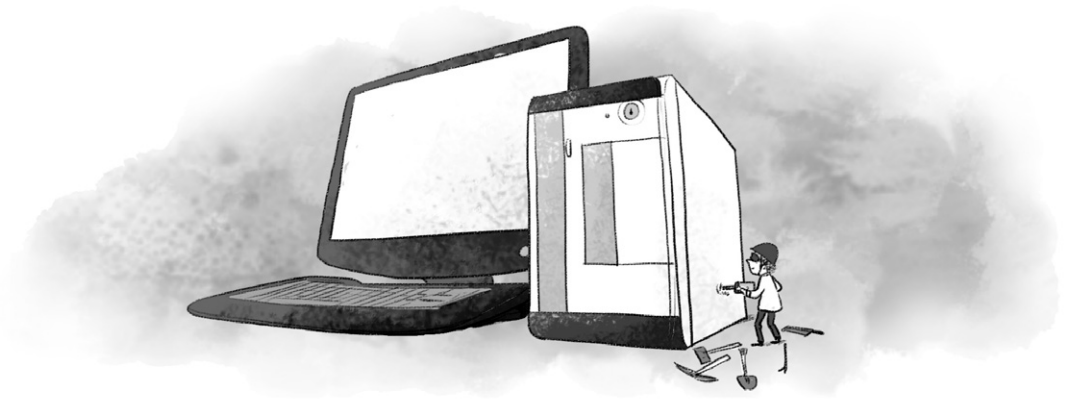
Part

02

시스템 해킹



시스템 해킹을 간단히 말하면 시스템의 권한을 장악하기 위한 모든 시도라 할 수 있다. 이런 시도 중에는 패스워드 크래킹이나 백도어 같은 비교적 단순한 것도 있지만, 버퍼 오버플로우나 포맷 스트링, 리버스 엔지니어링처럼 아주 복잡한 것도 있다. 이 파트는 시스템에서 일어나는 다양한 종류의 해킹을 이해하기 위해 알고 넘어가야 할 중요한 부분이다.

Chapter 04

패스워드 크래킹

* 학습목표

- 패스워드의 중요성을 이해한다.
- 윈도우와 리눅스/유닉스 시스템의 인증 구조를 이해한다.
- 패스워드 크래킹을 수행할 수 있다.
- 적절한 패스워드를 생성할 수 있다.
- 관리자 패스워드 분실 시 이를 복구할 수 있다.

Preview

인증(Authentication)은 '어떤 이가 그것을 할 만한 자격이 있는가?'를 확인하는 과정이다. 인증 수단은 크게 다음 네 가지로 나눌 수 있다.

첫째는 알고 있는 어떤 것(Something you know)으로 이는 지식 기반의 인증을 말한다. 대표적인 예로 패스워드를 들 수 있다. 둘째는 가지고 있는 어떤 것(Something you have)이다. 대표적인 예는 열쇠로, 집 열쇠는 소유하고 있는 것만으로도 해당 집에 들어갈 수 있는 자격을 증명한다. 셋째는 우리 여러분 자신(Something you are)으로 지문이나 망막과 같은 생체 조직을 이용하여 인증하는 것을 말한다. 마지막으로 우리가 있는 곳(Somewhere you are)도 최근 들어 인증 수단으로 사용되고 있다.

이렇게 다양한 인증 수단 중 가장 흔한 것은 패스워드로 대표되는 '알고 있는 어떤 것(Something you know)'이다. 패스워드는 우리 사회에서 매우 흔하게 사용되어 인터넷을 사용할 때, 금고를 열 때, 은행에서 돈을 찾을 때, 군대에서 보초를 교대할 때도 사용된다.

컴퓨터에서도 마찬가지다. 다양한 수단이 사용되고 있지만, 아직도 패스워드가 가장 일반적인 인증 수단이다. 그렇기에 패스워드는 해커에게 가장 구미가 당기는 침투 수단이며, 패스워드를 찾아내는 기술 하나만 가져도 뛰어난 해커가 될 수 있다. 이 장에서는 이런 패스워드 크래킹이 어떻게 가능한지 알아볼 것이다.

Section

01 패스워드 크래킹에 대한 이해

1 패스워드 관리

해커에게 패스워드 획득만큼 쉬운 침투 방법은 없다. 따라서 보안 관리자의 첫 번째 방어책은 패스워드를 통한 것이다. 물론 최근에는 네트워크 분리와 적절한 접근 제어로 인해 패스워드를 획득하더라도 침투하기 어려운 경우가 많아졌다. 그럼에도 불구하고 대부분의 사이트에서는 여전히 간단한 패스워드 크래킹만으로 운영 중인 시스템의 관리자 권한을 빼앗을 수 있다. 이와 같은 상황으로 볼 때 패스워드는 보안에서 가장 중요한 항목임이 분명하다.

서버에서 운영되는 시스템은 적으면 50여 대, 많게는 200여 대를 훌쩍 넘는다. 윈도우를 이용한 소형 서버의 클러스터링 등을 사용한 사이트의 경우 운영되는 시스템이 1500여 대를 넘기도 한다. 이렇게 1500여 대의 시스템이 운영되고 있다면 일부 시스템은 필요에 따라 계정을 생성하기도 한다. 하지만 대부분의 관리자는 생성했던 계정이 더 이상 필요 없어지더라도 예상치 못한 오류를 우려하여 계정을 삭제하지 않는 경우가 많다. 또한 임시로 생성한 계정에 기억하기 쉽도록 아주 간단한 패스워드를 입력해두거나 계정과 똑같은 패스워드를 설정한다. 이렇게 관리되는 시스템에는 시간이 지날수록 보안에 취약한 패스워드를 가진 계정이 많아지고 이에 따라 전체 서비스의 보안 위험도 커진다. 심지어 운영하는 시스템이 50여 대 이상인 사이트에 보안 책임자와 관리자가 없거나 부족한 경우도 많다.

패스워드 설정 문제는 서버 관리자만의 문제는 아니다. 사용자가 몇 개인지 기억하지 못할 만큼 수많은 인터넷 사이트에 가입하였고, 모든 사이트에 같은 패스워드를 사용한다면 패스워드 크래킹 공격에 취약하다 볼 수 있다. 패스워드를 자신의 이름이나 주민등록번호, 전화번호처럼 매우 간단한 것으로 설정해놓았다면, 인터넷에서 개인 정보가 누출되더라도 스스로 책임져야 할지 모른다. 패스워드 생성기를 사용한다면 크래킹이 어려운 패스워드를 생성할 수 있는 대신 외우기는 어렵다.

패스워드를 생성할 때 흔히 권고하는 기준이 있다. 권고 기준을 알아보기 전에 크래킹되기 쉬운 패스워드부터 살펴보자.

- **길이가 너무 짧거나 널(Null)인 패스워드 :** 패스워드 중 가장 짧으면서도 빈번하게 나타나는 것은 1과 a다. 이런 패스워드가 과연 효과가 있을까?
- **사전에 나오는 단어나 이들의 조합으로 이루어진 패스워드 :** 우리나라에서는 사전에 나오는 단어의 조합을 많이 사용하지 않는다. 우선 우리나라가 영어권이 아니기 때문에 패스워드를 굳이 영어 단어로 쓰지 않기 때문이다. 대신에 영문 자판에서 한글 단어를 써넣는 경우는 무척 많다. 예를 들면, ehtk(도사), godqhr(행복)과 같은 경우다.
- **키보드 자판을 일련순으로 나열한 패스워드 :** 흔히 쓰는 또 다른 패스워드는 asdf, qwer, 1234 등과 같이 키보드에 나열된 순서를 그대로 이용하는 경우다. 이런 경우 대부분 사전 파일에 등록되어 있으며, 매우 짧은 시간 동안 패스워드가 크래킹되고 시스템이 장악된다.
- **사용자 계정 정보에서 유추 가능한 단어들로 된 패스워드 :** wishfree라는 계정에 wishfree76로 패스워드를 등록한 경우다. 이렇게 사용자 계정 정보에서 유추 가능한 단어로 된 패스워드 역시 해커가 충분히 추측 가능하고 크래킹되기 쉽다.

그렇다면 좋은 패스워드란 어떤 것일까? 기억하기는 쉽지만 크래킹하기 어려운 패스워드다. 한국어를 쓰는 우리는 패스워드 설정에 있어 유리하다. 영문 자판으로 놓고 우리말을 모두 타자로 쳐서 워드 목록을 만들어 사용한다면 얘기가 조금 달라지겠으나, 'eodlf(대일)@!11'과 같은 패스워드는 거의 크래킹될 수 없다 봐도 좋다. 이름과 숫자가 간단하게 들어갔으나, 특수문자를 입력함으로써 그 조합의 수가 몇십 배로 늘어났기 때문이다. 따라서 이름 말고 자신이 좋아하는 단어와 간단한 숫자, 특수문자 한두 개를 조합하면 매우 훌륭한 패스워드를 만들 수 있다. 그리고 패스워드를 충분히 복잡하게 만들었으면, 쉽게 노출되지 않도록 관리를 잘 해야 한다.

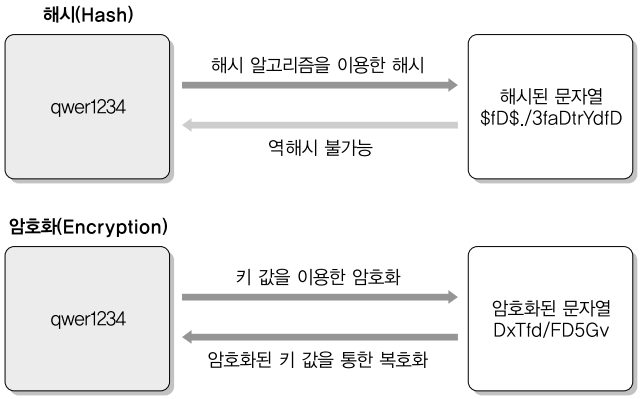
2. 해시와 암호화

운영체제는 어떤 형식으로든 패스워드의 내용을 숨겨야 한다. 패스워드를 숨기는 방법에는 기본적으로 2가지가 있다. 바로 해시와 암호화다.

해시(Hash)는 임의의 데이터로부터 일종의 짧은 '전자 지문'을 만들어 내는 방법이다. 해시 함수는 데이터를 자르고 치환하거나 위치를 바꾸는 등의 방법을 사용해 결과를 만들어 내며, 이 결과를 흔히 해시 값(hash value)이라 한다. 만약 해시 값이 다르면 그 해시 값에 대한 원래 데이터도 달라야 한다. 또한 해시 값에서 원래의 데이터를 구하는 것이 불가능해야 한다.

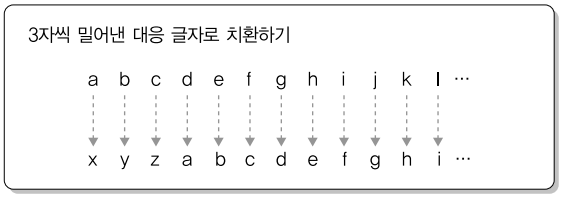
반면 암호화(Encryption)는 특별한 지식을 소유한 사람을 제외하고는 누구든지 읽어볼 수 없도록 알고리즘을 이용해 데이터를 전달하는 것이다. 이러한 과정을 통해 암호화된 정보를 생성할 수 있다. 그리고 암호화된 정보는 복호화(Decryption)하여 다시 읽을 수 있다.

해시와 암호화는 모두 패스워드를 효과적으로 숨기는 기술이나, 암호화 알고리즘이 암호화되지 않은 평문을 암호화한 뒤에 이를 다시 복호화할 수 있는 반면, 해시 알고리즘 결과물은 복호화가 불가능하다는 차이점이 있다. 이 차이를 이해하는 것은 패스워드 해킹과 관련하여 매우 중요하므로 [그림 4-1]을 통해 자세히 살펴보자.



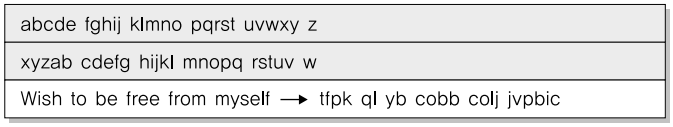
[그림 4-1] 해시와 암호화 알고리즘의 차이

해시를 처음 접한다면 바로 이해되지는 않을 것이다. 이해를 돕기 위해 간단한 해시와 암호화 예를 살펴보자. 먼저 로마시대에 사용된 암호화 방식을 살펴보자. 기본적인 치환(Substitution) 방식을 사용하는데, 다음과 같이 3자씩 알파벳을 밀어내 대응되는 글자로 치환한다.



[그림 4-2] 밀어내기식 치환 방법

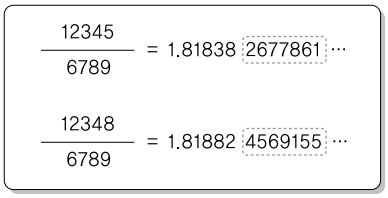
위 규칙에 따라 'Wish to be free from myself' 문구를 암호화해보자.



[그림 4-3] 밀어내기식 치환 방법 예

이 암호화로는 어떤 글을 암호화했는지 쉽게 알아볼 수 없다. 하지만 알파벳을 3자씩 밀어서 대응했다는 것을 알면 복호화할 수 있다. 여기에서 '알파벳을 밀어서 대응되는 글자로 치환' 하는 것은 암호화 알고리즘이 되고, 3이라는 숫자는 암호화 키(Key)가 된다.

이번에는 해시 예를 살펴보자. 먼저 123456789라는 수와 이것과 한 자리 수만 다른 123486789라는 수가 있다 가정해보자. 두 수를 다음과 같이 가운데를 기준으로 둘로 나누고 큰 수를 작은 수로 나눈다.



[그림 4-4] 나눗셈을 이용한 해시 값 획득

결과 값은 각각 1.818...이다. 앞 6자리 숫자를 버리고 나머지 값을 해시의 결과 값이라고 생각한다면 123456789의 해시 값은 2677861, 123486789의 해시 값은 4569155다. 하지만 두 해시 값만으로 해시 전의 원래 수를 알아내는 것은 불가능에 가깝다. 로직을 알더라도 버려진 1.81838과 1.81882를 알아낼 수 없기 때문이다.

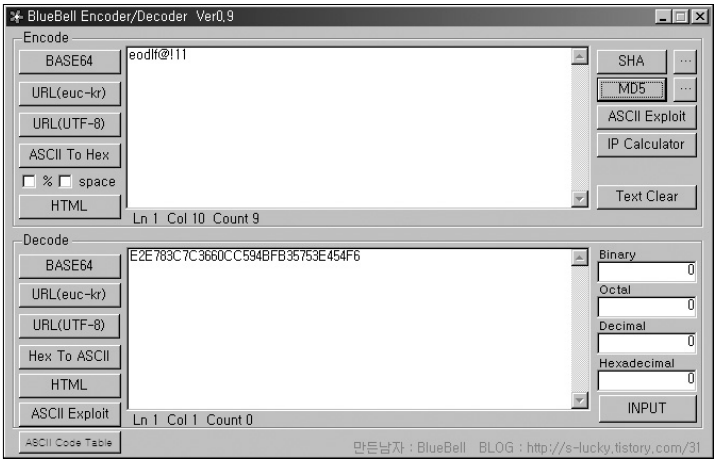
이처럼 해시는 로직을 알고 있을 경우 해시의 결과 값을 구하기가 쉽지만, 그 해시 결과 값을 통해 해시를 생성하기 전의 원래 값은 알기 어렵다. 그리고 값이 아주 조금만 다르더라도 해시 결과 값은 무척 상이하게 생성된다.

이제 암호화 알고리즘과 해시 알고리즘의 차이와 기본 원리를 이해했으리라 생각한다. 암호화와 해시 알고리즘은 패스워드 보안과 관련된 기본적인 사항이다. 이외에 패스워드 보안을 강화할 수 있는 기술을 알아보자.

❸ Salt

기본적으로 패스워드는 해시와 암호화 알고리즘으로 변경하여 저장한다. 예를 들어, root 사용자와 wishfree 사용자의 패스워드가 똑같이 'eodlf@!11' 라고 가정해보자. 해시나 암호화를 통해 패스워드를 저장하면 어떻게 저장될까? 해시를 사용하든 암호화를 사용하든 똑같은 패스워드는 똑같은 해시 값이나 똑같은 암호문으로 저장된다.

이렇듯 똑같은 해시 결과나 암호문은 똑같은 결과만으로도 패스워드를 노출시킨다는 약점이 있다. Salt는 이런 상황을 막기 위해 패스워드 해시와 암호화에 사용되는 첨가물의 일종으로, 운영체제별로 다양한 알고리즘이 존재한다. 간단한 Salt 예를 살펴보자. 필자는 MD5 해시 값을 생성하는 간단한 툴을 이용해 'eodlf@!11' 을 MD5 알고리즘으로 해시한 패스워드 문을 만들어 보았다(어떤 블로거가 간단한 실습용으로 만들어 공개한 것으로 해시 값을 생성하여 보여주는 편리한 툴이다).



