



응용보안

6. 패스워드 크래킹

경기대학교 AI컴퓨터공학부 이재흥
jhlee@kyonggi.ac.kr

CONTENTS

PRESENTATION



- 패스워드 크래킹의 이해
- 윈도우 인증과 패스워드
- 리눅스/유닉스 인증과 패스워드
- 서비스 데몬 패스워드
- 실습 FTZ Level 8. 리눅스 패스워드 파일 크랙



학습 목표

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



패스워드 크래킹의 이해



패스워드 크래킹의 이해

- 패스워드 관리
 - 보안 관리자의 첫번째 방어책
- 좋지 못한 패스워드
 - 길이가 너무 짧거나 널(Null)인 패스워드
 - 사전에 나오는 단어나 이들의 조합으로 이루어진 패스워드
 - 키보드에서 일렬로 키를 나열한 패스워드
 - 사용자 계정 정보에서 유추 가능한 단어들로 구성된 패스워드
- 좋은 패스워드
 - 기억하기는 쉽고 크래킹하기 어려운 패스워드



해시와 암호화

- 패스워드를 숨기는 기본적인 두 가지 방법
 - 해시(Hash)
 - 임의의 데이터에서 일종의 짧은 ‘전자 지문’을 만들어 내는 방법
 - 암호화(Encryption)
 - 특별한 알고리즘을 이용해 데이터를 전달하는 것

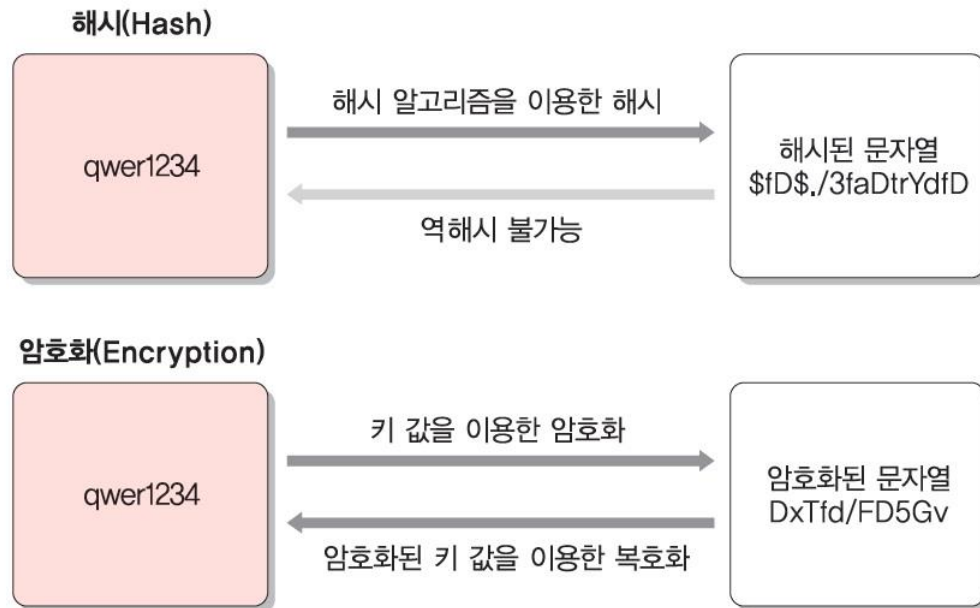


그림 4-1 해시와 암호화 알고리즘 차이점



해시와 암호화

- 암호화 예
 - 시저 암호 (Caesar cipher)
 - 로마 시대에 사용한 아주 오래된 암호화 방식
 - 세 글자씩 **알파벳을 밀어내 대응되는 글자로 치환**

세 글자씩 밀어낸 대응 글자로 치환하기

a	b	c	d	e	f	g	h	i	j	k	l	...
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
x	y	z	a	b	c	d	e	f	g	h	i	...

그림 4-2 밀어내기식 치환 방법

abcde fghij klmno pqrst uvwxy z

xyzab cdefg hijkl mnopq rstuv w

Wish to be free from myself → tfpe ql yb cobb colj jvpbic

그림 4-3 밀어내기식 치환 방법 예



해시와 암호화

- 해시 예

- 123456789 vs. 123486789

- 두 수를 가운데를 기준으로 둘로 나누고, 큰 수를 작은 수로 나눔
 - 그림 4-4 참조
 - 앞에서 6자리 숫자를 버리고 나머지 값을 해시의 결과 값으로 계산
 - 123456789의 해시 값은 2677861
 - 123486789의 해시 값은 4569155
 - 두 해시 값만으로 해시 전의 원래 수를 알아내는 것은 불가능에 가까움
 - 로직을 알더라도 버려진 1.81838과 1.81882를 알아낼 수 없기 때문
 - 해시는 로직을 알고 있을 경우 해시의 결과 값을 구하기 쉽지만, 그 해시의 결과 값으로 해시를 생성하기 전의 원래 값은 알기 어려움
 - 또한 값이 아주 조금만 달라도 해시의 결과 값은 무척 상이하게 생성됨

$$\frac{12345}{6789} = 1.81838 \text{ 2677861 } \dots$$
$$\frac{12348}{6789} = 1.81882 \text{ 4569155 } \dots$$

그림 4-4 나눗셈을 이용한 해시 값 획득



- 패스워드 저장
 - 해시나 암호화 알고리즘으로 변경해서 저장
 - 다른 사용자여도 같은 패스워드는 같은 해시 값, 같은 암호문으로 저장
- Salt
 - 이런 상황을 막기 위해 패스워드 해시와 암호화에 사용하는 첨가물의 일종
 - 운영체제별로 다양한 알고리즘 존재
 - 예: 'eodlf@!11'을 MD5 알고리즘으로 해시

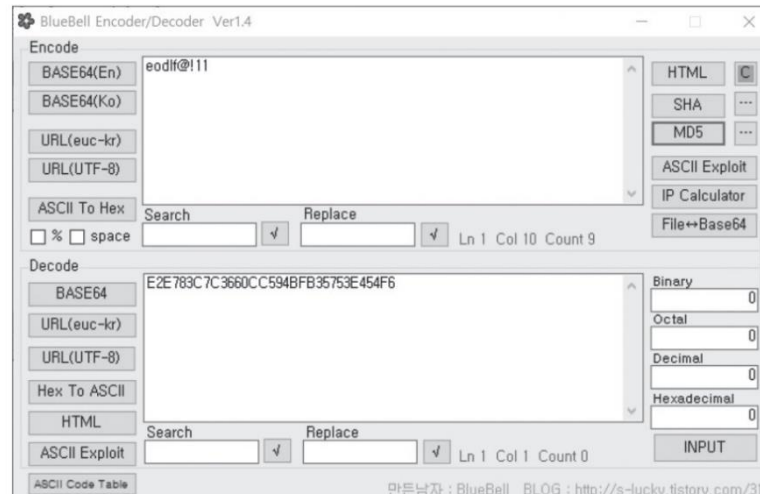


그림 4-5 eodlf@!11의 MD5 해시 값 생성



- Salt

표 4-1 Salt와 패스워드를 조합한 값에서 MD5 해시 값을 생성하는 예

계정	Salt	패스워드	Salt + 패스워드를 MD5로 해시한 결과 값
root	a2	eodlf@!11	707AF746F7B972C8FC5EC12142FAFB1C
wishfree	4F	eodlf@!11	818B2DDCBBD103987216563EDA219D3F

