MD5 considered harmful today

Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger

Presentation by: Hamidhasan G. Ahmed
Outline

● Why do we want third-party certification?
● Why such an attack has practical implications
● Recap of hashing functions and MD5
● Collision resistance, and its failure
● A closer look at certificate structure
● Constructing the fake CA
Why do we want third-party certification?

- Three players: user, merchant, and Certification Authority (CA)
- User wants to have secure transaction with merchant, and be sure that they are not on a spoofed website
- Merchant obtains certificate from third-party as evidence to demonstrate authenticity
- User’s browser can verify evidence using CA-issued verifier
- Result: User sleeps well at night, transaction secured.
How to attack the certification model

1a. The attacker requests a web site certificate from the Certification Authority (CA).
1b. The attacker creates a rogue CA certificate and signs it.
2. The attacker distributes the rogue CA certificate.
3. The user at PC with browser requests a secure web site certificate.

The diagram illustrates the steps involved in the attack and the different entities involved, such as the secure web site, rogue web site, user at PC with browser, browser vendor, and CA root certificate.
Recap: Hashing functions and desired traits

If \( m, m' \) are messages, \( H \) is a hashing function, and \( h = H(m) \), we want:

- **Efficient hashing**: \( H \) should be able to compute \( h \) in a reasonable amount of time.
- **Preimage resistance**: If we are given \( h \), it should be hard to calculate the initial string \( m \) (\( m \) is the "preimage" of \( h \)).
- **Second preimage resistance**: If we are given \( m \), it should be hard to calculate a second \( m' \neq m \) such that they both produce the same hashed value \( h = H(m) = H(m') \).
- **Collision resistance**: If we have \( H \), it should be hard to calculate a pair of messages \( m, m' \), such that \( H(m) = H(m') \) but \( m \neq m' \).
- Collision resistance is the vulnerability exploited in this attack.
Merkle-Damgård and MD5

- MD5 uses a deterministic compression function with several shifts, XORs, and other fun transformations at each stage.
- The compression function outputs an intermediate hash value - IHV - that is then fed as an input to the next iteration of the function.
The armor begins to crack: First vulnerability (2004)

- Discovered by Xiaoyun Wang and Hongbo Yu, in 2004.
- They constructed two 512-bit blocks C and C’, C=/=C’, such that, on an arbitrary iteration i of MD5, which has some intermediate value IHV_i, H(C, IHV_i) = H(C’, IHV_i) = IHV_i+1
- Due to the construction and “chunking” of MD5, this block can be inserted into a bigger message with equal prefixes and suffixes.
- Not practical, but ominous...
“Chosen-prefix” collisions: choosing doom for MD5 (2007)

- Expansion on previous collision by Marc Stevens
- Choose two arbitrary prefixes P and P'. Construct padding A and A' to align them, then use “birthdaying” to create B and B'. P,A,B and P',A',B' combined form i blocks.
- Birthday step gets the IHV’s into a special, close form, but not exact. The differences between the IHVs are then eliminated using “near-collision blocks”.
- Chalkboard Explanation Time
How do we utilize chosen-prefix collisions meaningfully?

- Need to find some way to change prefixes of a file (which has some specific formatting), and then construct near-collision blocks that will:
  - guide MD5 iterations towards a collision (previous slide)
  - appear meaningfully in the file format that you want to forge
  - be able to be controlled by you, and not have them be random
- Case study: Digital signatures and certificates.
A closer look at certificate structure

Two main parts:

- “To-be-signed” portion, the data/information about the certificate.
  - This has information such as the serial number, validity period, who is the issuer and what is his/her public key, who issued the certificate, etc.
  - This is the part an attacker wants to forge, by inserting their malicious information/website, but having it “issued by” a valid CA with a valid serial number, etc.

- “Signature” portion, containing a digital signature over the “to-be-signed” portion.
  - This is produced by the CA’s secret key.
  - It is checked by the browser, who has the CA’s public key.
  - Used by browser to check validity of “to-be-signed” portion.
“Since [the CAs using MD5] had ignored all previous warnings by the cryptographic community, we felt that it would be appropriate to attempt a practical attack to demonstrate the risk they present to everybody using a web browser that includes their root CA certificates.”

(mostly RapidSSL)
Forging a certificate

- “Prefix” portion is the “to-be-signed” part of the certificate.
- In order to forge it correctly, we first need a valid certificate.
- We need to know the entire prefix of this valid certificate beforehand...
- ...thus, it is necessary to predict valid start and end times, and predict a valid serial number
- Start and end times were simple: RapidSSL set the start time as exactly 6 seconds after you clicked the button, and the end time as 1 year and 6 seconds after
- Serial number a bit trickier:
  - RapidSSL keeps an internal, global counter. When it issues a certificate, its value becomes the serial number, and then increments it by one.
  - Needed to measure how many certificates were being issued over a 3-day period, then get the desired serial number by buying several certificates to hit the selected number.
Forging a certificate, continued

- That takes care of the prefix. But how will we construct the near collision blocks in the valid certificate?
- The certificate also encodes the client’s public key. Since the public key looks like random data to begin with, it is used to include (some of) the near collision blocks.
- The rest of them are constructed using “extension” information, which are essentially random bits of information that can be added onto the certificate.
- In reality, this public key will never really be utilized - it is just part of the signature that we will use to create a collision between a certificate that we generate.
An example breakdown

Complexity Analysis

- “Birthday” step: biggest computational obstacle to constructing a chosen-prefix collision
- Using 200 PS3 game consoles, it took “a little more than a day” for a single attempt at trying to construct one
- Around $2^{51.x}$ MD5 compression calls in total
Set the system clock back in time...

...and let’s check out the forged website!