[그림 4-5] eodlf@!11의 MD5 해시 값 생성

[그림 4-5]와 같이 'E2E783C7C3660CC594BFB35753E454F6'라는 해시 값이 나온다. 어떤 운영체제에서는 2개의 문자를 랜덤으로 생성하여 패스워드 앞에 붙이고 그것을 다시 해시하거나 암호화하기도 한다. 예로 [그림 4-5]의 툴을 이용하여 root 계정은 a2, wishfree 계정은 4F라는 문자 2개를 생성하여 해시 값을 생성해보자. 여기서 a2와 4F가 Salt다.

[표 4-1] Salt와 패스워드를 조합한 값에 대한 MD5 해시 값의 생성 예

계정	Salt	패스워드	Salt+패스워드를 MD5로 해시한 결과 값
root	a2	eodlf@!11	9EF83D5BEF4A7CBC6F7D4940D8447089
wishfree	4F	eodlf@!11	6B796B0DD16C30CCF0B7F02E6457F024

Salt와 패스워드를 합한 'a2eodlf@!11'과 '4Feodlf@!11'을 각각 해시한 결과 값은 전혀 다르다는 것을 알 수 있다. 하지만 패스워드 파일로 저장할 때는 MD5 해시 값만 저장할 수는 없다. 시스템이 패스워드와 어떤 것을 합해 해시를 구한 것인지 알 수 없기 때문이다. 따라서 패스워드 파일에 저장할 때는 간단한 인코딩을 통해 해시 결과 값 앞이나 뒤에 Salt를 붙인다. 다음은 [표 4-1]에서 구한 해시의 앞에 각각 Salt를 붙인 예다.

[표 4-2] 해시된 패스워드 값에 Salt 정보를 붙인 예

계정	Salt+MD5
root	a29EF83D5BEF4A7CBC6F7D4940D8447089
wishfree	4F6B796B0DD16C30CCF0B7F02E6457F024

이와 같이 적용된 Salt는 똑같은 패스워드를 숨길 뿐만 아니라 적용 수준에 따라 패스워드 크래킹을 매우 어렵게 만드는 요소가 된다.

4. 패스워드 크래킹 방법에 대한 이해

은행털이가 나오는 영화를 보면 이들이 은행 금고에 침투할 때 기계를 사용해 패스워드를 푸는 장면을 볼 수 있다. 패스워드가 한 자리 한 자리 풀려나가면서 마침내 이들은 침투에 성공하게 된다. 하지만 사실 패스워드를 조금만 아는 사람이라면 이것이 단지 허구적 요소라는 것을 알 수 있다. 패스워드 길이는 패스워드 강도를 결정하는 매우 중요한 요소다. 따라서 사용자가 설정한 패스워드에 해시나 암호화 알고리즘을 한 자리씩 별도로 적용해 패스워드의 강도를 약화시킬 바보는 없을 것이다. 즉 사용자가 만든 8자리 패스워드가 한 자리 8개와 같아지게 된다. 한 자리 패스워드는 알파벳, 특수문자, 숫자를 비롯해 아무리 경우의 수가 많아도 200가지를 넘지 못한다. 즉, 약 2,000개 정도의 문자를 적용하면 8자리의 패스워드는 크래킹되는 것이다. 그렇게 생각하면 영화에서는 고작 2,000개 정도의 문자를 대입하는 데 너무 많은 시간이 걸린다는 것을 알 수 있다.

물론 과거에는 한 글자씩 풀리는 암호화의 예가 있었다. 윈도우 95의 공유 폴더 패스워드가 그런 경우로 이 패스워드는 약 0.1초에 크래킹되었다. 하지만 근래의 대부분 패스워드는 전체 자리의 문자를 한 번에 적용하지 못하면 풀리지 않는다. 그렇다면 패스워드 크래킹을 하기 위해 어떤 방법을 적용해야 할까? 기본적으로는 여러 가지 패스워드를 임의로 적용해보는 방법이 있지만 레인보우 테이블을 이용한 기발한 방법도 있다. 하나씩 살펴보자.

사전 대입 공격

사전 대입 공격(Dictionary Attack)은 사용자가 설정하는 대부분의 패스워드에 특정 패턴이 있음에 착안한 것이다. 특정 패턴은 사전에 있는 단어일 수도 있고, qwer1234와 같이 키보드 자판을 일렬순으로 나열한 것일 수도 있으며, 주민등록번호나 이름, 회사 이름, 생일 등일 수 있다. 즉, 사전 대입 공격은 패스워드로 사용할 만한 것을 사전으로 만들어놓고 이를 하나씩 대입하여 패스워드 일치 여부를 확인하는 것이다. 공격 대상의 개인 정보 등을 충분히 알고 있다면 매우 효율적인 공격 방법이다. 실제로 모의해킹을 하는 경우 상당수의 패스워드가 추측에 의한 사전 대입 공격에 의해 밝혀졌다.

무작위 대입 공격

무작위 대입 공격(Brute Force Attack)은 사전 대입 공격에서 실패했을 때 주로 사용된다. 패스워드에 사용될 수 있는 문자열의 범위를 정하고, 그 범위 내에서 생성 가능한 모든 패스워드를 생성하여 패스워드로 입력해보는 것이다. 이 공격은 패스워드가 그다지 복잡하지 않거나 짧을 경우 단시간에 크래킹된 패스워드를 알아낼 수 있다.

레인보우 테이블을 이용한 공격

레인보우 테이블(Rainbow Table)의 개념은 1980년 마틴 헬만(Martin Hellman)에 의해 소개되었다. 하지만 당시에는 거의 사용되지 않았고, 2000년대에 들어 윈도우의 LM 패스워드를 몇 분 만에 크래킹하며 유명해지기 시작했다.

레인보우 테이블은 하나의 패스워드에서 시작해 특정한 변이 함수로 변이된 형태의 여러 패스워드를 생성한다. 그리고 변이된 각 패스워드의 해시를 고리처럼 연결하여 일정 수의 패스워드와 해시로 이루어진 체인(Chain)을 무수히 만들어 놓은 테이블이다. 실제로 레인보우 테이블을 이해하는 것은 매우 복잡하다. 아주 기본적인 이해부터 시작해보자.

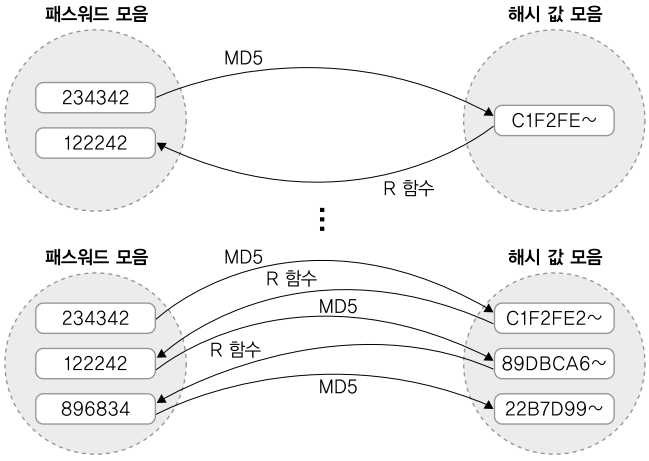
레인보우 테이블의 가장 기본적인 개념은 패스워드별로 해시 값을 미리 생성해놓는 것이다. 즉, 크래킹하고자 하는 해시 값을 테이블에서 검색하여, 거꾸로 원래 패스워드를 찾는 것이다. 예를 들어 패스워드가 '12qw' 고, 해시 값이 '123452323242' 라고 하자. [표 4-3]과 같이 각 패스워드별로 해시 값을 미리 구해둔 해시 테이블에서 '123452323242' 를 찾아 거꾸로 '12qw' 를 찾는 것이다.

[표 4-3] 미리 계산된 해시 테이블

패스워드	해시
1dww	551234523452
12qw	123452323242
21fe	523233452333
df32	234523232345

하지만 이런 식으로 사용 가능한 모든 패스워드에 대해 해시 값을 구해놓는다면, 엄청난 용량의 파일이 필요하다. 레인보우 테이블의 또 다른 핵심 아이디어는 대용량으로 생성될 수 있는 해시 테이블을 R(Reduction) 함수를 이용해 충분히 작은 크기로 줄이는 것이다. 물론 줄이더라도 레인보우 테이블은 보통 몇십 기가 바이트 정도의 용량으로 생성된다.

R 함수는 패스워드로 사용될 수 있는 문자열을 해시 값으로 만들어 내는 함수다. 예를 들어 패스워드가 6자리 숫자로 이루어진 '234342' 라면, 이 패스워드의 MD5 해시 값은 'C1F2FE224298D6E39EBA607D46F3D9CC' 다. R 함수는 이 해시 값에서 또 다른 형태의 무작위 패스워드를 추출한다. 여기서 R 함수가 MD5 해시 값 중 앞의 6개 숫자만 뽑아낸다고 가정해보면 R(C1F2FE224298D6E39EBA607D46F3D9CC)은 '122242' 가 된다. 그리고 레인보우 테이블을 생성하기 위해 MD5 해시 생성과 R 함수 동작을 [그림 4-6]과 같이 반복한다. 이를 체인(Chain)이라 한다.



[그림 4-6] MD5 해시와 R 함수의 동작

[그림 4-6]에서는 최초 패스워드인 '234342' 에서 MD 5 해시 값을 3번 구하고, R 함수가 2번 동작하였다(실제 레인보우 테이블을 생성할 때는 체인을 몇천 번에 걸쳐 형성한다). 이를 표로 정리하면 다음과 같다.

[표 4-4] 234342의 MD5 해시와 R 함수의 반복 실행 결과

패스워드		MD5 해시
최초 패스워드	234342	C1F2FE224298D6E39EBA607D46F3D9CC
첫 번째 R 함수 동작 결과	122242	89DBCA68BE341E03B5FB59777B93067E
두 번째 R 함수 동작 결과	896834	22B7D9922C994737D0D9DFCCF6B415B6

같은 방법으로 최초 패스워드가 '346343' 일 때와 '898232' 일 때 MD5 해시와 R 함수를 반복 실행한 결과를 정리해보자.

[표 4-5] 346343의 MD5와 R 함수의 반복 실행 결과

패스워드		MD5 해시
최초 패스워드	346343	A62798B2BFCF406BD76FCBC7A3678876
첫 번째 R 함수 결과	627982	570727EE4270E0C1A4D8FBB741926DB8
두 번째 R 함수 결과	570727	86AB6B3355F33F7CD62658FDDA5AF7D6

[표 4-6] 898232의 MD5와 R 함수의 반복 실행 결과

패스워드		MD5 해시
최초 패스워드	898232	91CF19DD04A05110A2D2A30D578DDA29
첫 번째 R 함수 결과	911904	3B8635770F22C17E9643441A3E49992E
두 번째 R 함수 결과	386357	E2038DD2A8315D9BF7F72AE5C07530F8

레인보우 테이블은 [표 4-4]~[표 4-6]에서 각 표의 최초 패스워드와 마지막 해시 값으로 다음과 같이 만든다.

[표 4-7] [표 4-4]~[표 4-6] 값을 이용해 생성한 레인보우 테이블

패스워드	MD5 해시
234342	22B7D9922C994737D0D9DFCCF6B415B6
346343	86AB6B3355F33F7CD62658FDDA5AF7D6
898232	E2038DD2A8315D9BF7F72AE5C07530F8

크래킹하고자 하는 패스워드의 해시 값을 '570727EE4270E0C1A4D8FBB741926DB8'이라 가정하면 레인보우 테이블을 이용한 패스워드 크래킹 과정은 다음과 같다.

❶ 레인보우 테이블에 크래킹하려는 해시 값과 같은 MD5 해시 값이 있는지 확인한다.
→ [표 4-7]에는 '570727EE4270E0C1A4D8FBB741926DB8' 해시 값이 없다.

❷ 레인보우 테이블에 크래킹하려는 해시 값이 없으면 크래킹할 해시 값에 R 함수를 적용하여 패스워드를 구하고 다시 해시 값을 구한다.
→ '570727EE4270E0C1A4D8FBB741926DB8'에 R 함수를 적용해 패스워드 '570727'을 구한다. '570727'의 해시 값을 구하면 '86AB6B3355F33F7CD62658FDDA5AF7D6'이다.

❸ 앞의 ❷에서 구한 해시 값 '86AB6B3355F33F7CD62658FDDA5AF7D6'이 레인보우 테이블에 있는지 다시 확인한다.
→ [표 4-7]에서 생성한 레인보우 테이블에 '86AB6B3355F33F7CD62658FDDA5AF7D6' 값이 있다.

❹ 레인보우 테이블에서 확인한 해시 값을 발견한 뒤 그 해시 값에 해당하는 최초 패스워드를 구한다(값이 없다면 같은 해시 값이 나올 때까지 ❷와 ❸ 과정을 해시 테이블 생성 시에 설정한 체인의 수만큼 반복한다.)
→ [표 4-7]에서 '86AB6B3355F33F7CD62658FDDA5AF7D6'에 해당하는 패스워드는 '346343'이다.

❺ 확인한 최초 패스워드에서 다시 패스워드와 일치하는 해시 값이 나올 때까지 MD5 해시와 R 함수를 반복 수행한다. 해당 해시 값이 확인되면 우리가 찾는 패스워드는 해당 해시 값을 생성한 문자열이 될 것이다.

[표 4-7]에서 '346343'와 '86AB6B3355F33F7CD62658FDDA5AF7D6'의 매칭 정보를 확인했으므로, 이제 '346343'과 '86AB6B3355F33F7CD62658FDDA5AF7D6' 사이의 체인을 확인해야 한다. 그 체인 중간 어디쯤에 패스워드가 있을 것이다. 그 체인은 레인보우 테이블에는 존재하지 않지만, 레인보우 테이블을 생성하는 과정에서 만들었다.

[표 4-5]가 그 체인이다. [표 4-5]에서 '346343'의 해시 값이 'A62798B2BFCF406BD76FCBC7A3678876'임을 알 수 있다. 이 해시 값에 R 함수를 적용한 결과는 '627982'다. '627982'를 다시 MD5 해시한 결과는 '570727EE4270E0C1A4D8FBB741926DB8'인데, 이 값은 크래킹하고자 했던 최초 해시 값이다. 따라서 크래킹하고자 했던 패스워드는 '570727EE4270E0C1A4D8FBB741926DB8'를 MD5 해시하기 바로 전 값인 '627982'임을 알 수 있다.

[표 4-4]~[표 4-6]에서는 체인 2개만을 사용했지만, 실제 레인보우 테이블에는 체인이 2,000개 이상이고, [표 4-7]과 같이 각 체인의 최초 패스워드와 최종 해시 값만 레인보우 테이블에 저장한다. 따라서 체인을 2,000개 사용하는 레인보우 테이블에서 해시 값을 10,000개 저장하고 있다면, 실제 그 레인보우 테이블에서 확인할 수 있는 패스워드의 종류는 20,000,000 (2000*10000)개가 된다. 보안 관련 일을 하면서 항상 느끼는 것이지만, 세상에는 천재가 참 많다.

Section

02 윈도우 인증과 패스워드

시스템마다 대부분 비슷한 방식으로 계정을 입력하고 패스워드를 입력하여 로그인한다. 하지만 이를 실제로 처리하고 인증(Authentication)하는 과정은 운영체제마다 다르며, 인증 과정은 패스워드 크래킹에 직접적인 영향을 주기도 한다. 먼저 윈도우 인증 과정을 살펴보자.

윈도우가 인증이라 할 만한 구조를 가지게 된 것은 NT 버전부터다. 윈도우 95나 98은 인증 체계를 갖추지 못했으며, 패스워드도 유명무실했다. 패스워드를 몰라도 로그인하는 데 문제가 없었고, 패스워드 삭제도 윈도우가 설치된 폴더에서 확장자가 PWL인 파일을 모두 지워버리면 됐다. 그리하여 공유 폴더의 패스워드가 아무리 복잡해도 간단히 크래킹되었다.

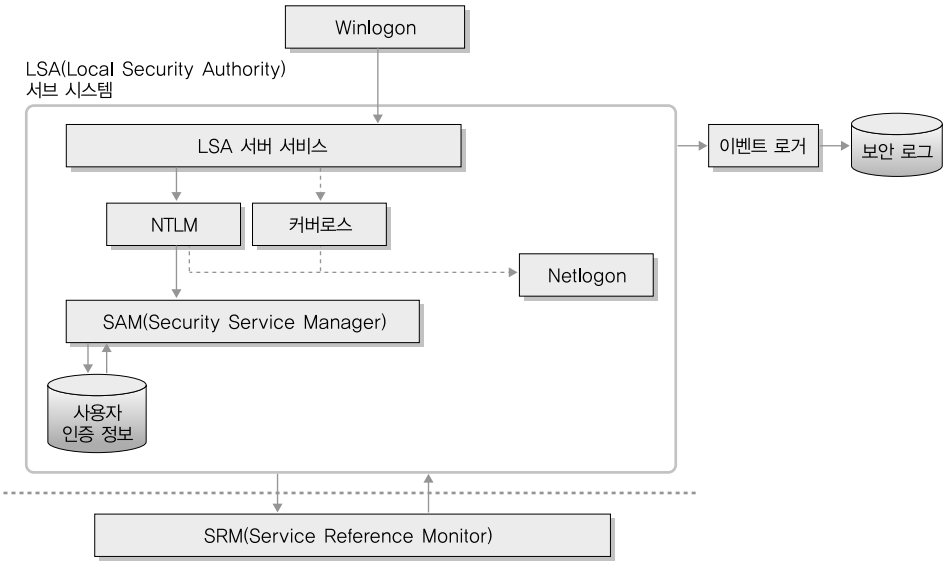
윈도우 NT 4.0부터 2008까지의 인증 체계는 기본적으로 비슷하지만, 인증 프로토콜과 암호화 로직은 꽤 많이 변화되었다. 윈도우 서버 2003 이전과 윈도우 비스타 이후로 구분하여 공통점을 중심으로 기본 내용을 학습하고, 개별적으로 차이점을 살펴보자.

