MD5 considered harmful today[1]

Creating a rogue CA certificate

By Alexander Sotirov 외 6 명
편역: vangelis(securityproof@gmail.com)

요약

안전한 웹 사이트에 대해 디지털 인증서(digital certificate)를 발급하는데 사용되는 PKI(Internet Public Key Infrastructure, 인터넷 공개키 기반구조)의 취약점 하나를 우리는 확인했다. 이 취약점을 입증하기 위해 우리는 실제 공격 시나리오를 실행했으며, 모든 일반 웹 브라우저들이 신뢰하는 가짜 CA(Certification Authority)[2]를 성공적으로 만들었다. 이 인증서는 HTTPS 프로토콜을 이용하는 안전한 금융 및 전자상거래 사이트들을 포함해 인터넷 상의 어떤 웹 사이트도 우리가 진짜인 것처럼 만들 수 있다.

우리의 공격은 같은 MD5 해쉬[3]와 함께 다른 메시지들을 만들 수 있도록 하는 MD5 암호 해쉬 함수에 존재하는 취약점을 이용한다. 이것은 'MD5 충돌'(MD5 collision)로 알려져 있다. MD5 충돌에 대한 2004년과 2007년 사이에 있었던 이전 작업들은 디지털 서명에서 MD5 해쉬 함수를 사용하는 것은 이론적으로 공격 시나리오로 이어질 수 있다는 것을 보여주었다. 우리의 현재 작업은 적어도 하나의 공격 시나리오가 실제로 가능하며, 그래서 웹 보안의 인프라구조가 실제 위협에 노출될 수 있다는 것이다.

성공적인 이 공격의 결과로서 우리는 현재 가짜 CA 인증서를 소유하고 있다. 이 인증서는 유효한 것으로 받아들여지고, 모든 일반 브라우저들이 신뢰할 것인데, 이는 브라우저가 기본적으로 신뢰하는 root CA들 중의 하나에 의해 서명되기 때문이다. 따라서 우리의 가짜 CA에 의해 서명된 웹 사이트의 인증서도 신뢰받을 것이다. 만약 의심을 받지 않는 어떤 사용자가 그와 같은

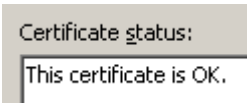
---

[1] (역자 주) 이 글은 Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger가 25회 Chaos Communication Congress에서 발표한 것이다. 용어 설명이나 독자의 이해를 위해 필요한 경우 역자가 별도의 각주를 달았다.
원문은 http://www.win.tue.nl/hashclash/rogue-ca/에서 볼 수 있으며, 오역(오자)이 있을 경우 메일 부탁 드립니다.

[2] CA(인증기관)은 보안적격 여부와 그리고 메시지의 암호화와 복원을 위한 공개키들을 발급하고 관리하는 네트웍 상의 기관이다. 공개키 기반구조의 일부로서, 인증기관은 디지털 인증서 요구자에 의해 제공되는 정보를 검증하기 위한 등록기관(RA)과 함께 안전성 등을 검사한다. 만약 등록기관이 요구자의 정보를 입증하면, 인증기관은 인증서를 발급할 수 있다. (http://www.terms.co.kr에서 발췌)

[3] (역자 주) 일반적으로 해쉬(hash(ing))란 하나의 문자열을 원래의 것을 상징하는 더 짧은 길이의 값이나 키로 변환하는 것을 가리킨다.

인증서를 사용하는 man-in-the-middle 공격의 희생자라면 모든 일반 보안 지시자들(주소 창의  "https://"나 인증서를 확인하기를 선택했다면 "This certificate is OK"와 같은 메시지)은 그 연결이 안전한 것으로 확인해줄 것이다.



이 성공적인 proof of concept은 브라우저에 의해 실행되는 인증서 유효성 확인이 전복될 수 있고, 안전한 웹 사이트로 보내지는 데이터를 악의적인 공격자들이 감시하거나 변경할 수 있다는 것을 보여준다. 금융권과 전자상거래 사이트들이 특히 위험에 빠질 수 있는데, 왜냐하면 이런 사이트에는 높은 가치를 가진 정보들이 HTTPS로 보안이 세워졌기 때문이다. 공격자는 가짜 CA 인증서를 이용해 그와 같은 사이트에 실제로 탐지가 되지 않는 phishing 공격을 할 수도 있다.

CA의 인프라구조는 이런 형태의 공격을 정확하게 막도록 되어 있다. 우리의 작업은 MD5 해쉬 함수에서 알려진 취약점이 실제 공격을 당할 수 있다는 것을 보여주는데, 이는 MD5의 보안 문제에 대해 몇 년 동안 경고를 해왔는데도 불구하고 몇몇 root CA들은 여전히 문제가 있는 해쉬 함수를 사용하고 있기 때문이다.

우리가 발표한 이 취약점은 SSL 프로토콜 또는 그것을 구현한 웹 서버나 브라우저에 있는 것이 아니라 공개키 기반구조에 있다. 이 공개키 기반구조는 웹 보다는 다른 영역에 적용되지만 우리는 다른 가능한 공격 시나리오들을 모두 연구하지는 않았다. 그래서 코드 서명, 이메일 보안에서, 그리고 디지털 서명을 가능하게 하는 인증서서와 공개키 암호화를 사용하는 다른 영역에서처럼 웹 이면의 다른 공격 시나리오들을 상상할 수 있다.

이 문서의 나머지는 우리의 작업과 그것의 밀접한 관계를 자세하게 설명할 것이다. 우리의 공격 기술을 이용하는 악의적인 공격들로부터 인터넷을 보호하기 위해 MD5 충돌을 연산하는 우리의 복잡하고도 고도로 활용될 수 있는 방법의 중요한 세부 내용은 생략했다. 우리가 개발한 방법에 대한 학술 논문이 준비 중이며, 몇 달 후에 공개될 것이고, 그래서 관련 인증 기관들이 이 취약점을 고칠 시간적 여유를 가질 것이다.

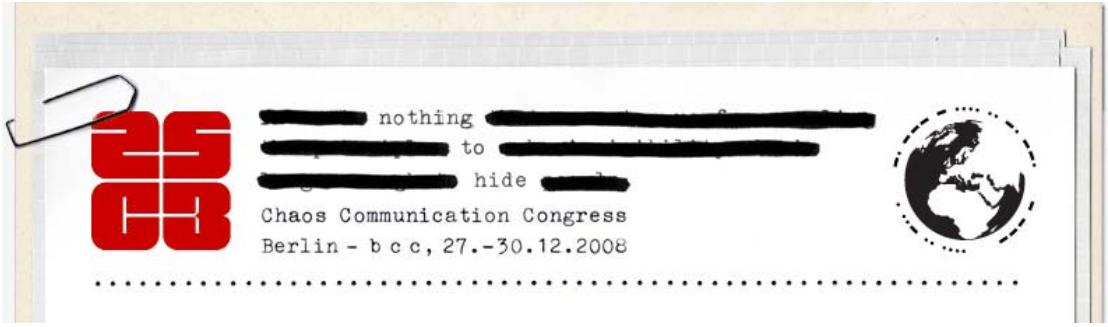
문서 구성

- 아래 섹션1은 우리 작업의 개요를 제시한다.
- 우리 연구의 결과를 완전히 이해하기 위해 몇 가지 기본적인 지식이 필요하다. 그래서 섹션2에서 우리는 인증서에 대한 몇 가지 배경 지식을 제공하고, 섹션3에서는 해쉬 함수에 대해 필요한 내용을 기술할 것이다. 이 주제들에 대해 잘 아는 독자라면 이 부분들을 그냥 건너뛰어도 된다.

- 현재 작업은 지난 몇 년간 우리 작업의 지속이라고 볼 수 있는데, 그 작업 과정이 섹션4에서 간단하게 기술될 것이다.
- 섹션5는 우리의 방법을 자세하게 기술할 것이다. 기술적인 주제에 대해 관심이 없는 독자들은 이 부분을 건너뛰어도 된다.
- 공격 시나리오들이 미칠 수 있는 영향과 대응책을 섹션6과 섹션7에서 각각 다룬다
- 섹션8에서는 FAQ로 결론을 내린다.

프리젠테이션

이 작업은 베를린에 열린 **25th Annual Chaos Communication Congress**[4] 에서 2008년 12월 30일에 발표되었다.



이 프리젠테이션은 원래 "Making the theoretical possible. Attacking a critical piece of Internet infrastructure"로 발표되었으나 최종적으로 "MD5 considered harmful today: creating a rogue CA certificate"[5]라는 제목을 붙였다.

[2009년 1월 8일 추가] 25C3에서의 발표 비디오와 오디오 파일들[6]은 CCC에서 다른 포맷으로 구할 수 있다. "25c3-3023-en-making_the_theoretical_possible.xyz"라는 이름을 가진 파일을 찾아보아라.

25c3에서 우리가 보여준 데모는 Moxie Marlinspike가 만든 sslsniff[7]에 기초를 두고 있다. 우리는 그것을 좀더 verbose하고(데모로서 좀더 시각적이길 바라면서), 시리얼 번호들을 랜덤화하기 위해(Firefox가 가진 문제를 해결하기 위해), 그리고 사용자가 선택한 해쉬 함수로 서명하기 위해(이것은 leaf certificate에서 SHA-1 서명을 허용했다) 약간 수정했다. Jacob Appelbaum이 발표 시작 때 이 툴을 그가 작성했다고 잘못 말했지만 동영상의 끝부분을 보면 이 툴의 출처에 대해 그가 정확하게 설명하고 있는 것을 볼 수 있다.

---

4 http://events.ccc.de/congress/2008/

5 http://www.win.tue.nl/hashclash/rogue-ca/downloads/md5-collisions-1.0.pdf

6 http://events.ccc.de/congress/2008/wiki/Conference_Recordings

7 http://thoughtcrime.org/software/sslsniff/index.html

1. 공격의 요점

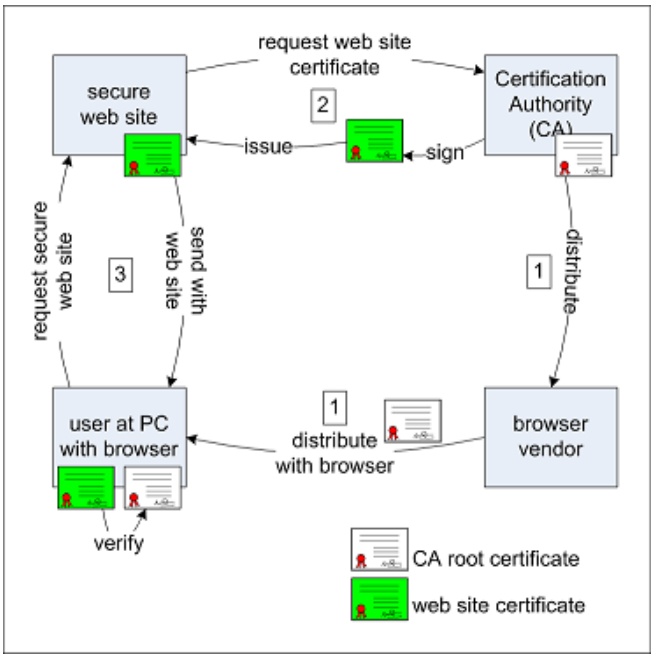
우리의 공격 시나리오는 기본적으로 다음과 같다. 우리는 모든 일반 브라우저들이 신뢰하는 상용 CA로부터 합법적인 웹 사이트의 인증서를 요청한다. 그 요청이 합법적이기 때문에 CA는 우리의 인증서에 서명을 하고, 그것을 우리에게 돌려준다. 인증서의 서명을 생성하기 위해 MD5 해쉬 함수를 사용하는 CA를 우리는 선택했는데, 우리의 인증서 요청이 두 번째 인증서를 가지고 MD5 충돌을 일으키도록 조작되었기 때문에 MD5 해쉬 함수를 사용하는 CA를 선택하는 것은 중요하다. 이 두 번째 인증서는 웹 사이트 인증서가 아니라 우리가 발행하고자 원하는 임의의 다른 웹 사이트 인증서를 서명하기 위해 사용될 수 있는 중계 CA 인증서이다. CA로부터 받은 합법적인 인증서와 가짜 인증서 둘 다 MD5 해쉬가 동일하기 때문에 상용 CA로부터 구한 디지털 서명은 가짜 CA 인증서로 복사될 수 있으며, 그것은 유효한 것으로 남게 될 것이다.

다음은 웹 사이트의 인증서가 작동하는 방식을 보여주는 도표와 그에 대한 설명이다.

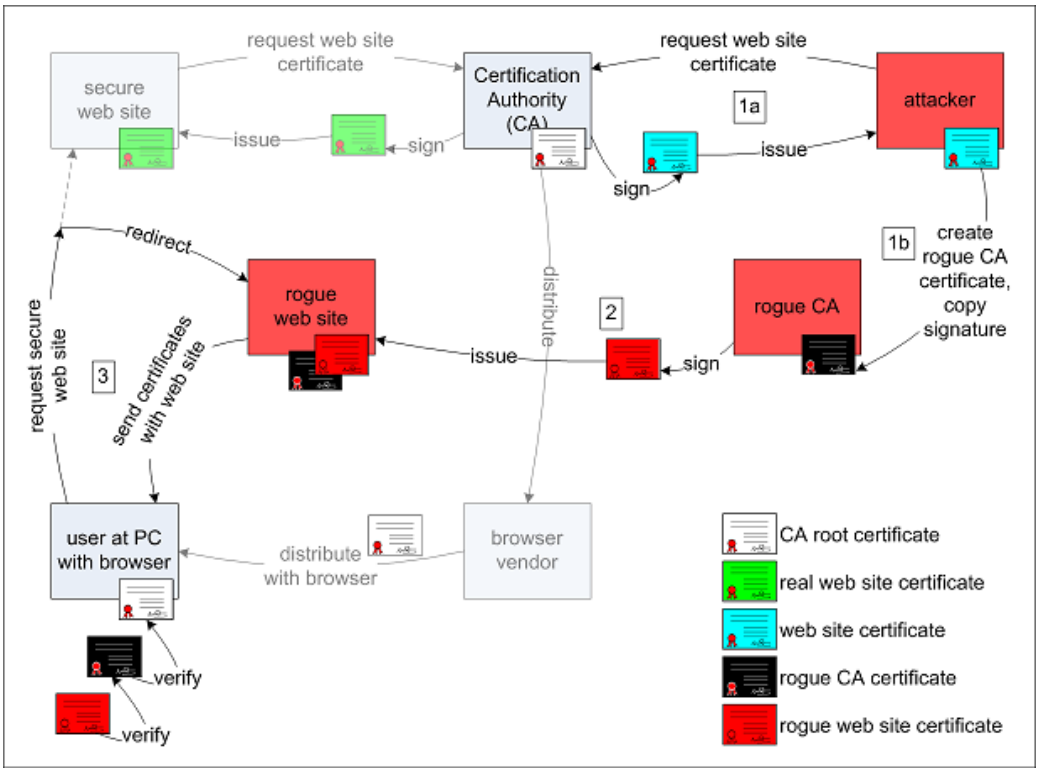
1. CA가 브라우저 벤더들을 통해 브라우저에 CA root 인증서(도표에서는 흰색)를 배포한다. 이 root 인증서는 사용자 PC의 '신뢰 목록'(trust list)에 상주한다. 이것은 이 CA에 의해 발급된 모든 인증서들을 사용자들이 기본적으로 신뢰할 것이라는 것을 의미한다.

2. 안전한 웹 사이트를 원하는 어떤 회사는 CA(도표에서는 녹색)에서 웹 사이트 인증서를 구입한다. 이 인증서는 CA에 의해 서명되었으며, 사용자들에게 그 웹 사이트가 진짜임을 보장한다.

3. 사용자가 어떤 웹 사이트가 안전한지를 확인한 후에 그 사이트를 방문하기를 원한다면 웹 브라우저는 먼저 웹 서버에 인증서를 요구한다. 만약 그 인증서의 서명이 신뢰 목록에 있는 CA 인증서로 확인될 수 있다면 그 웹 사이트의 인증서는 받아들여진다. 그런 다음 그 웹 사이트는 브라우저에 로딩될 것이며, 브라우저와 그 웹 사이트 사이의 모든 트래픽은 암호화를 사용해 안전해질 것이다.



다음은 우리의 공격 시나리오가 어떻게 기존의 웹 사이트를 사칭(impersonate)하는지를 보여주는 유사한 도표와 설명이다.



(1a) 상용 CA(도표에서는 푸른색)로부터 합법적인 웹 사이트의 인증서를 구입한다.

(1b) 가짜 CA 인증서(도표에서는 검은색)가 만들어진다. 그것은 웹 사이트의 인증서와 정확하게 같은 서명을 가지고 있다. 그래서 그것은 CA에 의해 발행된 것처럼 보이지만 사실 그 CA는 그것을 발행한 적이 결코 없다.

(2) 그런 다음, 진짜 웹 사이트의 아이덴터티이지만 또 다른 하나의 공개 키를 가진 웹 사이트의 인증서(도표에서 빨간색)가 만들어지고 가짜 CA에 의해 서명된다. 진짜 웹 사이트의 복사본이 만들어진 다음 다른 웹 서버에 올려지고, 결국 이 웹 사이트는 가짜 웹 사이트 인증서를 가지고 있다.

(3) 사용자가 안전한 웹 사이트를 방문하기를 원할 때 웹 브라우저는 인터넷 상의 진짜 웹 서버를 찾을 것이다. 하지만 "리다이렉션 공격(redirection attack)"을 통해 그 브라우저는 가짜 웹 사이트로 리다이렉트될 수 있다. 이 가짜 웹 사이트의 인증서는 사용자에게 가짜 CA 인증서와 함께 제공된다. 가짜 웹 사이트 인증서에 있는 서명은 가짜 CA 인증서로 확인될 수 있으며, 그것의 서명이 신뢰 목록에 있는 CA root 인증서로 확인될 수 있기 때문에 이 가짜 CA 인증서를 브라우저는 받아들일 것이다. 따라서 사용자는 어떤 것도 확인하지 못할 것이다.

이 취약점을 고치기 위한 효율적인 대책은 디지털 서명을 하는데 사용되고 있는 MD5를 사용하지 않는 것이다. MD5를 대체할 수 있는 SHA-1과 같은 대안들도 널리 보급되어 있으며, SHA-2와

같은 것을 이용하는 것도 빠르게 가능해지고 있다. 다행스럽게도 대부분의 CA들은 그와 같은 단계를 이미 취하고 있다. MD5를 지속적으로 사용한다면 심각한 보안 문제로 이어질 수 있기 때문에 인증서를 만들 때 MD5 사용을 멈추는 것이 최선이다. MD5 사용을 당장 멈출 수 없는 경우 최후의 수단으로 CA는 새로 발급된 인증서들의 시리얼 넘버를 랜덤화하는 것과 같은 간단한 대책을 취할 수 있다. 하지만 그와 같은 대책이 기본적인 해결책이라기보다는 단순히 미봉책에 불과하다는 것을 우리는 강조하고 싶다.

우리는 또한 SHA-1을 사용할 때도 역시 비슷한 주의를 기울여야 하며, 새로운 제품에 SHA-1을 채택하는 것을 피하기를 권고하는데, 왜냐하면 이론적으로 SHA-1은 MD5와 유사한 문제를 가지고 있는 것을 보여주었기 때문이다. 지금까지 알려진 SHA-1의 취약점이 지금 현재 MD5에 가능한 것과 같은 형태의 공격 시나리오에 적용되지는 않지만, 이 분야의 전문가들은 SHA-1가 오래지 않아 MD5와 같은 운명을 겪을 것이라고 믿고 있다.

2. 인증서 배경

2.1. 인증서의 필요성

점점 더 웹 서버는 민감한 정보를 교환하는데 사용되고 있다. 인터넷 뱅킹 웹 사이트, 로그인 패스워드를 필요로 하는 웹 사이트, 재정적 가치를 처리하는 웹 사이트 등에 대해 생각해보자. 그 민감한 정보는 권한이 없는 사람들에게 노출되지 말아야 한다. 민감한 정보를 교환하기 위해 어떤 웹 사이트의 사용자는 그가 방문하는 웹 사이트가 가짜 웹 사이트가 아니라 그가 업무를 처리해야 하는, 조직의 진짜 웹 사이트라는 확신을 원한다. 이것이 바로 인증서가 제공하는 것이다.

