 2F-3E 4D 9F 39
0120:  8B C9 5C 93-A5 EF A4 22-7D 9A 66 51-6E ED AF 70
0130:  32 90 D4 BD-67 92 38 9B-DC 15 0D BF-DC 71 72 27
0140:  E0 5B 43 FA-44 59 E8 60-F7 63 7F F0-73 0A D4 BE
0150:  33 28 AA 99-2C 90 2D D0-01 58 E3 8F-58 50 30 99
0160:  E8 60 DB 91-00 13 C9 1D-7A 61 9B 9A-5D 60 BD 71
0170:  23 1A D2 BD-A6 E0 38 66-0B 8C F5 99-56 79 63 D6
0180:  6E 5E D7 7E-C3 4E 9D 5F-65 23 C0 38-C9 55 5A A1
0190:  E2 3C CA 78-58 4D B5 3B-04 45 C3 B4-44 C8 87 26
01A0:  02 60 F6 62-91 34 70 FE-C3 34 54 6D-76 07 FF 1A
01B0:  73 53 E6 0B-08 FB 82 80-AD 5F 22 15-18 69 B5 6E
01C0:  BB 06 C3 A7-FF 39 15 52-BE FE D4 5C-D2 55 5A 71
01D0:  EC E9 BC 1A-B7 BB 08 61-C5 3E E7 89-7C 93 03 FC
01E0:  1F 8A 9A D8-42 BF 6C 01-6A 39 26 84-6C 58 E2 E4
01F0:  00 D4 67 7B-27 BD 93 6D-DF F0 10 4A-2B 00 7E 68
0200:  1D DE D5 8A-67 89 EA 52-0C 32 BD 30-A2 8C BE D0
0210:  A7 35 BA C6-BB 7D 07 80-49 22 EF E5-10 B2 83 6D
0220:  E6 18 6E E3-F0 52 E4 35-83 61 42 35-72 97 CD 8D
0230:  4F F7 93 68-5A 70 5F 5A-04 3A D5 42-C1 FA 0F E2
0240:  AE 57 DB AF-F1 51 B8 B7-38 18 EF 2E-B8 A6 A9 2C
0250:  81 87 FA FE-B2 C4 DC 45-A3 64 91 6D-B8 6E F5 D1
0260:  4F 9C FA 62-3D 42 46 59-67 32 EC 99-DA 89 7A 08
0270:  E7 AD E3 21-ED 3C 4B C0-4D 9F 83 3C-DC 7F B7 0A
```


A HASH COLLISION IS...
(IN THE CASE OF THESE MD5/SHA1 ATTACKS)
...A BIG PILE OF...
COMPUTED RANDOMNESS
WITH TINY DIFFERENCES.

Best attack on MD2: 2^{73} from 2008

These don't exist yet - not even for MD2 (from 1989!)

Generate a file X with a hash H:

given any H, make X so that hash(X) = H

(also called **pre-image attack**)

...and by extension:

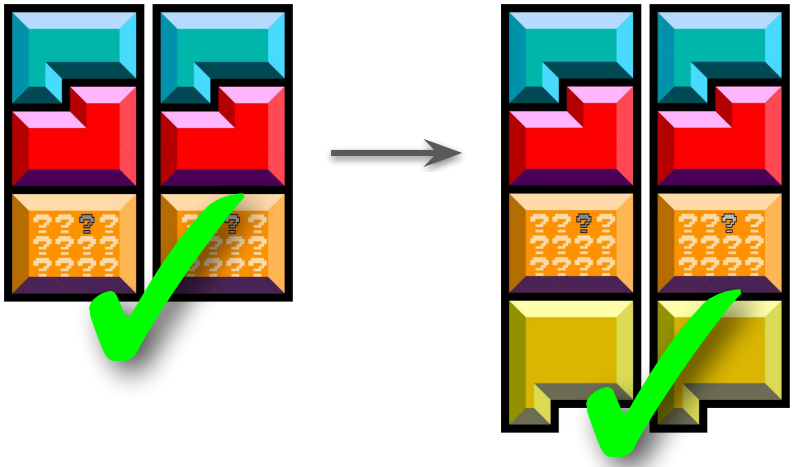
Given any file Y, generate a file X with the same hash

make X so that hash(X) = hash(Y) (with X != Y)

(**second pre-image attack**)

How hashes like MD5 or SHA1/2 work

1. Processing blocks, from start to end.
2. Appending the same thing to two files with the same hash will give files with the same hash.



First type of collision: Identical Prefix

IPC

Step 1/4 : the prefix (optional)

We define the start of the file.

The collision computation will depend on that.

The prefix can be empty.

Its content and size make **no** difference at all.

Step 2/4 : the padding (if needed)

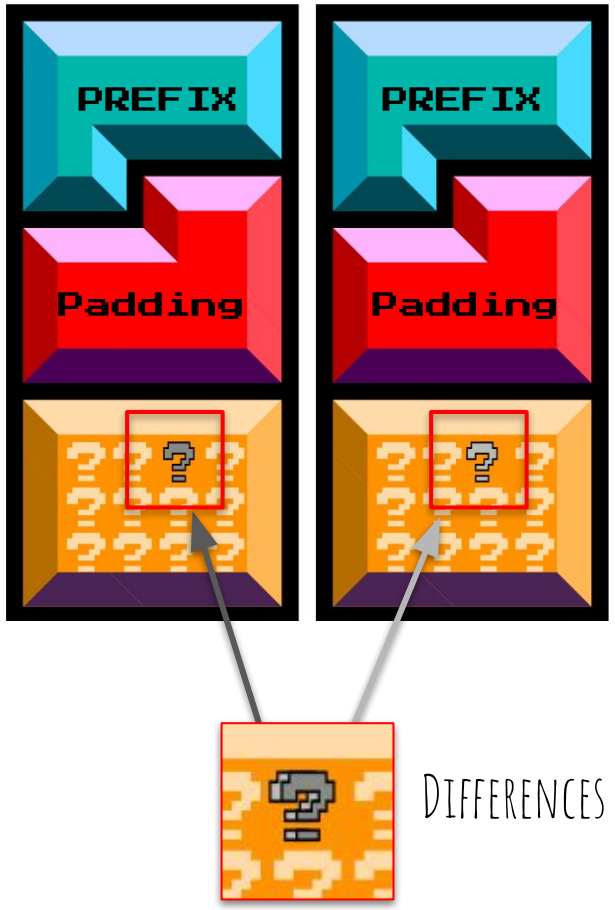
We add some data to the prefix
to get a rounded size (a multiple of 64).

Step 3/4 : the collision blocks

We compute a pair of blocks full of randomness
with tiny differences.

Despite the differences,
the hash of both files is the same.

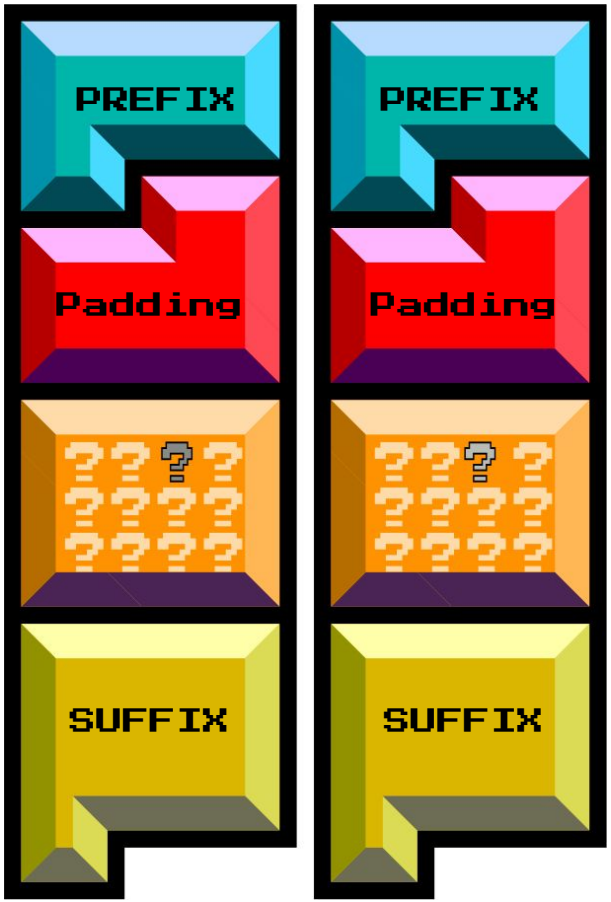
These collision blocks only work for that prefix.