1. 윈도우 인증의 구성 요소

윈도우의 인증 과정에서 가장 중요한 구성 요소는 LSA(Local Security Authority), SAM(Security Account Manager), SRM(Security Reference Monitor)다. SAM은 윈도우에서 패스워드를 암호화하여 보관하는 파일의 이름과 동일하다.

각 구성 요소들이 어떤 역할을 하며, 어떤 과정을 통해 로그인 인증을 수행하는지 살펴보자.

먼저 LSA는 모든 계정의 로그인에 대한 검증을 하고, 시스템 자원 및 파일 등에 대한 접근 권한을 검사한다. 물론 로컬, 원격 모두에 해당한다. 또한 이름과 SID를 매칭하며, SRM이 생성한 감사 로그를 기록하는 역할도 한다. 즉 LSA는 NT 보안의 중심 요소며, 보안 서브 시스템(Security Subsystem)이라 불리기도 한다.



[그림 4-7] 윈도우 인증 구조

SAM은 사용자/그룹 계정 정보에 대한 데이터베이스를 관리한다. 그리고 사용자의 로그인 입력 정보와 SAM 데이터베이스 정보를 비교해 인증 여부를 결정한다. 윈도우의 SAM 파일은 다음 경로에 위치한다(%systemroot%는 윈도우가 설치된 폴더로 보통 C:/Winnt 또는 C:/Windows다).

```
%systemroot%/system32/config/sam
```

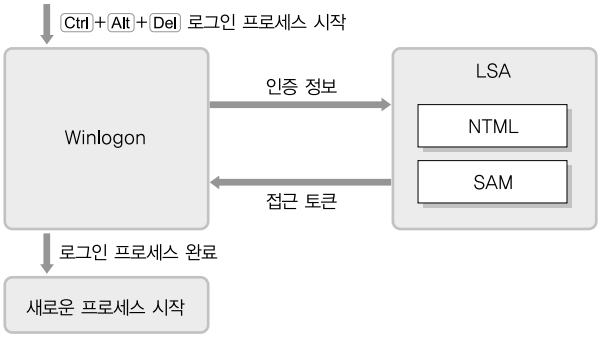
SAM이 사용자의 계정과 패스워드의 일치 여부를 확인하여 SRM에 알리면, SRM은 사용자에게 SID(Security Identifier)를 부여한다. 또한 SRM은 SID에 기반하여 파일이나 디렉터리에 대한 접근(access)을 허용할지를 결정하고, 이에 대한 감사 메시지를 생성한다.

❷ 로컬 인증과 도메인 인증

로컬 인증

윈도우를 부팅하면 운영체제 종류에 따라 로그인 창이 Ctrl + Alt + Delete 를 눌러야 나타나거나 자동으로 나타나는 경우가 있다. 이 창이 Winlogon 화면이다. 이 화면에서 아이디와 패스

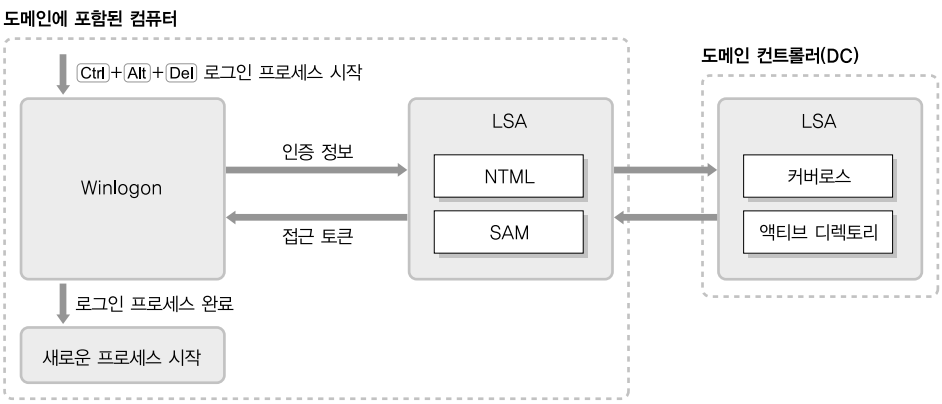
워드를 입력하면 [그림 4-8]과 같이 LSA 서브 시스템이 인증 정보를 받아 NTML 모듈에 아이디와 패스워드를 넘겨준다. 그리고 이를 다시 SAM이 받아 확인하고 로그인을 허용한다.



[그림 4-8] 윈도우 로컬 인증

도메인 인증

윈도우 도메인 인증에서는 로컬 인증처럼 Ctrl + Alt + Delete 를 눌러 로그인을 시도한다. Winlogon 화면에 인증 정보를 넣으면 로컬 인증과 마찬가지로 [그림 4-9]와 같이 해당 정보를 LSA 서브 시스템에 넘긴다. LSA 서브 시스템에서는 해당 인증 정보가 로컬 인증용인지 도메인 인증용인지 확인하고 커버로스(Kerberos) 프로토콜을 이용해, 도메인 컨트롤러에 인증을 요청한다. 도메인 인증에서는 기본적으로 풀 도메인 이름(FQDN: Full qualified domain name)과 커버로스 프로토콜을 이용하게 되어 있지만, IP를 이용해 접근을 시도할 경우 NTML을 사용한다.



[그림 4-9] 윈도우 원격 인증

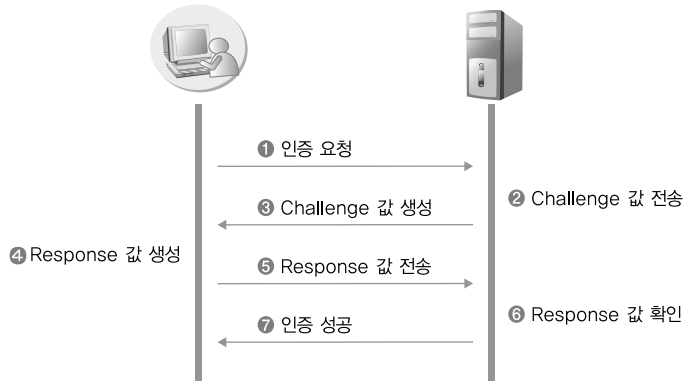
도메인 컨트롤러는 인증 정보를 확인하여 접속하고자 하는 사용자에게 접근 토큰을 부여하고 해당 권한으로 프로세스를 실행한다.

❸ 인증 구조

Challenge & Response 인증

패스워드 값을 인증 서버와 같은 인증 주체에 전달하여 올바른 패스워드임을 증명하는 가장 직관적이고 쉬운 방법은 패스워드 값을 직접 전달하는 것이다. 우리는 이미 이런 방식을 많이 접하고 있다. 텔넷이나 FTP가 아이디와 패스워드를 네트워크를 통해 직접 전달하고, 웹 포털 사이트에서 사용자 아이디와 패스워드로 로그인하는 것이 그러한 경우다.

하지만 운영체제 인증과 같이 높은 수준의 인증이 필요한 경우, 이런 단순 인증 방식은 패스워드 노출 또는 패스워드 재사용 공격에 매우 취약하다. 따라서 다음과 같은 Challenge & Response 방식으로 인증을 수행한다. 모든 Challenge & Response 방식의 인증 프로토콜은 기본 구조가 같다.



[그림 4-10] Challenge & Response 인증

각 단계를 간단히 살펴보자.

❶ 인증 요청

인증을 수행하고자 하는 주체가 인증 서버에 인증을 요청한다.

❷ Challenge 값 생성 / ❸ Challenge 값 전송

인증을 요청받은 인증 서버는 문자열 등의 값을 특정 규칙을 따르거나 혹은 랜덤하게 생성하여 인증 요구자에 전달한다.

❹ Response 값 생성

인증 요구자는 서버에서 전달받은 Challenge 값과 본인이 입력한 패스워드 정보 등을 이용해 서버에 보낼 Response 값을 생성한다. 대부분의 프로토콜이 Response 값을 생성하는 로직에서 차이가 난다.

❺ Response 값 전송 / ❻ Response 값 확인 / ❼ 인증 성공

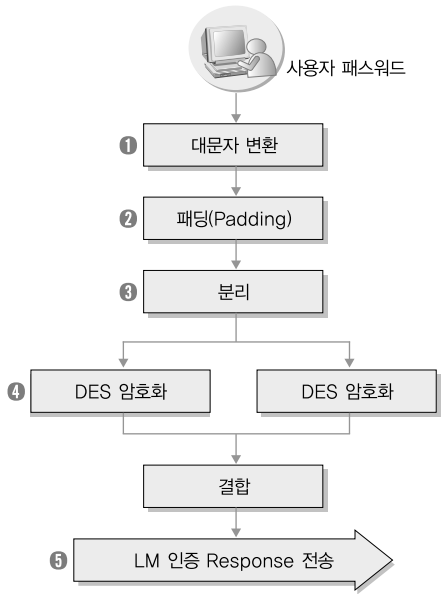
인증 요구자는 생성한 Response 값을 인증 서버에게 전달하고, 인증 서버는 이 Response 값을 확인하여 인증 요구자가 적절한 패스워드를 소유하고 있는지 확인한다. 그리고 확인된 Response가 적절하면 인증의 성공 여부를 인증 요구자에 알린다.

LM & NTML

서버 2003 이전의 윈도우는 인증에 LM과 NTML(NT 해시)을 함께 사용하였다. 즉, LM 해시와 NTLM 해시 모두 Response로 만들어 둘을 붙여 보냈다. 둘 다 만들어 보낸 것은 윈도우 시스템 간 호환성 때문이었으나, LM 해시가 비교적 크래킹이 쉽고 NTLM 해시도 해시 수준이 높지 않아 높은 보안 수준을 유지하기 어려웠다. LM 해시와 NTLM 해시의 내용을 간단히 살펴보자.

■ LM 해시

LM(Lan Manager) 해시는 1980년대에 만들어진 알고리즘으로, 본래 IBM의 OS 2에서 사용되다가 MS에서 1993년에 만든 윈도우 NT에 탑재되기 시작했다. LM은 구조적으로 취약한 알고리즘이지만, 윈도우 2000, XP의 기본 알고리즘이기도 하다. MS에서는 윈도우 비스타 이후 버전부터는 LM을 기본적으로 사용할 수 없게 했다. 하지만 사용하도록 설정을 변경할 수는 있다. LM의 알고리즘을 단계별로 살펴보자.

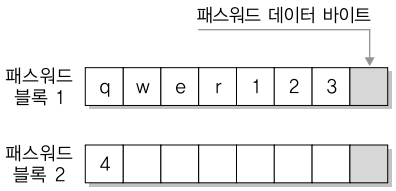


[그림 4-11] LM 해시 알고리즘

❶ **대문자 변환** : 우선 LM 방식은 사용자가 패스워드를 입력하면 모두 대문자로 바꾼다. 즉, LM은 패스워드의 대소문자를 구분하지 않는다.

❷ **패딩(Padding)** : LM은 기본적으로 14글자를 하나의 패스워드로 인식한다. 14글자가 되지 않는 패스워드는 뒤에 0을 붙여 14자리로 만든다.

❸ **분리** : LM은 패스워드 길이에 관계없이 8바이트가 블록 하나를 형성한다. 이 중 1바이트는 패스워드 블록에 대한 정보를 담고 있어 실질적으로 패스워드 문자열은 7바이트, 즉 문자 7개로 이루어져 있다. 따라서 패스워드가 qwer1234라면 8자이므로 패스워드 블록을 두 개 형성하게 된다.



[그림 4-12] 패스워드가 8자일 때의 패스워드 블록

❹ **DES 암호화** : 두 개의 블록으로 분리된 패스워드는 각각 "KGS!@#$%"라는 문자열을 암호화 키(Key)로 사용해 암호화한다.

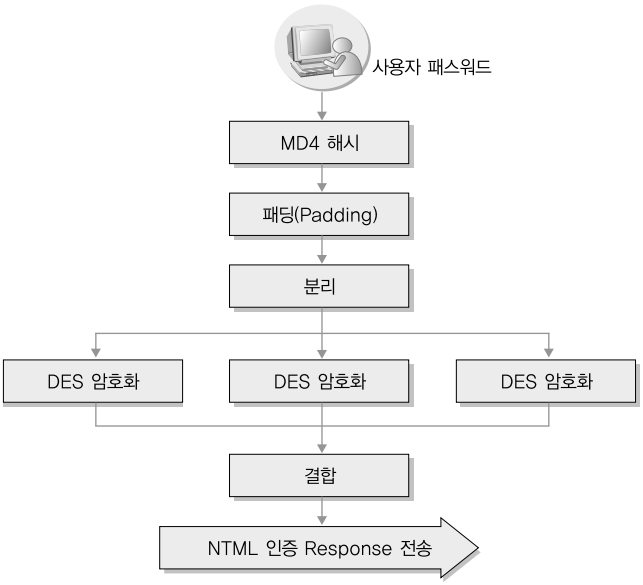
❺ **결합 :** "KGS!@#$%"로 각각 암호화한 두 결과 값을 합하여 SAM 파일에 저장한다.

LM 알고리즘 그림에서 확인한 바와 같이 패스워드 블록 하나는 별도로 운영된다. 따라서 패스워드를 아무리 길게 하더라도 패스워드 블록 하나인 7자리를 크래킹하는 것과 같은 노력으로 전부 풀 수 있다. qwer1234의 경우는 [그림 4-12]와 같이 qwer123과 4로 나뉜다. qwer123이 쉽게 크래킹되지 않을 수도 있겠으나, 4는 크래킹하는 데 몇 초 걸리지 않는다. Cain & Abel이라는 툴로 패스워드를 크래킹하는 방법을 [실습 4-1]에서 직접 학습해보자.

이렇게 윈도우는 문자열이 7개인 패스워드 블록을 이용해 패스워드를 구현하고 있기 때문에 7자 패스워드의 강도와 8자 패스워드의 강도가 사실상 같다. 이는 윈도우에서는 14자 패스워드를 크래킹하는 것이 7자 패스워드를 두 개 크래킹하는 것과 같은 노력이 필요하다는 의미다. 결국 14자 패스워드의 보안 강도는 7자 패스워드보다 겨우 2배 더 세다. 이것이 바로 LM의 구조적인 취약점이다.

■ NTML 해시

NTLM 해시는 [그림 4-11]과 같은 알고리즘으로 생성되는데, LM 해시에 MD4 해시가 추가된 것 외에는 큰 차이가 없다.

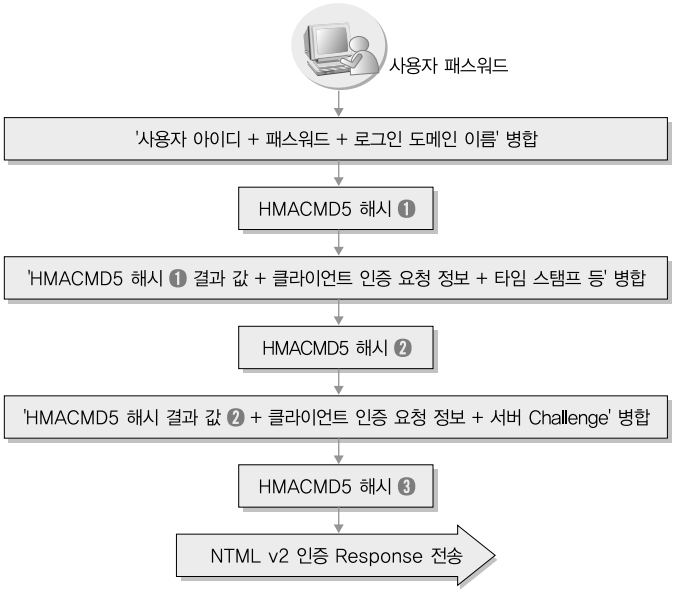


[그림 4-13] NTLM 해시 알고리즘

차이점을 간단히 살펴보면 사용자 패스워드를 입력할 때 MD4 해시가 수행되는데 그 결과 값이 16바이트다. 이 값에 5바이트의 패딩을 더해 21바이트를 만들고, 이를 LM 해시처럼 7바이트씩 3개 블록으로 나눈다. 그 다음은 LM 해시와 같다.