- Salt와 패스워드를 합한 ‘a2eodlf@!11’과 ‘4Feodlf@!11’을 각각 해시한 결과 값은 전혀 다르다는 것을 알 수 있음
- 패스워드 파일에 저장할 때는 간단한 인코딩을 통해 해시 결과 값 앞이나 뒤에 Salt를 붙임

표 4-2 해시된 패스워드 값에 Salt 정보를 붙인 예

계정	Salt + MD5
root	a2707AF746F7B972C8FC5EC12142FAFB1C
wishfree	4F818B2DDCBBD103987216563EDA219D3F

- 적용된 Salt는 똑같은 패스워드를 숨길 뿐만 아니라 적용 수준에 따라 패스워드 크래킹을 매우 어렵게 만드는 요소가 됨



패스워드 크래킹 방법 이해

- 사전대입공격(Dictionary Attack)
 - 사용자가 설정하는 대부분의 패스워드에는 특정한 패턴이 있다는 것에 착안
 - 패스워드로 사용할 만한 것을 사전으로 만들어 놓고 하나씩 대입하여 패스워드 일치 여부를 확인하는 것
- 무작위 대입 공격(Brute Force Attack)
 - 패스워드에 사용할 수 있는 문자열 범위를 정하고, 그 범위 안에서 생성 가능한 모든 패스워드를 입력해 보는 것
 - 패스워드가 그다지 복잡하지 않거나 짧을 경우 단시간에 크래킹 가능



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 1980년 마틴 헬만(Martin Hellman)에 의해 소개
 - 2000년대에 들어 윈도우 LM 패스워드를 몇 분 만에 크래킹하면서 유명해짐
 - 패스워드 하나에서 시작하여 특정한 변이 함수로 변이된 형태의 여러 패스워드를 생성
 - 변이된 각 패스워드의 해시를 고리처럼 연결하여 일정한 수의 패스워드와 해시로 된 체인(Chain)을 무수히 만들어 놓은 테이블



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 레인보우 테이블의 가장 기본적인 개념
 - 패스워드 별로 해시 값을 미리 생성
 - 크래킹하려는 해시 값을 테이블에서 검색하여 거꾸로 원래 패스워드를 찾음
 - 예
 - 패스워드가 '12qw'이고, 해시 값이 '123452323242'라고 할 때, 아래 해시 테이블에서 '123452323242'를 검색하여 거꾸로 '12qw'를 찾음

표 4-3 미리 계산된 해시 테이블

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



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 레인보우 테이블의 또 다른 핵심 아이디어
 - 대용량으로 생성할 수 있는 해시 테이블을 R(Reduction) 함수로 사용하기 충분한 작은 크기로 줄이는 것 (몇 십 GB 정도...)
 - 예
 - 패스워드가 6자리 숫자로 이루어진 ‘234342’
 - MD5 해시 값은 ‘C1F2FE224298D6E39EBA607D46F3D9CC’
 - R 함수는 이 해시 값에서 또 다른 형태의 무작위 패스워드를 추출
 - R 함수가 MD5 해시 값 중 앞에서 여섯 개 숫자만 뽑아낸다고 가정
 - R(C1F2FE224298D6E39EBA607D46F3D9CC)은 ‘122242’



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 예
 - 레인보우 테이블을 생성하기 위해 MD5 해시 생성과 R 함수 동작을 아래 그림과 같이 반복

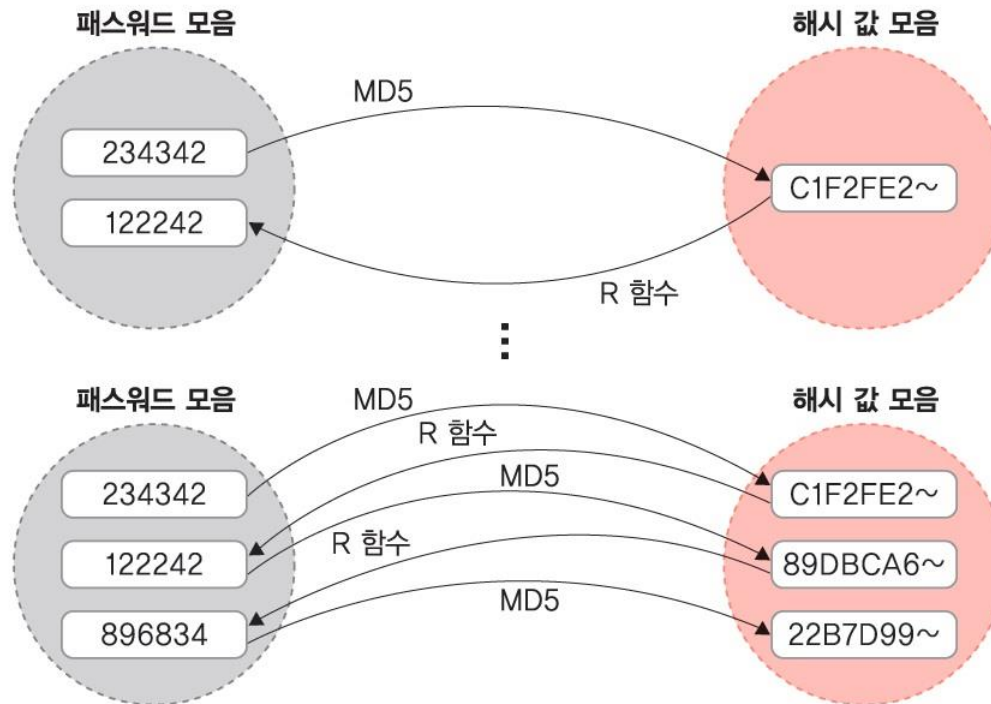


그림 4-6 MD5 해시와 R 함수의 동작



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 다양한 패스워드를 대상으로 이를 반복하여 레인보우 테이블 생성