인증서는 아이덴터티와 공개키를 디지털 서명에 의해 함께 묶은 것을 포함하고 있는 문서이다. 이 디지털 서명은 CA(Certification Authority)라는 조직에 의해 만들어진다. 이 조직은 디지털 서명을 만들자마자 그 공개키 소유자의 실체(예를 들어, 여권을 확인함으로써)를 확인했다는 것과 이 공개키 소유자가 대응하는 개인키를 가지고 있다는 것을 보증한다.
CA의 공개키를 소유한 사람은 그 인증서에 있는 CA의 서명을 확인할 수 있다. 이런 방식으로 그 인증서의 공개키가 그의 아이덴터티가 같은 인증서에 있는 개인의 것이라는 것을 그 CA는 보장한다.

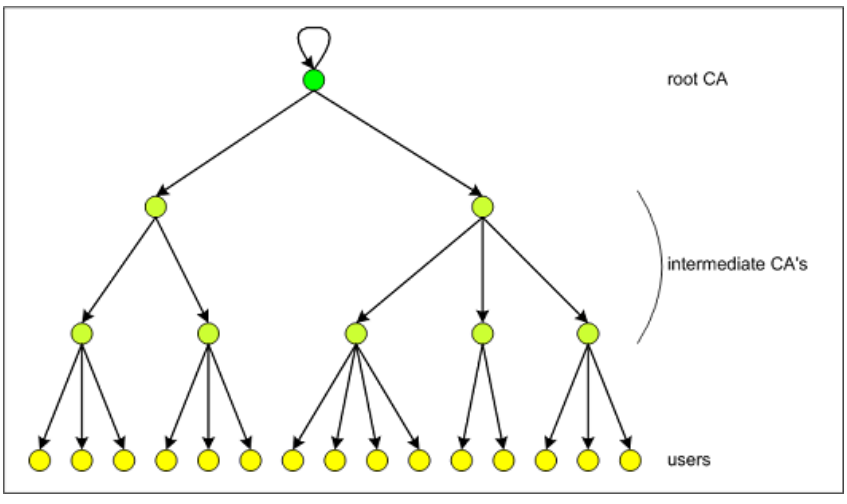
2.2. X.509 인증서

전세계적으로 받아들여지는 디지털 서명의 표준은 X.509이다. X.509 인증서는 다음 부분들을 포함하고 있다.

- 서명될("to-be-signed") 부분 구성:
    - 시리얼 번호
    - 유효기간
    - 발행자 이름 – 인증서를 발행하는 CA의 아이덴터티
    - 주체 이름(subject name) – 인증 받을 단체의 실체; 이것은 사용자(사람, 조직, 웹 사이트), 또 다른 CA, 또는 발행하는 CA 그 자신이 될 수 있음
    - 주체 공개키(subject public key) – 인증 받는 단체의 공개키
    - 기본 제한사항(basic constraint) 필드, 다음 사항을 포함하고 있음
        - CA 인증서 또는 사용자 인증서인지를 나타내는 bit(각각 다음과 같이 표시함 "CA = TRUE", "CA = FALSE")
        - "root CA"와 최종 사용자 사이에 허용되는 중개 CA의 최대수를 나타내는 경로 길이 필드(path length field)
- 서명(signature) 부분, 서명될 부분 위에 디지털 서명을 가지며, 이 디지털 서명은 인증서를 발행하는 CA의 개인키를 사용해 만들어진다.

인증서에 있는 디지털 서명을 이용해 CA는 인증서 안에 언급된 주체의 실체(subject identity)가 실제로 같은 인증서 안에 있는 공개키의 소유자이고, 이 공개키 소유자는 역시 상응하는 개인키를 가지고 있음을 그것의 정책에 따라 확인했다는 것을 보장한다. 인증서 안의 서명은 CA의 공개키를 이용해 누군가 확인할 수 있다.

2.3. 인증서 계층구조



X.509 인증서는 CA와 최종 사용자 인증서들의 계층에 적합하다. 데이터에 있는 서명은 서명자의 공개키에 의해 확인될 수 있다. 이 공개키는 인증서에 의해 그 소유자의 아이덴터티에 링크되어 있다. 이 링크는 발행자 CA의 공개키를 이용해 인증서의 서명을 확인함으로써 확인될 수 있다. 이 CA 공개키는 CA 인증서 내부에서 발견할 수 있는데, 한 층은 계층도에 위쪽에 있다. 이 CA 인증서 그 자체도 또 다른 층 위의 CA에 의해 서명될 것이다. 계층의 제일 꼭대기에 루트 인증서(root certificate)가 있고, 그것은 '자기 서명'이 되어 있으며, 그 자신을 위해 신뢰되어야 한다.

자기 서명된 많은 이 루트 인증서들은 브라우저나 운영체제에 설치되어 있다. 웹 브라우저와 같이 그 루트 인증서를 사용하는 어플리케이션들은 자동으로 그 인증서들을 신뢰할 것이며, 예를 들어, 어플리케이션들은 사용자에게 그래야 하는지 명시적으로 묻지도 않고 인증되고 진짜 인증서로 받아들일 것이다. 사용자에게는 브라우저 프레임에 있는 자물쇠(padlock) 심볼과 같은 작은 보안 표시를 통해 고지된다. 신뢰받은 루트 인증서가 있는 계층구조 내의 모든 인증서들 역시 자동으로 신뢰받는다.

3. Hash 함수에 대한 배경 지식

3.1. Hash 함수

데이터에 서명하기 위해 개인키를 사용하는 암호화 작업은 데이터 그 자체를 직접적으로 처리하지 않고 미리 결정된 고정된 길이를 가지고 있어 짧고 다루기 편리하며 독특한 것으로 알려진 데이터 표시 방식으로 이 데이터를 처리한다.

이것은 인간에 대해 독특한 것으로 알려져 있고, 짧으며, 편리한 표시 방법인 지문에 비유될 수 있다. 그와 같이 데이터를 짧게 표시하여 나타내는 과정을 "hashing"이라고 부른다. 유감스럽게도 hash의 고정된 길이 때문에 같은 해쉬 값을 가져오는 몇 쌍의 다른 입력(input)이 존재해야 한다. 하지만 비록 존재는 하겠지만 그와 같은 쌍을 찾는 것이 아주 힘들다는 속성을 좋은 해쉬 함수들은 가지고 있다. 이것이 정확하게 MD5가 중대한 취약점을 가지는 부분이다.

해쉬 함수는 일정 크기의 문자열을 입력 값으로 가지는 암호화 함수인데, 이 입력으로 결정론적인 알고리즘(deterministic algorithm)을 수행하고, 출력 값으로 고정된 크기의 문자열을 만들어낸다. 이 출력 값을 입력 값의 해쉬 값(hash value) 또는 해쉬(hash)라고 부른다. "hash value"에 대해 "fingerprint" 또는 "message digest"라는 용어를 사용하기도 한다.

만약 `H`가 해쉬 함수이면 `m`은 입력 비트 문자열(input bit string)이고, `h`는 입력 값 `m`에 적용된 `H`의 출력 값이기 때문에 `h = H(m)`라고 할 수 있다. 일반적이고 유용한 용어로 나타내면:

- 만약 `h = H(m)`라면
  - `h`는 `m`의 해쉬를 호출,
  - `m`은 `h`의 프리이미지(preimage)[8]를 호출,
- 주어진 입력 값 `m`에 대해, `m`의 두 번째 프리이미지는 `H(m) = H(m')`과 같은 다른 입력 값 `m'`이고,
- 만약 `m`과 `m'`이 `H(m) = H(m')`와 같이 다른 입력 값이라면 `{m,m'}` 쌍은 `H`에 대한 "충돌"(collision)이라고 부른다.

[8] (역자 주) f(x)라는 함수가 주어졌을 때, 만약 f(a) = b라면 a는 b의 preimage라고 한다.

두 번째 프리이미지와 충돌 사이의 개념 차이는 미묘하지만 중요하다.

3.2. 해쉬 함수의 필요조건들

해쉬 함수는 다음 조건들을 만족시켜야 한다:

- (**실용성**) 어떤 입력 값 m에 대한 해쉬 h(m)을 효율적으로 계산할 수 있어야 하고,
- (**프리이미지 저항성**, **preimage resistance**) 만약 h가 주어지면 h의 프리이미지를 계산하는 것이 어려운데, 즉, h = H(m)을 만족하는 m을 계산하는 것은 어렵고,
- (**두 번째 프리이미지 저항성**) 만약 m이 주어지면 m의 두 번째 프리이미지를 계산하는 것이 어려운데, 즉 m ≠ m' 그리고 H(m) = H(m')을 만족하는 m'을 계산하는 것이 어렵고,
- (충돌 저항성, **collision resistance**) H에 대한 충돌을 계산하는 것은 어려운데, 다시 말해서 m ≠ m' 그리고 H(m) = H(m')인 다른 임의의 두 입력 m과 m'을 계산하는 것은 어렵다.

K비트의 길이를 가진 해쉬를 만드는 해쉬 함수가 있다고 가정해보자. Brute force 방법은 일정 범위 내에서 가능한 모든 입력 값을 단순히 시도하고, 또는 (두 번째) 프리이미지 또는 충돌이 발견될 때까지 많은 수의 무작위 입력을 시도한다. (두 번째) 프리이미지 저항성에 대해 이것은 2^k와 동일한 예상된 수의 해쉬 계산을 요구한다. 충돌 저항성에 대해서는 해쉬 "birthday paradox"[9] 때문에 상황이 다른데, 계산결과의 수가 거의 $2^{k/2}$까지 줄어든다. 암호학자들은 길이가 k 비트인 해쉬 함수는 겨우 k/2 비트의 보안을 제공한다고 말한다.

암호의 해쉬 함수는 앞에서 언급된 brute force 형태의 방법들보다 더 나은 방법들이 알려져 있지 않을 때 "암호학적으로 강력하다"고 말할 수 있다.

3.3. 해쉬 함수의 역사

1990년 Rivest는 128 비트의 해쉬 길이를 가진 해쉬 함수 MD4를 디자인했다. MD4의 충돌이 1995년에 발견되었으며, 2005년 프리이미지를 찾아내는 방법이 공개되었다.

1991년 Rivest는 MD4의 계승자인 MD5(MD4보다 발전된 것이지만 해쉬 길이가 128 비트임)를 발표했다. 1993년에 MD5 구현상의 몇 가지 취약점들이 지적되었지만 2004년에서야 충돌이 발견되었다. 요즘 충돌을 일으키게 하는 것은 PC에서 몇 초면 가능하다. 우리가 고안한 방법에서

[9] (역자 주) *N*명 중 어떤 특정 사람을 선택하였을 때 그 사람과 생일이 같은 사람을 찾을 확률이 50%가 되기 위해서는 *N*이 183명 이상이 되어야 하지만 생일이 같은 임의의 두 사람을 찾을 확률이 50%가 되기 위해서는 23명만 있으면 된다는 것이 '생일 파라독스'이다.

사용된 더 발전된 형태의 충돌도 특별하지만 일반적인 하드웨어에서 몇 시간이면 만들어낼 수 있다.

1995년 NIST[10]는 해쉬 길이가 160 비트인 해쉬 함수 SHA-1를 제안했다. 그러나 2005년에 SHA-1 구현상의 취약점이 발견되었으며, 충돌을 발견하기 위해서는 brute force 복잡도 방법[11]을 사용했을 때 2^{80} 이상의 계산이 필수적임을 이것은 보여준다. 하지만 아직 충돌이 공개되지는 않았다. NIST는 2010년까지 SHA-1을 대체하라고 권고하고 있다.

소개부터 현재까지 해쉬 함수 MD5와 SHA-1는 많은 암호 시스템으로 사용되어 왔다. 일반적인 어플리케이션에서 MD5를 대체하는 것은 아직 완료되지 않았다. 그리고 SHA-1을 대체하는 것은 실제 시작도 되지 않았다.

SHA-1이 그것의 구현상의 목적을 만족시켰다 할지라도 출력 길이 160 비트는 SHA-1을 더 오래 사용하는 것을 정당화시키기에는 너무 작다. NIST는 초기 단계에 이것을 인정했으며, 2001년에 해쉬 함수 계열에 SHA-2를 새롭게 도입했다. 지금까지 이 해쉬 함수들은 모든 암호해독에 저항해왔다. 그럼에도 불구하고 NIST는 해쉬 함수 구현에 대해 더 깊게 이해하고, 다음 10년 동안 더 나은 해쉬 함수들을 고안하기 위해 암호 커뮤니티의 힘을 결집하는 것이 필요함을 깨달았다. 그래서 NIST는 우선 SHA-3을 계승자로 봉했고, SHA-2의 계승자를 선택하는 열린 경쟁도 시작되었다. 이 경쟁의 승리자는 2012년까지 선택될 것으로 예상되며, 아마도 다음 10년 동안 사실상의 해쉬 작업 표준이 될 것이다.

3.4. MD5 구현

MD5는 Merkle-Damgård의 반복적인 설계(iterative construction)[12]를 사용한다. 입력 비트 문자열은 다수의 512 비트에 덧붙여진다. 덧붙여지지 않은 비트 문자열의 길이는 이 덧붙여진 것에 통합된다. 그런 다음 그 덧붙여진 입력 비트 문자열은 각각 512 비트의 블록들로 나누어지는데, 그래서 지금부터는 이를 "입력 블록(input block)"라고 부른다.

MD5의 핵심은 압축(compression) 함수이다. 입력 블록들이 압축 함수에 연속적인 호출로 MD5에 공급되며, 128 비트의 상태를 업데이트할 때 각각의 입력 블록을 사용한다. 이 상태를 `IHV`라고 부르는데, 이것은 "Intermediate Hash Value"를 의미한다. 그래서 압축 함수는 입력으로 128 비트

---

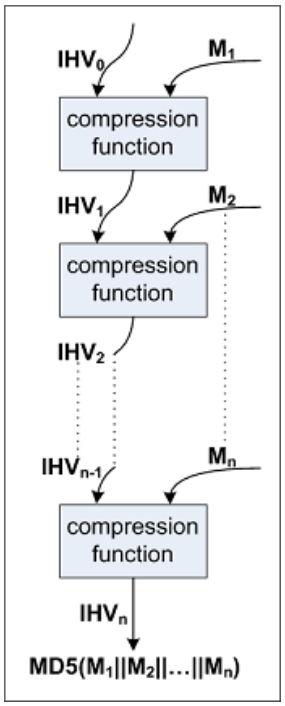
[10] http://www.nist.gov

[11] (역자 주) 알고리즘의 복잡도 분석 방법에는 시간 복잡도 분석과 공간 복잡도 분석이 있는데, 여기서 복잡도는 알고리즘의 수행 시간을 분석하는 시간 복잡도(time complexity) 분석을 가리킨다.

[12] (역자 주) Merkle-Damgård의 학위논문 http://www.merkle.com/papers/Thesis1979.pdf에 자세하게 기술되어 있으며, http://en.wikipedia.org/wiki/Merkle-Damgård_construction에는 그 핵심이 간단하게 기술되어 있다. 일반적으로 암호학에서 'Merkle-Damgård construction'이라고 부르는 Merkle-Damgård의 설계 방법을 http://en.wikipedia.org/wiki/File:Merkle-Damgard_hash_big.svg에서 그림으로 보여주고 있다.

상태 IHV와 512 비트 데이터 블록을 받아들이고, 출력으로 업데이트된 상태 IHV를 만든다. 초기 상태는 고정된 값이고, 최종 값은 해쉬 값이다.

그래서 만약 우리가 입력 블록을 M_1, M_2,..., M_n(각각 512 비트)으로, 초기 IHV를 IHV_0(128 비트)로, 그리고 압축 함수를 CF로 표시하면 연산의 시퀀시는 다음과 같다.



IHV_0 = fixed value

IHV_1 = CF(IHV_0, M_1)

IHV_2 = CF(IHV_1, M_2)

IHV_3 = CF(IHV_2, M_3)

:

:

:

IHV_{n-1} = CF(IHV_{n-2}, M_{n-1})

IHV_n = CF(IHV_{n-1}, M_n)

MD5 hash = IHV_n

3.5. MD5 충돌

2004년에 Xiaoyun Wang과 Hongbo Yu는 입력 비트 문자열을 512 비트에 덧붙이는 것을 무시하면서 2개의 입력 블록으로 구성된 MD5의 충돌을 제시했다. 충돌 설계 방법의 세부 내용은 **EuroCrypt 2005**[13]에서 발표[14]되었다. 그들의 방법은 초기 IHV_0의 값에 대해서도 적용된다.

다시 말해서, 그들의 방법은 어떤 128 비트 IHV_0의 입력에 대해 한 쌍의 {{M_1,M_2}, {M_1',M_2'}}을 만드는데, {M_1,M_2} ≠ {M_1',M_2'} 및 아래와 같이 각각은 두 개의 512 비트 입력 블록으로 구성되어 있다.

IHV_0 = IHV_0'

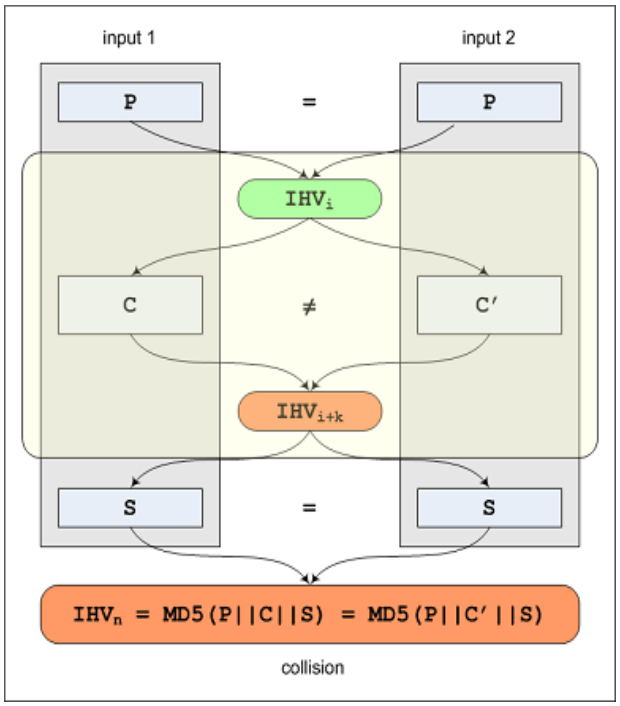
IHV_1 = CF(IHV_0,M_1) ≠ IHV_1' = CF(IHV_0',M_1')

IHV_2 = CF(IHV_1,M_2) = IHV_2' = CF(IHV_1',M_2')

---

[13] http://www.iacr.org/conferences/eurocrypt2005/

[14] Xiaoyun Wang and Hongbo Yu, "How to Break MD5 and Other Hash Functions"
In: Ronald Cramer (editor), "Advances in Cryptology - EUROCRYPT 2005", volume 3494 of Lecture Notes in Computer Science, pages 19-35, Springer Verlag, Berlin, 2005.

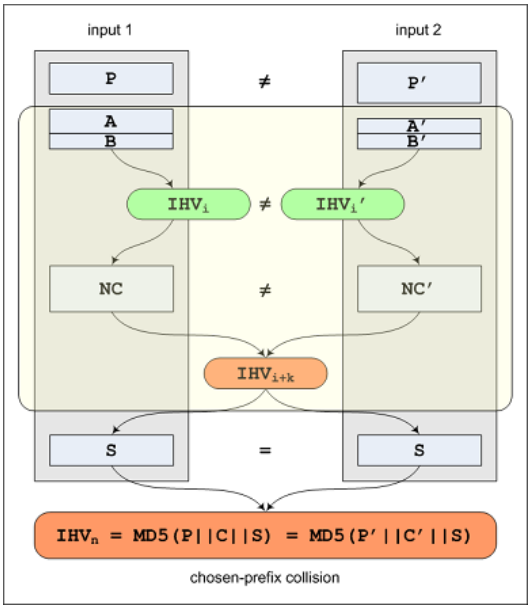


MD5의 반복 구조와 IHV_0가 어떤 128 비트 값을 가질 수 있다는 사실 때문에 그와 같은 충돌은 더 큰 입력으로 결합될 수 있다. 즉, 옆의 그림에서 보는 것과 같이 어떤 주어진 P(prefix)와 S(suffix)에 대해 한 쌍의 충돌 블록 {C,C'}는 MD5(P||C||S) = MD5(P||C'||S)와 같이 계산할 수 있다. 우리는 충돌을 성취하기 위해 다른 하나의 비트 문자열에 삽입된 특별히 조작된 비트 문자열에 대해서 "충돌 블록(collision block)"이라는 용어를 사용한다. 하나의 충돌 블록은 부분적인 입력 블록들을 포함해 몇 개의 입력 블록들로 구성될 수 있다. Xiaoyun Wang와 Hongbo Yu의 충돌 블록은 정확하게 두 개의 연속적인 입력 블록으로 구성되어 있다.