NTML v2

NTML v2는 윈도우 비스타 이후의 윈도우 시스템에서 기본 인증 프로토콜로 사용되며, 다음과 같이 인증한다. NTML v2는 LM/NTML과는 전혀 다른 알고리즘으로 해시 값을 생성하며, 현재 복잡도가 충분해 NTML v2 패스워드를 크래킹하기는 쉽지 않다.



[그림 4-14] NTMLv2 해시 알고리즘

저자 한마디

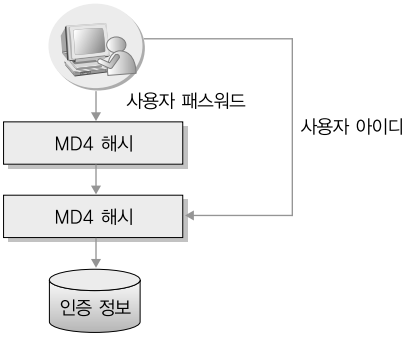
DES

DES는 1977년 IBM에서 만든 암호화 알고리즘으로, 56비트의 키 값으로 암호화한다. 개발 당시에는 매우 강력한 표준 알고리즘이었지만, 컴퓨터의 성능이 발달하면서 암호화 강도가 약해져 현재는 그다지 강한 암호화 알고리즘이 아니다. 1997년 초에는 RSA 암호화 알고리즘을 소유한 이들이 DES 메시지 해독에 10,000달러의 상금을 건 일이 있었다. 이때 약 3일 정도에 DES 알고리즘이 해독되었다고 한다.

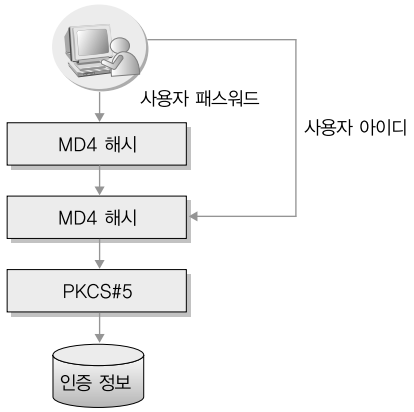
현재는 AES(Advanced Encryption Standard)의 라인델(Rijndael) 알고리즘이 사용되고 있다. 우리나라의 은행권에서 사용하는 국산 알고리즘인 SEED도 AES로 채택되기 위해 선별되었던 6개의 암호화 알고리즘 중 하나였다.

4. 자격 증명

일반 PC 사용자는 로그인할 때 로컬 계정을 이용한다. 하지만 회사 노트북이나 회사 PC처럼 도메인에 등록된 컴퓨터에 로그인할 때는 도메인 계정을 사용한다. 해당 컴퓨터가 도메인과 네트워크에 연결되어 있는 경우에는 NTML이나 커버로스를 이용해 로그인하는 것이 당연하다. 하지만 네트워크가 연결되지 않은 경우에도 도메인에 등록된 PC에 로그인할 때 도메인 계정을 사용해 로그인하는데, 이를 가능하게 만든 것이 자격 증명(Cache Credential)이다. 자격 증명을 'Password verifier' 라고도 하며 윈도우 서버 2003까지의 윈도우는 이 자격 증명을 [그림 4-15]과 같이 생성한다.



[그림 4-15] 윈도우 서버 2003까지의 자격 증명 생성



[그림 4-16] 윈도우 비스타 이후 자격 증명 생성

자격 증명에 대한 크래킹은 계속 시도되어 왔다. 윈도우 비스타 이후 버전에서는 [그림 4-16]와 같이 PKCS#5 암호화가 추가되어 초당 패스워드 크래킹을 10차례 정도밖에 시도하지 못하였다. 따라서 패스워드의 복잡도가 충분하다면 자격 증명에 대한 패스워드 크래킹은 사실상 거의 불가능하다.

여기서 잠깐

인터넷에서 안전한 정보를 교환할 수 있도록 산업계에서 사용하는 일련의 공개 키 기반 표준 프로토콜이다. 애플, MS, 썬 등이 공동 개발한 것으로, RSA 암호화, 패스워드 기반 암호화, 확장 인증서 구문법, 이메일 보안용으로 RSA사가 제안한 S/MIME을 위한 암호 메시지 구문법 등이 포함된다.

공개 키 암호 표준 (PKCS : Public Key Cryptography Standards)

실습 4-1 윈도우 XP, 2008, 7 패스워드 크래킹하기

윈도우 XP에서 LM/NTML 해시를 획득하여 일반적인 무작위 대입 공격(Brute-force 패스워드 크래킹)과 레인보우 패스워드 크래킹을 수행해보자.

실습 환경 구성

- 실습 환경 : 윈도우 XP가 설치된 시스템이나 VMWare에 윈도우 XP가 설치된 운영체제
- 필요 프로그램 : Cain & Abel과 Winrten(Cain & Abel과 Winrten은 www.oxid.it에서 다운로드 가능)

【윈도우 XP NT/NTML, 무작위 대입 공격】

1 테스트 계정 생성 및 패스워드 설정

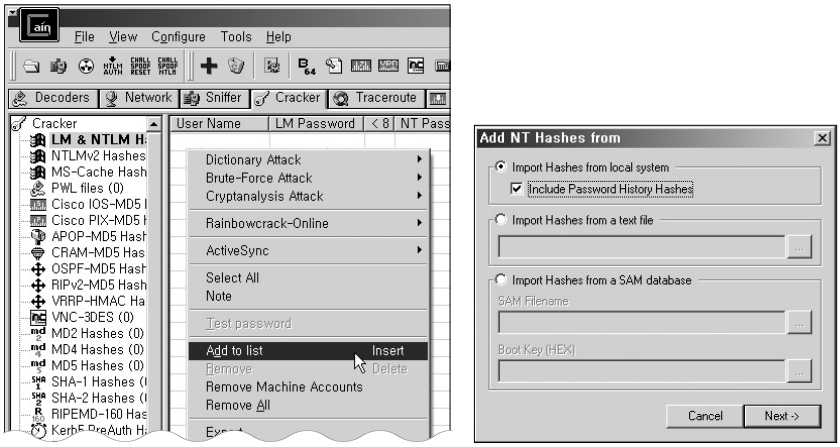
패스워드 크래킹에 사용할 계정 몇 개를 생성하고, 다양한 난이도의 패스워드 크래킹을 시도하기 위해, 숫자로만 된 패스워드, 짧은 패스워드, 영문자와 숫자로만 된 패스워드, 특수문자 등을 포함한 패스워드 등을 설정해본다. 윈도우 XP에서 계정 생성과 패스워드 설정은 [컴퓨터 관리]-[로컬 사용자 및 그룹]-[사용자]에서 가능하다. 필자는 Administrator 이외에 계정 4개를 생성하고, 각기 다른 패스워드를 설정하였다.

[컴퓨터 관리]는 탐색기의 [내컴퓨터]에서 마우스 오른쪽 버튼을 클릭하여 [관리] 메뉴를 선택하여 실행한다.

[그림 4-17] 테스트 계정 생성과 패스워드 설정

2 Cain & Abel을 이용한 LM/NTML 해시 덤프

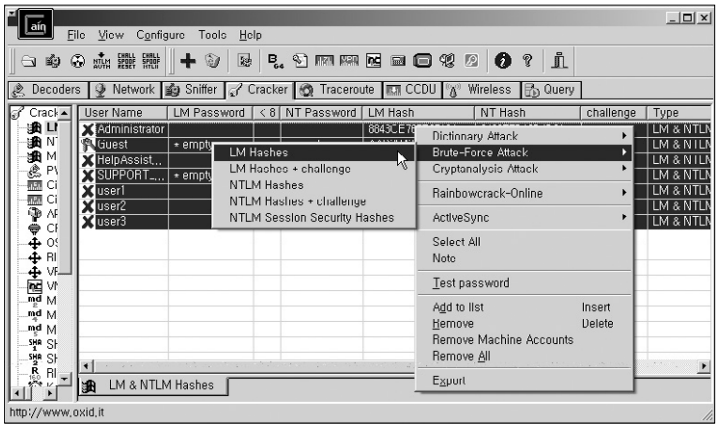
윈도우 XP에서 LM&NTML 해시를 덤프하기 위해 [Cracker] 탭의 'LM&NTML Hashes' 항목을 선택한 후 오른쪽 빈 창에서 마우스 오른쪽 버튼을 눌러 [Add to list] 메뉴를 선택한다. 오른쪽과 같이 LM/NTML 해시를 덤프하는 방법을 선택하는 창이 나타난다. 다른 시스템에서 획득한 LM/NTML 해시 값을 선택할 수 있고, SAM 파일을 직접 지정할 수도 있는데 여기서는 로컬 시스템에서 LM/NTML 해시를 획득하도록 선택한다.



[그림 4-18] 로컬 시스템에서 LM/NTML 해시 덤프

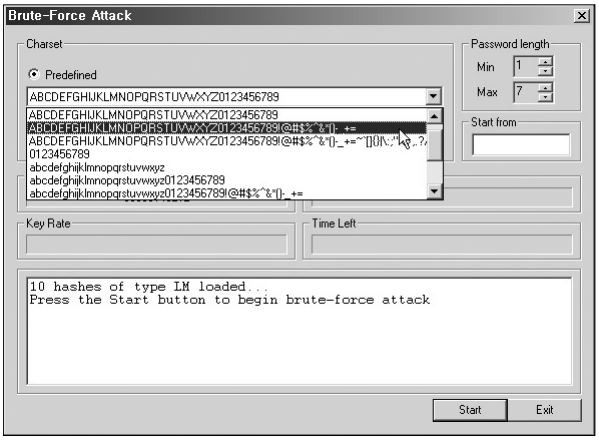
3 Cain & Abel을 이용한 무작위 대입 공격

로컬 시스템에서 NT/NTML 해시를 덤프하면, [그림 4-19]와 같이 사용자 이름과 LM 해시, NT (NTMLv2) 해시 값을 각각 확인할 수 있다. 윈도우 XP의 경우 LM 해시와 NTMLv2 해시 모두 크래킹 대상이 될 수 있지만, LM 해시가 더 취약하기 때문에 LM 해시를 대상으로 패스워드 크래킹을 수행한다. 확인된 계정을 모두 크래킹하기 위해 계정 하나에서 마우스 오른쪽 버튼을 누른 후 [Select All] 메뉴를 선택한다. 모든 계정이 선택되면 다시 마우스 오른쪽 버튼을 눌러 [Brute-Force Attack]-[LM Hashes] 메뉴를 선택하여 패스워드 크래킹 방법을 선택한다.



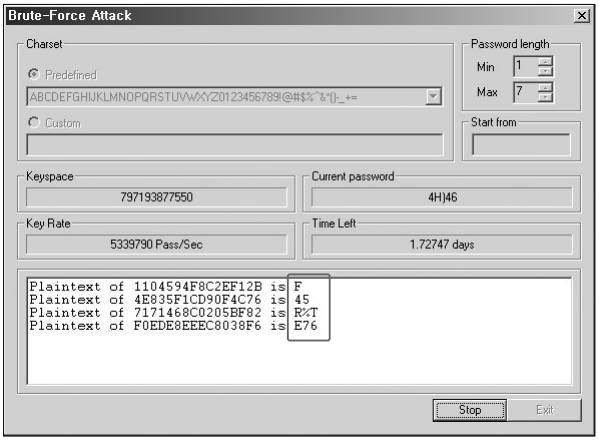
[그림 4-19] 패스워드 크래킹 방법 선택

다음으로 무작위 대입 공격을 수행하기 위한 상세 옵션을 선택한다. 패스워드 길이는 LM 해시 크래킹을 선택했으므로, 기본적으로 1~7 자리다. 그리고 무작위 대입 공격에 사용할 문자열을 선택할 수 있다. LM이 대소문자 구분하지 않으므로, 문자열은 대문자 알파벳과 숫자, 특수문자를 포함한다. 옵션 선택을 완료하였으면, 〈Start〉 버튼을 눌러 공격을 시작해보자.



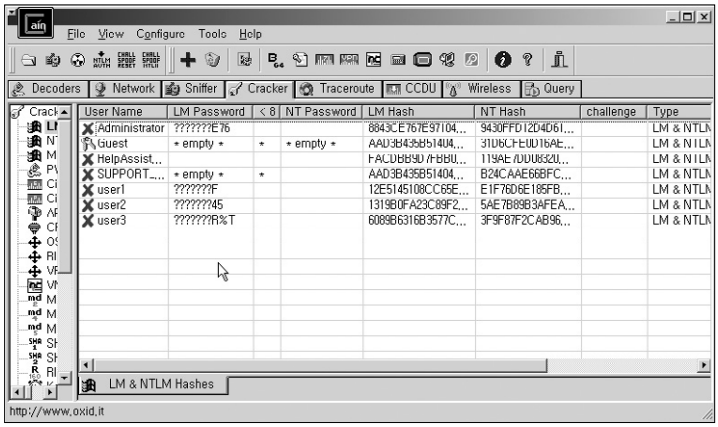
[그림 4-20] 패스워드 크래킹 옵션 선택

LM 패스워드는 패스워드가 7자리가 넘는 계정의 경우 7자리가 패스워드 블록 2개로 나누어지는데, [그림 4-21]과 같이 뒷 블록의 패스워드가 먼저 크래킹되어 나오는 것을 확인할 수 있다.



[그림 4-21] 크래킹되는 패스워드 확인

[그림 4-22]에서 〈Stop〉 버튼을 누르면 다음과 같이 패스워드 중 현재까지 크래킹된 부분을 확인할 수 있다.



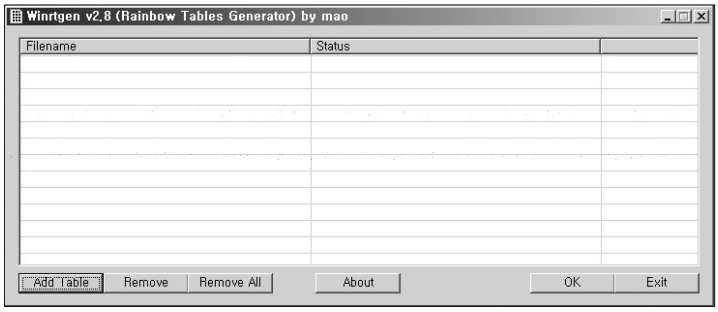
[그림 4-22] 크래킹되는 패스워드 확인

실제 패스워드 크래킹은 문자열과 숫자만으로 이루어진 패스워드의 경우 약 1.5일 시간 내에 크래킹이 가능하다.

【윈도우 XP NT/NTML, 레인보우 테이블을 이용한 패스워드 크래킹】

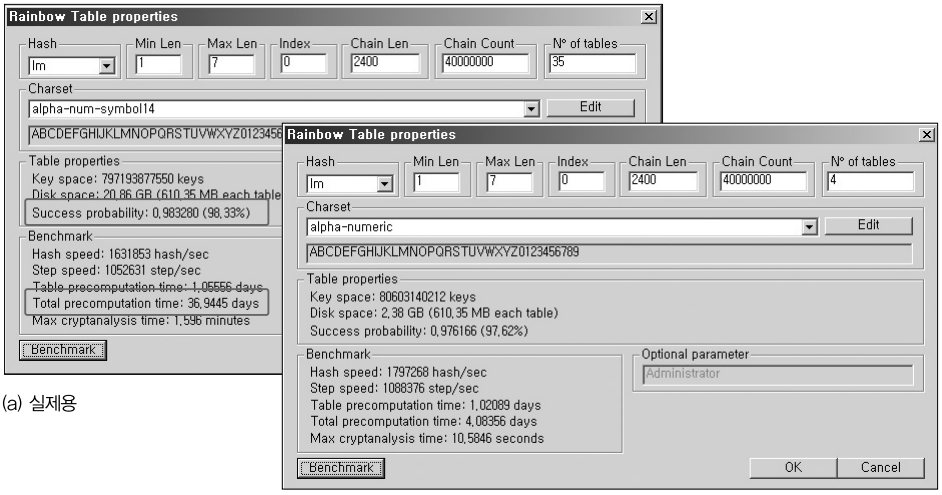
4 레인보우 테이블 생성

레인보우 테이블을 이용해 LM 패스워드를 쉽게 크래킹할 수 있다. Cain & Abel과 함께 내려받은 Winrtgen 프로그램을 이용해 레인보우 테이블을 생성해 사용할 수 있다. 먼저 Wintrgen을 실행한 뒤 〈Add Table〉 버튼을 누른다.



[그림 4-23] Winrtgen 실행

생성할 레인보우 테이블의 옵션을 선택한다. Hash는 'lm'로, Charset은 'alpha-num-symbol14'로 선택한다. No of tables에서 레인보우 테이블의 크기를 결정할 수 있는데, Charset을 'alpha-num-symbol14'로 선택한 경우 테이블이 35개 되어야 (a)와 같이 크래킹 성공률이 98.33%가 된다. 〈Benchmark〉 버튼을 누르면 레인보우 테이블을 생성하는 데 걸리는 시간 등을 확인할 수 있는데 약 37일이 걸린다. 이 시간은 시스템의 성능에 따라 차이가 난다. 소요 시간을 줄여 테스트하기 위해 레인보우 테이블의 옵션을 (b)와 같이 줄이고 〈OK〉 버튼을 눌러 실행하자.