표 4-4 '234342'에서 MD5 해시 값과 R 함수를 반복 실행한 결과

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

표 4-5 '346343'에서 MD5 해시 값과 R 함수를 반복 실행한 결과

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

표 4-6 '898232'에서 MD5 해시 값과 R 함수를 반복 실행한 결과

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

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

패스워드	MD5 해시 값
234342	22B7D9922C994737D0D9DFCCF6B415B6
346343	86AB6B3355F33F7CD62658FDDA5AF7D6
898232	E2038DD2A8315D9BF7F72AE5C07530F8



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 패스워드 해시 값 ‘570727EE4270E0C1A4D8FBB741926DB8’ 크래킹 과정
 - ① 레인보우 테이블에 크래킹하려는 해시 값과 같은 MD5 해시 값이 있는지 확인
 - [표 4-7]에는 해당하는 해시 값이 없음
 - ② 레인보우 테이블에 크래킹하려는 해시 값이 없으면 크래킹할 해시 값에 R 함수를 적용하여 패스워드를 구하고 다시 해시 값을 구함
 - ‘570727EE4270E0C1A4D8FBB741926DB8’에 R함수를 적용하여 패스워드 ‘570727’을 구하고, ‘570727’의 해시 값을 구하면 ‘86AB6B3355F33F7CD62658FDDA5AF7D6’임
 - ③ ②에서 구한 해시 값 ‘86AB6B3355F33F7CD62658FDDA5AF7D6’이 레인보우 테이블에 있는지 다시 확인
 - [표4-7]에서 생성한 레인보우 테이블에 해당 값이 있음



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 패스워드 해시 값 ‘570727EE4270E0C1A4D8FBB741926DB8’ 크래킹 과정
 - ④ 레인보우 테이블에서 확인한 해시 값을 발견하면, 그 해시 값에 해당하는 최초 패스워드를 구함 (값이 없다면 같은 해시 값이 나올 때까지 ②와 ③을 해시 테이블을 생성할 때 설정한 체인 개수만큼 반복함)
 - [표 4-7]에서 ‘86AB6B3355F33F7CD62658FDDA5AF7D6’에 해당하는 패스워드는 ‘346343’임
 - ⑤ 확인한 최초 패스워드에서 다시 패스워드와 일치하는 해시 값이 나올 때까지 MD5 해시와 R함수를 반복 수행함 (해당 해시 값이 확인되면 찾는 패스워드는 해당 해시 값을 생성한 문자열이 될 것임)



패스워드 크래킹 방법 이해

- 레인보우 테이블(Rainbow Table)을 이용한 공격
 - 실제 레인보우 테이블
 - 체인이 2,000개 이상 있음
 - 이러한 체인을 사용하는 레인보우 테이블에서 해시 값을 1만 개 저장하고 있다면, 레인보우 테이블에서 확인할 수 있는 패스워드 종류는 200만($= 2,000 \times 10,000$)개가 됨



윈도우 인증과 패스워드



윈도우 인증의 구성 요소

- 윈도우 인증 과정에서 가장 중요한 구성 요소
 - LSA(Local Security Authority)
 - SAM(Security Account Manager)
 - SRM(Security Reference Monitor)

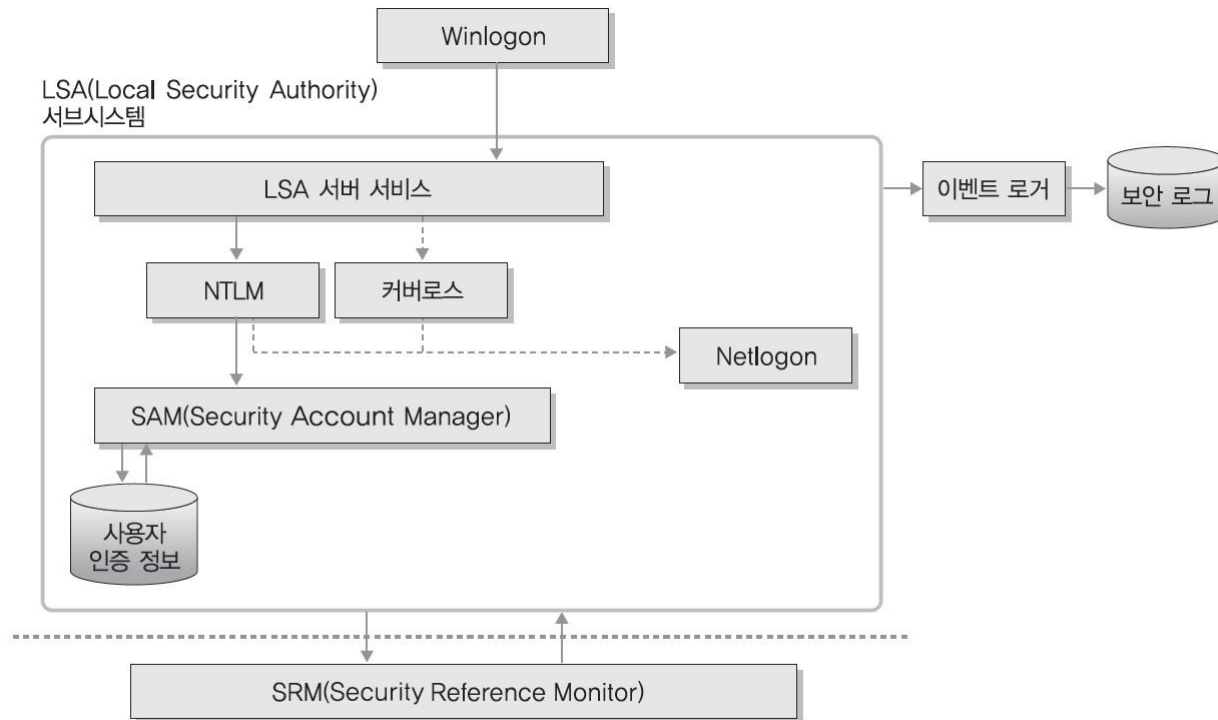


그림 4-7 윈도우 인증 구조



윈도우 인증의 구성 요소

- LSA
 - 모든 계정의 로그인을 검증하고, 시스템 자원 및 파일 등에 접근하는 권한을 검사
 - 로컬, 원격 모두에 해당
 - 이름과 SID를 매칭하며, SRM이 생성한 감사 로그를 기록하는 역할도 수행
- SAM
 - 사용자/그룹 계정 정보에 대한 데이터베이스를 관리
 - 사용자의 로그인 입력 정보와 SAM 데이터베이스 정보를 비교하여 인증 여부 결정
 - 윈도우의 SAM 파일은 다음 경로에 위치
 - %systemroot%는 윈도우가 설치된 폴더로, C:\Winnt 또는 C:\Windows가 일반적