2007년도 MSc 논문[15](다른 논문 "Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities"[16]도 참조)에서 Marc Stevens는 이 충돌 설계 방법을 소위 "chosen-prefix collision"로 확장했다. 이것은 충돌이 다른 임의의 초기 IHV들 사이에서 발견될 수 있다는 것을 의미한다. 그의 방법은 두 개 이상의 입력 블록을 요구할 수도 있다.

그래서, MD5의 반복 구조 때문에 chosen-prefix 충돌 설계 방법은 MD5(P||C||S) = MD5(P'||C'||S)와 같은 어떤 한 쌍의 선택된 prefix {P,P'}, suffix S, 그리고 한 쌍의 충돌 블록 {C,C'}을 만들 수 있다.

좀더 정확하게 설명하면, 주어진 {P,P'}에 대해 충돌 블록들은 다음과 같이 만들어진다. 먼저 P, P'는 P||A 및 P'||A'와 같은 길이를 만들기 위해 적절한 크기와 유용한 컨텐츠를 가진 비트 문자열 A, A'로 덧붙여진다. 그런 다음, "birthdaying" 단계를



[15] Marc Stevens, "On collisions for MD5", MSc Thesis, Eindhoven University of Technology, June 2007, (http://www.win.tue.nl/hashclash/)

[16] Marc Stevens, Arjen Lenstra and Benne de Weger, "Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities", In: Moni Naor (editor), "Advances in Cryptology - EUROCRYPT 2007", volume 4515 of Lecture Notes in Computer Science, pages 1-22, Springer Verlag, Berlin, 2007.

이용해 비트 문자열 B, B'가 만들어지는데, P||A||B와 P'||A'||B'는 같은 길이를 가지고, 그 길이는 다수의 512이며, 결과적으로 나오는 IHV들은 이 지점에서 미리 지정된 구조를 가진다. 이것은 충돌을 일으키는 한 쌍의 '근접 충돌 블록' {NC,NC'}를 만들게 한다. 각각의 근접 충돌 블록은 많은 512 비트 입력 블록들로 구성될 것이다. 그래서, C = A||B||NC 및 C' = A'||B'||NC'와 함께 결과적으로 나오는 IHV들은 동일하다. 결과적으로, MD5(P||C) = MD5(P'||C')가 되고, 그래서 어떤 suffix S에 대해서도 역시 MD5(P||C||S) = MD5(P'||C'||S)가 된다.

chosen-prefix 충돌 설계 방법이 실제 가능하지만 동일한 임의의 초기 IHV들을 사용하는 방법보다 필수적으로 더 복잡하다. 하지만 대신 prefix P와 P' 둘 다를 자유롭게 선택할 수 있다.

4. 충돌 인증서 연혁

4.1. 의미심장함

앞에서 언급했던 이용 가능한 방법들로 만들 수 있는 MD5에 대한 충돌 형태를 악용하는데 있어 주된 문제는 IHV들이 충돌하도록 하기 위해 계산된 입력 블록들에 대해 충분한 통제권을 가지고 있지 않다는 것이다. 이것은 충돌 블록이 어떤 의미 있는 파일에 맞도록 하는데 필요한 미리 규정된 포맷형성으로 충돌 블록을 획득하는 어떤 방법도 알려지지 않았다는 것을 의미한다.

다른 많은 파일 타입에 대해 충돌 어플리케이션과 chosen-prefix 충돌은 의미심장함의 문제를 다양한 방식으로 해결하는 것이 예를 들어 "The story of Alice and her Boss"[17]와 같은 문서에 기술되어 있다. 이제 우리는 인증서에 대한 적용에 집중할 것이다.

4.2. 다른 공개 키들

2005년 Arjen Lenstra, Benne de Weger, 그리고 Xiaoyun Wang은 동일한 초기 IHV 타입의 충돌이 어떻게 한 쌍의 X.509 인증서에 만들어질 수 있는지를 보여주었다[18]. 예를 한 가지 들면, "Colliding X.509 Certificates based on MD5-collisions".[19]를 보자. 이 글의 주된 아이디어는 공개 키에 랜덤하게 보이지만 주의 깊게 처리된 충돌 블록들을 숨기는 것이었다. 이 경우 공개 키들은

[17] http://th.informatik.uni-mannheim.de/People/lucks/HashCollisions/

[18] Arjen Lenstra and Benne de Weger, "On the possibility of constructing meaningful hash collisions for public keys", In: Colin Boyd and Juan Manuel Gonzalez Nieto (editors), "Information Security and Privacy - Proceedings of ACISP 2005", volume 3574 of Lecture Notes in Computer Science, pages 267-279, Springer Verlag, Berlin, 2005.

[19] http://www.win.tue.nl/~bdeweger/CollidingCertificates/

일반적으로 완전히 랜덤한 데이터처럼 보이기 때문에 아무런 의심을 일으키지 않을 것이며, 심지어 가짜 서명을 찾아내기 위해 인증서의 모든 부분을 주의 깊게 관찰한 사람에게도 마찬가지다. 계수(moduli) 쌍의 보안이 아주 큰 소수로 만들어진 것이라고 확신할 동안에도 RSA 계수 내에 충돌 블록들을 숨기는 것이 가능한 것으로 입증되었다. 그래서 서명될 부분들의 동일한 MD5 해쉬 값을 가지고, 그래서 동일한 CA 서명을 가진 한 쌍의 다른 X.509 인증서를 만드는 방법이 공개되었다. 이것은 인증서 인프라구조의 기본 원리를 침해하는 것이다. 하지만, 충돌 입력들의 prefix들이 같아야 하기 때문에 그 인증서들은 동일한 아이덴터티들을 보여주고, 실제 악용 잠재력을 엄격하게 제한시킨다.

4.3. 다른 아이덴터티

2007년에 Marc Stevens의 작업[20] 때문에 MD5의 chosen-prefix 충돌들이 이용 가능하게 되었다. 이것은 RSA 계수 이전의 인증서들에 있는 어떤 한 쌍의 prefix들을 선택하고, 다른 선택된 아이덴터티와 다른 공개 키(충돌 블록을 숨기면서), 그리고 서명될 부분의 동일한 MD5 해쉬 값들과, 그래서 동일한 CA 서명을 가진 인증서를 선택하는 것이 가능해졌다는 것을 의미한다. 예를 들어, "Colliding X.509 Certificates for Different Identities"[21]를 보아라. 이것은 이미 훨씬 더 흥미로운 적용이다. 하지만, 실제 Certification Authority에 적용될 때 많은 제한 사항이 이 공격을 복잡하게 한다.

공격자는 충돌 생성 과정이 시작될 수 있기 전에 전체 prefix를 선택해야 한다. 이것은 인증서의 유효 기간과 시리얼 넘버와 같이 공격자가 직접적으로 통제하지 않는 필드들을 포함하고 있다. 그 필드들을 성공적으로 예상하기 위해서는 성취하기에 힘든 것으로 생각되었던 CA의 작동 절차의 중요한 부분을 통제하는 것이 요구된다. 추가로 "Colliding X.509 Certificates for Different Identities" 에서 충돌하는 인증서들은 8192 비트의 RSA 키들을 사용했는데, 이는 모든 Certification Authority에 의해 받아들여지지는 않는다.

그럼에도 불구하고 2005년도의 충돌 설계 방법과 2007년도의 MD5를 기초로 한 인증서에 대한 더 나은 chosen-prefix 충돌 설계 방법 둘 다 MD5는 인증서에서 디지털 서명을 위해 더 이상 사용되어서는 안된다는 보안 커뮤니티에 엄격한 경고였다. 시간이 흐름에 따라 공격은 더 발전되기 때문에 2008년에 우리가 인증서를 충돌시키는 것에 대한 이전 작업을 개선시키고, 상용 Certification Authority에 의해 서명된 가짜 CA 인증서를 획득할 수 있는 것은 놀라운 일이 아니다.

5. 공격의 세부 내용

[20] Marc Stevens, "On collisions for MD5", MSc Thesis, Eindhoven University of Technology, June 2007, http://www.win.tue.nl/hashclash/

[21] http://www.win.tue.nl/hashclash/TargetCollidingCertificates/

5.1. 소개

2008년 2월에 우리는 인증서 충돌 공격이 실제 Certification Authority에 적용될 수 있는지 확인하기 위한 연구를 시작했다. 첫 번째 단계는 MD5를 여전히 사용하고 있는 CA들을 확인하는 것이었다. 하지만 CA 인증서로부터 CA가 사용하는 해쉬 함수를 확인하는 것이 가능하지 않았다. 대신 우리는 CA들에 의해 발행되는 (웹 사이트) 인증서들을 살펴보아야 했다. 1주일간의 과정 동안 우리는 웹을 검색해보고 Firebox가 신뢰하는 CA들에 의해 서명된 약 30,000개의 인증서들 중에서 100,000개 이상의 SSL 인증서들을 수집했다. 2008년에 MD5로 서명된 인증서를 발행한 CA는 다음 6개가 있었다:

*** RapidSSL**(http://www.rapidssl.com)
C=US, O=Equifax Secure Inc., CN=Equifax Secure Global eBusiness CA-1

* **FreeSSL**[22] (RapidSSL가 제공하는 무료 실험판 인증서)
C=US, ST=UT, L=Salt Lake City, O=The USERTRUST Network, OU=http://www.usertrust.com, CN=UTN-USERFirst-Network Applications

* **TC TrustCenter AG** (http://www.trustcenter.de)
C=DE, ST=Hamburg, L=Hamburg, O=TC TrustCenter for Security in Data Networks GmbH, OU=TC TrustCenter Class 3 CA/emailAddress=certificate@trustcenter.de

* **RSA Data Security** (https://www.verisign.com/repository/root.html#rsass)
C=US, O=RSA Data Security, Inc., OU=Secure Server Certification Authority

* **Thawte** (http://www.thawte.com)
C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc, OU=Certification Services Division, CN=Thawte Premium Server CA/emailAddress=premium-server@thawte.com

* **verisign.co.jp** (http://www.verisign.co.jp)
O=VeriSign Trust Network, OU=VeriSign, Inc., OU=VeriSign International Server CA - Class 3, OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign

수집된 30,000개의 인증서들 중에서 약 9,000개가 MD5를 이용해 서명되었고, 그것들 중 97%는 RapidSSL에 의해 발행된 것이었다.

2004년에 첫 충돌이 제기된 이후 MD5가 안전하지 않은 것으로 알려진 것을 생각해보면 그렇게 많은 CA들이 여전히 MD5를 이용하고 있는 것은 놀라울 일이다. 이 CA들이 암호 커뮤니티의

[22] http://www.rapidssl.com/ssl-certificate-products/free-ssl/freessl.htm

이전 경고들을 무시했기 때문에 root CA 인증서들을 가지고 있는 웹 브라우저를 사용하여 MD5를 이용하고 있는 CA들이 초래한 위험을 모든 사람들에게 입증하기 위해 우리는 실제 공격을 시도하는 것이 적절하다고 느꼈다.

우리는 서명될 부분의 전체 내용을 예상할 수 있도록 이 CA로부터 합법적인 웹 사이트 인증서를 구하기 위해 착수했다. 이 인증서는 RSA 계수에 특별히 조작된 충돌 블록을 가지고 있을 것이며, 이는 특별히 조작된 두 번째 인증서로 MD5 충돌이라는 결과가 나올 것이다. 아무리 주의 깊게 인증서를 살펴보아도 무지한 관찰자는 그것에 대해 의심스러운 어떤 것을 인지하는데 실패할 것이다.

일단 우리가 이 CA로부터 유효한 서명을 획득했기 때문에 우리는 그것을 우리의 두 번째 인증서로 복사한다. 두 인증서의 서명될 부분이 같은 MD5 해쉬를 가지고 있기 때문에 그 서명은 두 번째 인증서에 대해서도 유효할 것이다. 그래서 실제 CA는 그 사실을 모른 체 이전에 본적도 없고 결코 서명하지 말아야 하는 인증서에 대해 유효한 서명을 제공할 것이다.

이 공격 시나리오는 안전한 단일 웹 사이트에 대한 가짜 인증서를 획득하는 것보다 그 가능성이 더 크다. 이것은 우리의 가짜 인증서가 웹사이트의 인증서일 필요가 없기 때문이지만 그것은 중개 CA 인증서는 될 수 있다. 실제 CA에 의해 원래 서명된 인증서가 "기본 제한" 필드에 이 인증서는 인증서 체인에서 다른 인증서들을 유효화하는데 사용될 수 없다는 것을 나타내는 "CA = FALSE" 플래그를 비록 가지고 있다고 할지라도 우리의 가짜 인증서는 "CA = TRUE"에 설정되어 있는 같은 플래그를 가지고 있다. 우리는 이 가짜 CA 인증서에 있는 공개키에 상응하는 개인키를 가지고 있다. 그 결과 우리는 우리가 선택한 어떤 사람에게 일정 수의 인증서들을 발행할 수 있으며, 실제 CA를 신뢰하는 어떤 사람은 이 인증서들을 유효한 것으로 인정할 것이다.

5.2. 유효기간 및 시리얼 번호 예측

첫 번째 큰 도전은 Certification Authority에 의해 발행된 인증서들의 유효기간과 시리얼 번호를 예측하는 방법을 이해하는 것이었다. 왜냐하면 유효기간과 시리얼 번호는 chosen prefix에서 공격자가 직접적으로 통제하지 않는 유일한 두 필드이기 때문이다.

인증서의 유효기간을 예측하는 것은 쉬운 일로 입증되었다. RapidSSL와 FreeSSL가 인증서를 발행하는데 사용하는 시스템은 완전히 자동화되어 있으며, 각 인증서는 사용자가 구입을 완료하기 위해 최종 "OK" 버튼을 누른 후 정확하게 6초 후에 발행된다. 인증서 유효기간 필드에 있는 timestamp[23]는 몇 초 안에 입도(粒度, granularity)[24]를 사용하기 때문에 우리는 임의의 시간 **T-6**초

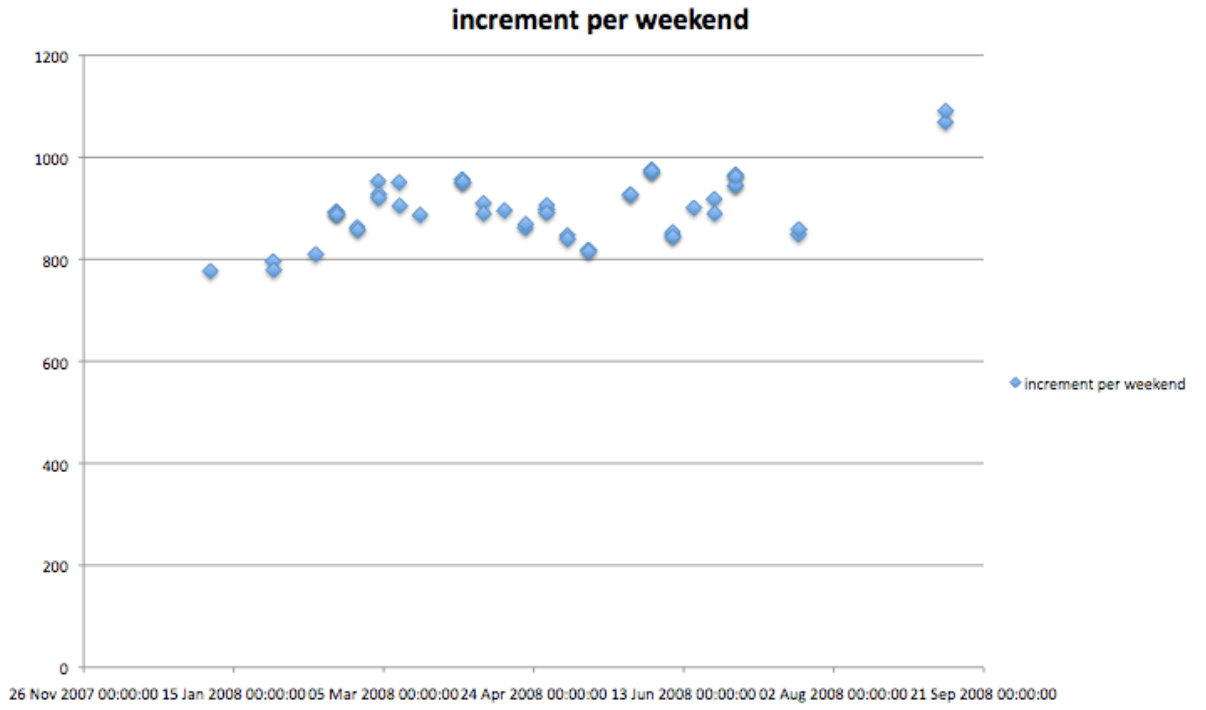
[23] (역자 주) Timestamp는 어떤 이벤트가 발행한 날짜와 시간을 나타내는 문자들의 연속을 가리킨다. Timestamp의 예를 들면, "2009-02-17 T 12:15 UTC"나 "Mon Feb 15 12:15:55 2009"와 같은 것이다.

[24] (역자 주) 입도는 어떤 객체나 활동을 특징짓는 상대적 크기, 비율, 자세함의 정도, 통찰의 깊이를 나타내는 말이다.

때 쉽게 이 버튼을 클릭할 수 있었고, 시간 **T**에서 **T+**(1년)까지 유효한 인증서를 획득할 수 있었다. 유효기간에 대한 완전한 통제권을 획득한 후 우리는 시리얼 번호 예측 문제를 해결하는 방향으로 나아갔다.

우리는 충돌하는 인증서 쌍을 생성하는데 3일 정도 걸릴 것으로 예상했는데, 이는 우리가 3일 먼저 인증서의 시리얼 번호를 예측할 필요가 있다는 것을 의미한다. 우리의 데이터 세트에 있는 대부분의 인증서들은 랜덤하게 보이고, 그래서 예측하기 힘든 시리얼 번호를 가지고 있었지만 RapidSSL와 FreeSSL는 순차적인 시리얼 번호들을 사용하고 있다는 것을 우리는 알아냈다.

사람들은 이 순차적인 시리얼 번호를 원격 카운터로 생각할 수 있다. 우리가 CA로부터 구입한 각 인증서는 그 카운터의 현재 값을 드러내고, 그것을 하나씩 증가시킨다. Certification Authority의 일반적인 작업 동안 그 카운터는 발행된 인증서들의 번호를 기반으로 해서 증가한다. 우리의 RapidSSL 데이터 세트에서 9,251개의 인증서들로부터 추출한 시리얼 번호를 통계학적으로 분석한 결과 평균적으로 이 번호는 아주 낮은 변화를 가지고 있다는 것을 드러냈다. 목요일 밤부터 토요일 밤까지 3일 동안 CA는 보통 800에서 1000개 사이의 인증서들을 발행한다:



위 그래프에서 마지막 데이터 지점의 예외는 9월 말 한 특정 주말 동안 우리가 구입한 100개의 인증서의 결과이다. 만약 우리가 관여하지 않았더라도 그 주말 동안 증가는 1,000개 이하였을 것이다.

이 정보를 가지고 우리는 다음 계획을 세웠다: 목요일 밤에 우리는 시리얼 번호 **S**의 현재 값을 구하기 위해 인증서 하나를 구입한다. 우리는 일요일 밤(시간 **T**)에 그 시리얼 번호가 **S+**1000보다 약간 작을 것이라고 예측한다. 시간 **T** 몇 시간 전에 우리는 목표 번호에 더욱 가까워진 시리얼 번호로 증가시키기 위해 인증서들을 구입하기 시작하고, 마침내 시간 **T** 약 30초 전에 시리얼 번호를 **S+**999에 도달하게 한다. 만약 운이 좋아 CA가 그 마지막 30초 동안 다른 인증서 요청을 받아들이지 않는다면 우리는 시간 **T**에 요청을 보내 우리가 예상한 시리얼 번호를 구할 수 있을 것이다.