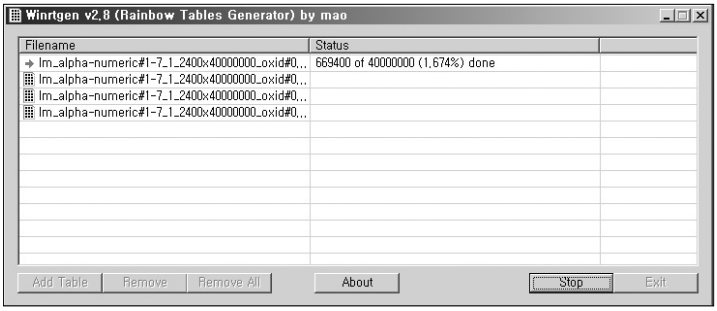


(a) 실제용

(b) 테스트용

[그림 4-24] 레인보우 테이블 실행 옵션

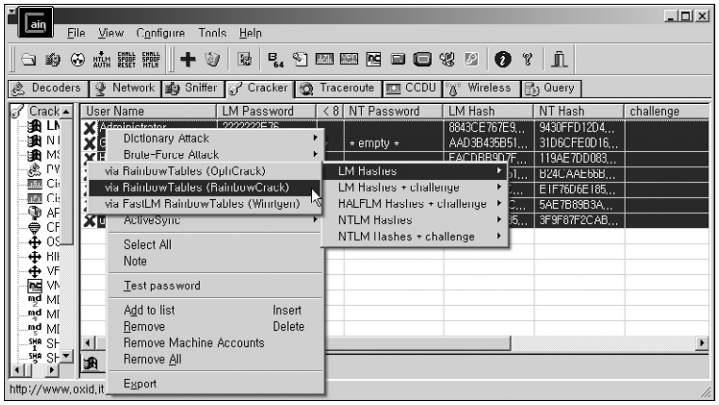
CPU의 성능이 좋다면 2일 이상 걸리지는 않을 것이다.



[그림 4-25] 레인보우 테이블 생성

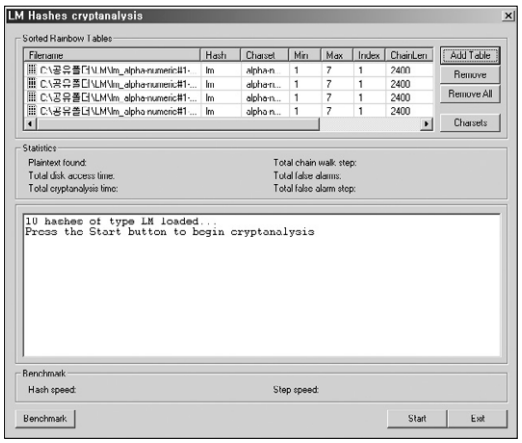
5 레인보우 테이블을 이용한 패스워드 크래킹

레인보우 테이블을 생성한 후 다시 패스워드 크래킹을 해보자. Cain & Abel에서 모든 계정을 선택한 뒤, 마우스 오른쪽 버튼을 눌러 [Cryptanalysis Attack]–[LM Hashes]–[via RainbowTables (RainbowCrack) 메뉴를 선택한다.



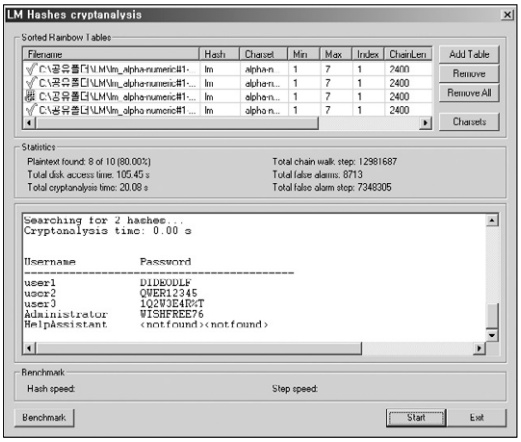
[그림 4-26] 레인보우 테이블 크래킹 선택

〈Add Table〉 버튼을 눌러 앞서 생성한 LM 레인보우 테이블을 선택하고, 〈Start〉 버튼으로 크래킹을 시작한다.



[그림 4-27] 레인보우 테이블 선택

약 20초의 시간이 지나면 다음과 같이 전체 패스워드들이 크래킹되어 나오기 시작한다. 전체 레인보우 테이블을 이용한 크래킹에는 약 3분의 시간이면 충분하다.

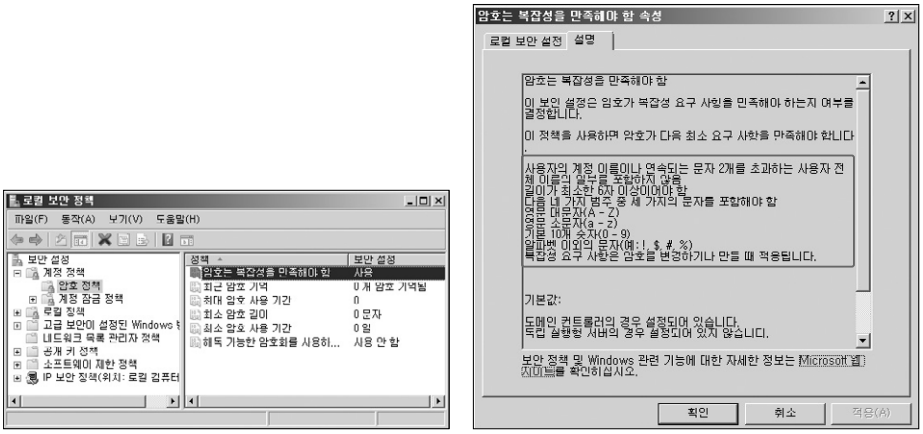


[그림 4-28] 레인보우 테이블을 이용한 패스워드 크래킹

【윈도우 2008/윈도우 7 NTMLv2, 무작위 대입 공격】

6 테스트 계정 생성 및 패스워드 설정

윈도우 2008에서도 윈도우 XP처럼 테스트 계정을 생성한다. 그런데 윈도우 2008에서는 [제어판]-[관리 도구]-[로컬 보안 설정]-[계정정책]-[암호 정책]에서 '암호는 복잡성을 만족해야 함' 항목 값이 '사용'으로 기본으로 설정되어 있어 간단한 암호는 설정할 수 없다. '암호는 복잡성을 만족해야 함' 항목에서 마우스 오른쪽 버튼을 누른 후 [속성] 메뉴를 선택하면 [설명] 탭에서 상세 내용을 확인할 수 있다.

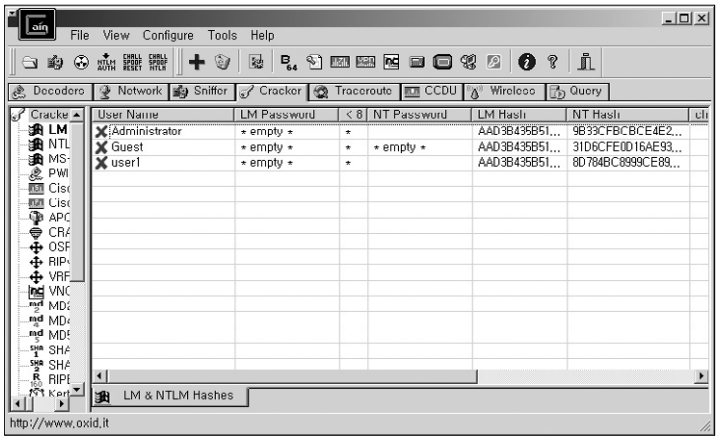


[그림 4-29] 로컬 보안 정책과 윈도우 2008 패스워드 설정 정책 확인

윈도우 7에서는 '암호의 복잡성을 만족해야 함' 항목 값이 '사용 안 함'으로 기본으로 설정되어 있어 패스워드 크래킹이 더 쉽다.

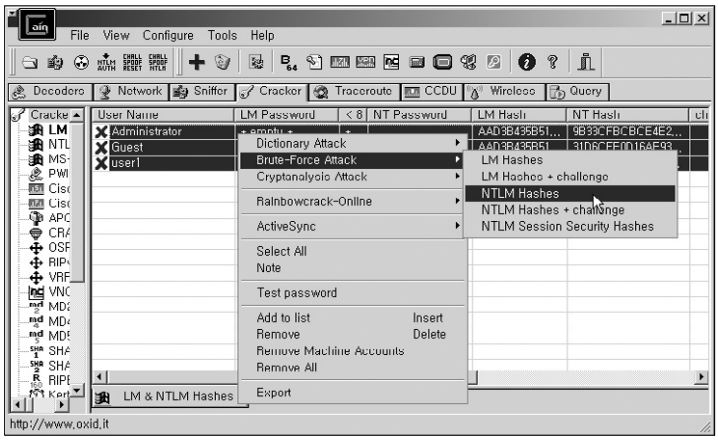
7 Cain & Abel을 이용한 NTMLv2 해시 덤프

윈도우 2008이나 윈도우 7에서 NTML 해시를 덤프하는 방법은 윈도우 XP와 같다. [그림 4-18]과 같이 [Cracker] 탭에서 'LM&NTML Hashes'를 선택한 뒤, 오른쪽 빈 창에서 마우스 오른쪽 버튼을 눌러 [Add to list] 메뉴를 선택한다. 다음 덤프 결과를 보면 LM 해시 값은 보이지만, LM Password에서 '* empty *'로 표시된 것처럼 이는 사용되지 않는 기본 값이다.



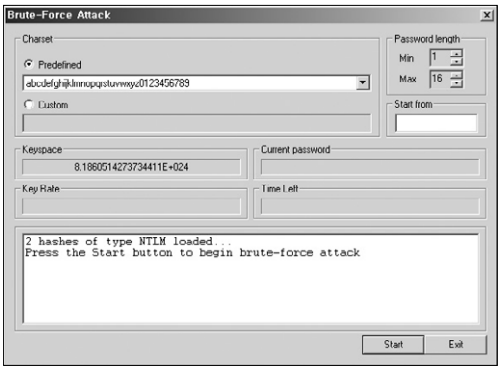
[그림 4-30] NTML 해시 덤프

따라서 윈도우 2008에서는 LM 해시를 이용한 크래킹을 수행할 수 없다. 다음과 같이 [Brute-Force Attack]-[NTLM Hashes] 메뉴로 크래킹해야 한다.



[그림 4-31] NTML 패스워드 크래킹 방법 선택

다음 화면에서 문자열을 선택하고, 〈Start〉 버튼을 누르면 패스워드 크래킹이 시작된다.



[그림 4-32] NTML 패스워드 크래킹 수행

윈도우 7의 패스워드에는 복잡도가 설정되어 있지 않지만, 윈도우 2008 패스워드는 복잡도 설정 환경으로 인해 간단한 패스워드가 존재할 수 없다. 하지만 복잡도가 일정 수준 이상인 패스워드를 크래킹하기 위해 계속 시간을 들여 패스워드 크래킹할 수는 없다. 여기서는 레인보우 테이블의 생성 시간으로 LM 패스워드와 NTML 패스워드 크래킹의 크래킹 시간을 가늠해보자.

[표 4-8] 해시 종류와 패스워드 설정 강도에 따른 레인보우 테이블 생성 조건

해시 종류	LM	NTMLv2	NTMLv2	NTMLv2
패스워드 최대 길이	7	7	10	10
구성 문자	알파벳(소문자), 숫자, 특수문자(14가지)			알파벳, 숫자
레인보우 테이블 숫자	30	30	3,700,000	1,400,000
레인보우 테이블 용량	17.88 GB	17.88 GB	2,205,371.85 GB	83,456.50 GB
성공률	97%	97%	97.16%	97.19%
레인보우 테이블 생성 시간	20일	12일	5140년	189년

[표 4-8]의 세 번째 NTMLv2 해시의 경우 패스워드의 길이를 10자리로 하고, 구성하는 문자를 알파벳(소문자), 숫자, 특수문자(14가지)로 설정하면 약 5140년의 시간이 소요된다. 따라서 패스워드 크래킹이 불가능하다고 할 수 있다 . 하지만 방법이 없는 것은 아니며, Winrtgen보다 효율적인 레인보우 테이블을 구매하여 크래킹하면 약 1시간 정도면 패스워드 크래킹할 수 있다.

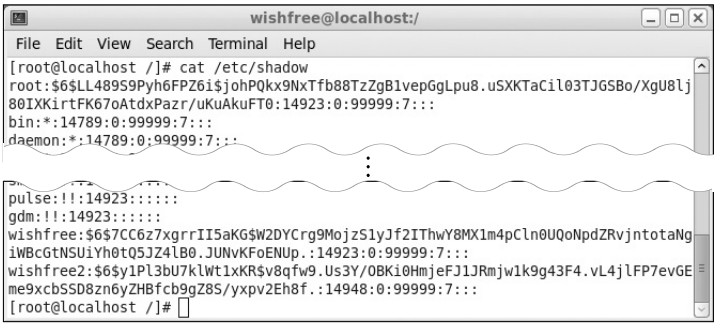
Section

03 리눅스/유닉스 인증과 패스워드

이제 리눅스와 유닉스의 인증 방식을 알아보자. 유닉스의 인증 방식은 윈도우의 인증 방식과 비교하면 훨씬 단순하지만 더 취약하지는 않다. 유닉스에서 인증에 가장 중요한 역할을 하는 것은 패스워드 파일과 shadow 파일이다. 패스워드 파일의 내용과 구조는 '3장. 계정과 권한' 에서 살펴본 바 있다.

패스워드 파일에는 계정 목록과 사용자 번호, 그룹 번호 등과 같은 정보가 존재하지만 패스워드와 직접 관련된 내용은 없다고 앞서 언급했다. 패스워드는 shadow 파일에 암호화되어 저장되어 있다. shadow 파일을 살펴보자.

```
cat /etc/shadow
```



[그림 4-33] /etc/shadow 파일

shadow 파일에서 root 계정에 대한 다음 정보를 확인할 수 있다. 각 정보는 패스워드 파일과 마찬가지로 :로 나뉘어 있다. 각각의 의미를 알아보자.

```
root : $6$LL489S99Pyh6~중략~Pazr/uKuAkuFT0/ : 14923 : 0 : 99999 : 7 : _ : _ :
  ❶                    ❷                     ❸    ❹     ❺     ❻   ❼   ❽
```

❶ 사용자 계정이다.

❷ 암호화된 사용자의 패스워드가 저장된다. 시스템마다 조금씩 다르며 페도라 14 버전에서는 SHA512 형식을 기본으로 저장되며, MD5, SHA256 등의 해시를 선택할 수 있다. '1' 로 시작하면 MD5, '5' 와 '6' 로 시작하면 각각 SHA256, SHA512를 나타낸다.

❸ 1970년 1월 1일부터 마지막으로 패스워드 변경한 날까지를 계산한 값이다. 14923일을 365로 나누면 약 41년이 된다.

❹ 패스워드 변경하기 전에 패스워드를 사용한 기간이다. 최초 설정 후 바꾸지 않았으므로 0이다.

❺ 패스워드 바꾸지 않고 최대한 사용할 수 있는 기간이다. 이 값은 보안 정책에 따라 달라질 수 있다. 보통 패스워드의 최대 사용 기간을 60일로 권고하고 있다.

❻ 패스워드 최대 사용 기간에 가까워질 경우 사용자에게 미리 그 사실을 알려야 한다. 그리고 패스워드 사용 기한 며칠 전에 경고를 보낼지 지정한다.

❼ 계정에 대한 사용 제한을 설정하고 며칠 후에 완전히 사용 정지할지 설정한다.

❽ 1970년 1월 1일부터 계정이 완전 사용 정지된 기간을 계산한 값이 기록된다.

❾ 관리자가 임의로 사용할 수 있는 부분이다.

shadow 파일의 필드 값에서 살펴볼 수 있듯, shadow 파일에는 암호화된 패스워드의 저장 기능 외에도 패스워드에 대한 보안 정책을 적용할 수 있다. 따라서 시스템에 shadow 파일이 존재하지 않고 passwd 파일에 암호화된 패스워드가 저장되어 있다면, 시스템에 계정에 대한 보안 정책이 적용되지 않았다고 간주해도 좋다.

리눅스나 유닉스에서 패스워드를 보호할 때는 로직이 정형화되어 있지 않다. 운영체제별로 조금 다르지만 기본적으로는 대부분 salt와 해시를 이용한다. 다만 해시에 사용되는 알고리즘이 다를 수 있다. 페도라 14에서는 [System]-[Administration]-[Authentication] 메뉴를 선택하면, [그림 4-34]와 같은 화면이 나오는데 'Advanced option' 에서 SHA512 해시 알고리즘을 사용하고 있음을 알 수 있다. 추가적으로 MD5, SHA256 등의 알고리즘을 선택할 수 있다.