%systemroot%/system32/config/sam [빈 박스]



윈도우 인증의 구성 요소

- SRM
 - SAM이 사용자 계정과 패스워드 일치 여부를 확인하여 알리면, 사용자에게 고유의 SID(Security Identifier)를 부여
 - SID를 기반으로 파일이나 디렉터리에 접근(access)을 허용할지 여부를 결정하고, 이에 대한 감사 메시지를 생성



로컬 인증과 도메인 인증

• 로컬인증

- ① 자동 또는 Ctrl+Alt+Delete 키를 입력하여 Winlogon 화면 생성
- ② ID와 패스워드 입력
- ③ LSA 서브 시스템이 인증 정보를 받아 NTLM 모듈에 ID와 패스워드를 넘겨줌
- ④ 이를 다시 SAM이 받아 확인하고, 로그인을 허용

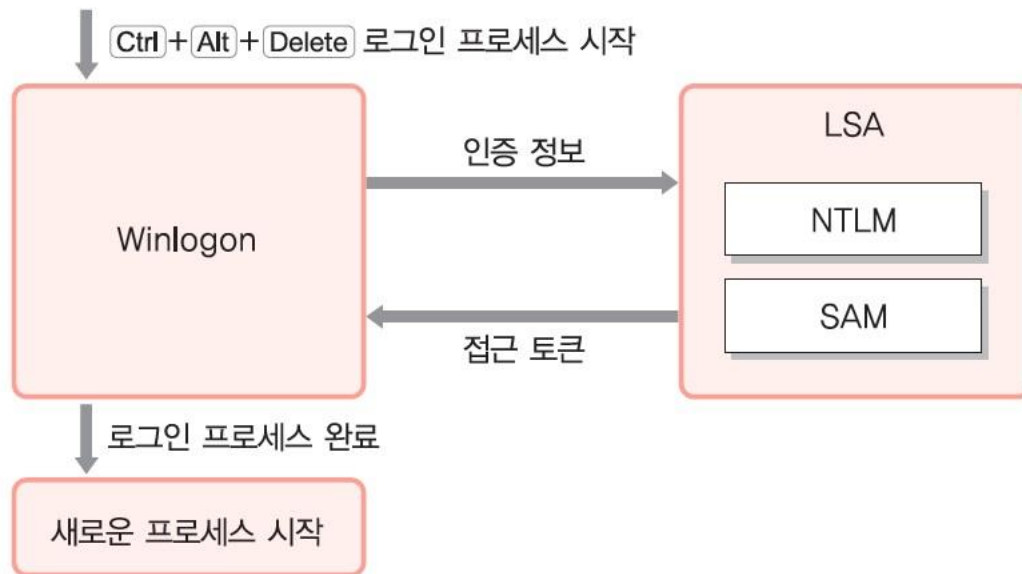


그림 4-8 윈도우 로컬 인증



로컬 인증과 도메인 인증

- 도메인 인증 (①, ②는 로컬 인증과 같음)
 - ③ LSA 서브 시스템이 인증 정보를 받아 해당 인증 정보가 로컬 인증용인지 도메인 인증용인지 확인하고 도메인 인증용이면 커버로스(Kerberos) 프로토콜을 이용하여 도메인 컨트롤러에 인증을 요청
 - 도메인 인증에서는 기본적으로 풀 도메인 이름(FQDN: Full Qualified Domain Name)과 커버로스 프로토콜을 이용하나, IP로 접근을 시도할 때는 NTLM을 사용

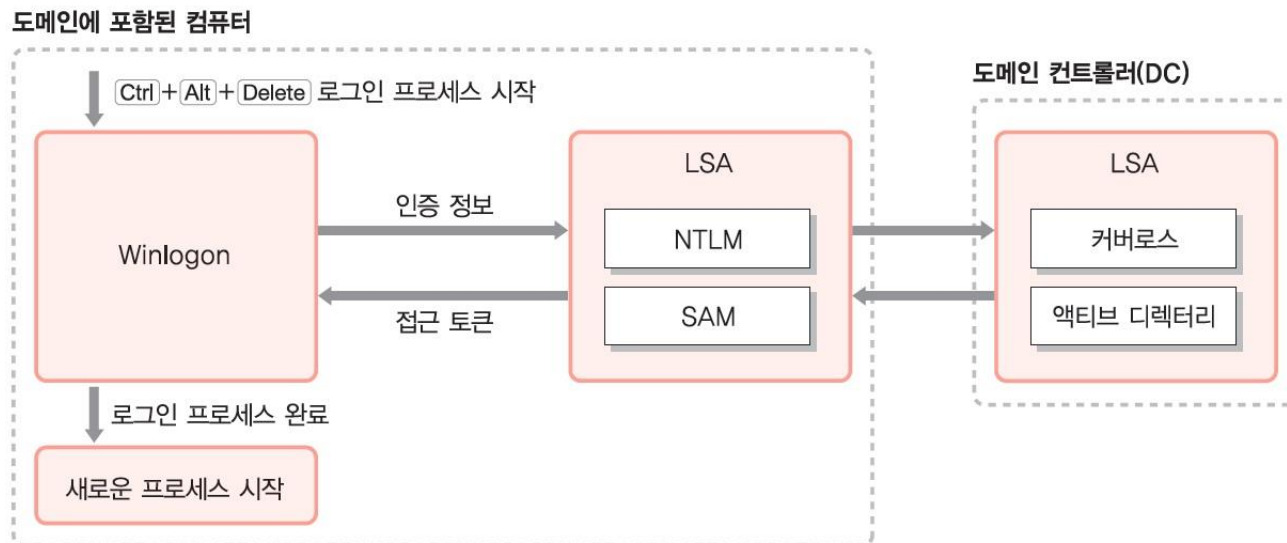


그림 4-9 윈도우 원격 인증

인증 구조

- Challenge & Response 인증

- 패스워드 값을 인증 서버와 동일한 인증 주체에 전달하여 올바른 패스워드라고 증명하는 가장 직관적이고 쉬운 방법은 패스워드 값을 직접 전달하는 것
 - 텔넷, FTP, 웹포털 사이트
 - 패스워드 노출 또는 패스워드 재사용 공격에 취약
- 운영체제 인증처럼 높은 수준의 인증이 필요한 경우
 - Challenge & Response 방식 사용