Step 4/4 : the suffix

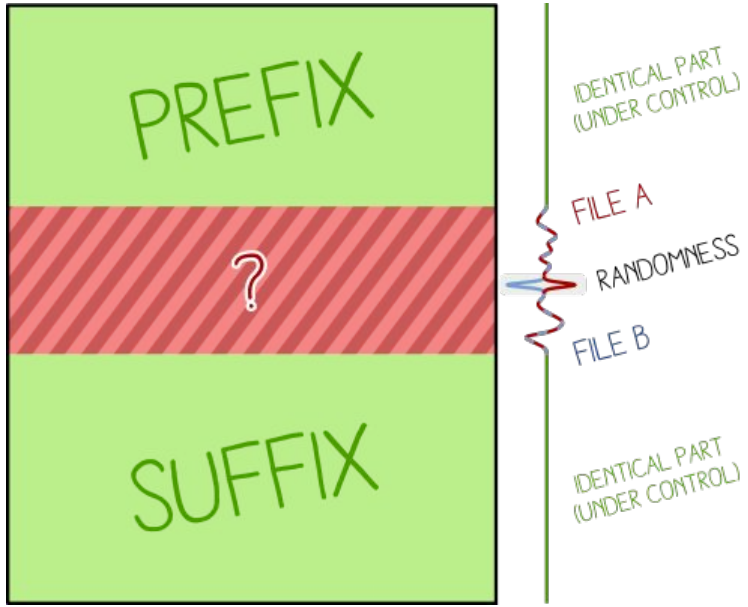
You can add anything to both sides (not required).

The hash value will remain the same.



Identical Prefix Collisions

Take a **single** optional input (the prefix)
Generate 2 different files with same hash.



The file content is identical
before and after the collision (prefix & suffix).
The only differences are in the collision blocks.

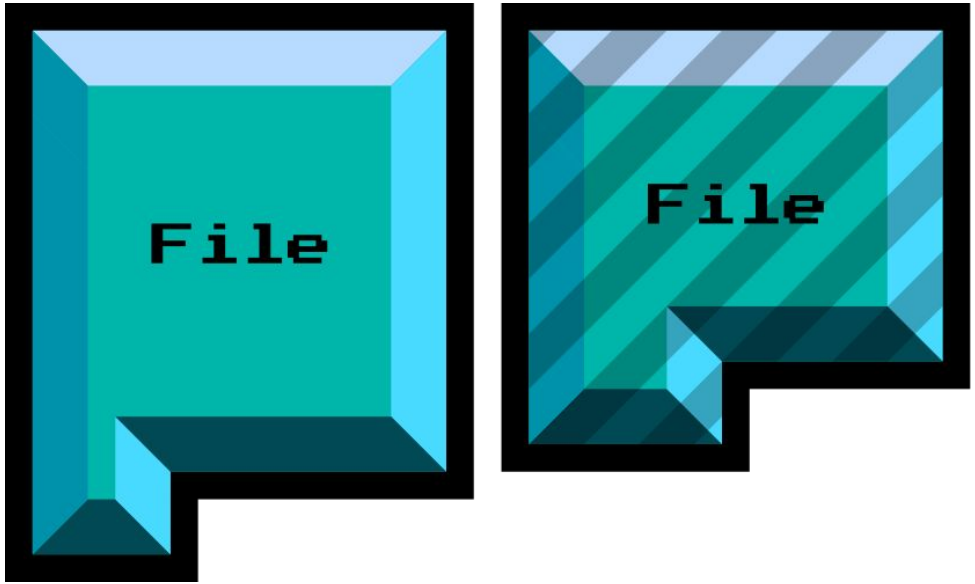
```
00:  .H .e .r .e .  .i .s .  .a .  .f .i .l .e .  .w
10:  .i .t .h .  .a .  .f .e .w .  .b .y .t .e .s 00
20:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40:  CE 84 07 61 4B BA 7A 3D 3A EA 8A AA F8 EE 1D E5
50:  44 17 9B F0 0A E0 D2 64 21 E2 38 E1 94 18 0A F6
60:  93 D2 B5 E4 FC 2F 3A 32 4F 50 46 01 F1 4B BF 02
70:  23 EE EF BF 92 B5 7C 29 D9 C5 66 08 31 5E 7A 1D
80:  2F 5A 9C 5C 12 8E DF F2 85 17 5B DD 67 25 05 78
90:  13 F2 BF D6 64 59 F2 C8 8B C3 00 6F 8B 5F 88 C6
A0:  CB 3D 80 E4 9F 48 91 5E 34 06 D0 3A 8B 03 FB E0
B0:  ED 18 67 0F C8 3A C9 A1 E7 48 F6 2A D2 5C 30 C0
```

```
00:  .H .e .r .e .  .i .s .  .a .  .f .i .l .e .  .w
10:  .i .t .h .  .a .  .f .e .w .  .b .y .t .e .s 00
20:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40:  CE 84 07 61 4B BA 7A 3D 3A EA 8A AA F8 EE 1D E5
50:  44 17 9B 70 0A E0 D2 64 21 E2 38 E1 94 18 0A F6
60:  93 D2 B5 E4 FC 2F 3A 32 4F 50 46 01 F1 CB BE 02
70:  23 EE EF BF 92 B5 7C 29 D9 C5 66 88 31 5E 7A 1D
80:  2F 5A 9C 5C 12 8E DF F2 85 17 5B DD 67 25 05 78
90:  13 F2 BF 56 64 59 F2 C8 8B C3 00 6F 8B 5F 88 C6
A0:  CB 3D 80 E4 9F 48 91 5E 34 06 D0 3A 8B 83 FB E0
B0:  ED 18 67 0F C8 3A C9 A1 E7 48 F6 AA D2 5C 30 C0
```

Example of an Identical-Prefix Collision - only a few differences.

Second type of collision: Chosen prefix

CPC



So, we have two files. Any pair of files.

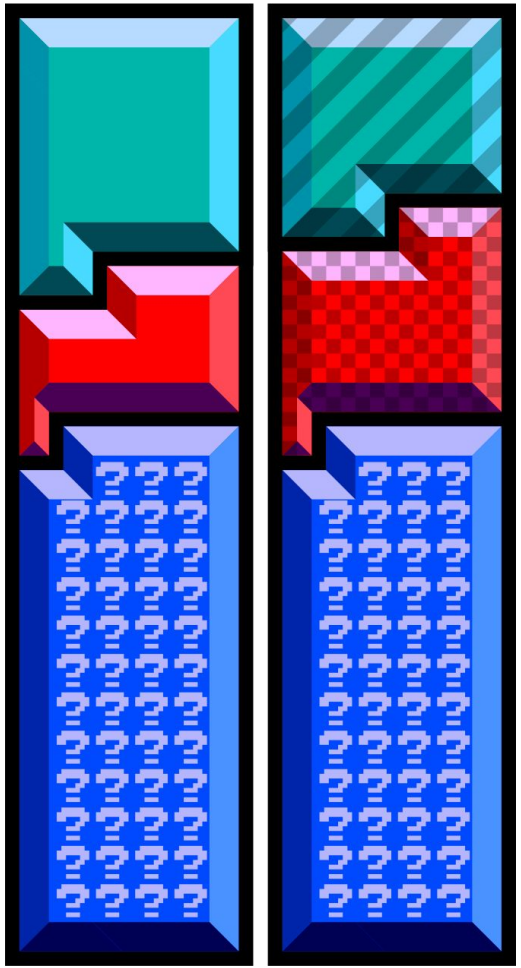
What a CPC does:

Pad both files to the same length.

Compute different blocks for each file.

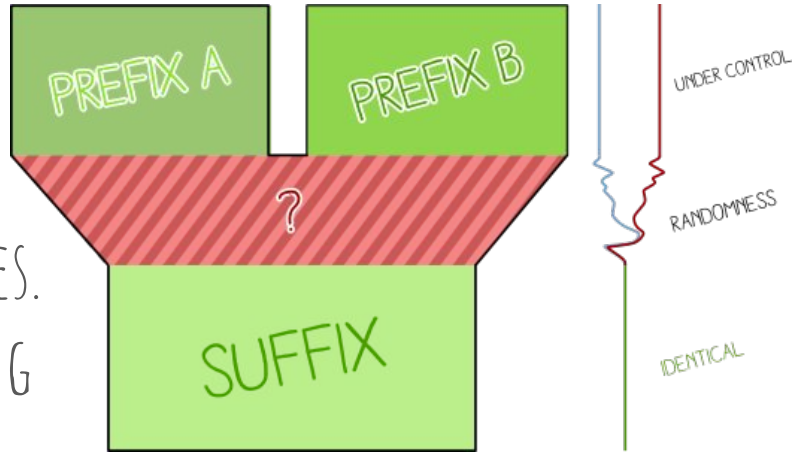
Append these blocks.

Suffix is optional once again.



Second type of collisions

take **two** prefixes, append something to both
to make them get the same hash.



It can work with **any** contents of any sizes.
Contents and sizes don't change anything
(Resulting files will have the same length).