[그림 4-34] Shadow 파일 해시 알고리즘 선택

리눅스의 경우에는 passwd 파일과 shadow 파일이 대부분 /etc/passwd와 /etc/shadow 파일로 생성된다. passwd 파일은 대부분 /etc/passwd 파일로 동일하나, shadow 파일은 다음과 같이 운영체제별로 고유한 경로와 파일명을 사용하는 경우도 많다.

[표 4-9] 운영체제별 passwd와 shadow 파일 위치

운영체제	shadow 파일의 위치
IBM AIX	/etc/security/passwd
IBM A/ux 3.0.3 (RS-6000)	/tcb/file/auth/?/*
BSD 4.3 - Reno	/etc/master.passwd
DEC DG/ux (Digital Unix)	/etc/tcb/aa/user
DEC EP/ux	/etc/shadow
HP/ux	/.secure/etc/passwd
IRIX 5	/etc/shadow
Free BSD	/etc/shadow
SunOS 4.1 + C2	/etc/security/passwd.adjunct
SunOS 5.x	/etc/shadow, passwd
System V Release 4.0	/etc/shadow, passwd

실습 4-2 리눅스 패스워드 크래킹하기

리눅스에서 패스워드 파일을 획득하여 일반적인 무작위 대입 공격(Brute-force 패스워드 크래킹)을 수행해본다.

실습 환경 구성

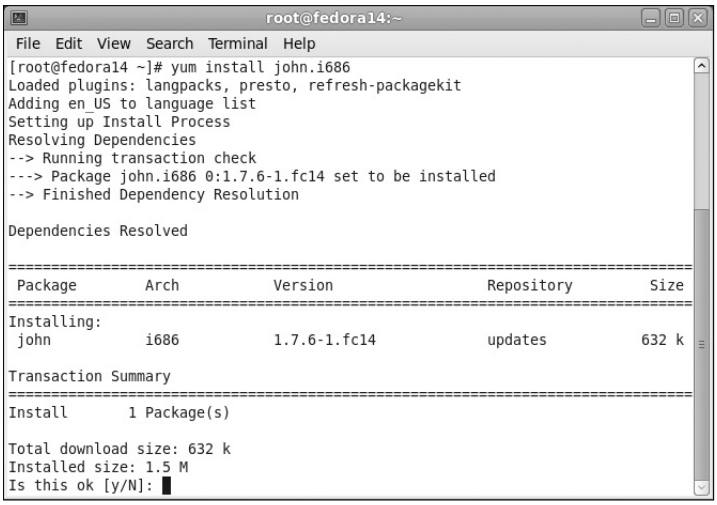
- 실습 환경 : 페도라 14
- 필요 프로그램 : John-the-ripper

【무작위 대입 공격】

1 John-the-ripper 설치

먼저 패스워드 크래킹에 사용할 John-the-ripper를 설치한다. 페도라 14에서 John-the-ripper는 다음과 같이 yum으로 간단히 설치할 수 있다.

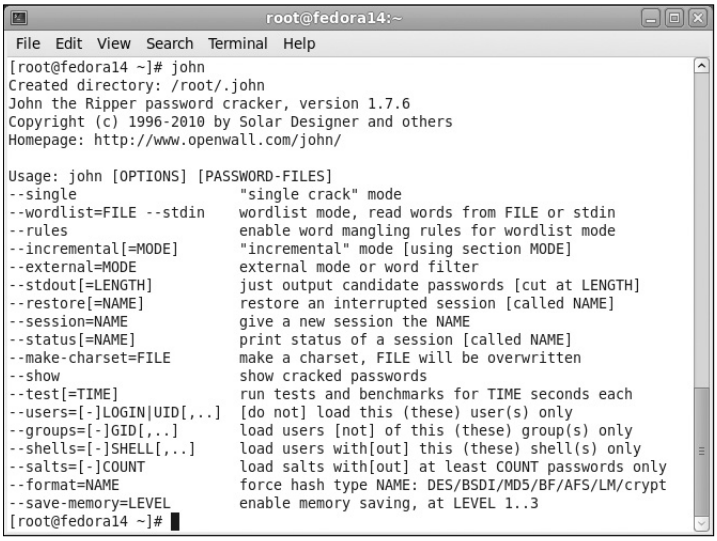
```
yum install john.i686
```



[그림 4-35] john the ripper의 설치

john the ripper 설치 후 john 명령으로 간단한 사용법 등을 확인할 수 있다.

```
john
```



[그림 4-36] john the ripper의 사용법 확인

2 테스트 계정 생성 및 패스워드 설정

윈도우에서처럼 패스워드 크래킹에 사용할 계정을 몇 개 생성한다. 다양한 난이도의 패스워드 크래킹 시도를 위해 숫자만으로 이뤄진 패스워드, 짧은 패스워드, 영문자와 숫자만으로 이루어진 패스워드, 특수문자 등을 포함한 패스워드 등으로 설정한다.

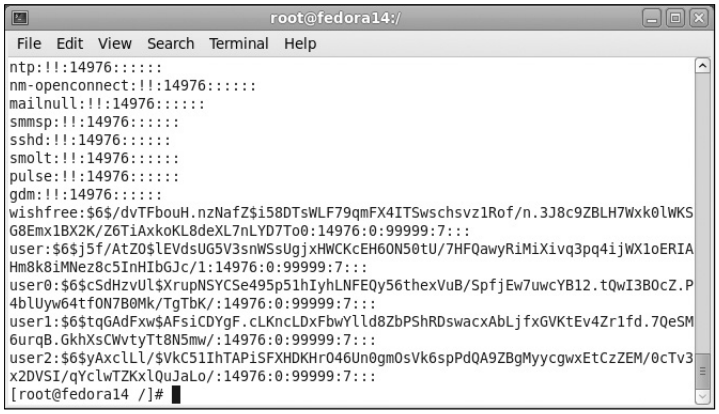
리눅스에서는 useradd 명령으로 계정을 생성할 수 있다.

```
useradd user
passwd user
```

[그림 4-37] 패스워드 크래킹을 위한 테스트 계정 추가

추가한 사용자의 패스워드 정보는 /etc/shadow 파일에서 확인할 수 있다. 필자는 패스워드 크래킹 테스트를 위해 user, user0, user1, user2 계정을 생성하였다.

```
cat /etc/shadow
```

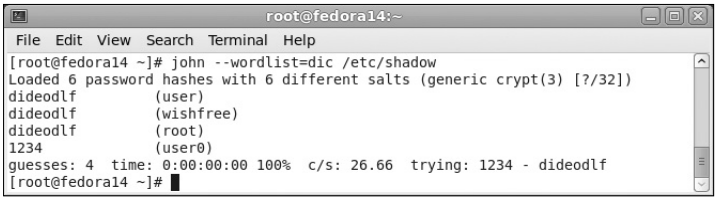


[그림 4-38] 추가한 계정의 SHA512로 해시된 패스워드 확인

3 패스워드 크래킹

리눅스에서의 패스워드 크래킹은 패스워드로 사용될 수 있는 사전 파일을 미리 만들어두고 이 사전 파일에 있는 패스워드를 대입해보는 것이다. 사전 파일에 'dideodlf(양대일)'이라는 패스워드를 미리 넣어둔 상태에서 john the ripper로 패스워드 크래킹을 시도해 보았다.

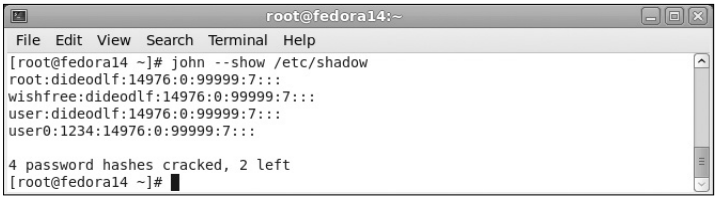
```
john --wordlist=dic /etc/shadow
```



[그림 4-39] 사전 대입법을 이용한 패스워드 크래킹

사전 대입 공격에 실패한 경우 무작위 대입법으로 패스워드 크래킹을 시도할 수 있다. 무작위 대입법을 이용한 패스워드 크래킹은 '--wordlist' 옵션 없이 바로 실행한다. 이때 패스워드가 크래킹되면 바로 확인할 수 있도록 '--show' 옵션을 사용하면 좋다.

```
john --show /etc/shadow
```



[그림 4-40] 사전 대입법을 이용한 패스워드 크래킹

앞서 사전 대입 공격에서 확인했던 패스워드를 다시 확인할 수 있다. 하지만 실제로 SHA512 해시를 사용한 shadow 파일을 무작위 대입법을 사용해 패스워드 크래킹하기는 매우 어렵다. SHA512 알고리즘을 사용한 해시 생성에 시간이 오래 걸려, 많은 경우의 수를 모두 대입하기 어렵기 때문이다. 따라서 SHA512와 같은 알고리즘으로 해시된 shadow 파일을 크래킹하기 위해서는 앞서 살펴보았던 레인보우 테이블을 이용해 크래킹하는 것이 훨씬 효율적이다.

Section

04 서비스 데몬 패스워드 크래킹

앞서 윈도우나 리눅스의 운영체제 중심으로 패스워드 크래킹을 하는 방법을 살펴보았다. 그러나, HTTP, FTP, 텔넷, SMB(NetBIOS) 데몬처럼 서버에서 제공하는 서비스 프로그램에 대해서도 패스워드 크래킹하는 것이 가능하다.

이러한 공격은 앞서 운영체제에서의 해킹처럼 운영체제에 대한 접근이 어느 정도 이뤄진 상태에서 하는 공격이 아니다. 여러 공격 대상 서버들 중 처음 비집고 들어갈 지점을 찾아서 하는 공격으로, 취약한 계정 하나를 찾기 위해 특정 네트워크에 대한 접근이 광범위하게 이뤄진다.

서비스 데몬 대부분은 운영체제와 동일한 아이디와 패스워드를 갖고 있다. 따라서 서비스 데몬을 통한 패스워드 크래킹 시도로 운영체제의 다른 서비스에 대한 접근 권한을 얻을 수도 있다. 윈도우의 파일 공유 서비스(SMB(NetBIOS))나 리눅스의 텔넷 서비스 등이 대표적인 경우다. 하지만 서비스 데몬에 대한 패스워드 크래킹에서 획득한 계정이 반드시 운영체제에 존재한다는 보장은 없다. 일부 윈도우 서비스의 경우 서비스 데몬에 대한 계정과 패스워드를 별도로 생성하여 관리하는 경우도 있다.

실습 4-3 서비스 데몬 패스워드 크래킹하기

NetBIOS, 텔넷, FTP 서비스에 대해 패스워드 크래킹을 수행한다.

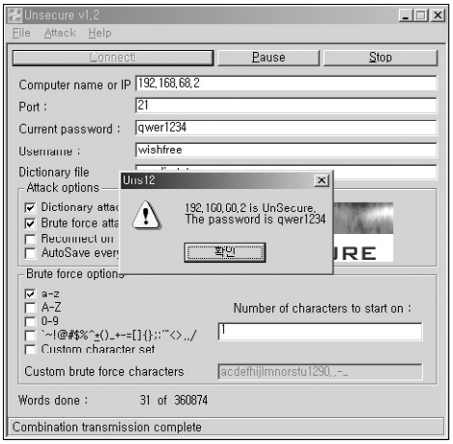
실습 환경 구성

- 실습 환경 : 공격자 (윈도우 XP), 공격 대상 시스템(윈도우 2008, 레드햇 6.2, 페도라 14 등)
- 필요 프로그램 : UNS, Brutus-AET

【UNS(Unsecure)를 이용한 FTP 계정 크래킹】

데몬 패스워드 크래킹 프로그램은 다양하다. 필자가 자주 쓰는 프로그램은 UNS와 BRUTUS-AET2다. 많은 툴이 사전 공격은 지원하나 무작위 대입 공격 기능은 충분히 제공하지

않는다. UNS는 무작위 대입 공격 성능이 좋고 다양한 선택이 가능하다. 또한 여러 개의 UNS를 동시에 띄워 패스워드를 크래킹할 수도 있다. UNS가 가장 좋은 성능을 보이는 것은 FTP 포트인 21번 포트에 대한 공격이다. [그림 4-41]에서는 wishfree 계정에 대한 FTP 패스워드가 qwer1234임을 확인할 수 있다.



[그림 4-41] FTP 계정 크래킹

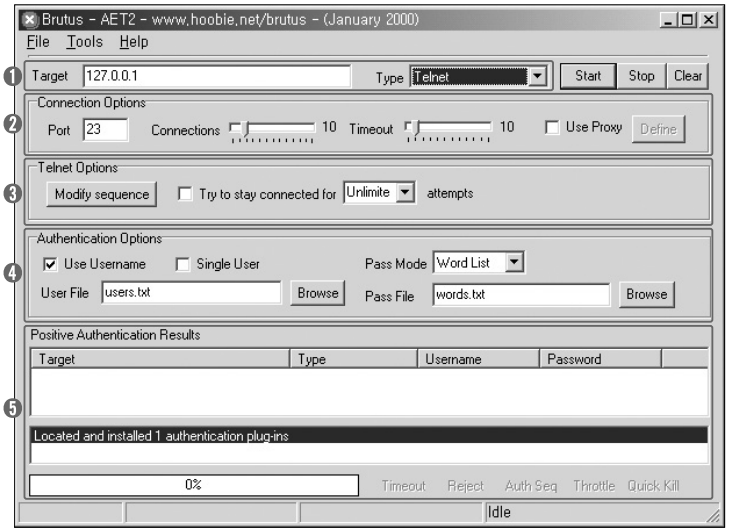
【BRUTUS-AET2를 이용한 텔넷 계정 크래킹】

BRUTUS(BRUTUS는 www.hoobie.net/brutus/brutus-download.html에서 다운로드 가능)는 HTTP, FTP, 텔넷, SMB(NetBIOS) 데몬의 패스워드를 크래킹할 수 있으며 백도어인 NetBUS 패스워드 크래킹도 가능하다. 그런데 NetBUS 패스워드 크래킹이 왜 패스워드 크래킹 툴에 포함되어 있을까? 이는 과거에 NetBUS가 설치된 컴퓨터가 매우 많았기 때문이다.

BRUTUS 툴로 텔넷의 패스워드를 크래킹해보자. 조금 복잡하지만 무척 신기하다. 텔넷은 FTP와 같이 단순한 버튼 하나로 크래킹되지는 않는다. 따라서 로그인할 때 텔넷 데몬이 보내는 문자열을 보고 순서도를 먼저 만들어야 한다. 페도라 14에서는 텔넷 데몬이 동작하지 않게 기본 설정되어 있으므로 레드햇 6.2를 사용하자.

1 BRUTUS-AET2 인터페이스 확인

BRUTUS-AET2를 처음 실행하면 다음과 같은 화면이 나온다. 각 부분의 기능을 살펴보자.



[그림 4-42] 인터페이스 확인

❶ 목표 시스템의 IP 주소와 크래킹할 데몬을 선택할 수 있다. 선택할 수 있는 데몬은 HTTP,FTP, 텔넷, SMB(NetBIOS), NetBUS다.

❷ 연결 포트와 1회 시도 횟수(Connections), 응답 대기 시간(Timeout)을 설정한다. 1회 시도 시 10개의 패스워드를 테스트하는 것이 기본이다. 시스템 응답 시간이 많이 걸릴 경우 응답 대기 시간을 충분한 값으로 설정한다.

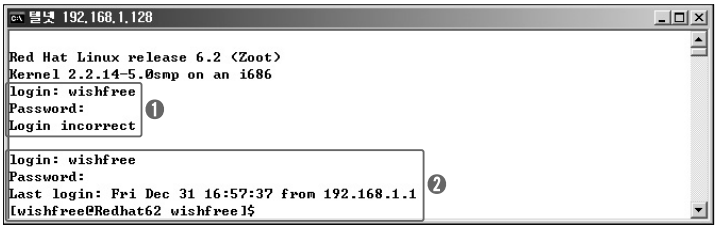
❸ 텔넷 데몬에 대한 특정 시퀀스를 입력하고 변경한다. 자세한 것은 다음 과정에서 살피자.

❹ 크래킹하고자 하는 계정과 패스워드 목록을 설정한다.

❺ 크래킹의 진행 과정 및 결과를 출력한다.

❷ 텔넷 서비스 인증 순서 확인하기

BRUTUS를 이용해 텔넷 서비스를 크래킹하려면 규칙성을 먼저 파악해야 한다. 규칙성을 파악한 뒤, [그림 4-42]의 ❸〈Modify sequence〉 버튼을 누르고 적절한 값을 입력한다. 로그인을 시도하여 한 번은 잘못된 패스워드 입력을 하였으며, 두 번째 시도에서 정상적으로 로그인하였다. 패스워드를 모르는데 이런 테스트를 어떻게 하느냐고 말하는 독자는 없으리라고 생각한다. 테스트 시스템을 만들어 수행할 수 있다.