5.3. 충돌하는 인증서 만들기

이 섹션에서는 우리가 지금 가지고 있는 동일한 서명을 가진 두 개의 인증서의 내용에 대해 자세하게 기술할 것이다. 우리가 상용 CA로부터 획득한 인증서는 지금까지 "실제 인증서"로 불릴 것이다. 그것의 형제(sibling) 인증서는 우리의 "가짜 CA 인증서"이고, MD5 해쉬 함수에 대해 충돌을 구성하기 위해 이 인증서들의 서명될 부분들이 함께 구성되었다는 의미에서 그것은 형제이다.

실제 인증서는 그것에 CSR(Certificate Signing Request, 인증서 서명 요청)을 보냄으로써 CA로부터 요청되었다. 우리가 해야 했던 초기 엔지니어링 작업은 CA에 의해 만들어진 실제 인증서와 우리가 만든 가짜 CA 인증서가 완벽하게 배열되는 방식으로 CSR의 내용과, 가짜 CA 인증서의 구조와 내용을 구성하는 것이다. 훨씬 더 복잡한 두 번째 엔지니어링 작업은 실제 인증서들에 있는 다양한 필드의 내용들을 예측하는 것이며, 이를 바탕으로 두 인증서의 다른 다양한 필드들의 내용을 MD5 충돌을 일으킬 수 있는 방식으로 구성하는 것이다.

X.509 표준[HPFS[25]]을 따르면 이 두 인증서의 각각은 다음과 같이 구성되어 있다:

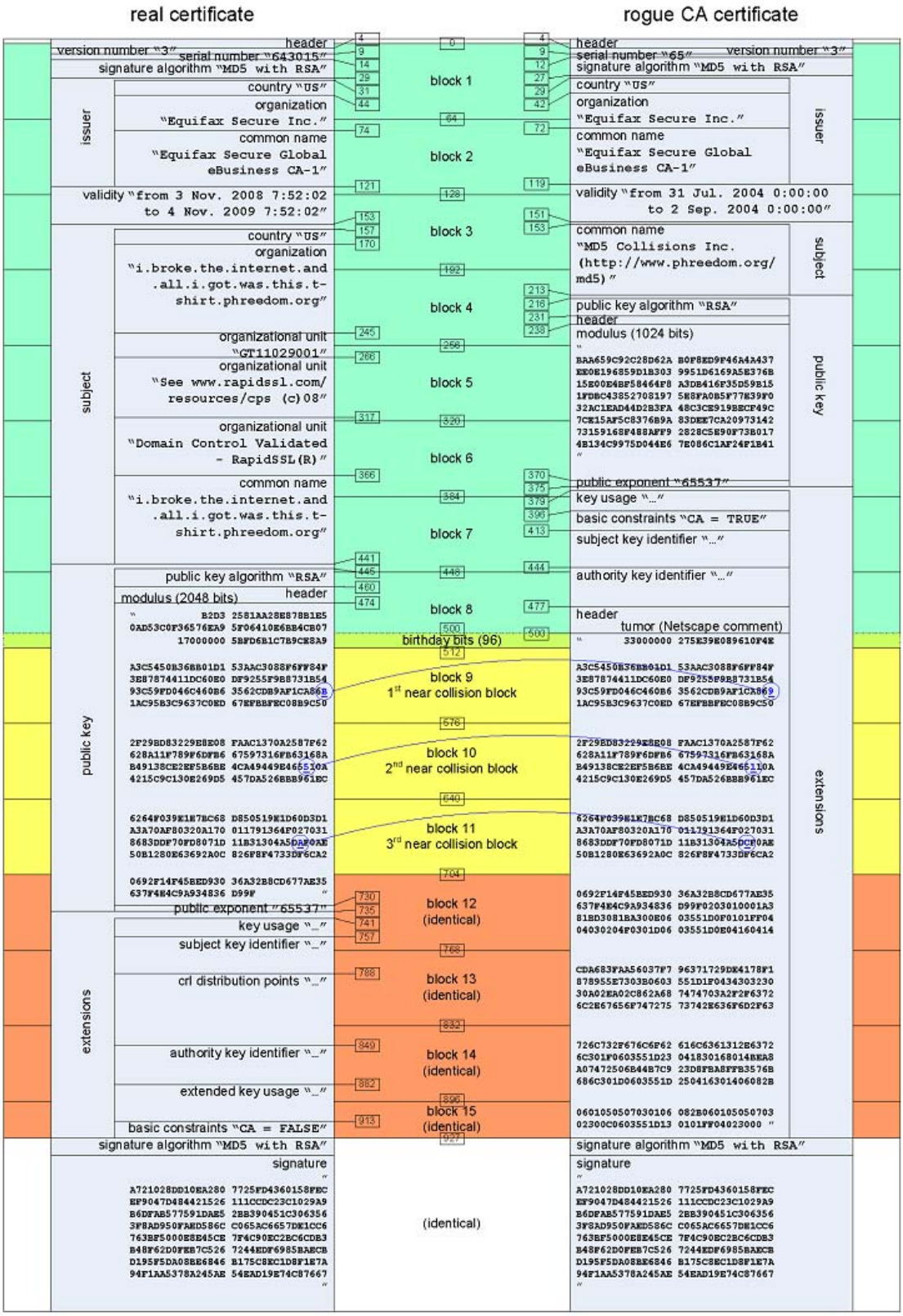
- 4 바이트의 헤더
- 927 바이트의 소위 "서명될"(to-be-signed) 부분
- 15 바이트의 소위 "서명 알고리즘"(signature algorithm) 필드
- 131 바이트의 "서명"(signature) 필드

아래 그림[26]은 두 인증서의 배열과 대부분의 내용을 자세하게 보여준다. 녹색, 노란색, 그리고 빨간색은 각각 chosen prefix, 충돌 블록, 그리고 suffix를 가리킨다.

---

[25] R. Housley, W. Polk, W. Ford, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", IETF RFC 3280, April 2002, (http://www.ietf.org/rfc/rfc3280.txt.)

[26] 이 그림은 http://www.win.tue.nl/hashclash/rogue-ca/downloads/alignment.pdf를 보면 더 자세하고 크게 볼 수 있다.


real certificate
rogue CA certificate
header
version number "3"
serial number "643015"
signature algorithm "MD5 with RSA"
issuer
country "US"
organization "Equifax Secure Inc."
common name "Equifax Secure Global eBusiness CA-1"
validity "from 3 Nov. 2008 7:52:02 to 4 Nov. 2009 7:52:02"
subject
country "US"
organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
organizational unit "GT11029001"
organizational unit "See www.rapidssl.com/resources/cps (c)08"
organizational unit "Domain Control Validated - RapidSSL(R)"
common name "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
public key
public key algorithm "RSA"
modulus (2048 bits)
B2D3 2581AA28E878B1E5 0AD53C0F36576EA9 5F06410E6BB4CB07 17000000 5BFD6B1C7B9CE8A9 A3C5450B36B01D1 53AAC3088F6FF84F 3E87874411DC60E0 DF9255F9B8731B54 93C59FD046C460B6 3562CDB9AF1CA86B 1AC95B3C9637C0ED 67EFBBFEC08B9C50 2F29BD83229E8E08 FAAC1370A2587F62 628A11F789F6DFB6 67597316FB63168A B49138CE2EF5B6BE 4CA49449E4655100 4215C9C130E269D5 457DA526BBB961EC 6264F039E1E7BC68 D850519E1D60D3D1 A3A70AF80320A170 011791364F027031 8683DDF70FD8071D 11B31304A5DA80AE 50B1280E63692A0C 826FBF4733DF6CA2 0692F14F45BED930 36A32B8CD677AE35 637F4E4C9A934836 D99F
public exponent "65537"
extensions
key usage "_"
subject key identifier "_"
crl distribution points "_"
authority key identifier "_"
extended key usage "_"
basic constraints "CA = FALSE"
signature algorithm "MD5 with RSA"
signature
A721028DD10EA280 7725FD4360158FEC EF9047D484421526 111CCDC23C1029A9 B6DFAB577591DAE5 2BB390451C306356 3F8AD950FAED586C C065AC6657DE1CC6 763BF5000E8E45CE 7F4C90EC2BC6CDB3 B48F62D0FEB7C526 7244EDF6985BAECB D195F5DA0BBE6846 B175C8EC1D8F1E7A 94F1AA5378A245AE 54EAD19E74C87667
block 1
block 2
block 3
block 4
block 5
block 6
block 7
block 8
birthday bits (96)
block 9
1st near collision block
block 10
2nd near collision block
block 11
3rd near collision block
block 12
(identical)
block 13
(identical)
block 14
(identical)
block 15
(identical)
(identical)
header
serial number "65"
version number "3"
signature algorithm "MD5 with RSA"
issuer
country "US"
organization "Equifax Secure Inc."
common name "Equifax Secure Global eBusiness CA-1"
validity "from 31 Jul. 2004 0:00:00 to 2 Sep. 2004 0:00:00"
subject
common name "MD5 Collisions Inc. (http://www.phreedom.org/md5)"
public key
public key algorithm "RSA"
header
modulus (1024 bits)
BAA659C92C28D62A B0F8ED9F46A4A437 EE0E196859D1B303 9951D6169A5E376B 15E00E4BF58464F8 A3DB416F35D59B15 1FDBC43852708197 5E8FA0B5F77E39F0 32AC1EAD44D2B3FA 48C3CE919BECF49C 7CE15AF5C8376B9A 83DEE7CA20973142 73159168F48BAFF9 2828C5E90F73B017 4B134C9975D044E6 7E086C1AF24F1B41
public exponent "65537"
extensions
key usage "..."
basic constraints "CA = TRUE"
subject key identifier "..."
authority key identifier "..."
header
tumor (Netscape comment)
33000000 275E39E089610F4E A3C5450B36B01D1 53AAC3088F6FF84F 3E87874411DC60E0 DF9255F9B8731B54 93C59FD046C460B6 3562CDB9AF1CA869 1AC95B3C9637C0ED 67EFBBFEC08B9C50 2F29BD83229E8E08 FAAC1370A2587F62 628A11F789F6DFB6 67597316FB63168A B49138CE2EF5B6BE 4CA49449E4655100 4215C9C130E269D5 457DA526BBB961EC 6264F039E1E7BC68 D850519E1D60D3D1 A3A70AF80320A170 011791364F027031 8683DDF70FD8071D 11B31304A5DC80AE 50B1280E63692A0C 826FBF4733DF6CA2 0692F14F45BED930 36A32B8CD677AE35 637F4E4C9A934836 D99F0203010001A3 81BD3081BA300E06 03551D0F0101FF04 04030204F0301D06 03551D0E04160414 CDA683FAA56037F7 96371729DE4178F1 878955E7303B0603 551D1F0434303230 30A02EA02C862A68 7474703A2F2F6372 6C2E67656F747275 73742E636F6D2F63 726C732F676C6F62 616C6361312E6372 6C301F0603551D23 04183016801488EA8 A07472506B44B7C9 23D8FBA8FFB3576B 686C301D0603551D 2504163014060882B 0601050507030106 082B060105050703 02300C0603551D13 0101FF04023000
signature algorithm "MD5 with RSA"
signature
A721028DD10EA280 7725FD4360158FEC EF9047D484421526 111CCDC23C1029A9 B6DFAB577591DAE5 2BB390451C306356 3F8AD950FAED586C C065AC6657DE1CC6 763BF5000E8E45CE 7F4C90EC2BC6CDB3 B48F62D0FEB7C526 7244EDF6985BAECB D195F5DA0BBE6846 B175C8EC1D8F1E7A 94F1AA5378A245AE 54EAD19E74C87667

5.3.1. 실제 인증서와 CSR

실제 인증서의 "to-be-signed" 부분의 구조는 CA에 의해 규정되어 있다. 우리는 몇몇 필드의 내용과 길이만을 통제할 수 있다. 구조는 다음과 같다:

byte 0 - 3: 헤더

byte 4 - 8: 버전 번호(3, 기본 값)

byte 9 - 13: 시리얼 번호(643015, CA에 의해 설정되고 우리가 예측한 것)

byte 14 - 28: 서명 알고리즘("md5withRSAEncryption", CA가 설정)

byte 29 - 120: 발행자 구분명(issuer Distinguished Name) (CA 기본값)

byte 121 - 152: 유효기간("3 Nov. 2008 7:52:02에서 4 Nov. 2009 7:52:02까지", CA에 의해 설정되고 우리가 예측한 것)

byte 153 - 440: 주체 구분명(subject Distinguished Name)
(Country = "US",
Organisation = "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org",
Organisational Unit (CA에 의해 설정된 3개의 필드)
Common Name = "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org",
CSR의 우리에 의해 설정된 Country, Organisation 및 Common Name[27])

byte 441 - 734: 주체 공개키 정보:

- byte 441 – 444: 헤더
- byte 445 – 459: 공개키 알고리즘("RSAEncryption", CSR에서 우리에 의해 설정됨)
- byte 460 – 468: 헤더들
- byte 469 – 729: RSA 계수(2048 비트 값, CSR에서 우리에 의해 설정됨)
- byte 730 – 734: RSA 공개 지수(exponent) (65537, CSR에서 우리에 의해 설정됨)

byte 735 - 926: 확장된 부분:

- byte 735 - 740: 헤더
- byte 741 - 756: 키 사용법(CA가 설정한 "디지털 서명, nonrepudiation[28], 키 암호화, 데이터 암호화")
- byte 757 - 881: 주체 키 식별자, crl 배포 지점, 권한 키 식별자(CA가 설정했지만 흥미 없는 필드)
- byte 882 - 912: 확장된 키 사용법(CA가 설정한 "서버 인증, 클라이언트 인증")
- byte 913 - 926: 기본 제한사항("CA = FALSE, Path Length = None", CA가 설정)

---

[27] (역자 주) Common Name에는 유효한 도메인 주소가 들어간다.

[28] nonrepudiation은 계약 또는 통신의 상대가 문서나 또는 보내진 메시지에 첨부된 서명의 확실성을 부정할 수 없도록 보증하는 능력을 가리킨다(terms.co.kr 참고).

이 인증서를 구하기 위해 우리는 필수적인 PKCS#10 포맷에 맞는 CSR를 구성했다. 이 요청은 다음 데이터를 포함하고 있다:

- "to-be-signed" 부분:
  - 버전 번호 (1)
  - 주체 국가 = "US"
  - 주체 Common Name = "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
    (Organisation과 Common Name 필드에 대해 CA에 의해 사용됨)
  - 공개키 알고리즘 ("RSAEncryption")
  - 공개키 (RSA 계수와 RSA 지수로 구성)

- 서명 알고리즘("md5withRSAEncryption")
- 서명("to-be-signed" 부분에서 우리의 개인키로 연산된 2048 바이트의 값, 이 서명은 그 요청 내의 공개키로 확인될 수 있으며, 이와 같이 CA는 요청자가 상응하는 개인키를 가지고 있다는 것을 확신할 수 있다)

실제 인증서에 대한 key pair를 구성하는데 필요한 것은 다음과 같다:

- 계수는 2,048비트 값이어야 한다,
- 보통 때처럼 우리는 공개 지수로 65537을 선택했다.
- 우리는 대응하는 개인키를 알고 있어야 하는데, 이는 CSR이 그 개인키로 서명되어야 하기 때문이다.
- 충돌 블록은 계수에서 숨겨져 있다.

그 정렬은 256 바이트 계수가 474 바이트에서 시작하는 것으로 입증되었다. 이것은 남은 현재 512 비트 MD5 입력 블록의 38 바이트를 우리가 가지고 있다는 것을 의미한다. 그 38 바이트 중에서 첫 26 바이트는 랜덤하게 선택되었다. 나머지 12 바이트는 "birthday bit"로 사용되었는데, 추가 설명은 아래를 참고하길 바란다.

그런 다음 우리는 각각 512 비트(64 바이트)의 "근접 충돌 블록" 3개를 가졌다. 그래서 전체 "충돌 블록"은 96 + 3 x 512 = 1632 비트(204 바이트)로 구성되어 있다.

마지막 근접 충돌 블록 끝에 MD5 충돌이 확립된다. 그 충돌 블록은 아래 기술된 충돌 발견 방법에 의해 계산되었다. 이것은 우리가 그것의 내용들에 대해 많은 통제권을 가지고 있지 않고, 게임 콘솔의 클러스터로부터 나오는 랜덤하게 보이는 데이터를 우리가 단순히 받아들여야 한다는 것을 의미한다.

그런 다음 계수에 남은 256 - 26 - 204 = 26 바이트(208 비트)를 가진다. 우리는 그 계수가 우리에게 알려진 두 소수의 곱(product) 형태를 가질 것이라는 것을 확인하기 위해 그 26 바이트를 사용했으며, 이와 같은 방법으로 우리는 상응하는 개인 지수도 역시 계산할 수 있다. 이것은 다음과 같이 이루어진다.

우리는 1840 비트(230 바이트, 즉 MD5 입력 블록들로 정렬에 도달하기 위한 38 바이트와, 3개의 근접 충돌 블록에 대한 3 x 64 = 192 바이트)의 고정된 비트 문자열 B로 시작한다. 우리는 이제 또 다른 208비트(26 바이트)로 된 비트 문자열 S를 추가하고자 하며, 결과적으로 나오는 2048 비트의 $B||S$는 온전히 안전한 RSA 계수인 정수를 나타내는데, 즉, 이는 현재 인수분해(factoring) 방법들에 저항하는 소수 p, q를 가진 $n=p\times q$를 가리킨다. 그와 같은 S를 발견하기 위해 우리는 랜덤한 224 비트 정수 q를 선택하고, 그런 다음 $B||S$가 수 $n=p\times q$를 나타낼 때의 정수 p가 있기를 희망한다. 그런 다음 우리는 p와 q 둘 다 소수이기를 희망한다. 만약 우리의 희망이 잘못된 것으로 판명된다면 우리는 성공할 때까지 다른 랜덤한 값 q를 시도할 것이다.

이 방법은 조잡하게 보이지만 더 좋은 방법을 우리가 알지 못하고, 그렇더라도 이 방법은 실제 잘 작동한다. 그 이유는 다음과 같다. 우리가 찾고 있는 n의 값은 $B\times 2^{208}\le n<(B+1)\times 2^{208}$의 간격에 놓여있다. 주어진 k 비트로 된 q에 대해 이것은 $(B/q)\times 2^{208}$로부터 $((B+1)/q)\times 2^{208}$까지의 간격은 정수 p를 가지고 있어야 하고, 더 나아가 정수 p와 q 둘 다 소수라는 것을 의미한다. 이 간격의 넓이가 $2^{208}/q$(대략 2^{208-k})이기 때문에 소수이어야 한다는 것과는 상관 없이 적절한 간격에 놓여있는 정수 p와 같은 것을 발견하기 전에 우리는 대략 2^{k-208}번을 시도해야 한다. q가 소수일 가능성은 대략 $\log(2^k)=k\times\log(2)$당 1이다. 다음으로, p가 대략 $2048-k$ 비트임을 주목해라. 그와 같은 수가 소수일 가능성은 대략 $\log(2^{2048-k})=(2048-k)\times\log(2)$당 1이다. 이것은 우리가 한번의 히트를 예상할 수 있기 전에 k 비트로 된 대략 $2^{k-208}\times k\times(2048-k)\times(\log(2))^2$ 수 q를 생성해야만 한다는 것을 의미한다. 단지 $k\times(2048-k)\times(\log(2))^2$의 경우에 우리는 소수성(primality)에 대해 k 비트 수를 테스트해야 하고, 단지 $(2048-k)\times\log(2)$의 경우에 소수성에 대해 $2048-$ 비트 수를 테스트해야만 한다. 우리는 $k=224$를 선택했는데, 그것은 실제로 할 수 있는 것으로 입증되었다. 이 안전한 계수를 계산하는데 필요한 시간은 표준 PC에서 몇 분이면 되었다.