A CHOSEN PREFIX HASH COLLISION OF YES AND NO

```
0000:  .y .e .s 00-00 00 00 00-00 00 00 00-00 00 00 00
0010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0030:  00 00 00 00-00 00 00 00-B7 46 38 09-8A 46 F1 7B
0040:  F3 45 26 13-66 60 C8 01-B9 2A 75 25-5A 67 23 A6
0050:  92 3D EB 8D-B0 B7 57 F1-45 9F 22 95-BE C0 43 75
0060:  91 98 A2 D3-E0 FD 59 ED-D1 C5 FA 0B-79 65 97 4D
0070:  B3 B3 E4 0C-11 0C 90 32-DE 4B A1 4B-B8 1B 5E C8
0080:  25 D3 8F 19-CD 10 43 07-D9 BB FF 8C-B7 5A 23 F9
0090:  4D D8 13 14-58 A3 35 97-C5 D1 D4 A9-9A E2 FD 1F
00A0:  BA 78 40 00-C3 7E 93 B2-31 A3 6E 2D-34 6A 4A C9
00B0:  53 4E C0 45-36 1E C8 6A-56 98 E6 F0-57 1D 61 98
00C0:  13 FC FF CD-4D 83 A2 D2-BB B8 DC 04-2B E2 B8 83
00D0:  DB 53 80 D7-3D E9 97 D3-23 5A 27 F9-98 9A E7 56
00E0:  7D 86 E4 35-1E B8 33 EE-EA 15 D1 81-BA 96 62 EC
00F0:  75 31 FB DA-4F AE 24 6F-67 D6 AF 10-96 29 FB C7
0100:  A3 32 BB A9-EA D5 E4 AE-1F C2 FB 23-41 22 B2 E0
0110:  69 1E 29 20-6F 5B 20 1E-5E 3D 11 2F-3E 4D 9F 39
0120:  8B C9 5C 93-A5 EF A4 22-7D 9A 66 51-6E ED AD 70
0130:  32 90 D4 BD-67 92 38 9B-DC 15 0D BF-DC 71 72 27
0140:  E0 5B 43 FA-44 59 E8 60-F7 63 7F F0-73 0A D4 BE
0150:  33 28 AA 99-2C 90 2D D0-01 58 E3 8F-58 50 30 99
0160:  E8 60 DB 91-00 13 C9 1D-7A 61 9B 9A-5D 5E BD 71
0170:  23 1A D2 BD-A6 E0 38 66-0B 8C F5 99-56 79 63 D6
0180:  6E 5E D7 7E-C3 4E 9D 5F-65 23 C0 38-C9 55 5A A1
0190:  E2 3C CA 78-58 4D B5 3B-04 45 C3 B4-44 C8 87 26
01A0:  02 60 F6 62-91 34 70 FE-C3 34 54 6D-76 07 7F 1A
01B0:  73 53 E6 0B-08 FB 82 80-AD 5F 22 15-18 69 B5 6E
01C0:  BB 06 C3 A7-FF 39 15 52-BE FE D4 5C-D2 55 5A 71
01D0:  EC E9 BC 1A-B7 BB 08 61-C5 3E E7 89-7C 93 03 FC
01E0:  1F 8A 9A D8-42 BF 6C 01-6A 39 26 84-74 58 E2 E4
01F0:  00 D4 67 7B-27 BD 93 6D-DF F0 10 4A-2B 00 7E 68
0200:  1D DE D5 8A-67 89 EA 52-0C 32 BD 30-A2 8C BE D0
0210:  A7 35 BA C6-BB 7D 07 80-49 22 EF E5-10 B2 83 6D
0220:  E6 18 6E E3-F0 52 E4 35-83 61 42 35-72 97 C5 8D
0230:  4F F7 93 68-5A 70 5F 5A-04 3A D5 42-C1 FA 0F E2
0240:  AE 57 DB AF-F1 51 B8 B7-38 18 EF 2E-B8 A6 A9 2C
0250:  81 87 FA FE-B2 C4 DC 45-A3 64 91 6D-B8 6E F5 D1
0260:  4F 9C FA 62-3D 42 46 59-67 32 EC 99-DA 89 7A 88
0270:  E7 AD E3 21-ED 3C 4B C0-4D 9F 83 3C-DC 7F B7 0A
```

Padding

Random buffer

(partial birthday attack bits)

Collision blocks

```
0000:  .n .o 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0030:  00 00 00 00-00 00 00 00-19 71 E7 F7-09 72 FB 0C
0040:  F3 45 26 13-66 60 C8 01-B9 2A 75 25-5A 67 23 A6
0050:  92 3D EB 8D-B0 B7 57 F1-45 9F 22 95-BE C0 43 75
0060:  91 98 A2 D3-E0 FD 59 ED-D1 C5 FA 0B-79 65 97 51
0070:  B3 B3 E4 0C-11 0C 90 32-DE 4B A1 4B-B8 1B 5E C8
0080:  25 D3 8F 19-CD 10 43 07-D9 BB FF 8C-B7 5A 23 F9
0090:  4D D8 13 14-58 A3 35 97-C5 D1 D4 A9-9A E2 FD 1F
00A0:  BA 78 40 00-C3 7E 93 B2-31 A3 6E 2D-34 72 4A C9
00B0:  53 4E C0 45-36 1E C8 6A-56 98 E6 F0-57 1D 61 98
00C0:  13 FC FF CD-4D 83 A2 D2-BB B8 DC 04-2B E2 B8 83
00D0:  DB 53 80 D7-3D E9 97 D3-23 5A 27 F9-98 9A E7 56
00E0:  7D 86 E4 35-1E B8 33 EE-EA 15 D1 81-FA 96 62 EC
00F0:  75 31 FB DA-4F AE 24 6F-67 D6 AF 10-96 29 FB C7
0100:  A3 32 BB A9-EA D5 E4 AE-1F C2 FB 23-41 22 B2 E0
0110:  69 1E 29 20-6F 5B 20 1E-5E 3D 11 2F-3E 4D 9F 39
0120:  8B C9 5C 93-A5 EF A4 22-7D 9A 66 51-6E ED AF 70
0130:  32 90 D4 BD-67 92 38 9B-DC 15 0D BF-DC 71 72 27
0140:  E0 5B 43 FA-44 59 E8 60-F7 63 7F F0-73 0A D4 BE
0150:  33 28 AA 99-2C 90 2D D0-01 58 E3 8F-58 50 30 99
0160:  E8 60 DB 91-00 13 C9 1D-7A 61 9B 9A-5D 60 BD 71
0170:  23 1A D2 BD-A6 E0 38 66-0B 8C F5 99-56 79 63 D6
0180:  6E 5E D7 7E-C3 4E 9D 5F-65 23 C0 38-C9 55 5A A1
0190:  E2 3C CA 78-58 4D B5 3B-04 45 C3 B4-44 C8 87 26
01A0:  02 60 F6 62-91 34 70 FE-C3 34 54 6D-76 07 FF 1A
01B0:  73 53 E6 0B-08 FB 82 80-AD 5F 22 15-18 69 B5 6E
01C0:  BB 06 C3 A7-FF 39 15 52-BE FE D4 5C-D2 55 5A 71
01D0:  EC E9 BC 1A-B7 BB 08 61-C5 3E E7 89-7C 93 03 FC
01E0:  1F 8A 9A D8-42 BF 6C 01-6A 39 26 84-6C 58 E2 E4
01F0:  00 D4 67 7B-27 BD 93 6D-DF F0 10 4A-2B 00 7E 68
0200:  1D DE D5 8A-67 89 EA 52-0C 32 BD 30-A2 8C BE D0
0210:  A7 35 BA C6-BB 7D 07 80-49 22 EF E5-10 B2 83 6D
0220:  E6 18 6E E3-F0 52 E4 35-83 61 42 35-72 97 CD 8D
0230:  4F F7 93 68-5A 70 5F 5A-04 3A D5 42-C1 FA 0F E2
0240:  AE 57 DB AF-F1 51 B8 B7-38 18 EF 2E-B8 A6 A9 2C
0250:  81 87 FA FE-B2 C4 DC 45-A3 64 91 6D-B8 6E F5 D1
0260:  4F 9C FA 62-3D 42 46 59-67 32 EC 99-DA 89 7A 08
0270:  E7 AD E3 21-ED 3C 4B C0-4D 9F 83 3C-DC 7F B7 0A
```

KILL MD5

Wasn't it... killed long ago?