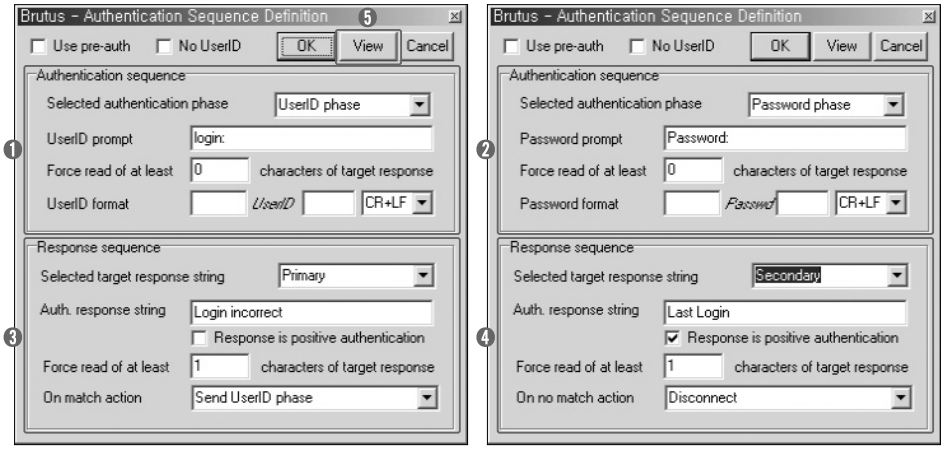
[그림 4-43] 인증 순서 확인

❶ 'login:' 뒤에 계정을 입력하고, Enter(CR, Carriage Return)를 입력한다. 줄이 바뀌면(LF, Line Feed) 'password:' 문자열이 나온다. 잘못된 패스워드를 입력한 경우에는 다시 'CR+LF' 과정을 거치고 'Login incorrect' 문자열이 뜬다.

❷ 로그인을 성공할 때도 정상적인 패스워드를 입력한 뒤, 'CR+LF' 과정을 거치면 된다. 비슷한 과정으로 단지 'Last login: Tue .... '만 다르다.

❸ 텔넷 서비스 규칙성 설정

이제 위의 과정을 BRUTUS에 입력하자. [그림 4-42]의 ❸ 〈Modify Sequence〉 버튼을 눌러보자. 먼저 [그림 4-44]의 (a)와 같이 'Selected authentication phase'를 'UserID phase'로 설정하고 처음 아이디를 로그인하는 과정과 아이디 입력에 따른 응답 규칙을 입력한다. 'Selected authentication phase'를 'Password phase'로 바꾸면 (b)와 같은 창을 확인할 수 있다. (b)에서는 패스워드를 입력하는 규칙과 그에 대한 응답 규칙을 입력한다.



(a) UserID phase 설정 화면 (b) Password phase 설정 화면

[그림 4-44] 규칙성 설정

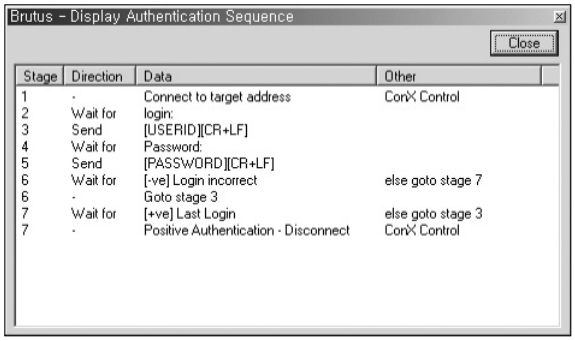
❶ **Authentication Sequence** : UserID phase – 문자열 'login :' 에 대한 입력을 설정한다.

❷ **Authentication Sequence** : Password phase – 문자열 'Password :' 에 대한 입력을 설정한다.

❸ **Response Sequence** : Primary – 위 테스트에서 잘못된 패스워드를 입력했을 때 'Login incorrect' 문자열을 응답으로 받았다. 그대로 입력하고 해당 문자열을 만났을 때 On match action을 Send UserID phase로 한다. 이로써 ❶ 과정으로 돌아가 처음부터 계정과 패스워드를 입력받게 한다.

❹ **Response Sequence** : Secondary – 올바른 패스워드를 입력했을 때 응답받는 'Last Login ...' 문자열을 입력한다. 해당 문자열을 받을 경우 우리가 올바른 패스워드를 입력한 것이므로, 'Response is positive authentication' 에 체크 표시를 해준다. On no match action은 잘못된 패스워드를 입력한 경우므로 'Disconnect' 로 설정한다.

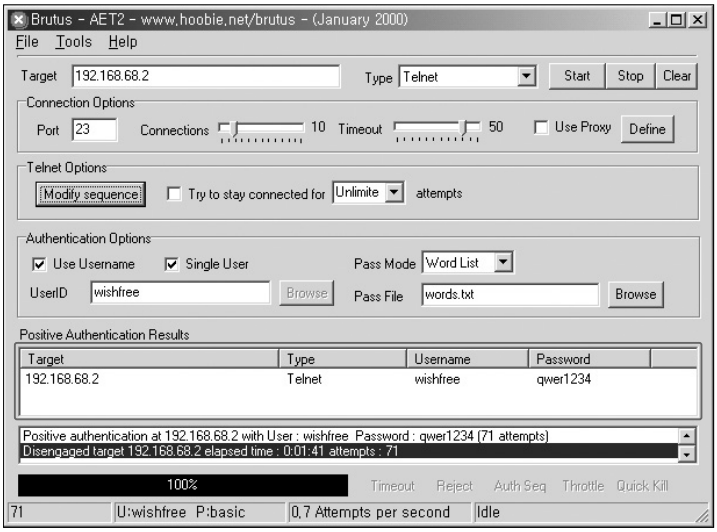
❺ [그림 4-45]에서는 설정한 규칙성을 확인할 수 있다. Stage2에서 'login:' 문자열을 받으면 Stage3에서 크래킹하고자 하는 계정을 입력한다. 그리고 Stage 4에서 'Password :' 문자열을 받으면 Stage5에서 크래킹할 패스워드를 입력한다. Stage6에서는 Login incorrect를 만나면 Stage3으로 돌아가 계정을 입력하는 과정을 반복한다.



[그림 4-45] 설정한 규칙성 확인

4 패스워드 크래킹

규칙성을 올바르게 설정하였으면 텔넷 계정을 크래킹해보자. 필자의 시스템은 응답 시간이 길어 'Timeout' 을 50초로 설정하였으며, 'wishfree' 계정 하나에 대해서만 테스트를 실시하였다. 크래킹된 패스워드를 확인할 수 있다.



[그림 4-46] 패스워드 크래킹

그럼 패스워드 크래킹에 대한 보안 대책은 무엇일까? 거듭 강조했듯이 기본은 적절한 패스워드를 설정하는 것이다. 그리고 암호화된 SAM 파일이나 shadow 파일을 잘 보호해야 한다. 하지만 이러한 파일을 획득하는 방법은 너무 다양하기에 결국 패스워드를 보호하려면 보안의 전반적인 내용을 알아야 한다.

Section

05 운영체제별 패스워드 복구

윈도우 95, 98을 쓰던 시절에는 패스워드 분실에 대한 걱정이 없었다. 설정한 패스워드를 잊어도 컴퓨터를 쓰는 데는 아무런 지장이 없었기 때문이다. 그럼에도 개인용 컴퓨터를 쓰면서 패스워드 때문에 골치가 아팠다면 BIOS(Basic input/output system) 패스워드를 잊어 시스템의 기본 설정을 바꿀 수 없었을 때일 것이다.

BIOS 패스워드를 복구하려면 우선 메인 보드 매뉴얼이 있어야 한다. 매뉴얼의 메인보드에서 BIOS 패스워드 복구용 점퍼(Jumper)의 위치를 찾을 수 있을 것이다. 메인보드에서 점퍼의 모양은 파란색이나 검은색의 아주 조그만 사각형이다. 점퍼는 회로의 연결을 쉽게 조절하기 위한 도구로 메인보드에 꽤 많이 부착되어 있다. BIOS 패스워드 복구용 점퍼를 찾았으면 다시 회로를 연결하는 방향으로 점퍼를 꼽은 후 부팅한다. 처음에는 화면에 아무 것도 나타나지 않을 것이다. 이런 상태에서 약 10초 후에 시스템의 전원을 끄고 점퍼를 원위치로 바꾼 뒤 재부팅하면 BIOS 패스워드가 사라졌음을 확인할 수 있다.

윈도우 2000이나 윈도우 XP, 리눅스 등을 쓰고 있다면 패스워드를 잊어 운영체제를 다시 설치한 경험이 한 번쯤은 있을 것이다. 특히 시스템 운영자라면 운영 중인 시스템의 패스워드를 더욱 철저히 관리해야 한다. 만약 시스템이 해킹당해 해커가 패스워드를 바꿔놓았다면 문제가 커진다. 하지만 다행히 로컬에서 각 시스템의 패스워드를 복구하는 것은 원격보다 훨씬 쉽다.

실습 4-4 윈도우 패스워드 복구하기

분실한 윈도우 패스워드를 알아내려 할 때 먼저 시도할 수 있는 방법은 패스워드 크래킹이다. 패스워드 크래킹을 위한 SAM 파일을 얻기 위해 NTFS를 읽을 수 있는 부팅 디스켓으로 부팅하고, SAM 파일을 복사한 뒤, 앞 절에서처럼 패스워드 크래킹 툴로 패스워드를 크래킹한다. 하지만 윈도우 2008처럼 충분히 복잡한 패스워드로 설정해둔 경우라면 이런 시도는 무모하며 패스워드를 초기화하는 것이 현명하다.

윈도우 시스템에서 패스워드를 초기화하는 방법은 CD의 미리 만들어진 이미지로 부팅 후, NTFS를 무시하고 하드웨어 기반에서 패스워드가 저장된 섹터를 임의로 바꾸는 것이다. 리눅스의 부팅 디스켓이나 CD를 부팅 이미지로 이용할 수 있다.

실습 환경 구성

- 실습 환경 : 윈도우 2008
- 필요 프로그램 : 윈도우 복구용 부팅 이미지(http://pogostick.net/~pnh/ntpasswd에서 다운 가능)

1 패스워드 복구용 CD를 이용한 부팅

다음은 윈도우 2008을 패스워드 복구용 CD를 이용해 부팅한 후 초기 화면에서 별다른 명령어 없이 Enter를 누른다.

2 파티션 마운트

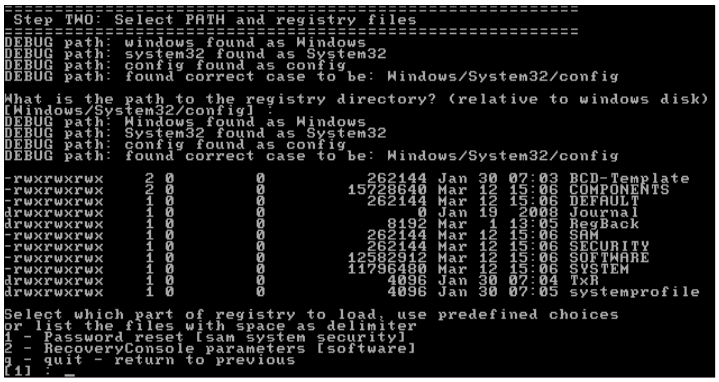
보통의 IDE 하드디스크를 사용하는 경우에는 파티션을 따로 마운트하는 과정이 필요없다. 그러나 SCSI 하드디스크를 사용한다면 다음과 같이 파티션을 마운트하는 작업이 필요하다. 윈도우가 설치되어 있는 /dev/sda1을 선택한다.

```
* Windows Registry Edit Utility Floppy / chntpw                              *
* (c) 1997 - 2010 Petter N Hagen - pnordahl@eunet.no                          *
* GNU GPL v2 license, see files on CD                                         *
*                                                                            *
* This utility will enable you to change or blank the password of            *
* any user (incl. administrator) on an Windows NT/2k/XP/Vista                *
* WITHOUT knowing the old password.                                          *
* Unlocking locked/disabled accounts also supported.                         *
*                                                                            *
* It also has a registry editor, and there is now support for                *
* adding and deleting keys and values.                                       *
*                                                                            *
* Tested on: NT3.51 & NT4: Workstation, Server, PDC.                         *
*            Win2k Prof & Server to SP4. Cannot change AD.                   *
*            XP Home & Prof: up to SP3                                       *
*            Win 2003 Server (cannot change AD passwords)                    *
*            Vista & Win7 32 and 64 bit, Server 2008 32+64 bit               *
*                                                                            *
* HINT: If things scroll by too fast, press SHIFT-PGUP/PGDOWN                *
******************************************************************************

============================================================
There are several steps to go through:
- Disk select with optional loading of disk drivers
- PATH select, where are the Windows systems files stored
- File-select, what parts of registry we need
- Then finally the password change or registry edit itself
- If changes were made, write them back to disk

DON'T PANIC! Usually the defaults are OK, just press enter
             all the way through the questions

============================================================
- Step ONE: Select disk where the Windows installation is -
============================================================

Disks:
Disk /dev/sda: 42.9 GB, 42949672960 bytes

Candidate Windows partitions found:
 1 :             /dev/sda1   40958MB BOOT

Please select partition by number or
 q = quit
 d = automatically start disk drivers
 m = manually select disk drivers to load
 f = fetch additional drivers from floppy / usb
 a = show all partitions found
 l = show probable Windows (NTFS) partitions only
Select: [1] ^P_
```

[그림 4-47] 윈도우 파티션 마운트

3 파티션 마운트와 윈도우 디렉터리 지정

윈도우의 기본 디렉터리를 지정한다. 기본 값은 '/windowns/system32/config'다. 다음으로 어떤 작업과 관련된 레지스트리를 수정할 것인지를 물을 때, '1 – Password reset [sam system security]' 항목을 실행한다.



[그림 4-48] 윈도우 디렉터리 지정

4 SAM 파일 편집

계속 Enter를 누르면 기본적으로 SAM 파일이 읽힌다. 그 후 사용자 계정을 편집할 것인지에 대한 메뉴를 확인할 수 있다. 1번의 'Edit user data and passwords'를 선택한다.

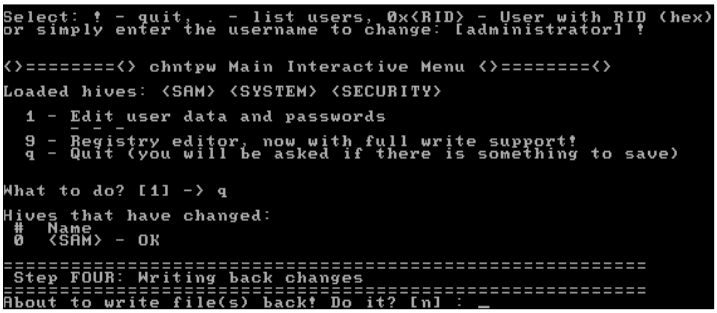
[그림 4-49] SAM 파일 선택 및 편집 메뉴 선택

해당 시스템의 계정 목록을 확인할 수 있다. 관리자 계정(Administrator)를 선택하고, User edit menu에서 '1. clear (blank) user passwd'을 선택하여 해당 계정의 패스워드를 삭제할 수 있다.

[그림 4-50] 계정 선택 후 계정에 대한 패스워드 초기화

5 설정사항 저장

다시 계정 선택 메뉴가 나오면 '!(quit)'를 누르고, 다음 메뉴에서 'q(quit)'로 프로그램을 끝낸다. 마지막으로 'Step FOUR : Writing back changes'에서 'y'를 누르면 패스워드 초기화가 끝난다.



[그림 4-51] 패스워드 변경 후 저장

재부팅을 하면, 패스워드가 삭제되어 있는 것을 확인할 수 있을 것이다.

실습 4-5 리눅스 패스워드 복구하기

리눅스의 부팅 과정에서 싱글 모드(Single Mode)에 대해 살펴봤다. 리눅스의 패스워드 복구법은 매우 간단하다. LILO와 GRUB이 약간 다르나, 최근에 많이 쓰고 있는 GRUB를 기준으로 살펴보겠다.

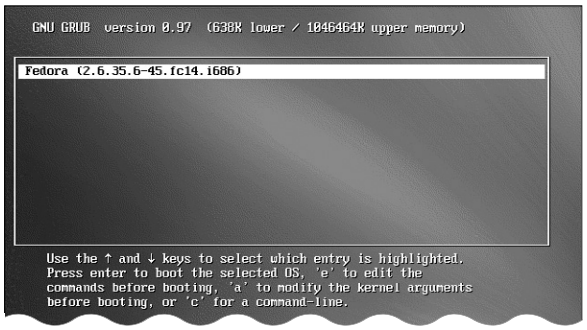
실습 환경 구성

• 실습 환경 : 페도라 14

1 부팅 이미지 선택

페도라 시스템을 최초 부팅하면 다음과 같은 화면을 확인할 수 있다. GRUB 화면에서 'e(edit)'를 누른다.

이런 GRUB 화면을 확인할 수 없는 독자는 부팅이 시작되면 바로 Esc를 누른다.