결과적으로 나오는 RSA 계수는 그래서 다소 균형이 맞지 않은데, 그것의 소인수(prime factor)는 각각 1,824와 224 비트이다. 이것은 특정 ECM에서 현재 인수분해 방법들의 영역 밖인 것으로 평가된다.

실제 인증서에 상응하는 개인키가 CSR을 서명하기 위해서만 사용되고, 그 실제 인증서 그 자체는 우리에게 유효한 서명을 우리에게 제공하기 위해서만 사용된다는 것을 주목해라. 이 인증서와 그에 상응하는 key pair는 비록 원칙 상으로는 그 누구에게도 해를 끼치지는 않겠지만 몇몇 실제 어플리케이션에서는 결코 사용되지 않을 것이다. 이것의 흥미로운 결과는 안전한 RSA 계수를 만드는 것이 전혀 필요하지 않다는 것이며, 우리는 $\varphi(n)$이 우리의 공개 지수 65537에 대한

서로소(coprime)라는 것처럼, 우리가 알고 있는 인수분해 n에 대한 어떤 수를 가질 수 있어야 했다. 우리가 인수분해를 필요로 하는 이유는 CSR을 서명해야 하는 것처럼 우리가 개인 지수를 계산할 필요가 있기 때문이다. 하나의 소수를 가지거나 또는 어떤 종류의 인수분해가 역시 하는 것처럼 n이 두 소수의 곱일 필요는 없다. 그럼에도 불구하고 우리는 여기서 역시 주의를 했으며, 그리고 그것의 안전한 RSA 계수를 만들기로 위해 개인키도 역시 비밀로 유지될 것이다.

흥미로운 것 하나는 CA root 인증서의 "기본 제한사항" 필드에 있는 "경로 길이" 필드가 설정되어 있지 않다는 것이다. 경로 길이 필드는 CA 계층에서 이 CA가 그 자신의 레벨과 최종 사용자 레벨 사이를 허용하는 중개 CA 레벨의 최대 수이다. 만약 그 CA가 경로 길이를 0으로 설정했다면 그 경로 길이를 점검하는 어플리케이션들은 우리의 가짜 CA에 의해 서명된 인증서들을 거부했을 것이며, 경로 길이가 1이기 때문에 우리의 가짜 CA는 최종 사용자 인증서와 root CA 인증서 사이의 레벨에 있다.

충돌 설계를 위해 우리는 충돌 블록이 시작하는 지점까지 인증서의 서명될 부분의 전체 내용을 비트 수준에서 예측할 수 있는 것이 절대적으로 필요하다. 지속적이지 않거나 또는 우리에 의해 직접적으로 통제되지 않는 유일한 필드들은 유효 기간과 시리얼 번호이다. 하지만, 섹션 5.2에서 기술된 테크닉을 사용하여 이 두 필드의 값을 우리는 예측할 수 있었다.

5.3.2. 가짜 CA 인증서

가짜 CA 인증서의 "서명될" 부분의 구조는 충돌을 만드는 과정을 용이하게 하기 위해 우리가 (하지만 물론 X.509 표준의 한계 내에서) 선택했다. 여기서 CA에 의해 규정된 유일한 필드는 발행자의 구분명(Distinguished Name)이다. 가짜 CA 인증서의 구조는 다음과 같다.

byte 0 - 3:	헤더
byte 4 - 8:	버전 번호(3, 기본 값)
byte 9 - 11:	시리얼 번호(65, 임의의 선택)
byte 12 - 26:	서명 알고리즘("md5withRSAEncryption", 우리가 CSR에 설정한 것)
byte 27 - 118:	발행자 구분명(CA 기본값)
byte 119 - 150:	유효기간("31 Jul. 2004 0:00:00부터 2 Sep. 2004 0:00:00까지")
byte 153 - 212:	주체 구분명(subject Distinguished Name) (Common Name = "MD5 Collisions Inc. (http://www.phreedom.org/md5)")
byte 213 - 374:	주체 공개키 정보:
	byte 213 - 215: 헤더
	byte 216 - 230: 공개키 알고리즘("RSAEncryption")

byte 231 - 237: 헤더들

byte 238 - 369: RSA 계수 (1024 비트 값)

byte 370 - 374: RSA 공개 지수(65537)

byte 375 - 926: 확장된 부분:

byte 375 - 378: 헤더들

byte 379 - 395: 키 사용법("디지털 서명, nonrepudiation, 인증서 서명, 오프라인 CRL 서명, CRL 서명")

byte 396 - 412: 기본 제한사항("CA = TRUE, Path Length = None")

byte 413 - 476: 주체 키 식별자, 권한 키 식별자(흥미 없는 필드)

byte 477 - 926: tumor[29] (Netscape 확장)

이 인증서를 'CA 인증서'로 만드는 필드는 "CA = TRUE" 값을 가진 "기본 제한사항" 필드이다.

공개키가 가짜 CA 인증서 내에 있는 것에 대해 RSA key pair는 표준 key pair 생성 과정에서 OpenSSL에 의해 생성되었다. 개인키는 안전한 곳에 보관되어 있고, 우리는 그것을 공개적으로 발표할 계획은 없다. 이 개인키는 가짜 CA에 의해 서명된 최종 사용자 인증서를 발행하는데 필요하며, 그래서 이것은 아주 민감하다.

이 위험을 최소화하기 위해 가짜 CA 인증서의 유효 기간은 2004년(Xiaoyun Wang이 Santa Barbara에서 열린 Crypto Conference에서 첫 MD5 충돌을 발표했던 것이 2004년 8월임)의 상대적으로 짧은 기간으로 설정되어 있다. 일부러 날짜를 이렇게 소급하여 설정하였다. 이는 우리의 가짜 CA 인증서에 대한 개인키가 만약 나쁜 사람의 손에 넘어갈지라도 사용자 컴퓨터의 시스템 시간이 2004년의 이 짧은 기간으로 소급 설정될 때를 제외하고는 그것의 만기일 때문에 그 인증서는 모든 브라우저들에 의해 거절될 것이다. 이것이 부지부식간에 일어날 개연성은 그렇게 크지 않다.

우리가 선택한 시리얼 번호가 다소 낮은 것이라는 것을 주목해라. 이 시리얼 번호로 이 CA에 의해 발행된 실제 인증서가 이미 존재했는지 모른다. 우리는 낮은 번호를 선택했는데, 이는 대부분 그 시리얼 번호를 가진 가능한 다른 인증서가 오래 전에 이미 폐기되었다는 것을 아마도 의미하기 때문이다.

5.3.3. 정렬

실제 인증서에서 바이트 0-473(계수까지의 필드, 그리고 예측 가능한 헤더인 계수 필드의 첫 5 바이트)은 CA의 요구조건에 의해 아주 많이 고정되어 있다. 그 474 바이트는 실제 인증서 쪽에서

[29] (역자 주) tumor란 가짜 CA 인증서에 들어가 있는 427 바이트의 무의미한 데이터 블록을 가리킨다.

"chosen prefix"를 구성한다. 이 인증서에 대해 우리는 2048 비트의 RSA key를 가지도록 선택했다. 이 크기의 주된 이유는 거기서 충돌 블록을 우리가 숨겨야 한다는 사실 때문이다. 우리의 충동 설계 방법은 우리가 1632 비트의 충돌 블록을 만들 수 있도록 하며, 그래서 2048 비트는 적당한 선택으로 보인다. 더욱이 2048 비트 RSA 계수는 아주 일반적이어서 아무런 의심도 일으키지 않는다.

가짜 인증서의 측면에서 우리는 충돌 블록을 숨기기 위해 공개키 계수를 사용할 수 없었다. 이것의 이유는 우리가 실제 인증서에서 상응하는 필드와 반대로 "기본적인 제한사항" 필드의 CA 플래그에 "TRUE" 값을 설정하기를 원하기 때문이다. 문제는 이 "기본적인 제한사항" 필드가 공개키 다음에 나오고, 우리가 가진 충돌 형태로 충돌 블록 다음에 나오는 어떤 비트 차이도 허용할 수 없다는 것이다. 그래서 가짜 CA 인증서에서 공개키와 "기본적인 제한사항" 필드까지의 확장 둘 다 충돌 블록이 시작되기 전에 위치해야 한다.

이것을 성취하기 위해 실제 인증서의 subject Distinguished Name이 아주 큰 길이(적절한 크기의 Common Name을 선택함으로써 쉽게 확장될 수 있는)를 가지는 것이 도움이 되었다. 동시에 우리는 전체 1024 공개키와 "기본적인 제한사항" 필드를 위한 충분한 공간을 확보하기 위해 가짜 CA 인증서의 subject Distinguished Name을 가능한 짧게 잡았는데, 이 모든 것은 실제 인증서의 subject Distinguished Name 필드에 상응하는 영역에 위치할 것이다. 이와 같이 가짜 CA 인증서의 "chosen prefix"에 속하는 모든 필요한 필드들은 "to-be-signed" 부분의 첫 477 바이트에 위치할 수 있었다.

해결해야 할 다음 문제는 충돌 블록의 적어도 3 × 512 + 96 = 1632 비트(204 바이트)를 숨겨야 하는 가짜 CA 인증서의 확장에서 필드 하나가 필요하다는 것이다. 이 데이터가 그 충돌 설계 방법으로부터 나오고, 또한 우리는 몇몇 실제 세계 해석으로 그와 같은 블록을 만들 수 있을 만큼 충분한 통제력을 가지고 있지 않기 때문에 이 데이터는 랜덤하게 보인다. 그리고 우리는 문제를 한가지 더 가지고 있는데, 이는 실제 인증서로부터 나온 공개키 다음의 모든 데이터가 비트 단위로 가짜 CA 인증서에 복사될 필요가 있기 때문이다. 이 데이터는 실제 인증서와 관련된 해석을 가지고 있지만 이 해석은 그 가짜 CA 인증서에 대해 대부분 관련이 없고, 심지어 방해가 될 수 있다. 더불어 그 필드들은 427 바이트의 큰 크기를 가지고 있다.

이 두 문제는 427 바이트의 데이터(일부는 랜덤하게 보이고, 일부는 인증서를 보는 사람에게 보여서는 안되는 해석을 가지고 있는)를 가짜 CA 인증서에 숨길 공간을 동시에 요구한다. 우리가 채택한 해결책은 소위 "Netscape Comment"라는 블록을 정의하는 것이다. 이것은 어떤 데이터가 저장될 수 있는 속성 확장인데, 주요 브라우저들을 포함해 대부분의 인증서 처리 어플리케이션들에 의해 무시될 것이다. 그런데 이 필드의 내용들은 공식적으로 IA5String 형태여야 하는데 우리가 가진 내용은 그런 형태가 아니라는 작은 문제가 여기에 있다. 엄격하게 표준을 따르는 ASN.1 분석기(parser)는 Peter Gutmann의 프로그램 "dumpasn1"의 예에서처럼 이에 대해 불평할 것이다.

하지만 대부분의 어플리케이션 소프트웨어가 어떤 식으로던 그 확장을 무시하기 때문에 표준을 위반하는 필드를 가진 인증서도 실제로 여전히 받아들여질 것이다. 그 427 바이트가 다른 확장 필드에 숨겨졌을 수도 있다는 것을 우리가 인지할 수 있다(심지어 안에 영화를 숨긴 X.509 인증서[30]도 있다).

Netscape Comment 확장은 23 바이트의 헤더를 요구한다. 그래서 그 확장자의 내용은 양 인증서에서 500 바이트에서 시작하고, 그것은 이제 양 인증서에서 birthday 비트들이 시작하는 그 지점이 정확하게 될 수 있다. 이것은 계수가 474 바이트에서 시작하는 실제 인증서에는 그 공간을 채우기 위해 랜덤한 26 바이트가 처음에 있다는 사실을 설명한다. 기술적인 의미에서 그 두 인증서의 "chosen prefixe"들은 499 바이트에서 끝난다. 충돌 블록은 500 바이트에서 시작한다. 이 충돌 블록은 앞에서 말한 것처럼 96 + 3 × 512 = 1632 비트(204 바이트)로 구성되어 있고, 그래서 703 바이트에서 끝난다.

그래서 500 바이트부터 926 바이트까지 인증서 그 자체에는 아주 무의미한 427 바이트가 가짜 CA 인증서 안에 들어가 있다. 우리는 이 크고 분명히 불필요한 데이터 블록을 "tumor"라고 부른다. 우리의 관점에서 그것은 양호한 것이다.

실제 인증서의 공개키 계수를 474 바이트에서 시작한다. 앞에서 설명한대로 가짜 CA 인증서의 배열에 대해 우리는 첫 26 바이트를 어떻게 해서든 채울 필요가 있고, 그래서 우리는 그 계수가 시작할 랜덤한 26 바이트를 선택했다. MD5가 512 비트 블록으로 입력을 쪼개기 때문에 우리는 64 바이트 블록으로 생각해야 한다. 그래서 다음 512 비트 블록이 새로운 `IHV` 값으로 시작할 때까지 이제 12 바이트가 남아 있다.

그래서 "to-be-signed" 부분의 500 바이트에서 우리는 실제 인증서에서는 계수 안에 있고, 가짜 CA 인증서에서는 tumor 안에 있는 단계에 도달했다.

우리가 사용한 MD5 충돌 설계 방법은 "birthdaying" 단계와 "근접 충돌" 단계 두 개로 구성되어 있다. Birthdaying 단계는 두 개의 512 비트 블록 사이의 경계에서 두 개의 인증서 사이의 `IHV` 상의 다음 차이점을 준비하는 것을 의미한다. 그래서 우리는 그 birthday 비트들에 대해 이용 가능한 12 바이트(96 비트)를 가진다. Birthdaying 절차가 개선되었기 때문에 이것은 충분하고도 남는다. 사실, 우리의 새로운 birthdaying 테크닉 때문에 72 비트만 필요로 했고, 그래서 이용 가능한 그 12 바이트 중에서 3 바이트는 단순히 0으로 설정되었다. 양 인증서의 본질상 이 72비트의 birthday 비트는 양 인증서에서 완전히 다르며, 완전히 랜덤하게 보인다. 이 birthday bit는 우리의 충돌 발견 프로그램들에 의해 계산되었는데, 자세한 것은 아래를 보아라.

그런 다음 512 비트로 된 MD5 입력 블록 3개(각각 3x64 바이트)가 있는데, 이는 바이트 번호 512에서 703까지다. 이 3개의 입력 블록 각각은 `IHV`상의 어떤 차이점들을 일소하기 위해 우리의

---

[30] http://www.cs.auckland.ac.nz/~pgut001/pubs/dave.der

충돌 발견 방법들에 의해 설계되었다. Birthday bit와는 반대로 이 근접 충돌 블록들은 1 또는 2 비트에서만 다르다. 세 번째 블록의 끝에서 그 차이는 완전히 없어지고, 그래서 MD5 압축 함수에 대한 충돌을 구성한다. 이 인증서들의 704 바이트부터 정확하게 비트 단위로 동일하다.

실제 인증서에서 우리는 이제 26 + 12 + 3 × 64 = 230 바이트의 계수를 가지고 있다. 나머지 26 바이트의 계수는 앞에서 설명한 것처럼 그것이 안전한 RSA 계수라는 것을 확신하기 위해 계산되었다.

그런 다음 실제 인증서에는 공개 지수와 모든 "버전 3 확장"들이 따르는데, 전체적으로 197 바이트가 필요했다. 그것들은 CA에 의해 실제 인증서에서 생성되었고, 그런 다음 가짜 CA 인증서에 있는 tumor 안으로 복사된다. 거기에 있는 필드들 중의 하나는 "기본적인 제한사항" 필드이며, "CA = FALSE"가 설정되어 있다. 흥미롭게도 이 필드의 바이트들 역시 tumor에 존재하지만 그 인증서 처리 소프트웨어는 그것들이 tumor의 일부이기 때문에 그것들에 첨부된 어떤 해석을 가진 것으로 그것들을 인정하지 않을 것이다.

실제 인증서에 확장과 가짜 CA 인증서에서 tumor가 끝난 다음 926 바이트에서 이 두 인증서의 "to-be-signed" 부분이 끝나고, MD5 해쉬 함수는 충돌하는 값을 만들어내기 위해 이 926 바이트 길이의 문자열에서 호출될 수 있다.

그 인증서 안에서 최종적으로 따르는 것은 "서명 알고리즘"과 "서명" 필드이다. 당연히 이 필드들은 양 인증서에서 동일하다.

양 "to-be-signed" 부분을 가지고 있는 바이너리 파일들은 다음에서 다운받을 수 있다.

- 실제 인증서의 "to-be-signed" 부분, 바이너리 파일
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/real.cert.tbs.bin)

- 실제 인증서의 "to-be-signed" 부분, 사람이 읽을 수 있는 16 진수 파일
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/real.cert.tbs.hex)

- 가짜 CA 인증서의 "to-be-signed" 부분, 바이너리 파일
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/rogue_ca.cert.tbs.bin)

- 가짜 CA 인증서의 "to-be-signed" 부분, 사람이 읽을 수 있는 16 진수 파일
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/rogue_ca.cert.tbs.hex)

위의 두 "to-be-signed" 부분이 충돌하는 것을 확인하기 위해 위의 바이너리 파일들에 대해 MD5 프로그램을 실행시켜라. 그리고 이 두 "to-be-signed" 부분이 다르다는 것을 확인하기 위해 바이너리 파일들에 대해 SHA-1 프로그램을 실행해라.

다음은 MD5 Intermediate Hash Value이다:

IHV	**실제 인증서**	**가짜 인증서**
IHV_0	0123456789ABCDEFFEDCBA9876543210	0123456789ABCDEFFEDCBA9876543210
IHV_1	058484A77F07A36382AAECF2DFE207A2	713F764E78B5C9B03F8878F7A440551B
IHV_2	D52743425C3DAC23A9E62C6C9670622E	2AC9681DDB3B72D29A1422A515C9E4F4
IHV_3	7789E58E3B45621A3E46A64CA9D7AC3A	104DD09F9F651E554C528578AC1F6885
IHV_4	CDA2CB5673D3D32092C7F1EF80CE5729	15ADC95447929A2AC0EACF9E618E14EB
IHV_5	F08E24604482508B959A0B5762207A3F	D6D6E59C0BDB1F701CB04C29A0573EA0
IHV_6	A83EA6CCCC50B41A4BFADBC6D856B338	3AAB0CE98F1E9B2AC270A5A2C60FF605
IHV_7	0B42EAAB4258AACA8C30BDB8192A1BC0	DE3CCC11526732CA0FD8B9F5992A7673
IHV_8	D21CED8CC56726B6BF2AE4A93D742C3A	D21CED8CCE3D7EB4C82ADCA94674243A
IHV_9	DC1EDBFFF3C3E9E7BCEB3F9E2D0705BD	DC1EDBFFF49941E8BDEB379E2E07FDBC
IHV_{10}	F0D655805A71A74EF8A6A630D11977D8	F0D655805A479F4EF8A69E30D1196FD8
IHV_{11}	9808B5471E7130CC5A30A2ABF2BE4B4D	9808B5471E7130CC5A30A2ABF2BE4B4D
IHV_{12}	AA1F57B21A8732130CB0CAEF4BB9C746	AA1F57B21A8732130CB0CAEF4BB9C746
IHV_{13}	151754FA2FCC5914E72B71B4300B6485	151754FA2FCC5914E72B71B4300B6485
IHV_{14}	271EECDC4DAC9E9C471C34C833917E26	271EECDC4DAC9E9C471C34C833917E26
IHV_{15}	9ED7B966BD815C141B899DC64B528564	9ED7B966BD815C141B899DC64B528564
MD5	9ED7B966BD815C141B899DC64B528564	9ED7B966BD815C141B899DC64B528564

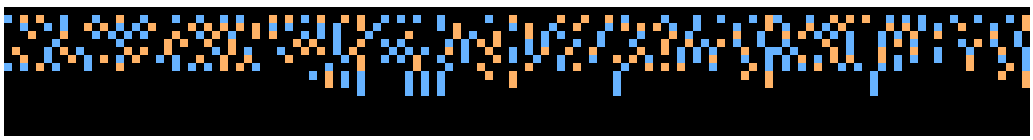
다음은 MD5 Intermediate Hash Value의 XOR의 차이점이다:

IHV	XOR 차이점
IHV_0	00000000000000000000000000000000
IHV_1	74BBF2E907B26AD3BD2294057BA252B9
IHV_2	FFEE2B5F8706DEF133F20EC983B986DA
IHV_3	67C43511A4207C4F7214233405C8C4BF
IHV_4	D80F02023441490A522D3E71E14043C2
IHV_5	2658C1FC4F594FFB892A477EC277449F
IHV_6	9295AA25434E2F30898A7E641E59453D
IHV_7	D57E26BA103F980083E8044D80006DB3
IHV_8	000000000B5A5802770038007B000800
IHV_9	00000000075AA80F010008000300F801
IHV_{10}	00000000036380000003800000001800
IHV_{11}	00000000000000000000000000000000
IHV_{12}	00000000000000000000000000000000
IHV_{13}	00000000000000000000000000000000
IHV_{14}	00000000000000000000000000000000
IHV_{15}	00000000000000000000000000000000
MD5	00000000000000000000000000000000

SHA-1 해쉬 값들은 각각 다음과 같다:

=> 1AA8050F29EECAAFE1F6815487FF164783129E63과 5D23230D1B36C8DDD21CB5ABDAED46967D1ED4AD

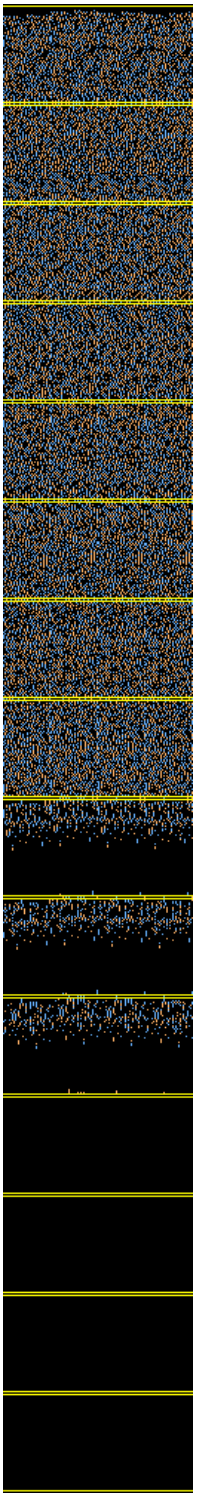
MD5 IHV들의 차이점은 다음 이미지에 나타나 있다.