M:I 2008
MD5 SSL certificate

Since 2008, MD5 was considered dead for good

An outstanding attack:

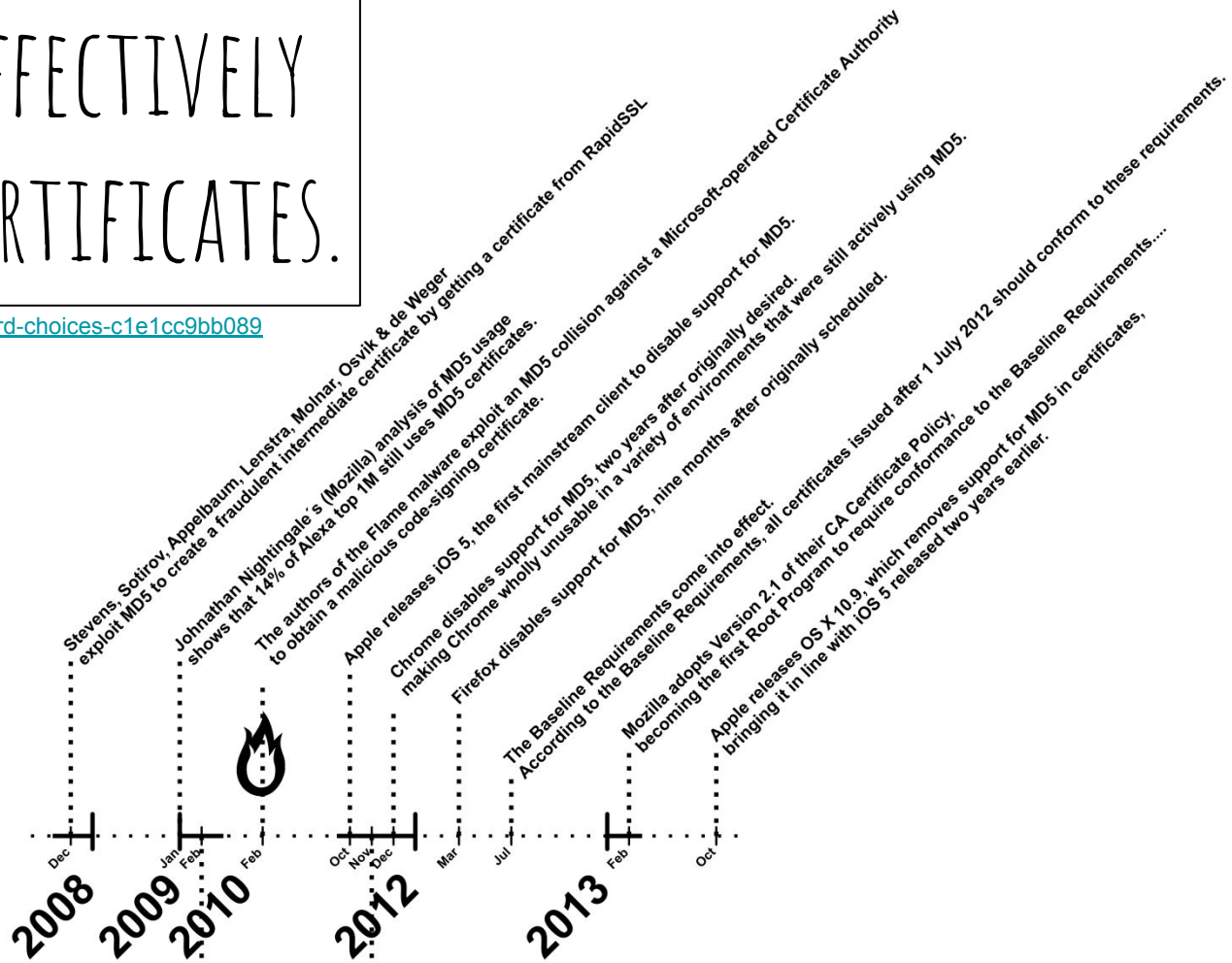
200 Playstation 3 and signing at an exact second

with 2 days of computations for each of the 4 attempts.

2004: first MD5 collision 2006: first practical impact 2008: rogue SSL certificate

MD5 has been effectively banned from certificates.

https://medium.com/@sleevi_/a-history-of-hard-choices-c1e1cc9bb089



Sure, MD5 is weak against such kinds of attack.

Since 2009, no more attacks on MD5 nor research (regarding files):

it was considered dead for good by experts.

So it's dead and buried, right?

Attack on protocols:

CVE-2015-7575: SLOTH **S**ecurity **L**osses from **O**bsolete and **T**runcated Transcript **H**ashes

https://www.mitls.org/pages/attacks/SLOTH


MD5
1991-20??

https://www.swgde.org/documents/Released%20Docs%20Public%20Comment/SWGDE%20Position%20on%20the%20Use%20of%20MD5%20and%20SHA1%20Hash%20Algorithms%20in%20Digital%20and%20Multimedia%20Forensics
74ce36b71d161cb5ab35627bd5a7f1070ff475f42da397bf17e6c8c6a897657a

MD5 IS NOT DEAD

Scientific Working Group on Digital Evidence

SWGDE Position on the Use of MD5 and SHA1 Hash Algorithms in Digital and Multimedia Forensics

5. Recommendations for the Appropriate Uses of MD5 and SHA1

Because MD5 and SHA1 have proven to be susceptible to engineered collisions, they should only be used for certain functions. It is still appropriate to use MD5 and SHA1 for the following situations:

- **Integrity Verification**
  It is appropriate to use both MD5 and SHA1 for integrity verification provided the hash is securely stored or recorded in examination documentation. This will prevent an individual from substituting a different file and its hash. This is true for all hash algorithms.
- **File Identification**
  Since there are no preimage attacks against MD5 and SHA1, it is appropriate to use both algorithms for file identification.

It's still used to index files or validate integrity:

"It's still better than CRC32!"

The big question

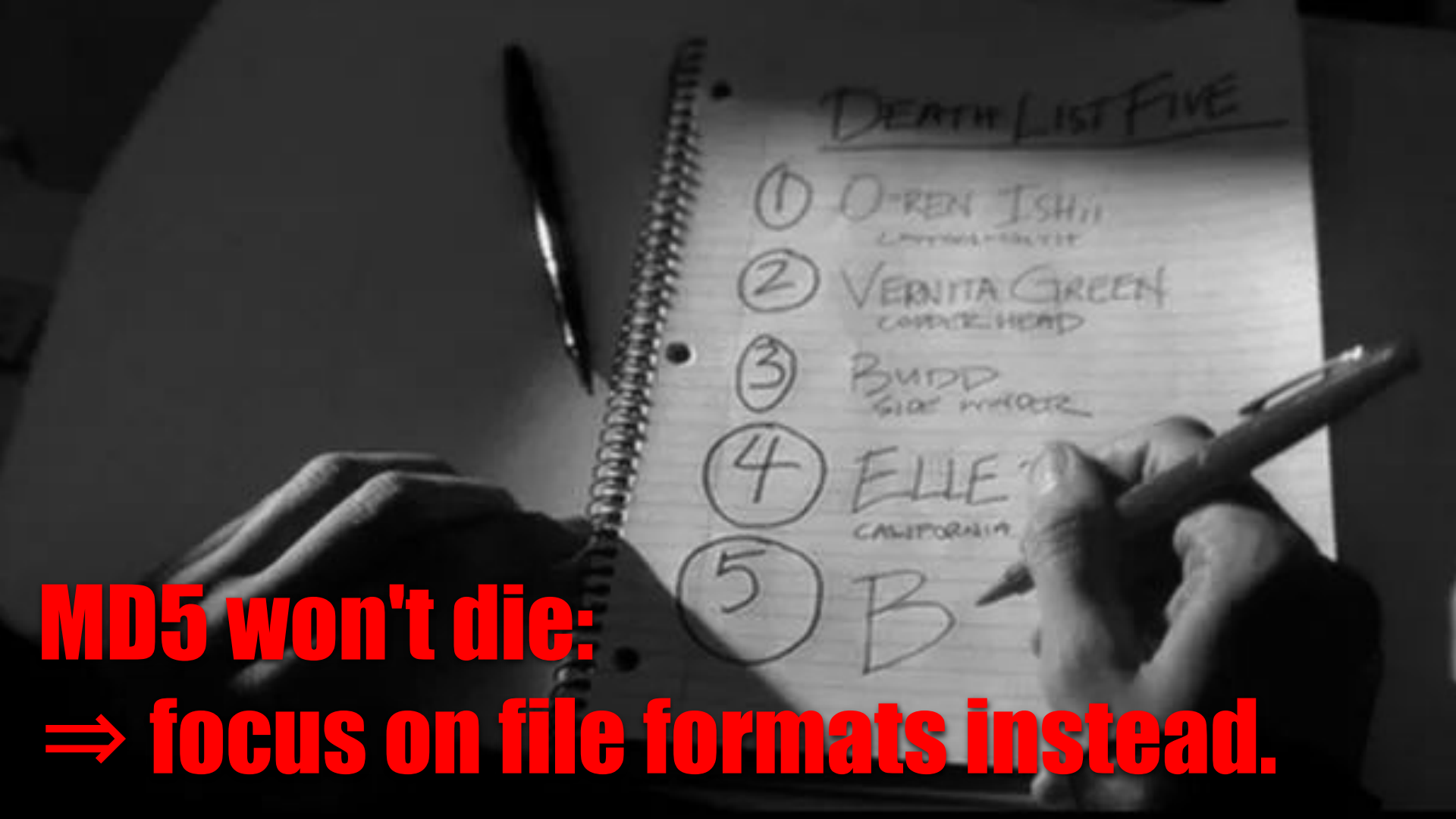
Since current attacks aren't enough to kill MD5....

How efficiently can one make collisions w/ standard file formats?

By any possible means:

with file tricks and pre-computed prefixes

with any existing attacks.