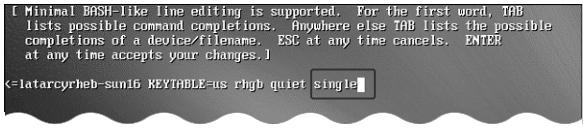
[그림 4-52] GRUB 시작 화면

다음과 같은 메뉴로 바뀌면 kernel 항목을 선택한 후 다시 'e'를 누른다.

[그림 4-53] 부트 이미지 선택

2 싱글모드 부팅 설정

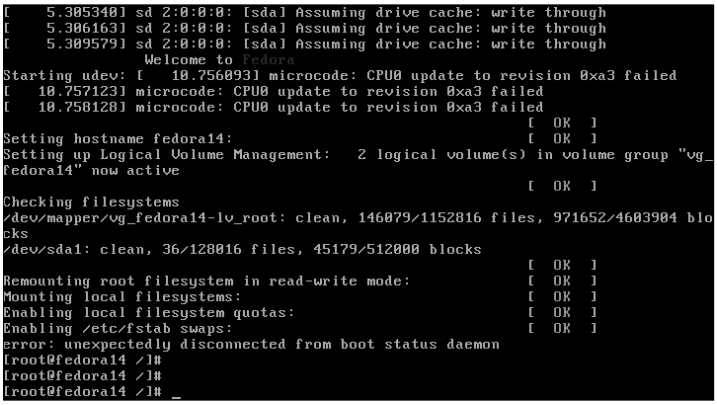
다음과 같은 셸이 확인되면 설정을 변경하지 말고 마지막에 'single' 이란 단어만 추가한 뒤 Enter를 누른다. 그리고 'b(boot)'를 누른다.



[그림 4-54] 싱글 모드 부팅 설정

3 싱글모드 부팅

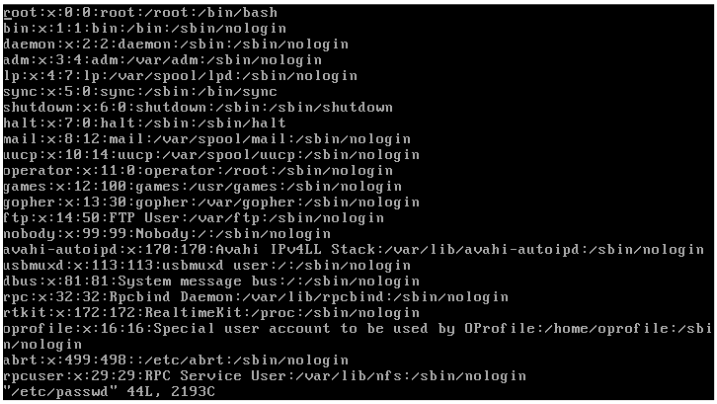
부팅 과정이 진행되면 다음과 같이 계정과 패스워드를 묻는 과정 없이 셸을 획득할 수 있다.



[그림 4-55] 싱글모드 셸 획득

4 패스워드 파일 수정

이제 패스워드 파일을 수정해보자. 패스워드 파일의 shadow에 암호화되어 있음을 알리는 'x' 표시를 지우고, 저장한 후 재부팅한다. shadow에 암호화된 패스워드가 있더라도 'x' 표시를 지우면 이를 참조하지 않는다. 또한 로그인할 때 계정에 root만을 입력하면 패스워드를 묻지 않는다.



[그림 4-56] 패스워드 파일 편집

요약

1 크래킹되기 쉬운 패스워드

- 길이가 너무 짧거나 널(Null)인 패스워드
- 사전에 나오는 단어나 이들의 조합 패스워드
- 키보드 자판을 일렬순으로 나열한 패스워드
- 사용자 계정 정보에서 유추 가능한 단어들로 된 패스워드

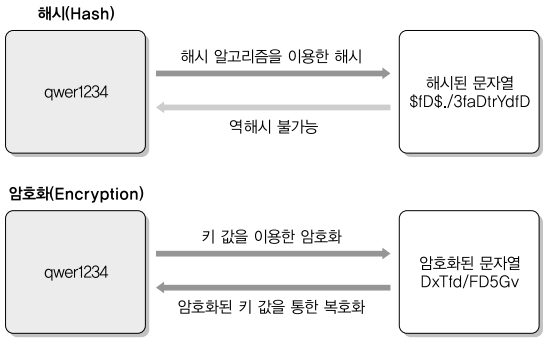
2 크래킹되기 어려운 패스워드

기억하기는 쉽지만 크래킹하기는 어려운 패스워드다. 숫자와 특수문자를 조합하면 아주 훌륭한 패스워드를 만들 수 있다.

3 해시와 암호화 알고리즘의 차이

❶ **해시** : 임의의 데이터로부터 일종의 짧은 '전자 지문'을 만들어 내는 방법이다. 해시 함수는 데이터를 자르고 치환하거나 위치를 바꾸는 등의 방법을 사용해 결과를 만들어내며, 이 결과를 흔히 해시 값(hash value)이라 한다. 해시 값이 다르면 그 해시 값에 대한 원래 데이터도 달라야 하며, 해시 값으로부터 원래의 데이터를 구하는 것은 불가능해야 한다. MD1~5, SHA 등이 있다.

❷ **암호화(Encryption)** : 특별한 지식을 소유한 사람을 제외하고는 누구나 읽어볼 수 없도록 알고리즘을 이용하여 데이터를 전달하는 것이다. 이러한 과정을 통해 암호화된 정보를 생성한다. 암호화된 정보는 다시 복호화(Decryption)되어 읽을 수 있다. DES, AES(Rijndael), SEED, RSA 등이 있다.



4 Salt

똑같은 해시 결과나 암호문은 같은 결과만으로도 패스워드를 노출시키는 약점이 있다. Salt는 이런 상황이 막기 위해 패스워드에 추가되는 문자열 부분이다.

요약

5 사전 대입 공격(Dictionary Attack)

패스워드로 사용할 만한 것을 사전으로 만들어놓고 이를 하나씩 대입하여 패스워드 일치 여부를 확인하는 패스워드 크래킹 방법이다.

6 무작위 대입 공격(Brute Force Attack)

패스워드에 사용될 수 있는 문자열의 범위를 정하고, 그 범위 내에서 생성 가능한 모든 패스워드를 생성하여 패스워드로 입력하는 패스워크 크래킹 방법이다.

7 레인보우 테이블(Rainbow Table)을 이용한 공격

레인보우 테이블은 하나의 패스워드에서 시작해 특정한 변이 함수를 이용해 여러 변이된 형태의 패스워드를 생성한다. 그리고 각 변이된 패스워드의 해시를 고리처럼 연결하여 일정 수의 패스워드와 해시로 이루어진 체인(Chain)을 무수히 만들어 놓은 테이블이다.

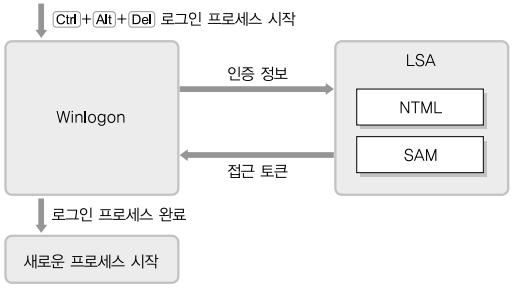
8 윈도우 인증의 구성 요소

❶ **LSA(Local Security Authority)** : 모든 계정의 로그인에 대한 검증을 하고, 시스템 자원 및 파일 등에 대한 접근 권한을 검사한다.

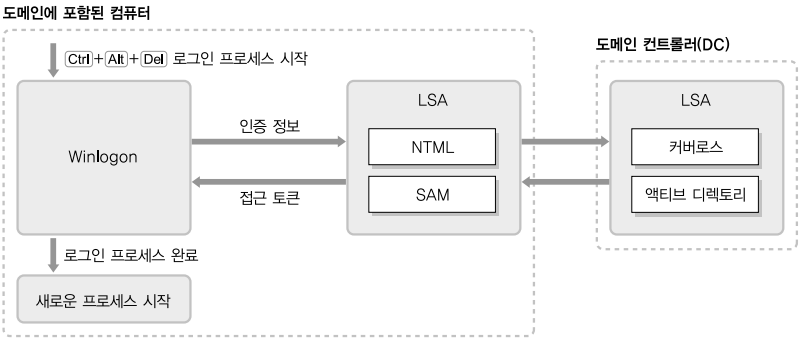
❷ **SAM(Security Account Manager)** : 사용자/그룹 계정 정보에 대한 데이터베이스를 관리한다. 사용자의 로그인 입력 정보와 SAM 데이터베이스 정보를 비교해 인증 여부를 결정하도록 해준다.

❸ **SRM(Security Reference Monitor)** : 사용자의 계정과 패스워드가 일치하는 사용자에게 고유의 SID(Security Identifier)를 부여한다. 또한 파일이나 디렉터리에 대한 접근(access)의 허용 여부를 결정하고 이에 대한 감사 메시지를 생성한다.

9 윈도우 로컬 인증과 도메인 인증



(a) 윈도우 로컬 인증

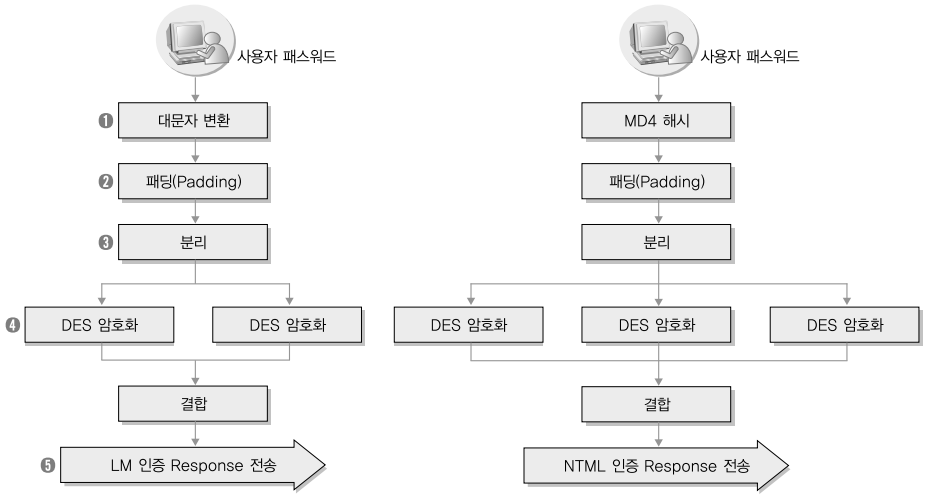


(b) 윈도우 원격 인증

10 Challenge & Response 인증

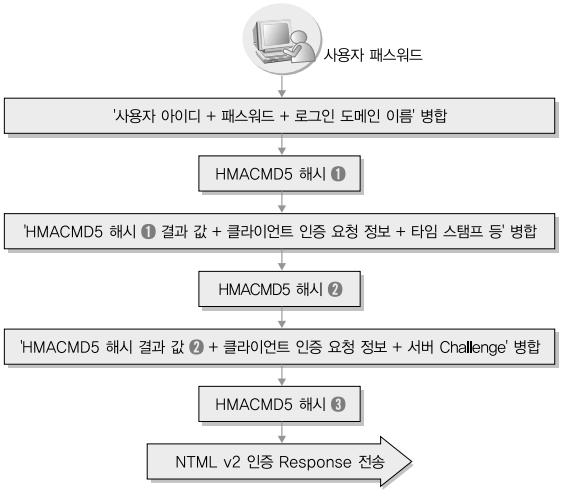
❶ 인증 요청
❷ Challenge 값 생성과 ❸ Challenge 값 전송
❹ Response 값 생성
❺ Response 값 전송 / ❻ Response 값 확인 / ❼ 인증 성공

11 LM(Lan Manager) / NTML / NTML v2 해시



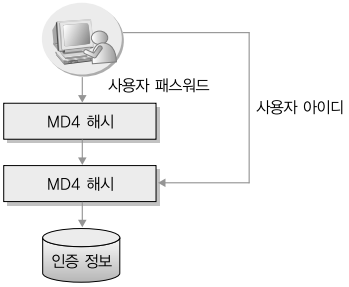
(a) LM(Lan Manager)

(b) 윈도우 로컬 인증 NTML

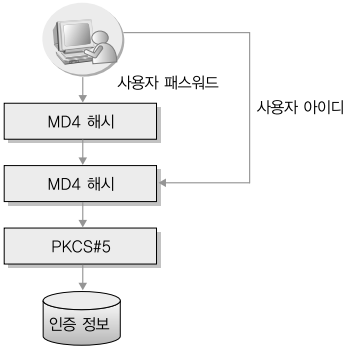


(c) NTML v2 해시

12 자격 증명(Cache Credential)



(a) 윈도우 서버 2003까지의 자격 증명



(b) 윈도우 비스타 이후 자격 증명

13 리눅스/유닉스 shadow 파일 구조

```
root : $6$LL489S99Pyh6~중략~Pazr/uKuAkuFT0/ : 14923 : 0 : 99999 : 7 : _ : _ :
 ❶                    ❷                      ❸    ❹    ❺     ❻   ❼   ❽
```

❶ 사용자 계정이다.
❷ 암호화된 사용자의 패스워드가 저장된다.

❸ 1970년 1월 1일부터 마지막으로 패스워드 변경한 날까지 계산한 값이다.
❹ 패스워드 변경하기 전에 패스워드를 사용한 기간이다.
❺ 패스워드 바꾸지 않고 최대한 사용할 수 있는 기간이다.
❻ 패스워드 최대 사용 기간에 가까워질 경우 사용자에게 미리 그 사실을 알려야 하며, 여기에 패스워드 사용기한 며칠 전에 경고를 보낼 것인지 지정한다.
❼ 계정에 대한 사용 제한을 설정하고 며칠 후에 완전히 사용 정지되게 할 것인지를 설정한다.
❽ 계정이 완전 사용 정지된 기간을 1970년 1월 1일부터 계산한 값이 기록된다.
❾ 관리자가 임의로 사용할 수 있는 부분이다.

【 연습문제 】

1 패스워드는 다음 중 무엇을 이용한 인증 방법인가?

① Something you know ② Something you have
③ Something you are ④ Somewhere you are

2 다음 중 가장 강력한 패스워드는?

① eodlf76 ② qwer7543 ③ soRj1352rt!@ ④ rjfdma1101

3 패스워드에 사용될 수 있는 문자열의 범위를 정하고, 그 범위 내에서 생성 가능한 모든 패스워드를 생성하여 패스워드로 입력해보는 패스워드 크래킹 방법은?

① Salt 크래킹 ② 사전 대입 공격
③ 레인보우 테이블을 이용한 공격 ④ 무작위 대입 공격

4 윈도우의 SAM(Security Accounts Manager) 파일이 위치하는 곳은?

① %systemroot%system/device/sam ② %systemroot%system32/config/sam
③ %systemroot%system32/etc/sam ④ %systemroot%system/etc/sam

5 다음 각 설명에 해당하는 항목을 맵핑하여라.

사용자에게 고유의 SID를 부여 •	• LSA(Local Security Authority)
모든 계정의 로그인에 대한 검증을 하고, 시스템 자원 및 파일 등에 대한 접근 권한을 검사 •	• SAM(Security Accounts Manager)
사용자/그룹 계정 정보의 데이터베이스 관리 •	• SRM(Security Reference Monitor)

6 윈도우의 인증 방법 중 도전/응답(Challenge/Response) 방식을 사용하는 NTML의 장점은?

7 다음 중 DES 암호화 알고리즘을 사용하는 알고리즘 두 개는?
① LM ② NTML ③ NTMLv2 ④ MD5

8 일반 PC 사용자들은 로그인할 때 로컬 계정을 이용한다. 하지만 회사 노트북이나 회사 PC처럼 도메인에 등록된 컴퓨터에 로그인할 때는 도메인 계정을 사용한다. 해당 컴퓨터가 도메인과 네트워크에 연결되어 있을 때는 NTML이나 커버로스를 이용해 로그인하는 것이 당연하겠지만, 네트워크가 연결되지 않을 때도 도메인에 등록된 PC에 로그인할 때도 도메인 계정을 사용해 로그인한다. 이런 것을 가능하게 하는 것이 ( )이다.

9 레인보우 테이블의 핵심 아이디어는 대용량으로 생성될 수 있는 해시 테이블을 ( )를 이용해 사용하기 충분한 작은 크기로 줄이는 것이다.

【심화문제】

10 Salt에 대해 설명하시오.

11 좋지 않은 패스워드와 좋은 패스워드에 대해 설명하시오.

12 윈도우의 LM 해시의 문제점을 패스워드 블록과 연관지어 설명하시오.

13 리눅스/유닉스 시스템의 shadow 파일의 각 부분에 대해서 설명하시오.

```
root : $1$pS/cfFID$pzmD10T5rjrx : 12364 : 0 : 99999 : 7 : _ : _ : _
 ❶              ❷                  ❸    ❹    ❺    ❻   ❼   ❽   ❾
```

14 암호화와 해시의 차이점은?

15 리눅스의 패스워드 복구 방법에 대해서 간단히 설명하시오.