각 라인은 128비트로 된 IHV 하나의 차이(이번에는 XOR 연산의 차이가 아니라 32비트 word에 대한 근접하지 않은 형태의 차이며; 검정색 픽셀은 아무런 차이가 없음을 의미하고, 푸른색 픽셀은 a+1 차이를 의미하고, 오렌지 픽셀은 a-1 차이를 의미한다)를 나타낸다. IHV_1부터 IHV_7까지에서 IHV 차이는 랜덤하게 보이고, IHV_8에서는 특별해지고(이것은 birthdaying 때문이다), 다음 3개의 입력 블록에서는 하나씩 취소되며, 그래서 IHV_{11}에서는 완전히 사라졌다.

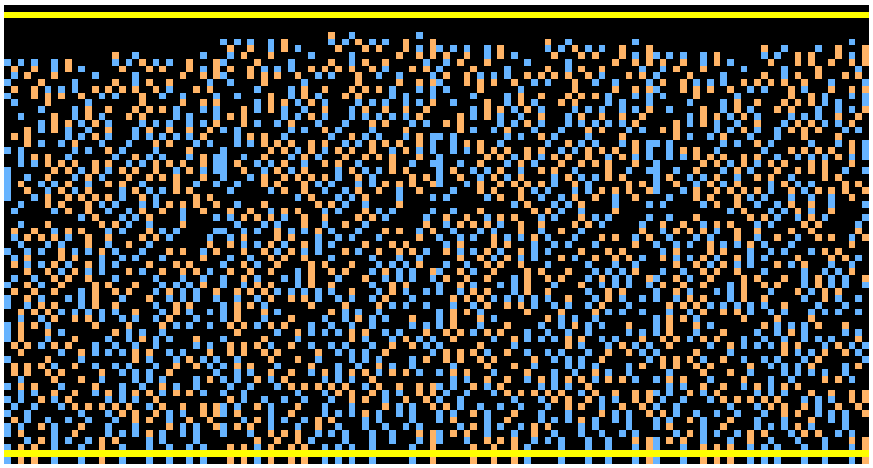
5.3.4. 충돌 설계

이 chosen-prefix 충돌 설계 방법은 기본적으로 우리의 EuroCrypt 2007 문서[31]에 기술되어 있다. 하지만, 현재 적용이 가능하게 만들어진 이 방법에 몇 가지 중요한 개선책이 개발되었다. 그 개선책에 대한 세부사항들은 곧 발표될 학술 논문에서 공개될 것이다. 그 방법의 기본적인 아이디어는 두 부분으로 되어 있다. 이 방법의 두 번째 부분에서 IHV들에 있는 특별한 비트 차이점들을 제거할 수 있는 차별적인 경로들이 설계된다. 이것은 근접 충돌을 찾는데 호출될 수 있다. 그래서 연속적인 몇몇 "근접 충돌 블록"들에서 특별한 조합의 비트 차이점들이 제거될 수 있다. 각 비트 차이에 대해 새로운 차별적 경로가 설계되어야 하고, 그래서 차별적 경로들을 만드는 자동화된 방법을 가지는 것이 중요하다.



이 방법의 첫 부분은 IHV 차이를 준비하고 그래서 그것은 두 번째 부분에 맞게 된다. 이것은 적절한 구조를 가진 IHV 차이를 가진 입력 쌍에 대한 birthday 검색에 의해 간단히 이루어진다. Birthday 비트 수와 근접 충돌 블록들의 수 사이의 거래(trade-off)를 최적화하는 것은 앞으로 발표될 논문에서 논의될 또 다른 하나의 항목이다. 당장의 경우 우리는 72 birthday 비트와 3개의 근접 충돌 블록들에 대해 선택하기로 결정했다. 계산상의 병목현상이 일어나는 것은 birthday 검색이다.

충돌 설계를 예증하기 위해 우리는 MD5의 내부 상태에서의 bit 차이들에 대한 몇 가지 그림[32]을 만들었다.



[31] Marc Stevens, Arjen Lenstra and Benne de Weger, "Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities", In: Moni Naor (editor), "Advances in Cryptology - EUROCRYPT 2007", volume 4515 of Lecture Notes in Computer Science, pages 1-22, Springer Verlag, Berlin, 2007.

[32] 이 이미지의 움직임 부분을 포함해서 전체 이미지를 제대로 보기 위해서는 다음 주소를 참고하라. http://www.win.tue.nl/hashclash/rogue-ca/images/botsing.gif

MD5에 대한 전체 입력은 각각 512 비트로 된 15개의 입력 블록들의 쌍으로 구성되어 있다. 각각의 압축 함수 호출에서 입력 IHV는 128 비트의 내부 상태(internal state) 속으로 입력된다. 이 압축 함수는 64번 수행되고, 매회 내부 상태를 업데이트하며, 입력 블록으로부터 몇 비트를 사용한다. 64번 째 수행 후 그 내부 상태는 출력 IHV를 만들기 위해 입력 IHV로 XOR된다.

위의 그림 오른쪽에서 매 실행 사이에서 IHV들과 64번째 내부 상태 사이의 차이점들이 보인다. 여기서 매번 두 입력 파일들 사이의 차이가 발생한다. 이 차이점들은 32비트 word에 대해 비근접 형태로 계산된다. 그림에서 각 라인은 128 비트의 IHV 또는 내부 상태 차이를 보여준다. 위의 애니메이션화된 그림[33](앞에서 왼쪽 그림)에서, 제일 위에 있는 노란색 구분선 위의 첫 라인은 입력 IHV 차이를 보여주고, 제일 아래 노란색 구분선 아래에 있는 마지막 라인은 출력 IHV 차이를 보여주고, 노란색 구분선들 사이에 있는 64개의 라인들은 매번 실행 후 64개의 내부 상태 차이들을 보여준다. 이 애니메이션은 한번에 하나씩 15개의 압축 함수 호출 과정을 보여준다. 오른쪽 세로 그림은 애니메이션화된 것보다 큰 하나의 열에 있는 같은 데이터를 보여준다. 각 출력 IHV는 다음 압축 함수 호출에 대한 입력 IHV임을 주목해라.

chosen-prefix 충돌을 설계하기 위해 한번 시도할 때 약 하루 이상이 걸린다. birthday 검색을 구성하는 첫 번째 단계는 계산상 가장 비싸다. 운 좋게도 그것은 Sony PlayStation 3이 사용하는 Cell Processor의 특별한 SPU core에 아주 적합하다. 우리는 스위스 Lausanne의 EPFL에 있는 Arjen Lenstra's Laboratory for Cryptologic Algorithms의 "PlayStation Lab"[34]에 우리가 처리할 수 있는 200개 이상의 PlayStation 3을 가지고 있었다(그림 참고; 이 클러스터를 획득하는데 EPFL DIT의 후원을 받았고, Swiss National Science Foundation이 필요한 장비를 지원해주었다). Birthday 과정은 각각 30GB의 메모리를 사용하는 200개의 PS3에서 약 18시간이 걸린다.

두 번째 단계는 첫 단계 다음에 남아 있는 IHV 차이점들을 제거하는 3개의 충돌 블록들을 계산하는데, 고성능 quadcore pc에서 약 3~10시간 정도 걸린다. 이 부분은 많은 메모리가 필요하고, 소프트웨어 실행 흐름에서 높은 수의 분기(branch) 때문에 PS3 SPU core에는 적합하지

---

[33] http://www.win.tue.nl/hashclash/rogue-ca/images/botsing.gif

[34] http://www.win.tue.nl/~bdeweger/PS3Lab/

않다. 이 두 단계가 다른 하드웨어에서 실행되기 때문에 우리는 여러 시도를 연결(pipeline)했고, 그래서 일주일(72시간) 동안 우리는 쉽게 3번(3 × 18 + 10 = 64시간) 시도할 수 있었다. 충돌 설계의 전체 복잡성은 30GB의 메모리가 가능할 때 $2^{51.x}$번의 MD5 압축 함수 호출로 평가될 수 있다.

5.4. 실제 실행

한번의 단일 시도로 CA 서명을 획득하기 위해 우리는 시리얼 번호와 CA가 며칠 미리 사용할 유효기간을 추측해야만 했다. 이 정보는 실제 인증서의 "to-be-signed" 부분에 대한 chosen-prefix를 획득하기 위해 중요하며, 이것은 충돌 설계 방법에 대한 입력이다. 일단 충돌이 발견되면 공개키 계수가 계산되어야 한다. 그런 다음 Certificate Signing Request가 구성되어 정확한 시간에 CA로 보내질 수 있고, 그런 다음 우리는 CA가 서명된 인증서를 반환하기를 기다려야 하며, 그 시리얼 번호와 유효기간이 정확하게 추측되었는지 확인해야 한다.

우리는 CA의 더 낮은 로드(load)를 이용하기 위해 일주일 이상 시도를 했다. 매주 우리는 예상된 범위 내의 시리얼 번호를 가진 하나에서 세 개의 충돌을 생성하려고 했으며, 원하는 값까지 시리얼 번호를 증가시키기 위해 충분한 인증서들을 구입하려고 했다. 하지만 첫 세 번의 시도는 타이밍 문제 때문에 실패했다.

우리는 겨우 1초 늦게 서명된 몇 개의 인증서와 올바른 시리얼 번호를 구하기 위한 우리의 시도 때문에 방해를 받은 CA의 다른 고객들에 의한 여러 번의 요구를 받았다. 최종적으로, 넷째 주에 우리는 적절한 시간에 서명된 올바른 시리얼 번호를 가진 인증서를 구하는데 성공했다.

우리가 RapidSSL로부터 인증서를 구입하는데 USD 45의 비용이 들었다. 하지만, CA는 우리가 무료로 20번까지 각 인증서를 다시 발행하는 것을 허용했으며, 이것은 인증서 하나를 요청하는데 겨우 2.25 달러가 필요하다는 것을 의미했다. 이 프로젝트를 위한 인증서를 구입하는데 총 657달러를 사용했다.

5.5. 결과

우리의 시나리오를 실행한 결과 두 개의 인증서를 획득했다. 우리는 .cer와 .pem 포맷으로 다운로드 받을 수 있게 했다. 더욱이 우리는 인증서의 구조를 더 읽기 편한 형태로 나타내기 위해 "dumpasn1"[35] 프로그램을 인증서에 적용한 출력 결과를 제공한다.

- 실제 인증서(.cer)

[35] http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c

(http://www.win.tue.nl/hashclash/rogue-ca/downloads/real.cert.cer)

- 실제 인증서(dumpasn1 의 출력물)
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/real.cert.txt)

- 가짜 CA 인증서(.cer)
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/rogue_ca.cert.cer)

- 가짜 CA 인증서(.pem)
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/rogue_ca.cert.pem)

- 가짜 CA 인증서(dumpasn1 의 출력물)
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/rogue_ca.cert.txt)

그 인증서에 대한 개인키의 다운로드가 가능하지는 않지만 이것이 우리 인증서의 신뢰성을 확인하는데 방해가 되지는 않을 것이다.

5.6. 데모

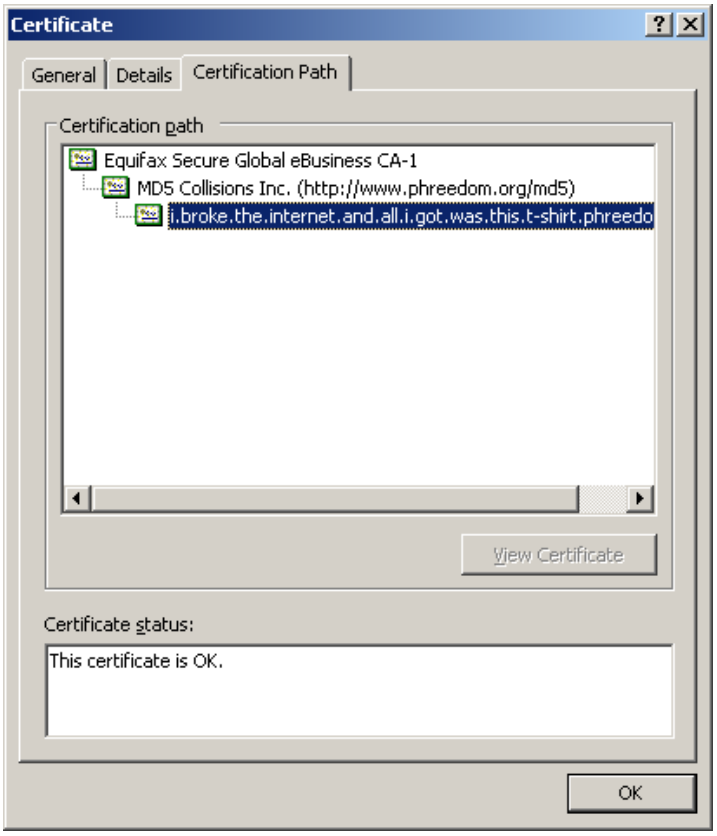
우리의 가짜 CA 인증서가 제대로 작동하는지를 보기 위해서는 컴퓨터 시스템 시계를 2004년 8월로 설정한 다음 https://i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org를 클릭해라. 제대로 작동하는지 그 여부를 확인했다면 여러분의 시스템 시계를 다시 설정하는 것을 잊지 마라.

우리의 가짜 CA에 의해 서명된 예제 웹 사이트의 인증서를 다른 포맷으로 다운로드 받을 수 있다:

- 가짜 인증서에 의해 서명된 예제 웹 사이트 인증서(.cer)
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/website_localhost.cer)

- 가짜 인증서에 의해 서명된 예제 웹 사이트 인증서(.pem)
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/website_localhost.pem)

- 가짜 인증서에 의해 서명된 예제 웹 사이트 인증서(dumpasn1 의 출력물)
  (http://www.win.tue.nl/hashclash/rogue-ca/downloads/website_localhost.txt)



6. 영향

우리 시나리오의 결과를 간단히 요약하면, 우리는 인증서가 대부분의 브라우저들에 의해 기본적으로 받아들여질 가짜 Certification Authority를 소유하게 되었다는 것이다. 그래서 우리는 합법적인 웹 사이트라고 주장하는 가짜 웹 사이트들을 포함해 우리가 원하는 어떤 웹 사이트에도 SSL 인증서를 발행할 수 있다.

이것은 다음 취약점들을 공격함으로써 가능했다:

- MD5 해쉬 함수에 대한 "chosen-prefix 충돌"을 설계하기 위한 효율적인 방법
- 다음과 같은 작업을 하는 적어도 하나의 상용 Certification Authority가 있다:

  - MD5 해쉬 함수를 이용해 만들어진 서명을 가진 인증서 발급
  - 자동화된 방법으로 인증서들에 대한 온라인 요청을 처리
  - 변칙적인 요청에 대해서는 점검하지 않음
  - 시리얼 번호와 유효 기간의 효과적인 혼합에 대해 합리적인 성공 개연성으로 예측하는 것을 허용
  - 일련의 인증서 길이에 대해 기술적으로 강제된 제한을 가지고 있지 않음

안전하던(SSL을 사용하기 때문에) 안전하지 않던, MD5, SHA-1, SHA-256 기반이던 또는 어떤 다른 형태의 인증서 기반이던, 인증서를 발행한 Certification Authority와는 상관 없이 어떤 웹 사이트도 진짜 인증서를 가진 것처럼 보일 수 있고, 특히 MD5 기반의 인증서를 가진 웹 사이트들은 이에 취약하다.

우리가 가진 가짜 Certification Authority는 데모 목적으로만 설치되었다. 이 가짜 Certification Authority의 인증서 날짜는 일부러 뒤로 조정되었으며, 그래서 브라우저들은 그 인증서의 유효기간이 4년 전에 만기된 것으로 인식할 것이다. 더욱이 인증서를 발행하는데 필요한 상응하는 개인키도 남용을 막기 위해 안전한 곳에 보관될 것이다. 그래서 이 특정 가짜 Certification Authority 때문에 피해가 발생하지는 않을 것 같다.

하지만 우리는 그와 같은 가짜 Certification Authority를 설정하는 것이 우리의 상당한 노력으로 실제 가능하다는 것을 보여주었다. 만약 우리가 지금 그것을 할 수 있다면 다른 사람들도 멀지 않은 미래에 그것을 역시 할 수 있을 것이다. 이론상 다른 그룹이나 사람들이 이미 비슷한 기술들을 개발했거나 우리가 가지고 있는 지식을 가지고 있어 이미 그들의 의도대로 웹 사이트들을 사칭할 수 있을지도 모른다. 우리는 이것이 실제로 그런지 아무런 징후도 가지고 있지 않다. 이 때문에 우리는 가능한 빨리 MD5 사용을 멈추는 것이 중요하다고 믿는다.

Kaminsky의 "DNS 결함" [36] 과 같은 DNS 프로토콜의 알려진 취약점과 조합해 우리가 공개한 취약점은 실제로 탐지할 수 없는 피싱 공격을 가능하게 했다.

피싱 공격에 대해 알지 못한 체 사용자들은 그들이 방문하고자 하는 실제 신뢰할 수 있는 금융 및 상거래 웹 사이트와 완전히 같은 것처럼 보이는 악의적인 사이트로 리다이렉트될 수 있다. 그래서 사용자의 패스워드와 다른 개인 데이터가 나쁜 사람의 손에 들어갈 수 있게 된다.

우리는 범죄 의도를 가진 사람들을 돕기를 원하지 않는다. 그래서 우리는 가짜 Certification Authority 인증서를 획득하는 방법에 대한 전체 세부 내용을 당분간 발표하지 않을 것이다. 하지만 MD5에 대한 "chosen-prefix 충돌"을 설계하는 기본 원칙들은 2007년 5월에 이미 공개되었다는 것을 주목해야 하며, 이에 대한 것은 "Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities" [37] 와 "Chosen-prefix 충돌" 웹 사이트 [38] 를

---

[36] Dan Kaminsky, "Black Ops 2008: It's the end of the cache as we know it" http://www.doxpara.com/DMK_BO2K8.ppt, August 2008.