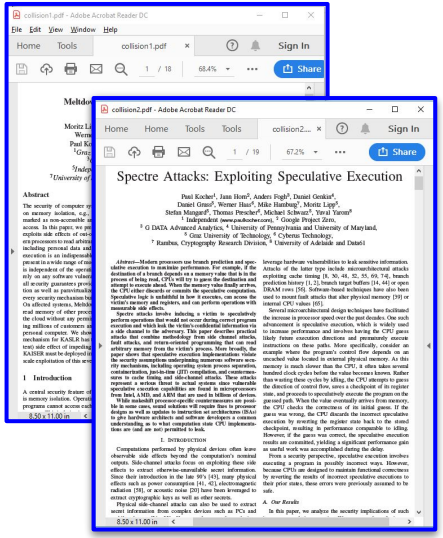
MD5 won't die:

⇒ focus on file formats instead.

Our contributions - 1/2

Instant MD5 collisions, with no recomputation

(collision data is pre-computed)

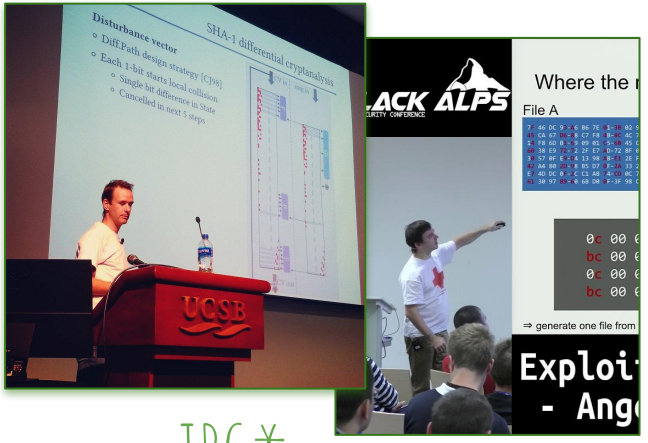


PDF

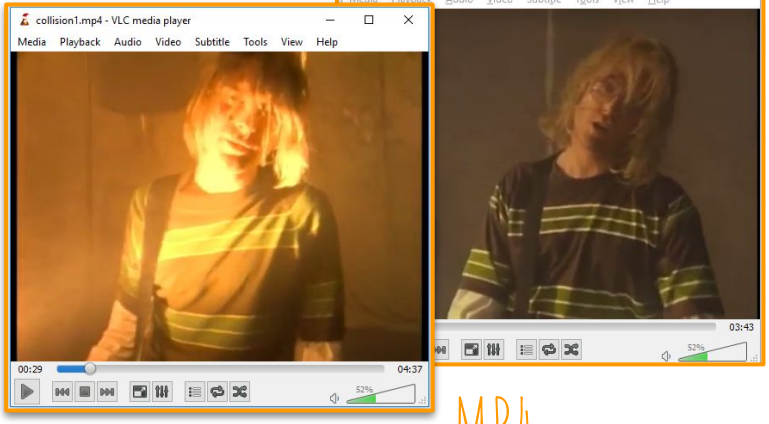
Sega

Nintendo

PNG*



JPG*



MP4

https://github.com/corkami/collisions

*some limitations

Our contributions - 2/2

*some limitations

GIF*

PE(Windows executables)

JP2

Just new collisions?

Instant, re-usable and generic collisions:

take any pair of files, run script, get colliding files.

For example, the colliding PDFs are 100% standard.

From a parser perspective,

the contents are unmodified: only the files' structures are.

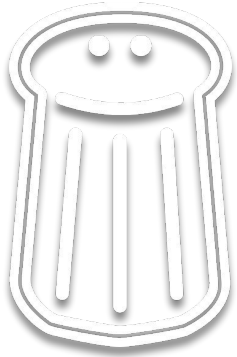
These pictures come from the conference website.

```
22:30:39.00>jpg.py 1.jpg 5.jpg

22:30:39.17>png.py salt-2019-wh.png ubicast.png

22:30:39.32>md5sum collision*
1c3d329b9243e49ead76b9c972752761 *collision1.jpg
1c3d329b9243e49ead76b9c972752761 *collision2.jpg
1d0ea9d09a2b562533a1de5e7d2be2a5 *collision1.png
1d0ea9d09a2b562533a1de5e7d2be2a5 *collision2.png
```

UBICAST



Less than 1 s to collide PNG, JPG, PE, PDF, MP4...

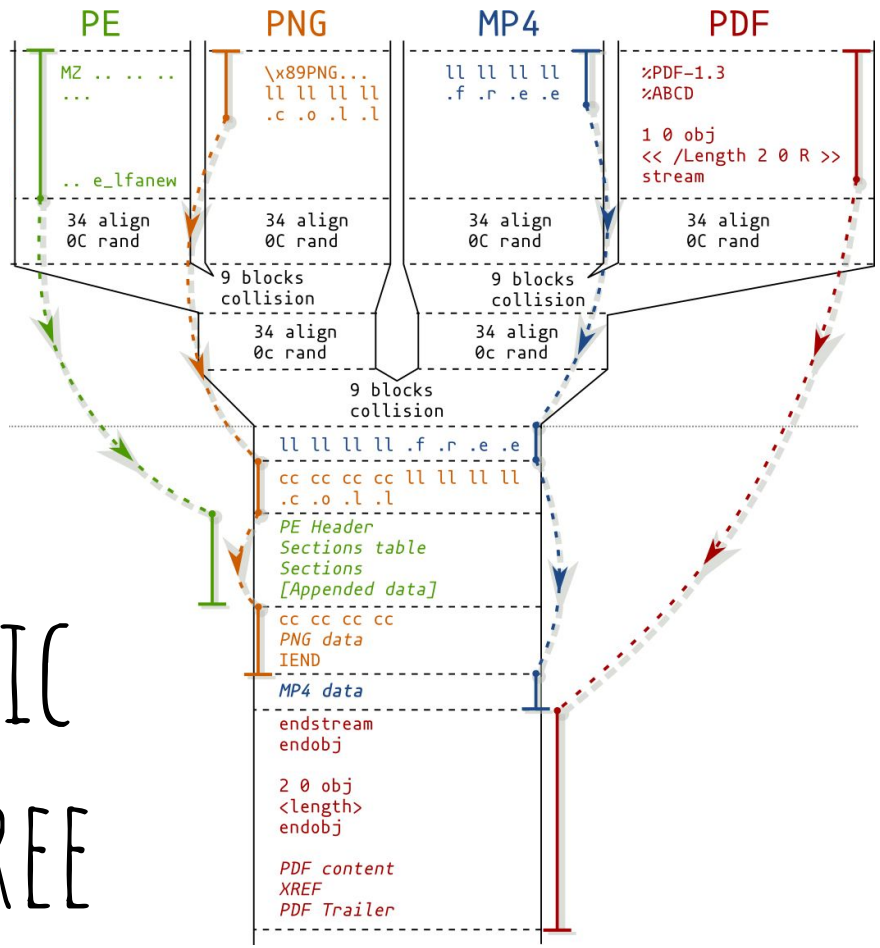
Kill some long-lasting myths

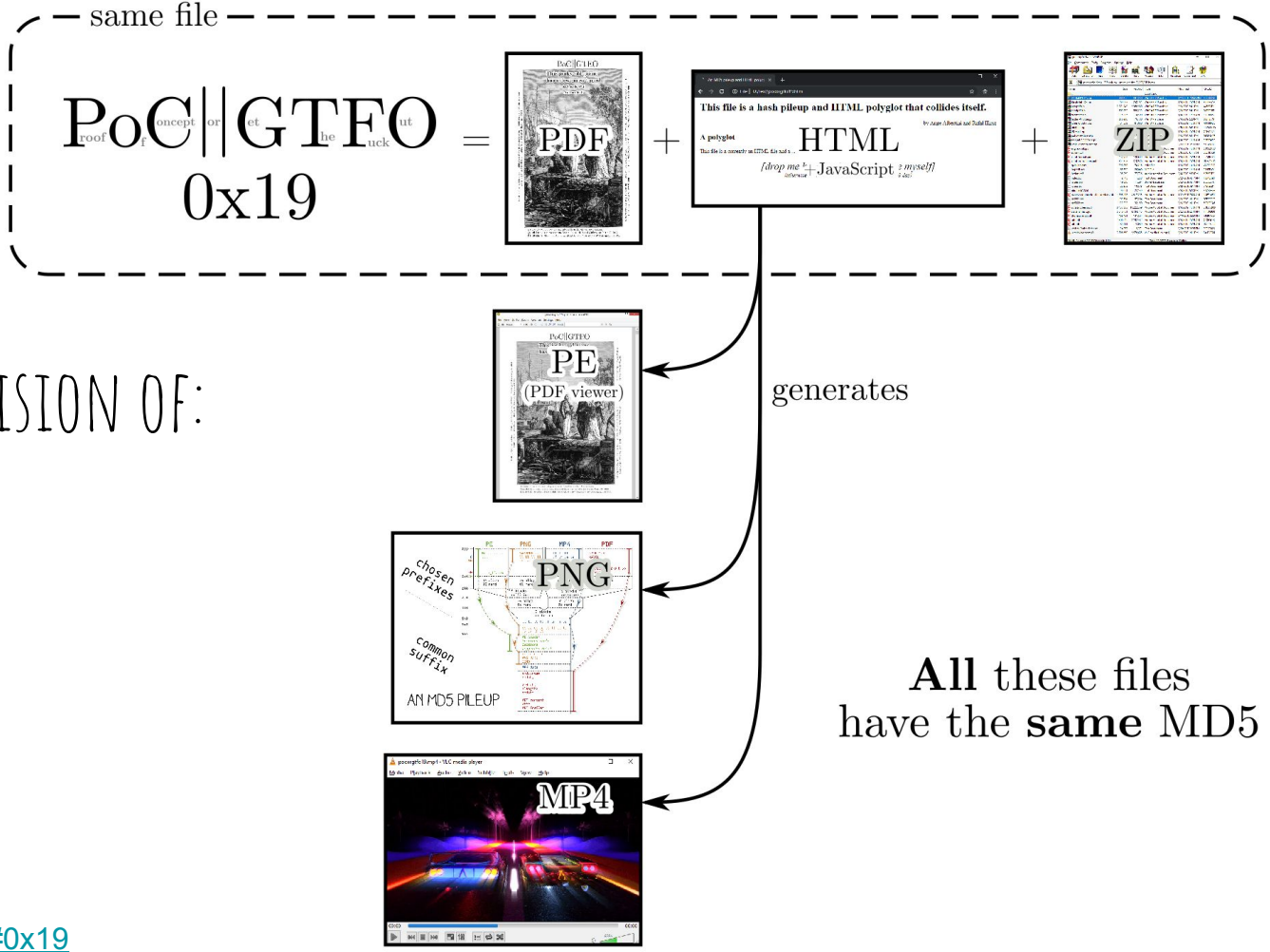
Hash collisions are sometimes perceived as:

- only applying to a pair of files.