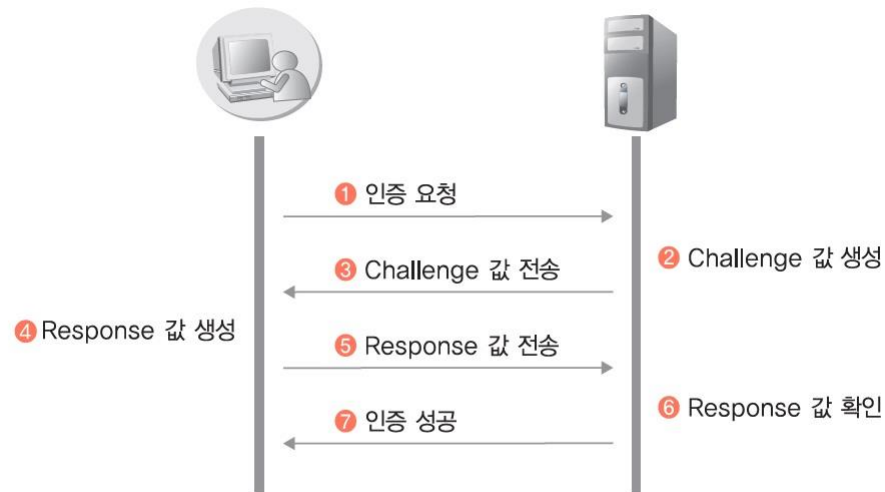


그림 4-10 Challenge & Response 인증



인증 구조

- Challenge & Response 인증 단계

- ① 인증 요청

- 인증을 수행하고자 하는 주체가 인증 서버에 인증을 요청

- ② Challenge 값 생성 / ③ Challenge 값 전송

- 인증을 요청 받은 인증 서버는 문자열 등의 값을 특정 규칙에 따르거나 랜덤하게 생성하여 인증 요구자에게 전달

- ④ Response 값 생성

- 인증 요구자는 서버에서 전달받은 Challenge 값과 자신이 입력한 패스워드 정보 등을 이용하여 서버에 보낼 Response 값을 생성

- ⑤ Response 값 전송 / ⑥ Response 값 확인 / ⑦ 인증 성공

- 인증 요구자는 생성한 Response 값을 인증 서버에 전달하고, 인증 서버는 이 Response 값을 확인하여 인증 요구자가 적절한 패스워드를 소유하고 있는지 확인
 - 확인된 Response가 적절하면 인증 성공 여부를 인증 요구자에게 알림



LM & NTLM

- LM(Lan Manager) 해시
 - 1980년대에 만든 알고리즘
 - 원래는 IBM의 OS/2에서 사용
 - MS에서 1993년에 만든 윈도우 NT에 탑재하기 시작
 - 윈도우 2000, XP의 기본 알고리즘으로 사용
 - 구조적으로 취약

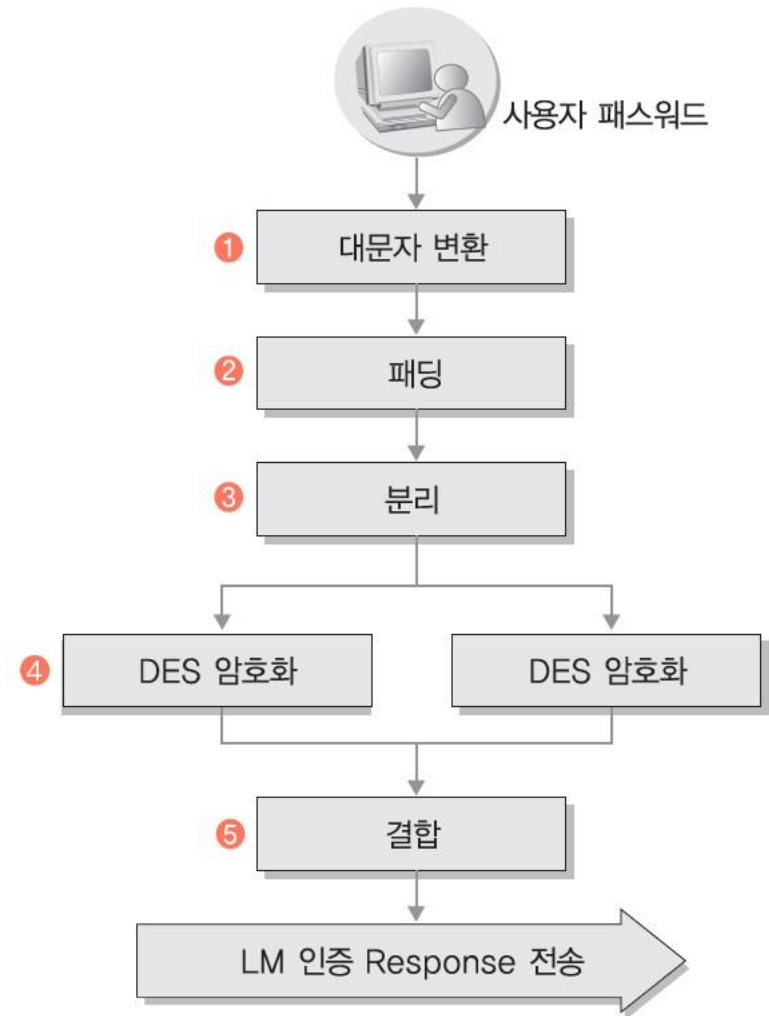


그림 4-11 LM 해시 알고리즘



LM & NTLM

- LM(Lan Manager) 해시

- ① 대문자 변환

- 사용자가 패스워드를 입력하면 모두 대문자로 바꿈
 - 따라서 대소문자를 구분하지 않음

- ② 패딩(Padding)

- 기본적으로 14 글자를 패스워드 하나로 인식
 - 14 글자가 되지 않는 패스워드는 뒤에 0을 붙여 14자리로 만듦

- ③ 분리

- 패스워드 길이에 관계없이 8바이트가 블록 하나를 형성
 - 이 중 1바이트는 패스워드 블록 정보를 담고 있어 실질적으로 패스워드 문자열은 7바이트, 즉 문자 7개로 구성
 - 패스워드가 qwer1234라면 8 글자이므로 패스워드 블록을 두 개 형성함