다음도 역시 참고해라.
Matthew Olney, Patrick Mullen, Kevin Miklavcic, "Dan Kaminsky's 2008 DNS Vulnerability", Snort Whitepaper, http://www.snort.org/vrt/docs/white_papers/DNS_Vulnerability.pdf, July 2008.

[37] Marc Stevens, Arjen Lenstra and Benne de Weger, "Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities", In: Moni Naor (editor), "Advances in Cryptology - EUROCRYPT 2007", volume 4515 of Lecture Notes in Computer Science, pages 1-22, Springer Verlag, Berlin, 2007.

참고해라. 우리의 현재 결과를 가능하게 만들기 위해 공개적으로 알려진 이 기술들은 중요한 점에서 개선되어 왔다. 이 개선점들은 학술 논문에서 곧 발표될 것이다. 이와는 별개로 우리가 했던 작업을 다시 하고자 원하는 사람은 엄청난 개선과 최적화 노력을 해야만 할 것이다. 당분간 우리는 그와 같이 개선된 세부 내용들을 발표할 계획을 가지고 있지는 않다.

SSL을 사용하는 안전한 웹 커뮤니케이션 이외의 다른 어플리케이션들도 역시 취약할 수 있다. MD5 기반의 인증서들에 대한 요청에 응하고 충분히 예측 가능한 시리얼 번호와 유효기간을 가진 모든 Certification Authority도 비슷한 공격에 취약할 수 있다. 이것은 이메일 서명, 암호, 소프트웨어 서명, 부인-봉쇄(non-repudiation) 서비스[39] 등의 영역의 Certification Authority들도 포함할 수 있다.

우리 작업의 영향은 우리의 가짜 CA가 어떤 사람(우리 자신들을 포함해)에 의해 남용될 가능성이 높지는 않다고 생각하는 의미에서 아주 제한적이라고 말할 수 있을 것 같다. 우리는 우리가 실제 공격 시나리오를 가지고 있다는 것을 보여주기 위해 우리가 사용한 방법들의 세부내용을 충분히 공개했다. 우리가 만든 충돌하는 인증서들은 우리 작업의 물질적 보수의 증거가 될 수 있으며, 그래서 모든 사람들이 우리의 주장을 확인할 수 있고, 그리고 이에 대해서는 데모에서 이미 제시했다.

우리의 의도는 MD5가 철저하게 깨져서 디지털 서명 체계와 인증서에서 MD5를 지속적으로 사용할 경우 실질적인 위협을 초래할 것에 대한 인식을 높이는 것이다.

우리가 원하는 것은 CA들이 새로운 인증서를 발행하는데 있어 MD5 사용을 멈추는 것이다. 또한 다른 어플리케이션에서 MD5를 사용하는 것 역시 재고되기를 희망한다. 다른 적절한 대안이 이미 가능하기 때문이다.

6.1. 파기(revocation) 문제

우리 작업을 통해 관찰한 흥미로운 점 하나는 일반 브라우저에서 이용 가능한 파기 메커니즘을 이용해 우리가 만든 가짜 인증서를 폐기하는 것이 아주 어렵다는 것이다. 인증서 파기를 위한 프로토콜에는 CRL과 OSCP 두 가지가 있다. Firefox 3과 IE 7 이전까지 인증서 파기는 기본적으로 비활성화되어 있었다. 심지어 가장 최신 버전에서도 그 브라우저들은 파기 서버를 가리키는 URL을 포함시키기 위해 인증서에 의존하고 있다. 우리의 가짜 CA 인증서는 아주 제한된 공간을 가지고 있으며, 파기 서버와 같은 URL을 포함하는 것이 불가능했는데, 이것은 기본 설정상

---

[38] http://www.win.tue.nl/hashclash/ChosenPrefixCollisions/

[39] (역자 주) 앞에서도 언급했지만, 통신의 일부 속성이 발생한 사실과 그 내용이 변경 없이 제대로 전달된 것을 거부하는 것을 방지하거나 규제하는 서비스(예를 들어, A와 B가 서로 온라인 거래를 했는데, A는 입금을 완료했지만 B가 아직 물건을 배송하지 않고 있어 거래 시 지정한 날짜보다 배송이 늦어지고 있다는 것을 이야기 하며 신속한 배송을 요구하자 B는 배송 날짜를 A가 다음 주로 지정했다고 주장하는 것 등을 방지하는 것)

Internet Explorer와 Firefox 둘 다 우리의 인증서를 확인하기 위한 파기 서버를 발견할 수 없다는 것을 의미한다.

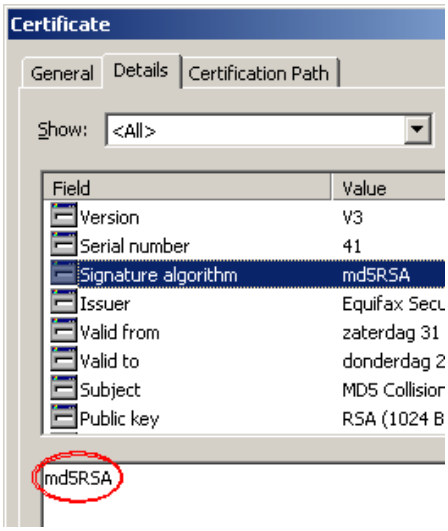
해결책의 하나로 시스템 관리자들은 파기 정보를 위해 회사가 지정한 특정 OSCP 서버를 브라우저들이 항상 질의하도록 설정할 수는 있지만 이 옵션을 최종 사용자들이 쉽게 이용할 수는 없다. 이것은 비슷한 공격을 통해 미래에 파기될 필요가 있는 인증서를 만들어내는 경우에 여전히 해결되어야 할 한가지 중요한 문제를 지적하고 있다.

또 다른 흥미로운 시나리오는 우리의 결과가 기존의 Certification Authority들에 대한 DoS 공격의 가능성을 열었다는 것이다. 만약 공격자가 어떤 사람(예를 들어 경쟁 웹 상점)의 인증서가 폐기되기를 원한다면 그는 공격 대상 인증서와 같은 시리얼 번호를 가진 같은 Certification Authority로부터 가짜 인증서를 획득하기 위해 우리의 테크닉을 이용할 수 있다. 이 가짜 인증서가 공개된다면 Certification Authority는 그것을 즉시 폐기하는 것 이외의 그 어떤 선택도 없다. 파기가 인증서의 시리얼 번호를 바탕으로만 하기 때문에 희생자의 합법적인 인증서도 동시에 폐기될 것이다. 이 공격은 만약 Certification Authority의 root 인증서가 그 공격 대상이 된다면 아주 강력한 것이 될 것이다. 공격자는 그래서 이 root 인증서의 파기를 강제하고, 그 root 인증서에 의존하는 전체 인증서 트리도 강제로 파기할 수 있다.

7. 대응책

무엇보다 먼저 SHA-2와 같은 충분히 강력한 대안들을 쉽게 이용할 수 있을 때 깨진 암호 원형(cryptographic primitive)을 계속 사용하는 것에 대한 그 어떤 적절한 변명도 없을 것이다.

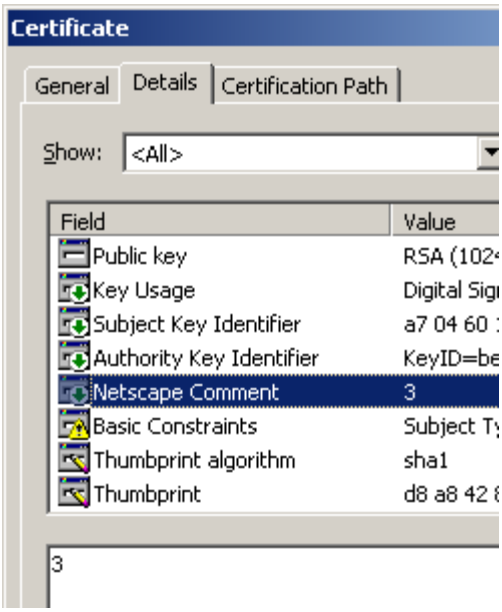
두 번째, 보안 의식을 대체할 수 있는 것은 없다. 보안 문제, 취약점, 그리고 기술적 가능성에 대한 개방의식은 인터넷을 더 안전한 곳으로 만들기 위한 아주 소중한 것이다. 전문가들의 충고는 그 과정에서 초기에 진지하게 고려되어야 한다. 이 경우 MD5는 2004년 이후에 단계적으로 곧 폐지되어야만 했다.



대응책에 대해 말하는 동안에도 다른 이해관계자들이 채택할 수 있는 대응책들을 우리는 지속적으로 토론할 것이다.

웹을 검색하는 **사용자**들은 잘못 신뢰된 SSL 인증서를 가진 가짜 웹 사이트로 리다이렉트될 수도 있다. 하지만 사용자는 이것이 일어나는 것을 막기 위한 어떤 것도 할 수 없다. 사용자가 그것을 탐지하려고 시도는 하겠지만 평균적인 웹 사용자들에게 이것은 다소 성가신 것임을 우리는 알고 있다. 그 사용자는 그가 방문하고 있는 웹

사이트에 대한 인증서의 세부내용들을 보기 위해 브라우저의 관련 버튼이나 메뉴 아이템을 클릭할 수 있다. 사용된 해쉬 함수는 "Signature algorithm" 필드에서 볼 수 있는데, 위의 그림에서 "md5RSA"는 MD5가 인증서를 서명하는데 사용되었다는 것을 의미한다. 체인에서 root CA 인증서까지의 모든 인증서들이 MD5보다는 SHA-1과 같은 다른 해쉬 함수들을 사용할 때 우리의 공격은 사용되지 않았다.



MD5가 사용되었을 때 비트 레벨에서 인증서를 정밀 검사할 경우 부정 행위는 탐지될 수도 있다. 그 인증서가 내부에 "tumor"를 가지고 있을 때 사용자 친화적인 인증서 뷰어로 이것을 볼 수도 있고 못 볼 수도 있다. 왼쪽의 그림을 보면, 3이라는 값을 가진 이상한 "Netscape comment" 필드가 있다. 실제로 이것은 기술적인 이유들 때문에 이 뷰어에서 스크린 상에 보이지 않는 427 바이트 필드이다. "dumpasn1"[40]과 같은 툴을 이용해 그 인증서를 검사할 수 있는 전문가는 인증서에 있는 그와 같은 이상한 필드를 확인할 수 있을 것이다. 하지만 표준 사용자는 어떤 것도 목격하지 못할 것 같다. 그러므로, 인증서를 검사하는 것은 강력한 대책은 아니다.

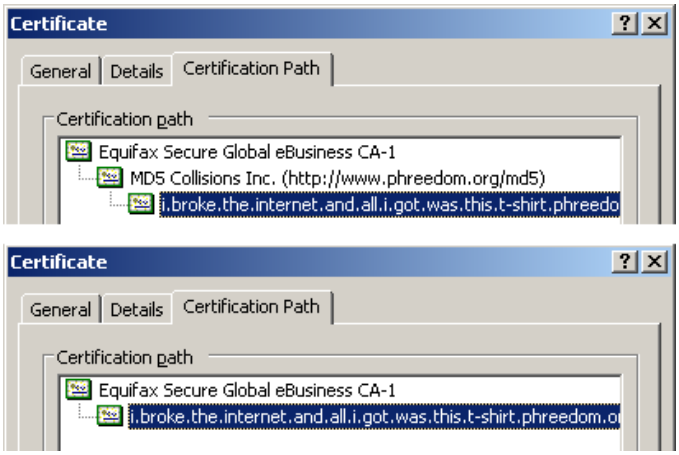
예방책들은 훨씬 더 강력하다. 우리는 이제 CA들과 브라우저 벤더들이 취할 수 있지만 사용자들은 취할 수 없는 예방책을 기술할 것이다.

Certification Authority들은 다 함께 MD5 사용을 멈추기를 권장한다. Chosen prefix 공격을 막기 위해 CA들은 인증서 필드들(가능하다면 인증서의 시작 부분이 더 좋음)에 충분한 양의 랜덤화를 추가할 수 있다. 시리얼 번호는 이런 목적을 위해 사용하기 좋은 필드인데, 왜냐하면 시리얼 번호는 인증서의 시작부분에 아주 가까이 있고, 20바이트의 랜덤화를 허용하기 때문이다. 많은 CA들은 아마도 다른 이유들 때문이기는 하지만 이미 무작위적인 시리얼 번호를 사용하고 있는 것처럼 보인다. 독일 서명법(German Signature Law)은 해쉬 입력에 몇 가지 랜덤화를 제공하기 위해 검증을 거친 SHA-1 인증서[41] 들에 대해 랜덤한 시리얼 번호들을 규정하고 있다. 하지만 우리는 시리얼 번호에 랜덤화를 이렇게 사용하는 것이 X.509 표준에서 운 좋은 선택들에 의해 가능해진 차선책임을 유의해야 한다. 공격자가 해쉬 입력을 선택할 수 있을 때 그 입력에 랜덤화를 추가하는 것은 일반적으로 나쁜 생각은 아니다. 설계된 이후로 훨씬 더 신뢰할 수 있는

[40] http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c

[41] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, "Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen)", http://www.bundesnetzagentur.de/media/archive/14953.pdf, November 2008.

해결책은 랜덤화된 해쉬를 사용하는 것인데, 이에 관한 논문[42]과 Firefox에 구현한 것[43]을 참고해라. 그와 같은 해결책은 랜덤화를 해쉬 함수들에 대한 "작동 모드"로 소개했으며, 이것은 비보안적인 이유들 때문에 또는 아무런 이유 없이 기존의 표준들에 우연히 존재하게 된 기능들에 의존하는 것 보다 그 문제에 대해 훨씬 더 기본적인 접근 방법이다.



Certification Authority가 취할 수 있는 다른 대책들은 "기본적인 제한사항들" 필드에 있는 "pathlength" 파라미터를 적극적으로 사용하는 것이다. 어떤 CA가 최종 사용자의 인증서를 발행하는 것에 대한 정책만을 가지고 있을 때 path length를 0으로 설정함으로써 이 정책은 기술적으로 강화될 수 있다. 우리의 특정 공격 시나리오는 3차원(three-level)적인 계층구조 때문에 탐지될 수 있다(오른쪽 그림에서 위의 그림 참조). 하지만 우리의 공격 시나리오를 조그만 변형해도 이 탐지를 회피할 수 있고, 여기서 충돌하는 가짜 인증서는 CA 인증서가 아니라 원래 웹 사이트 인증서이다(오른쪽 그림에서 아래 그림 참고). 우리는 그와 같은 충돌하는 가짜 최종 사용자 인증서가 tumor 없이 설계될 수 있다는 것을 주목하는데, 충돌 블록은 공개키 내부에서 다시 숨겨질 수도 있다. 이것은 탐지를 훨씬 더 어렵게 만든다. 그 인증서를 구성하는 다른 방법들도 가능할 수 있다.

한가지 흥미로운 의문은 CA들이 MD5를 사용해 서명된 기존의 인증서들을 파기해야 하는지 여부이다. 현재 공격 시나리오가 2007년 5월 이후 원칙적으로 가능해졌고, 그래서 이 날 이후 발행된 MD5로 서명된 모든 인증서(또는 모든 CA 인증서들)들은 공격을 받았을지도 모른다고 논쟁할 수 있다. 그 인증서들이 실제로 공격을 당했는지 여부는 관계 없다. 중요한 것은 그 인증서들을 신뢰할 필요가 있는 관계자들이 그 인증서가 신뢰된 것인지 아닌지를 확인하기 위한 적절한 방법을 가지고 있지 않다는 것이다. 가짜 인증서들을 만드는 공격자가 인증서의 날짜를 그가 원하는 과거의 어느 날로 조정하는 것이 쉽고, 그래서 MD5 기반의 어떤 인증서가 위조된 것일지도 모르기 때문에 MD5 기반의 오래된 모든 인증서들이 폐기되어야 한다고 주장할 수 있다. 반면, 이 시나리오들의 가능성이 아주 낮고, 많은 MD5 기반의 인증서들을 대체하는 비용은 많을 수 있기 때문에 기존의 MD5 기반의 인증서들을 계속 사용하는 것은 위험할 수 있다고도 주장할 수 있다. 그러나 이와 상관 없이 MD5는 새로운 인증서들에 대해 더 이상 사용되어서는 안된다.

---

[42] Shai Halevi and Hugo Krawczyk, "Strengthening Digital Signatures via Randomized Hashing", http://www.ee.technion.ac.il/~hugo/rhash/rhash.pdf, January, 2007.

[43] Dan Boneh and Weidong Shao, "Randomized Hashing for Digital Certificates: Halevi-Krawczyk Hash, An implementation in Firefox", http://crypto.stanford.edu/firefox-rhash/, January, 2007.

마지막으로, 같은 사용자에 의한 빠르면서도 연속적인 많은 요청과 같은 일련의 비정상적인 요청에 대해 그들이 받는 Certificate Signing Request들의 흐름을 모니터할 수 있다.

Microsft(Windows와 Internet Explorer의 벤더)와 Mozilla(Firefox의 벤더) 같은 **브라우저 및 운영체제 벤더들**은 MD5 기반의 인증서를 만날 때 사용자들에게 팝업 경고를 구현할 수 있다. MD5 기반의 인증서를 막는 것 역시 가능하지만 다소 극단적이다. 브라우저 벤더들은 경로 길이 점검을 구현할 수 있다. 더욱이 브라우저나 운영체제 내의 신뢰 목록에 어떤 CA들이 존재하는지를 결정하는 것은 브라우저 벤더들이다. 이 때문에 적절한 절차를 채택하고 강력한 암호 원형을 사용하도록 CA들에게 벤더들이 압력을 가하기가 용이하다. 우리는 앞에서 언급한 브라우저 벤더들과 접촉했으며, 그래서 그들은 이 문제에 대해 알고 있다.

웹 사이트 소유자들은 그들의 CA가 적절한 절차를 가지고 있는지, MD5와 같은 받아들일 수 없는 해쉬 함수를 명시적으로 사용하지 않는지 그 여부를 확인할 수 있다. 웹 사이트 소유자들은 그들의 CA들에게 SHA-2와 같은 더 안전한 해쉬 함수들로 교체하기를 요구할 수 있다.

7.1. 공개 후 반응들

[2008년 12월 31일 추가] **Verisign**(RapidSSL 브랜드 소유)는 우리의 작업이 공개되었을 때 즉시 반응했다. Tim Callan의 "This morning's MD5 attack - resolved"[44]를 참고해라. 다음은 Tim Callan의 블로그로부터 몇 가지 흥미로운 부분을 인용한 것이다.

- "우리는 이런 종류의 보안 연구에 대해 박수를 보내며, "MD5 Collision Inc." 그룹과 같은 화이트 해커들이 온라인 보안을 위한 연구를 주장하는 것에 대해 기쁘게 생각한다."
- "우리는 우리가 RapidSSL 인증서들을 발행할 때 MD5를 이용하는 것을 중지했으며, 우리가 판매하는 모든 다른 SSL 인증서들은 이 공격에 취약하지 않다는 것을 확인했다. 우리는 2009년 1월 말까지 모든 최종 실재 인증서에 MD5 사용을 중지하기 위한 방침을 지속할 것이다. "
- "... 그렇게 하고 싶어하는 고객은 무료로 MD5로 해쉬된 인증서를 대체할 수 있다. "

우리는 이 신속하고도 적절한 대응에 대해 기쁘게 생각한다.

[2008년 12월 31일 추가] Verisign의 언론 발표[45]와 RapidSSL 권고문[46] 참고

---

[44] https://blogs.verisign.com/ssl-blog/2008/12/on_md5_vulnerabilities_and_mit.php

[45] https://press.verisign.com/easyir/customrel.do?easyirid=AFC0FF0DB5C560D3&version=live&prid=463259&releasejsp=custom_97

[46] https://knowledge.rapidssl.com/support/ssl-certificate-support/index?page=content&id=AD125

[2008년 12월 31일 추가] **Microsoft**는 보안 권고문(961509) "Research proves feasibility of collision attacks against MD5"[47]를 발표했으며, Damian Hasse가 Microsoft Technet 블로그에 MD5 충돌 문제에 대한 정보[48]를 발표했다.

[2008년 12월 31일 추가] **Mozilla**는 "Mozilla Security Blog"에 Johnathan Nightingale이 "MD5 Weaknesses Could Lead to Certificate Forgery"[49]라는 짧은 글을 올렸다.