- only applying to the same file type.

So what about...

An instant & generic polyglot collision tree





An instant collision of:

- a document
- an executable
- an image
- a video.

https://github.com/angea/pocorgtfo#0x19

Don't be fooled: shortcuts are necessary

Instant & reusable collisions rely on attacks **and** file formats tricks.

Some formats have no suitable tricks.

-> no generic collisions for ELF, Mach-O, ZIP, TAR, Class.

These tricks will be re-usable with **future** collision attacks:

the same JPEG trick was re-used with 3 hash collisions (MD5, MalSHA1, SHA1)

HOW?

INSTANT COLLISIONS COMBINES STANDARD ABUSES TECHNICS.

(NOT EXCLUSIVE TO COLLISONS)

Normalizing content.

Hosting 'parasite' data.

Abusing parsers tolerance.

It's a good exercise for your hacking skills.

All existing hash collision attacks

	Type	Exploitability	Definition	Implementation
MD5				
- FastColl: a few seconds.	IPC	hard	2009	2009
- UniColl: a few minutes.	IPC	easy	2012	**2017**
- HashClash: a few hours.	CPC	easy	2009	2009
SHA1				
- Shattered: a few thousand years	IPC	easy	2013	**2017**
- Stevens13: ?	CPC	easy	2013	?

FastColl: The instant collision

(0.3s at best)

We can put whatever we want **before** and **after** the collision.

We need the following from the target file format:

⇤ ⇥ **Padding**, for alignments

& % ! @ collision blocks' **randomness** need to be ignored

...?... **Differences** need to be taken into account

...  ? **Appended** data needs to be ignored

```
00:  .H .e .r .e .  .i .s .  .a .  .f .i .l .e .  .w
10:  .i .t .h .  .a .  .f .e .w .  .b .y .t .e .s 00
20:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40:  CE 84 07 61 4B BA 7A 3D 3A EA 8A AA F8 EE 1D E5
50:  44 17 9B F0 0A E0 D2 64 21 E2 38 E1 94 18 0A F6
60:  93 D2 B5 E4 FC 2F 3A 32 4F 50 46 01 F1 4B BF 02
70:  23 EE EF BF 92 B5 7C 29 D9 C5 66 08 31 5E 7A 1D
80:  2F 5A 9C 5C 12 8E DF F2 85 17 5B DD 67 25 05 78
90:  13 F2 BF D6 64 59 F2 C8 8B C3 00 6F 8B 5F 88 C6
A0:  CB 3D 80 E4 9F 48 91 5E 34 06 D0 3A 8B 03 FB E0
B0:  ED 18 67 0F C8 3A C9 A1 E7 48 F6 2A D2 5C 30 C0
C0:  we can put whatever we want here, but identical
D0:  ......
```

```
00:  .H .e .r .e .  .i .s .  .a .  .f .i .l .e .  .w
10:  .i .t .h .  .a .  .f .e .w .  .b .y .t .e .s 00
20:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40:  CE 84 07 61 4B BA 7A 3D 3A EA 8A AA F8 EE 1D E5
50:  44 17 9B 70 0A E0 D2 64 21 E2 38 E1 94 18 0A F6
60:  93 D2 B5 E4 FC 2F 3A 32 4F 50 46 01 F1 CB BE 02
70:  23 EE EF BF 92 B5 7C 29 D9 C5 66 88 31 5E 7A 1D
80:  2F 5A 9C 5C 12 8E DF F2 85 17 5B DD 67 25 05 78
90:  13 F2 BF 56 64 59 F2 C8 8B C3 00 6F 8B 5F 88 C6
A0:  CB 3D 80 E4 9F 48 91 5E 34 06 D0 3A 8B 83 FB E0
B0:  ED 18 67 0F C8 3A C9 A1 E7 48 F6 AA D2 5C 30 C0
C0:  we can put whatever we want here, but identical
D0:  ......
```

Instant *computation* is not enough.

The only instant collision computation
generates too much randomness.

-> too many restrictions for most file formats.

-> instant collision needs more than instant computation.

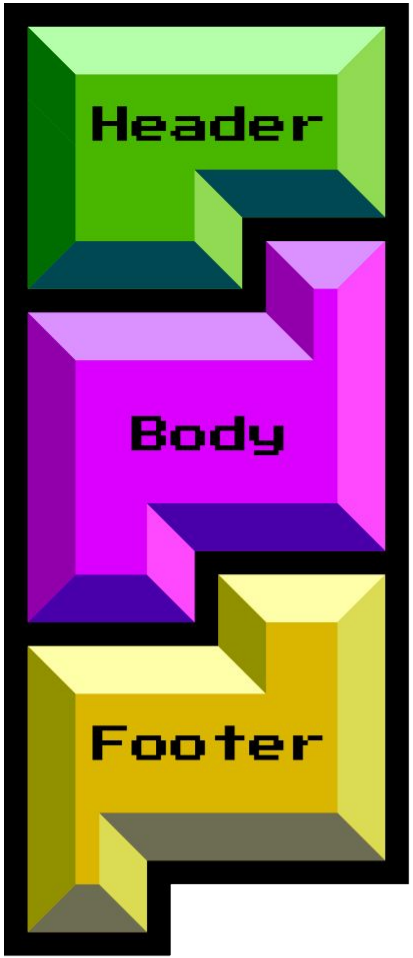
Plan something re-usable with pre-computed values.

The general structure of file formats

header: at the start of the file.
It defines the file type, versions, and metadata.

body comes after. made of several sub-elements.

footer follows the body.
indicates that the file is complete.
Any data is usually ignored after.

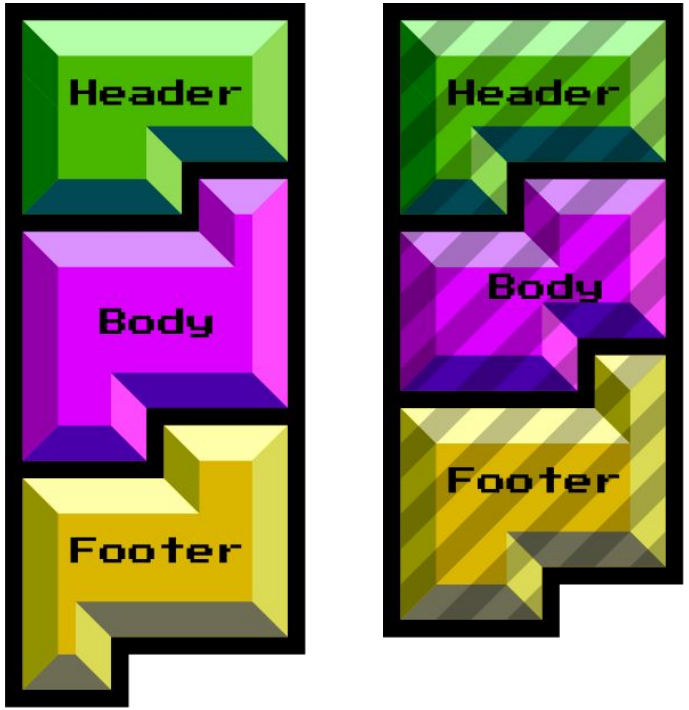


How to make a reusable collision attack

1. Pick a specific file format.
2. Find a normalized form of the file format (same **header** structure):
   most files can be turned into this form but still render the same.