LM & NTLM

- LM(Lan Manager) 해시



그림 4-12 패스워드가 여덟 글자일 때 패스워드 블록

④ DES 암호화

- 블록 두 개로 분리된 패스워드는 각각 “KGS!@#%” 문자열을 암호화하는데 키로 사용

⑤ 결합

- 키 두 개로 “KGS!@#%”를 각각 암호화한 결과 값을 합하여 SAM 파일에 저장



LM & NTLM

- LM(Lan Manager) 해시
 - 앞의 그림에서 확인한 바와 같이 패스워드 블록 하나는 별도로 운영
 - 패스워드를 아무리 길게 설정하더라도 패스워드 블록 하나인 일곱 글자를 크래킹하는 데 필요한 노력으로 전부를 풀 수 있음
 - 예
 - qwer1234는 qwer123과 4로 나뉨
 - qwer123은 쉽게 크래킹하지 못할 수도 있으나, 4는 크래킹하는데 몇 초도 걸리지 않음
 - 윈도우는 문자열이 일곱 개인 패스워드 블록을 이용하여 패스워드를 구현하기 때문에 일곱 글자 패스워드의 강도와 여덟 글자 패스워드의 강도가 사실상 같음
 - 윈도우에서 14자 패스워드 크래킹이 7자 패스워드 두 개 크래킹과 같은 노력 필요
 - 14자 패스워드의 보안 강도는 7자 패스워드보다 겨우 2배 더 셈



LM & NTLM

- NTLM 해시
 - LM 해시에 MD4 해시가 추가된 것 외에는 큰 차이가 없음

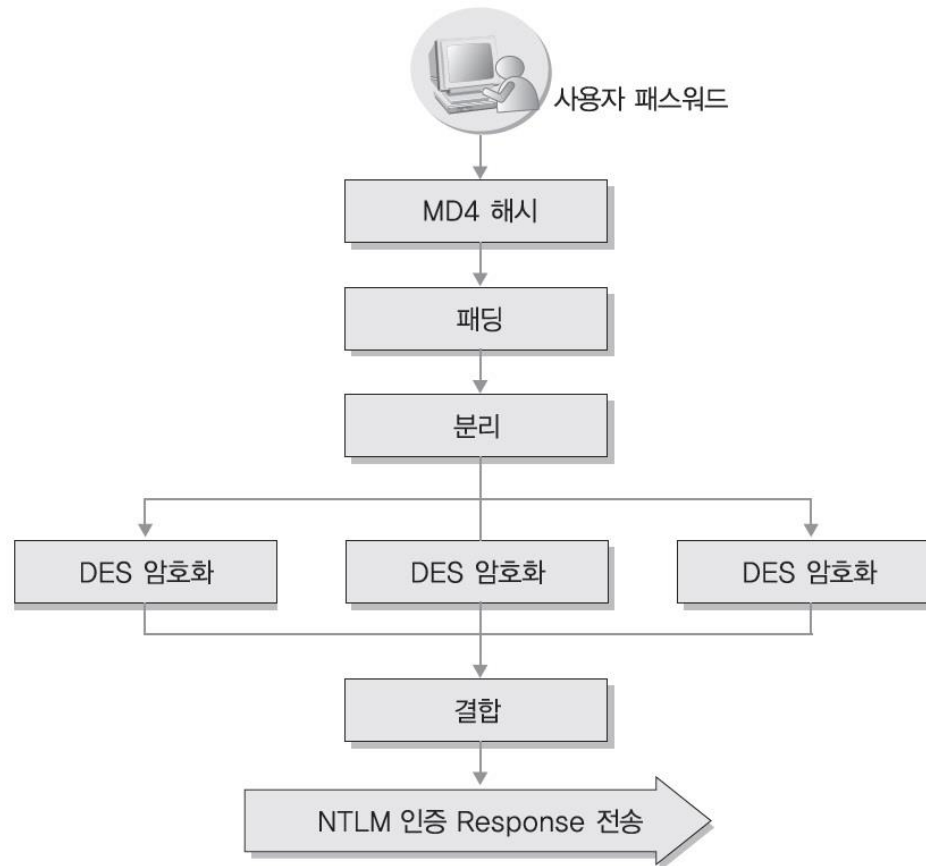


그림 4-13 NTLM 해시 알고리즘

LM & NTLM

- NTLMv2 해시

- NTLMv2는 윈도우 비스타 이후의 윈도우 시스템에서 기본 인증 프로토콜로 사용

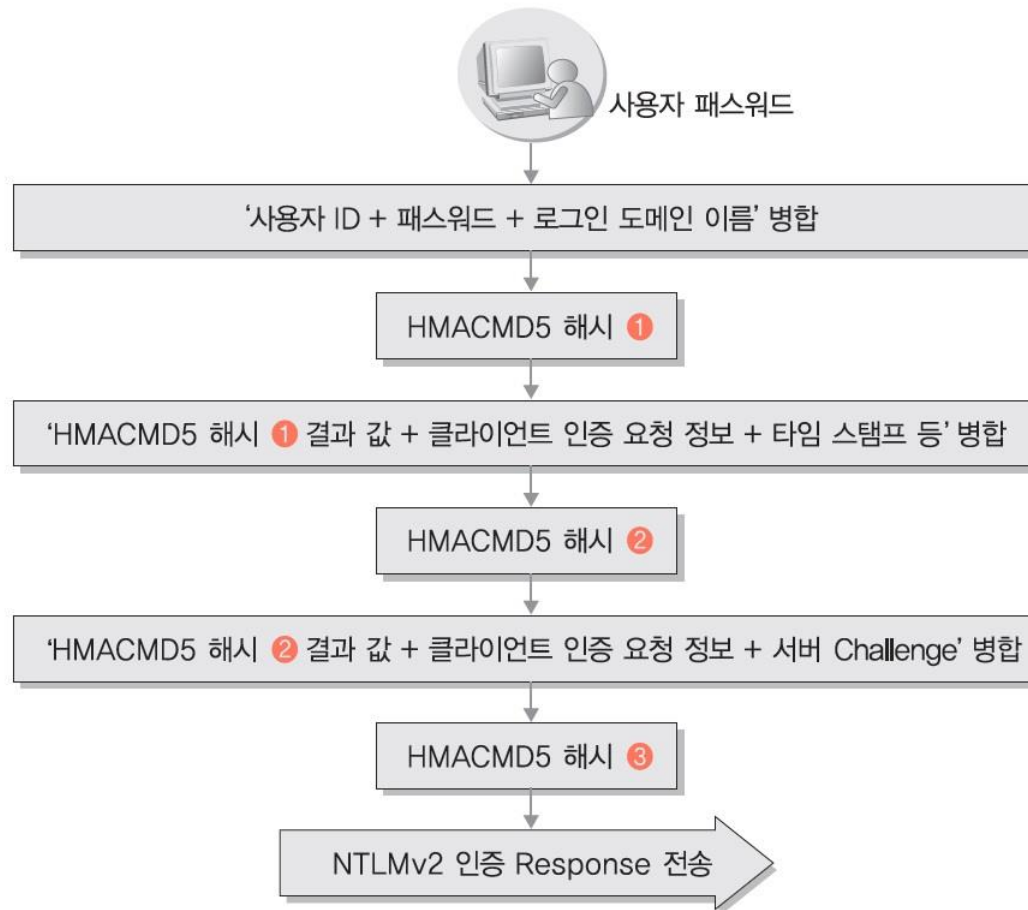


그림 4-14 NTLMv2 해시 알고리즘

자격 증명

- 자격 증명(Cache Credential)