[2009년 1월 2일 추가] **TC TrustCenter**는 영문[50]과 독일어[51]로 반응을 보였다. 그들은 자동화된 온라인 인증서 요청 과정의 사용을 중지했고, 랜덤한 시리얼 번호들을 사용하기 때문에 우리의 공격 방법은 그들에게 적용되지 않는다고 하였다.

[2009년 1월 2일 추가] **RSA**는 "Speaking of Security blog"에 Sam Curry가 "A Real New Year's Hash"[52]란 글을 올렸다.

[2009년 1월 2일 추가] **US-CERT**의 US Department of Homeland Security's Computer Emergency Readiness Team은 Vulnerability Note VU#836068: "MD5 vulnerable to collision attacks"[53]를 발표했다. 여기서 몇 가지 흥미로운 것을 인용하면 다음과 같다.

- **MD5 알고리즘을 사용하지 마라**
  "소프트웨어 개발자들이나 CA들, 웹 사이트 소유자들, 그리고 사용자들은 가능한 MD5 알고리즘 사용을 피해야 한다. 앞의 연구가 입증한 것처럼 MD5는 암호학적으로 깨진 것이며, 앞으로 사용하기에는 적절하지 않은 것으로 간주되어야 한다."

- **MD5 알고리즘을 이용한 인증서에 의해 서명된 SSL 인증서들을 철저히 검사해라.**
  "사용자들이 웹 사이트 인증서의 속성들을 직접 분석하기를 원할 수 있다 (...) md5RSA 또는 비슷한 것으로 목록에 오른 인증서들이 영향을 받는다. 이상하고 의심스러운 필드들 또는 다른 예외적인 것들을 포함하고 있는 그런 인증서들은 가짜일 수 있다. 부정 수단을 쓰는 것에 대한 신뢰할만한 조짐이 없기 때문에 이 해결책은 오류 경향을 가지고 있으며, 대부분의 사용자들에게 실용적이지 않다는 것을 주목해야 한다."

[2009년 1월 15일 추가] **Cisco**는 그들의 제품들 중 몇 가지가 역시 취약할지도 모른다고 생각하기 때문에 "Cisco Security Response: MD5 Hashes May Allow for Certificate Spoofing "[54]를 발표했다.

---

47 http://www.microsoft.com/technet/security/advisory/961509.mspx

48 http://blogs.technet.com/swi/archive/2008/12/30/information-regarding-md5-collisions-problem.aspx

49 http://blog.mozilla.com/security/2008/12/30/md5-weaknesses-could-lead-to-certificate-forgery/

50 http://www.trustcenter.de/media/TC_response_to_MD5_vulnerability_paper.pdf

51 http://www.trustcenter.de/media/Stellungnahme_der_TC_TrustCenter_GmbH_zum_Artikel_von-5.pdf

52 http://www.rsa.com/blog/blog_entry.aspx?id=1411

53 http://www.kb.cert.org/vuls/id/836068

8. FAQ(자주 하는 질문)

Q: 우리의 proof of concept용 가짜 CA 인증서는 악용될 수 있는가?
A: 그럴 가능성은 아주 적다. 우리는 우리의 가짜 CA 인증서의 폐기 날짜를 일부러 2004년 8월로 설정해두었다. 우리의 가짜 CA에 의해 서명된 어떤 인증서도 2004년 8월 이후에는 유효하지 않을 것이다. 2004년 8월로 시스템 날짜를 설정함으로써 우리의 proof of concept 인증서를 테스트할 수 있게 해주지만 나쁜 사람들의 손에 들어가게 되더라도 악용될 수 있는 것으로부터 막아준다. 만약 여러분의 시스템 날짜가 현재로 정확하게 설정되어 있다면 여러분의 웹 브라우저는 그 웹 사이트를 신뢰해서는 안된다고 경고하며 그 인증서의 날짜가 이미 폐기되었기 때문에 유효하지 않다고 알려줄 것이다. 그 가짜 인증서의 유효기간이 너무 짧아 컴퓨터의 시스템 시계가 우연히 과거로 거슬러 설정되어 있을 것 같지 않게 한다.

Q: 모든 디지털 인증서와 서명이 깨진 것인가?
A: 아니다. 디지털 인증서와 서명은 안전한 암호 해쉬 함수들에 기초를 두고 있을 때는 우리의 작업 때문에 디지털 인증서와 서명의 보안에 대해 의문을 가질 아무런 이유도 없다. 우리의 연구 결과는 디지털 인증서들이 깨진 것으로 알려진 해쉬 함수 MD5를 이용해 서명될 때에만 적용된다. 우리의 방법을 이용하면 두 개의 특별히 제작된 디지털 인증서들 사이에서 MD5에 기반한 디지털 서명을 복사하는 것만이 가능하다. 만약 그 인증서들이 특별히 제작되지 않다면 디지털 인증서들로부터 MD5에 기초를 둔 디지털 서명들을 복사하는 것은 가능하지 않다. 비록 그렇다 할지라도, 우리의 연구 결과는 MD5가 디지털 서명에 적합하지 **않다**는 것을 보여준다. CA들은 깨진 MD5 사용을 멈추고, SHA-2와 같이 광범위하게 이용할 수 있고 더 안전한 대안으로 옮겨가야 한다.

Q: SHA-1에 대한 충돌 공격의 경우 상황은 어떤가?
A: 우리가 알고 있는 한 SHA-1 충돌 공격에 대해 진지하게 작업 중인 그룹은 오스트리아의 Graz에 있는데, 그들의 웹 사이트에서 "SHA-1 Collision Search"[55]를 참고해라.

<이론의 상태>: Crypto 2007의 Rump Session에서 그들은 동일한 초기 IHV들을 가진 충돌에 대한 공격의 복잡도가 압축 함수에 대해 $<2^{61}$번의 호출이 될 것으로 평가했다. chosen-prefix 충돌들에 대해 그들은 birthday 바로 아래의 복잡도가 2^{80} 정도 될 것으로 2006년에 평가했다. 후자 결과에 대한 개선들이 아마도 가능하겠지만 누구도 이것에 대해 조사해보지 않았다.
<실험의 상태>: 2007년 중반 이후 Graz 사람들은 동일한 초기 IHV들을 가진 SHA-1 충돌을 찾아내는데 목적을 둔 광범위한 "SHA-1 Collision Search BOINC project"[56]를 운영하고 있다. 그들은 지금(2008년 12월) 연구 범위의 10% 이하 정도만을 다루고 있다고 평가하고 있다. 그래서 그들은 여러분들의 도움이 필요한 상태다.

---

[54] http://www.cisco.com/en/US/products/products_security_response09186a0080a5d24a.html

[55] http://www.iaik.tugraz.at/content/research/krypto/sha1/

[56] http://boinc.iaik.tugraz.at/

Q: 범죄자가 우리의 방법을 이용해 모든 브라우저가 신뢰하는 그 자신의 가짜 CA를 만들었다고 가정해보자. 그가 사용한 서명을 가진 CA로부터 발행된 인증서를 가진 웹 사이트들만이 사칭(impersonation) 공격에 취약한가? MD5에 기초를 둔 인증서를 가진 웹 사이트들이 사칭 공격에 취약한가?
A: 아니다. 범죄자가 가짜지만 신뢰받은 CA 인증서를 혼합해 리다이렉션 공격을 사용할 때 모든 웹 사이트들은 동등하게 취약하다.

Q: MD5로 서명된 디지털 인증서를 가진 웹 사이트들은 무엇을 해야 하는가?
A: 현재로서는 할 것이 없다. 모든 CA들로부터 합법적으로 획득한 디지털 인증서들은 MD5로 서명되었다 할지라도 안전하고 신뢰받을 수 있는 것으로 믿어질 수 있다. 우리의 방법은 특별히 조작된 디지털 인증서를 CA로부터 구입하는 것이 필요했으며, 다른 어떤 일반 웹 사이트에 발행된 인증서들에는 영향을 주지 않는다.

Q: 왜 이 글의 제목을 "MD5 considered harmful today"라고 했는가?
A: 2004년 12월 MD5 충돌이 발견된 직후 Dan Kaminsky는 그것에 대한 exploit을 만들었다. 그의 글 제목이 "MD5 to be considered harmful someday"[57]이었다. 우리는 그 날이 바로 오늘이라고 생각한다.

Q: chosen-prefix 충돌 설계 방법에 대한 세부내용을 가진 글은 언제 어디서 공개될 것인가?
A: 이 논문은 아직 끝나지 않았다. 그 글은 학술 저널에 공개될 것이다. 웹에서는 2009년 상반기쯤에 공개될 수도 있다. 그렇게 되면 공개 장소에 대해 여기서 공지될 것이다.

Q: chosen-prefix 충돌 설계 방법의 소스와 실행 코드를 공개할 것인가?
A: 당분간 그런 계획은 없다.

Q: MD5를 여전히 사용하고 있는 CA들이 이미 누군가에 의해 공격을 당하지 않았다는 것을 어떻게 알 수 있나?
A: 우리의 proof of concept을 만드는데 필요한 시간과 자원, 그리고 우리의 방법을 구현하는데 요구되는 발전된 수학이 있다고 할지라도 공격이 이미 발생했을 것 같지는 않다는 것이 우리의 의견이다. 하지만 우리의 proof of concept은 그것이 가능하다는 것을 보여주고 있고, 공격을 당하지 않았다는 것에 대한 100% 보장은 없다.

Q: 언론 발표에서 언급한 DNS의 알려진 취약점은 무엇인가?
A: 2008년 Dan Kaminsky에 의해 발견된 DNS spoofing 취약점은 어떤 웹 사이트에 대한 요청을 공격자가 통제하는 웹 사이트로 리다이렉트시키는 것을 허용한다. 더 자세한 것은 Kaminsky의

---

[57] Dan Kaminsky, "MD5 to be considered harmful someday" http://www.doxpara.com/md5_someday.pdf, December 2004.

"Black Ops 2008: It's the end of the cache as we know it"[58]와 Matthew Olney 등이 작성한 "Dan Kaminsky's 2008 DNS Vulnerability" [59] 를 참고해라. 안전한 웹 사이트를 위해 사용되는 SSL 프로토콜은 공격자가 하이재킹된 웹 사이트에 대한 합법적인 디지털 인증서를 가지고 있지 않기 때문에 이런 종류의 공격을 탐지하는 것으로 생각된다. 하지만, 우리가 발견한 MD5 충돌 공격은 공격자가 신뢰된 임의의 디지털 인증서를 만드는 것을 허용하고, 만약 사용자의 DNS 서버가 DNS 스푸핑 취약점에 대해 패치가 되어 있지 않다면 공격자가 인터넷 상의 어떤 안전한 웹 사이트도 하이재킹하는 것을 허용한다.

Q: 우리의 proof of concept 연구에는 얼마나 많은 비용이 들었는가?
A: 우리가 지난 2년 이상 개발한 많은 지식과 기술에 기초를 두고 우리의 공격 방법을 기획하고 구현하는데 몇 달이 걸렸다. 우리는 CA로부터 테스트용 인증서들을 구입하는데 700 달러를 지불했다. 우리의 작업을 위해 필요한 계산은 EPFL에 있는 암호분석 연구실에서 약 200개의 PlayStation 3의 게임 콘솔을 연결한 것에서 실행되었다.

Q: 왜 게임 콘솔이 사용되었는가? 다른 적절한 하드웨어는 무엇이 있는가?
A: 게임 콘솔은 게임에서 세밀한 3D 그래픽의 연산 요구를 위해 전문화된 하드웨어를 사용한다. 이 하드웨어는 암호 알고리즘에 사용되는 기본 계산에도 역시 절절하며, brute-force 계산에서는 일반적인 목적의 컴퓨터들을 능가한다. 하나의 PlayStation 3 게임 콘솔은 약 40개의 현대 단일 코어 프로세서들과 맞먹는다. 우리가 사용한 방법에서 계산적으로 가장 집중적인 부분은 200개 이상의 게임 콘솔로 약 3일간의 작업을 요구했는데, 이는 일반적인 데스크탑 컴퓨터에서 32년간의 계산에 맞먹는 것이다. 일반적인 그래픽 카드들이 MD5 암호 분석을 위해 몇몇 소프트웨어[60]에 의해 역시 사용되었다.

Q: 우리의 작업을 누군가 모방하는데 얼마나 걸릴 것인가?
A: 우리 작업 이면의 기본적인 이론은 2007년에 공개되었지만 우리는 그 문서에는 아직 없었던 많은 것을 개선했다. 우리는 이 분야에 사전 지식을 가진 기술적으로 아주 뛰어난 연구자들과 프로그래머들로 구성된 한 팀이 우리의 작업을 되풀이하는데 적어도 한 달은 걸릴 것이라고 추정하고 있다. MD5 충돌 기술에 대한 사전 이해 없이 막무가내로 시작하는 것은 더 많은 도전을 요구할 것이고 훨씬 더 오랜 시간이 걸릴 수도 있다. 암호 연구와 개발에 필요한 전문지식에 덧붙여 중요한 계산 자원을 보유하는 것도 필요하다. 우리의 작업을 재생산하는데 걸리는 시간과는 상관없이 MD5는 인증서를 위해서는 더 이상 사용되어서는 안된다.

Q: 우리가 개발한 공격 시나리오가 미래에는 가능하지 않도록 하는 가장 좋은 방법은 무엇인가?
A: 가능한 빨리 MD5 사용을 멈추고, 더 안전한 암호 해쉬 함수로 바꾸어야 한다.

---

58 http://www.doxpara.com/DMK_BO2K8.ppt

59 http://www.snort.org/vrt/docs/white_papers/DNS_Vulnerability.pdf

60 http://www.elcomsoft.com/edpr.html

필자들 소개

(왼쪽에서 오른쪽방향으로: Benne, Arjen, Marc, Jake, David, Alex (Dag Arne는 빠짐))

Alexander Sotirov

Independent security researcher,

New York, USA

http://www.phreedom.org/

Marc Stevens

Centrum Wiskunde & Informatica (CWI),

Amsterdam, The Netherlands

http://homepages.cwi.nl/~stevens/

News at the CWI website: English, Nederlands.

Jacob Appelbaum

Noisebridge, The Tor Project,

San Francisco, USA

http://www.appelbaum.net/

Arjen Lenstra

LACAL - LAboratory for Cryptologic ALgorithms,

EPFL - École Polytechnique Fédérale de Lausanne,

Lausanne, Switzerland

http://people.epfl.ch/arjen.lenstra

David Molnar

Computer Science Division,

University of California at Berkeley,

Berkeley, USA

http://www.cs.berkeley.edu/~dmolnar/

Dag Arne Osvik

LACAL - LAboratory for Cryptologic ALgorithms,

EPFL - École Polytechnique Fédérale de Lausanne

Lausanne, Switzerland

http://people.epfl.ch/dagarne.osvik

Benne de Weger

EiPSI - Eindhoven Institute for the Protection of Systems and Information,

TU/e - Eindhoven University of Technology,

Eindhoven, The Netherlands

http://www.win.tue.nl/~bdeweger/

관련 정보

언론과 일반적인 질문을 위해 md5-collisions@phreedom.org로 메일을 보내라.

이 작업을 설명하는 Marc의 비디오[61](독일어로 되어 있음)가 있다.

이 프로젝트에 대해 더 많은 정보를 원한다면 다음 프로젝트 웹 사이트들을 참고해라.

- http://www.win.tue.nl/hashclash/rogue-ca/ (이 문서가 있는 현재 사이트)
- http://www.phreedom.org/research/rogue-ca/ (Alex의 사이트)
- https://i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org/

(데모 사이트, 컴퓨터의 시스템 시계를 2004년 8월로 설정했을 때만 작동함)

[2009년 1월 8일 추가] 25C3에서의 발표 비디오와 오디오 파일들[62]은 CCC에서 다른 포맷으로 구할 수 있다. "25c3-3023-en-making_the_theoretical_possible.xyz"라는 이름을 가진 파일을 찾아보아라. 흥미롭게도 파일 다운로드의 무결성을 확인하는 수단으로 MD5 해쉬를 제공하고 있다.

---

[61] http://www.youtube.com/watch?v=Sx5-e5w9ais

[62] http://events.ccc.de/congress/2008/wiki/Conference_Recordings

링크

- IAIK **SHA-1 Collision Search BOINC project** - http://boinc.iaik.tugraz.at/
  및 그들의 "SHA-1 Collision Search" 웹 사이트 - http://www.iaik.tugraz.at/content/research/krypto/sha1/

- "**The story of Alice and her Boss**", Magnus Daum and Stephan Lucks, 2005
  - http://th.informatik.uni-mannheim.de/People/lucks/HashCollisions/

- Peter Gutmann의 **dumpasn1** 프로그램 - http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c
  Windows GUI 버전 - http://geminisecurity.com/features-downloads/tools/guidumpasn
  온라인 버전 - http://geminisecurity.com/parse.php
  Peter Gutmann의 **X.509 Style Guide** - http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt
  내부에 MPEG 동영상을 가진 인증서 - http://www.cs.auckland.ac.nz/~pgut001/pubs/dave.der

- "**Colliding X.509 Certificates based on MD5-collisions**" 웹 사이트, Arjen Lenstra, Benne de Weger, Xiaoyun Wang, March 2005.
  - http://www.win.tue.nl/~bdeweger/CollidingCertificates/

- "**Colliding X.509 Certificates for Different Identities**" 웹 사이트, Marc Stevens, Arjen Lenstra, Benne de Weger, October 2006.
  - http://www.win.tue.nl/hashclash/TargetCollidingCertificates/

- "**Chosen-prefix collisions**" 웹 사이트, Marc Stevens, Arjen Lenstra, Benne de Weger, February 2006.
  - http://www.win.tue.nl/hashclash/ChosenPrefixCollisions/

- "**Predicting the winner of the 2008 US Presidential Elections using a Sony PlayStation 3**", 웹 사이트, Marc Stevens, Arjen Lenstra and Benne de Weger, November 2007.
  - http://www.win.tue.nl/hashclash/Nostradamus/

- "**Vulnerability of software integrity and code signing applications to chosen-prefix collisions for MD5**", 웹 사이트, Marc Stevens, Arjen Lenstra and Benne de Weger, November 2007.
  - http://www.win.tue.nl/hashclash/SoftIntCodeSign/

- Verisign 의 응답: "**This morning's MD5 attack - resolved**"(by Tim Callan)
  - https://blogs.verisign.com/ssl-blog/2008/12/on_md5_vulnerabilities_and_mit.php
  Verisign 언론 발표
  - https://press.verisign.com/easyir/customrel.do?easyirid=AFC0FF0DB5C560D3&version=live&prid=463259&releasejsp=custom_97
  RapidSSL 보안권고문

- https://knowledge.rapidssl.com/support/ssl-certificate-support/index?page=content&id=AD125

- Microsoft 보안 권고문(961509): "**Research proves feasibility of collision attacks against MD5**"
  - http://www.microsoft.com/technet/security/advisory/961509.mspx

  Microsoft Technet blog item, **Information regarding MD5 collisions problem**(by Damian Hasse)
  - http://blogs.technet.com/swi/archive/2008/12/30/information-regarding-md5-collisions-problem.aspx

- Mozilla Security Blog(by Johnathan Nightingale)
  - http://blog.mozilla.com/security/2008/12/30/md5-weaknesses-could-lead-to-certificate-forgery/

- TC TrustCenter의 반응
  - http://www.trustcenter.de/media/TC_response_to_MD5_vulnerability_paper.pdf
  - http://www.trustcenter.de/media/Stellungnahme_der_TC_TrustCenter_GmbH_zum_Artikel_von-5.pdf

- RSA 블로그 항목: "**A Real New Year's Hash**"(by Sam Curry)
  - http://www.rsa.com/blog/blog_entry.aspx?id=1411

- US-CERT Vulnerability Note VU#836068: "**MD5 vulnerable to collision attacks**"
  - http://www.kb.cert.org/vuls/id/836068

- 25C3 프리젠테이션 파일과 동영상
  - http://events.ccc.de/congress/2008/wiki/Conference_Recordings

- Cisco Security Response: "**MD5 Hashes May Allow for Certificate Spoofing**"
  - http://www.cisco.com/en/US/products/products_security_response09186a0080a5d24a.html

[최근 수정일: 2009년 1월 26일]