3. Pre-compute the start of the files to match this form.
4. use the differences in the computed collision
   to hide the different **bodies** of each files.

Take two files.

(of the same file type)



Plan a special common header.

Same images dimensions? Color space?
Remove some features.
Flatten content.

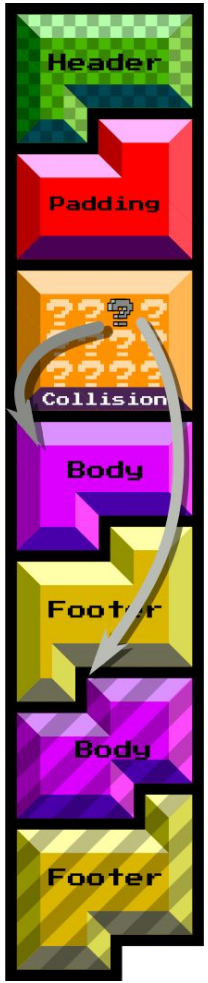
...

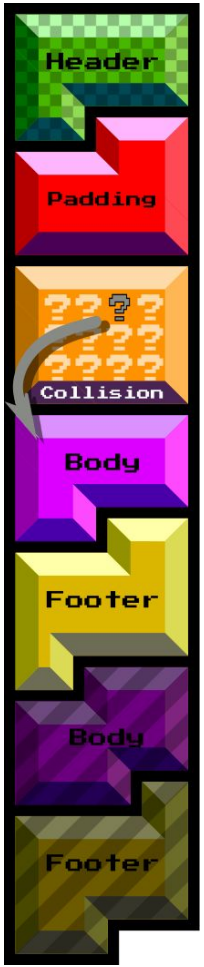
Compute the collision for this header.

Padding and randomness with tiny differences.
These differences follow some patterns
that will be abused.
Margin errors have to be mitigated.

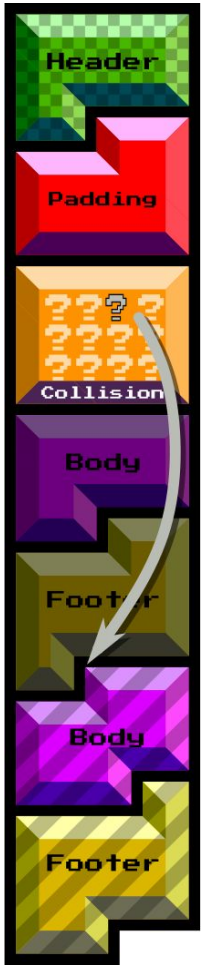
Create a super file combining two files' data.

Both files' Body and Footer are kept original.
The header has to be a common ground.

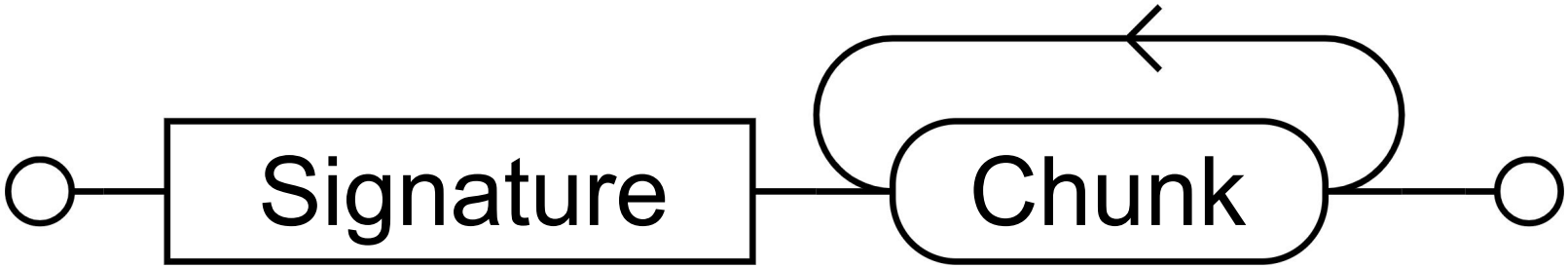




Find a way
to make the collision
work with the file format.



Formats are made with specific structures



For example, a PNG image is made of:
a signature then a sequence of chunks.

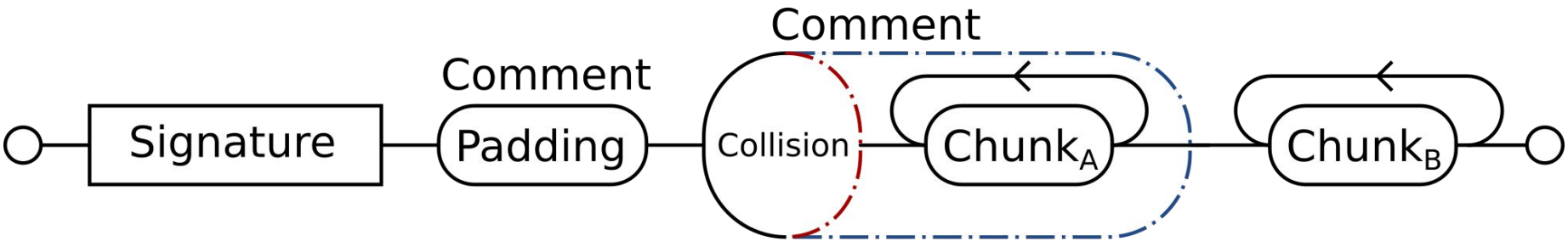
Comment chunks

Abuse comment chunks as placeholders for foreign data.

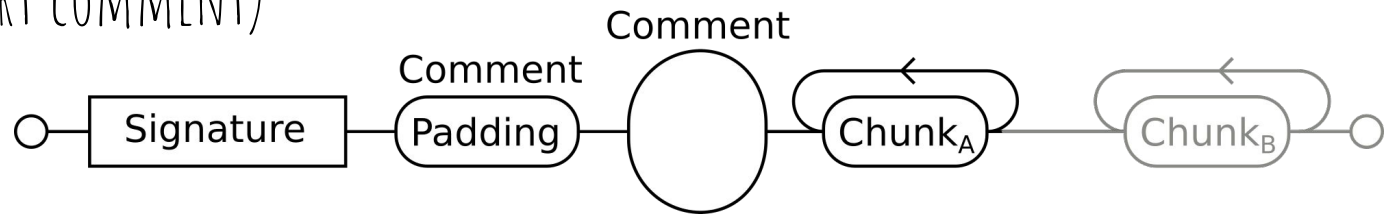
Their length is declared before their content.
-> "ignore the next X bytes please".

A variable-length comment chunk

Overlap the declared length of one comment and one of the collision differences.

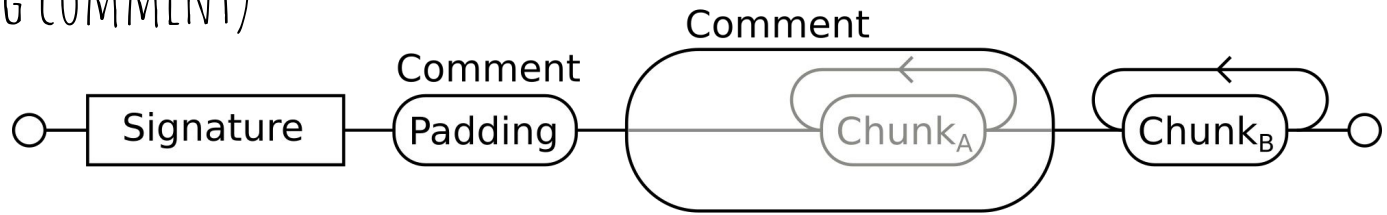


Case A (short comment)



Since $Chunk_A$ defines a complete file, $Chunk_B$ is ignored.

Case B (long comment)



$Chunk_A$ is commented out.