- 일반 PC 사용자는 로그인할 때 로컬 계정 이용
- 회사 PC처럼 도메인에 등록된 컴퓨터에 로그인할 때는 도메인 계정을 이용
- 해당 컴퓨터가 도메인과 네트워크에 연결되어 있는 경우
 - NTLM이나 커버로스를 이용하여 로그인
- 네트워크에 연결되어 있지 않는데 도메인에 등록된 PC에 로그인할 때
 - 도메인 계정으로 로그인 (자격 증명 때문에 가능)

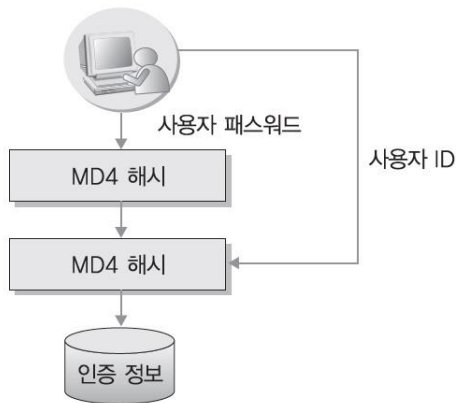


그림 4-15 윈도우 서버 2003까지 자격 증명 생성

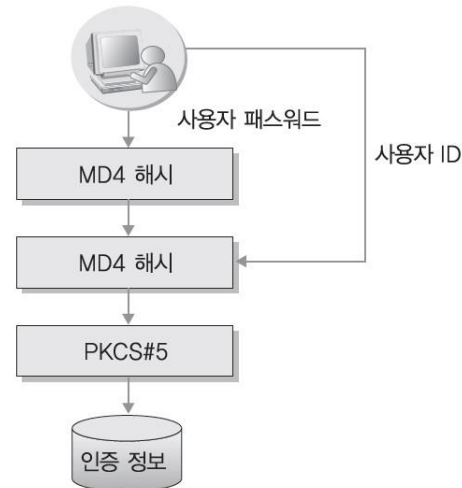


그림 4-16 윈도우 비스타 이후 자격 증명 생성



윈도우 패스워드 크래킹하기

- 윈도우 패스워드 크래킹하기
 - 윈도우 서버 2016에서 NTLM 해시를 획득하여 일반적인 무작위 대입 공격 (Brute Force 패스워드 크래킹)과 레인보우 패스워드 크래킹 수행
 - 실습 환경 구성
 - 실습 환경
 - 윈도우 서버 2016
 - 필요 프로그램
 - L0phtCrack 7 (<https://l0phtcrack.gitlab.io/>)
 - Ophcrack (<https://ophcrack.sourceforge.io/>)
 - 레인보우 테이블 (Vista Special)
 - Cain & Abel (WinRTGen)



윈도우 패스워드 크래킹하기

1. 테스트 계정 생성 및 패스워드 설정하기

- 패스워드 크래킹에 사용할 계정 몇 개를 생성
- 다양한 난이도의 패스워드 크래킹을 시도하기 위해, 숫자로만 된 패스워드, 짧은 패스워드, 영문자와 숫자로만 된 패스워드, 특수 문자 등을 포함한 패스워드 등을 설정
- 윈도우 서버 2016에서는 기본적으로 암호의 복잡도가 일정 수준 이상이어야 하기 때문에 [제어판] - [관리 도구] - [로컬 보안 정책]의 계정 정책 > 암호 정책을 선택한 후 ‘암호는 복잡성을 만족해야 함’을 ‘사용 안 함’으로 설정

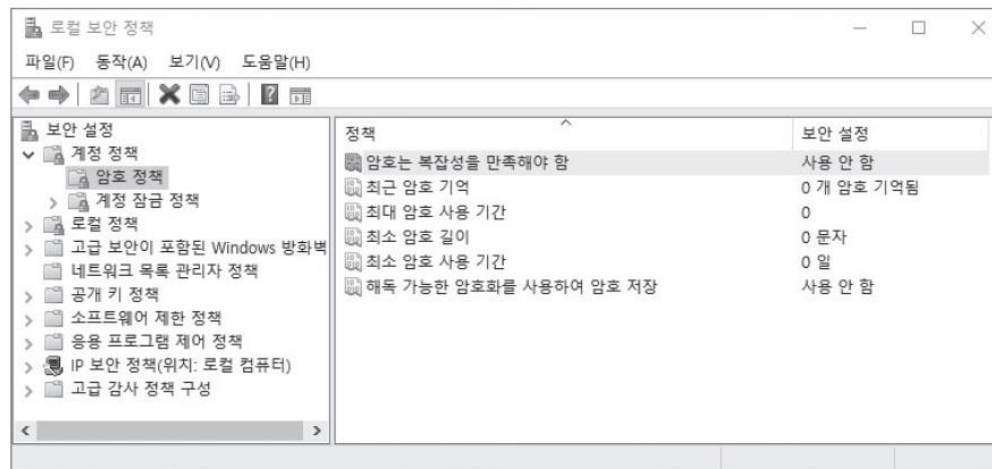


그림 4-17 암호 정책 변경



윈도우 패스워드 크래킹하기

1. 테스트 계정 생성 및 패스워드 설정하기

- 계정과 패스워드는 [컴퓨터 관리]의 로컬 사용자 그룹 > 사용자에서 설정할 수 있음



그림 4-18 테스트 계정 생성과 패스워드 설정

표 4-8 테스트 계정별로 설정한 패스워드

계정	설정한 패스워드
test01	1234
test02	qwer1234
test03	QwEr1234
test04	QwEr!2#4
test05	!@2q4
test06	2b567gho

윈도우 패스워드 크래킹하기

2. NTLM 해시 추출 및 패스워드 크랙하기

- 윈도우의 NTLM 해시는 C:\Windows\System32\config\SAM 파일과 경로가 동일한 SYSTEM 파일에서 추출
- 주로 pwdump, creddump, fgdump 같은 툴을 사용할 수 있음
- 여기에서는 15일간 무료로 사용할 수 있는 l0phtcrack 7을 사용
- 웹 사이트에서 다운로드하여 설치한 후 실행하고 “Proceed with Trial”을 선택
- 다음 창이 나타나면 <Password Auditing Wizard> 버튼을 누름