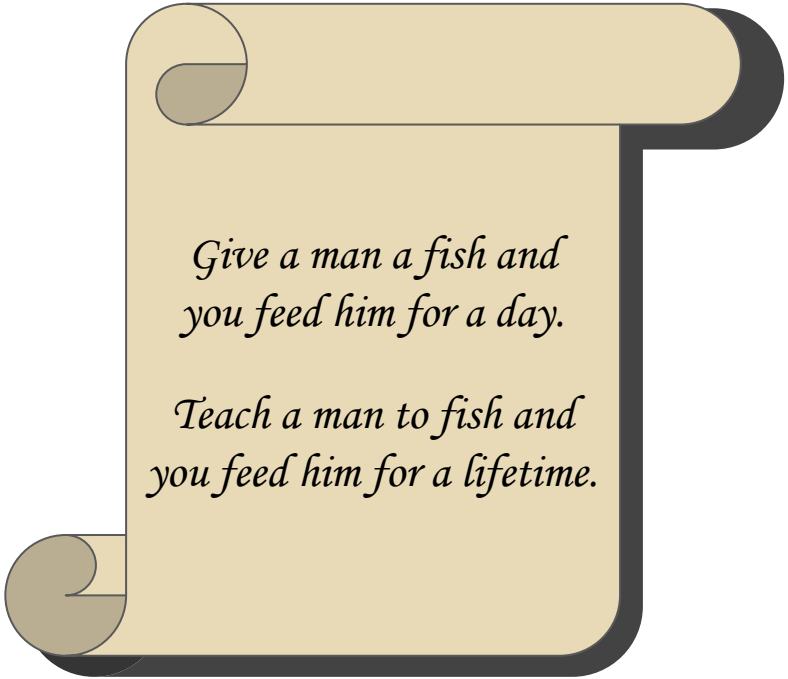
How to prevent such exploits

- **At specs level** (for the next format)
  - Enforced file size / structure length / parent length / CRC
  - Comments only once, after all critical structures.
- **At parser/sanitizer level** (still implementable)
  - Limit comments: AlphaNum/UTF8-only. Size limited.
  - Forbid appended data.


Need some practice?

Theory < PoCs < scripts < workshop

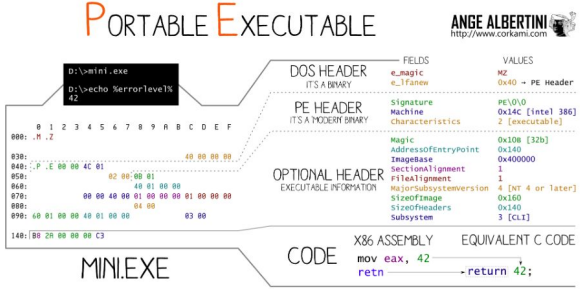
if it's free, open and accessible,
it will reach a lot more people!



https://github.com/corkami/collisions

- Attacks
- Exploitations
- Strategies
- Use cases
- Failures
- Test files

Portable Executable



The Portable Executable has a peculiar structure:

- the old DOS header is almost useless, and points to the next structure, the PE header. The DOS headers has no other role. DOS headers can be exchanged between executables.
- the DOS header has to be at offset 0, and has a fixed length of a full block, and the pointer is at the end of the structure, beyond UniColl's reach: so only chosen-prefix collision is useful to collide PE files this way.
- The PE header and what follows defines the whole file.

So the strategy is:

1. the PE header can be moved down to leave room for collision blocks after the DOS header.
2. The DOS header can be exploited (via chosen-prefix collisions) to point to two different offsets, where two different PE headers will be moved.
3. The sections can be put next to each other, after the `DOS/Collisions/Header1/Header2` structure. You just need to apply a delta to the offsets of the two section tables.

This means that it's possible to instantly collide any pair of PE executables. Even if they use different subsystems or architecture.

While executables collisions is usually trivial via any loader, this kind of exploitation here is transparent: the code is identical and loaded at the same address.

Examples: tweakPNG.exe (GUI) ↔ fastcoll.exe (CLI)

Here is a script to generate instant MD5 collisions of Windows Executables.

Collisions examples

MD5

FastColl

single frame GIF: collision1.gif / collision2.gif

UniColl

JPG: collision1.jpg / collision2.jpg - tldr-1.jpg / tldr-2.jpg

PDF: collision1.pdf / collision2.pdf

PNG:

- generic headers (not OS X compatible): collision1.png / collision2.png
- specific headers (same metadata): 0a959025-1.png / 0a959025-2.png - aac2423a-1.png / aac242

JP2: collision1.jp2 / collision2.jp2

MP4:

- generic header, 32b length (LTV): collision1.mp4 / collision2.mp4
- generic header, 64b length (TLV): collisionl1.mp4 / collisionl2.mp4
  - specific header: collisions1.mp4 / collisions2.mp4

Strategies:

- Good/bad contents (gotta catch 'em all): gcea1.png / gcea2.png
- Valid/Invalid: png-valid.png / png-invalid.png

Multiple UniColl

poeMD5 (not Adobe compatible by accident): poeMD5_A.pdf / poeMD5_B.pdf

ZIP: collision1.zip / collision2.zip

HashClash

PE: collision1.exe / collision2.exe

polycolls

- JPG / PE: jpg-pe.exe / jpg-pe.jpg
- PE / PDF: pepdf.exe / pepdf.pdf
- PNG / PDF: png-pdf.pdf / png-pdf.png

My page about hash collisions

docs, scripts+precomputed collisions, test PoCs...

My (free) workshop on the topic

http://speakerdeck.com/ange/colltris

HASH COLLISION EXPLOITATION

A workshop by

Ange Albertini

with the help of

Marc Stevens

A.K.A.

CONCLUSION

Hash

A big fixed-size value associated to any content.

One way only: can't find content from hash.

Very different with tiny changes.

used to index stuff.

ex: your pictures in the cloud.

used to check passwords:

take input, compute hash,

compare with previously stored value.

Hash collision

In case you just jumped to the conclusion

Creating 2 files with the same hash.

Hash collision attack:

Collide 😇 with 😈.

Now you have a 😇 and a 😈 with the same hash.

Send 😇 to your target, get it whitelisted.

(its hash is now stored on a "good" list).

Now 😈 can be used transparently.

Its hash is already on the list!

You could even collide **any** file on the fly.

Hash collisions FAQ

Collisions are full of randomness: it's **impossible to match a given hash**.

The **final** hash of a collision is **unknown** in advance.

The sizes of the files to be collided have **no** influence on the computation.

MD5 can be instant. SHA1 is doable but expensive. MD5+SHA1 is not much better.

SHA2 family is still much stronger.

2^{61} on SHA1 -> 2^{69} on MD5+SHA1 (cf Joux04)

Colliding standard files can be trivial and instant.

Don't play with fire, don't use MD5.

https://gunshowcomic.com/648

MD5 IS

~~a cryptographic hash~~

a toy function

...have fun!

Mako's "Toy MD5 Collider" for the Mega Drive

https://www.makomk.com/~aidan/selfmd5-release.zip

dd49d7eb47db5c970ccab1746f3233cb272f25d884b53f899cb47460d3aa7f1b

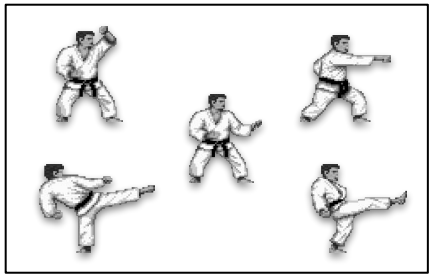
1988: Sega Mega Drive/Genesis - 1992: MD5

"Even..."! A Chosen-Prefix Collision is not enough to kill a hash.

Threats? theory...
Exploits PoCs? reality!

immediate threat

Theoretical attacks to put in practice



Old is not useless

Older attacks can be reused
with new tricks and have new **IMPACT**!

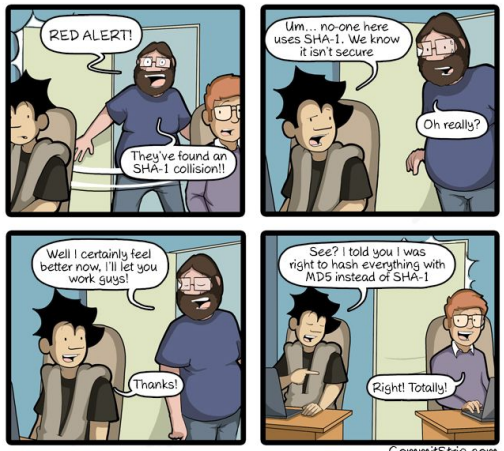
New tricks can be reused with several attacks.

(including future ones)

Remember

It's *our* job to go out there,
to *show* the risks
and educate users & devs.

Kill MD5, wherever it may hide!



http://www.commitstrip.com/en/2017/02/27/the-sha-1-alternative/

Thank you!

Any feedback is welcome!

KILL MD5

Special thanks to:

Doegox, BarbieAuglend, Slurdge, Cryptax, Cryptopathe, Noutoff, Agarri.

ANGE ALBERTINI

reverse engineering

VISUAL DOCUMENTATIONS

@angealbertini

ange@corkami.com

http://www.corkami.com

To get the workshop slides,
take this deck file.

Rename it as .html,
open it in a browser.
(it's a polyglot)

Drop the file on itself,
get the workshop slide deck.
(Both decks have the same MD5)