그림 4-19 l0phtcrack 7 실행 옵션 선택



윈도우 패스워드 크래킹하기

2. NTLM 해시 추출 및 패스워드 크랙하기

– 몇 가지 설정 사항

- [Choose Target System]: 패스워드 크랙을 수행할 운영체제 종류 선택 (윈도우 선택)
- [Windows Import]: 패스워드 목록을 로컬 시스템에서 가져올지, 원격 시스템에서 가져올지, pwdump, fgdump 결과를 사용할지 묻음 (로컬 선택)
- [Windows Import From Local Machine]: 어떤 계정의 권한으로 NTLM 해시를 추출할지 묻음 (Administrator 계정으로 로그인한 상태이므로 'Use Logged-In User Credential'을 그대로 둠)
- [Choose Audit Type]: 어떤 수준의 패스워드 크랙을 수행할지 묻음 (여기에서는 간단히 동작을 확인할 예정이므로 'Quick Password Audit'을 선택)
- [Reporting Option]: 패스워드 크랙 결과를 어떤 형태로 보여 주고 저장할지 묻음 (기본 설정을 사용)
- [Job Scheduling]: 바로 실행할지, 작업을 예약할지 확인 (그대로 실행)
- [Summary]: 설정한 내용을 확인

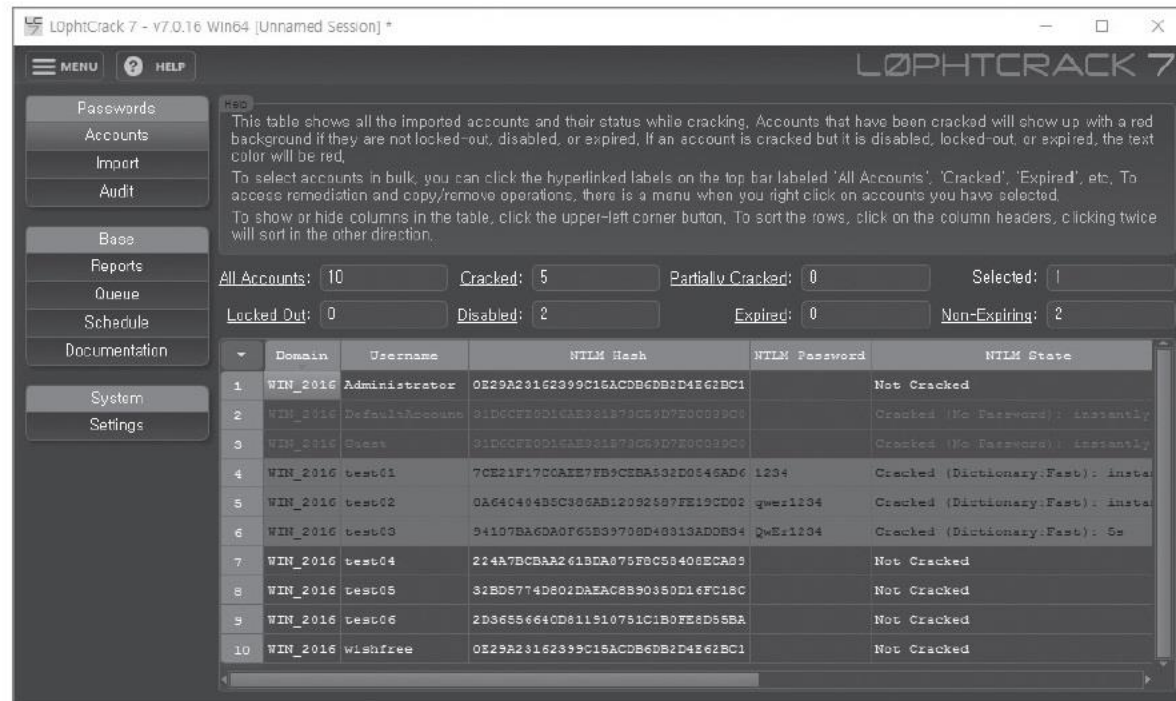


윈도우 패스워드 크래킹하기

2. NTLM 해시 추출 및 패스워드 크랙하기

- 실행 결과 확인

- test01, test02, test03 계정의 패스워드가 크랙된 것을 확인할 수 있음



The screenshot shows the L0phtCrack 7 interface. On the left is a sidebar with menu items: Passwords, Accounts, Import, Audit, Base, Reports, Queue, Schedule, Documentation, System, and Settings. The main window displays a table of accounts and their status. Above the table, there are summary statistics: All Accounts: 10, Cracked: 5, Partially Cracked: 0, Selected: 1, Locked Out: 0, Disabled: 2, Expired: 0, Non-Expiring: 2. The table has columns for ID, Domain, Username, NTLM Hash, NTLM Password, and NTLM State. The first three rows (Administrator, DefaultAccount, Guest) are marked as 'Not Cracked'. Rows 4, 5, and 6 (test01, test02, test03) are marked as 'Cracked (Dictionary.Fast): instantly'. Rows 7 through 10 (test04 through wishfree) are marked as 'Not Cracked'.

ID	Domain	Username	NTLM Hash	NTLM Password	NTLM State
1	WIN_2016	Administrator	0E29A23162399C15ACDB6DB2D4E62BC1		Not Cracked
2	WIN_2016	DefaultAccount	31D6CFF2D1C3E331373CE9D7E0C039C0		Cracked (No Password): instantly
3	WIN_2016	Guest	31D6CFF2D1C3E331373CE9D7E0C039C0		Cracked (No Password): instantly
4	WIN_2016	test01	7CE21F17CCAIE7FB9CEBA532D0546AD6	1234	Cracked (Dictionary.Fast): instantly
5	WIN_2016	test02	0A640494B5C386AD12032587FE19CD02	qwer1234	Cracked (Dictionary.Fast): instantly
6	WIN_2016	test03	94107BA6DA0F65B93738D48313AD0B34	QwEr1234	Cracked (Dictionary.Fast): instantly
7	WIN_2016	test04	244A7BCDAA261BDA075F8C58408ECAA93		Not Cracked
8	WIN_2016	test05	32BD5774D802DAEAC8590350D16FC18C		Not Cracked
9	WIN_2016	test06	2D3655664CD811910751C1B0FE2D55BA		Not Cracked
10	WIN_2016	wishfree	0E29A23162399C15ACDB6DB2D4E62BC1		Not Cracked

그림 4-20 l0phtcrack 7 실행 결과



윈도우 패스워드 크래킹하기

3. 레인보우 테이블을 이용한 패스워드 크랙하기

- 앞에서 확인한 NTML 해시를 다른 툴에서 읽을 수 있도록 아래와 같이 pwddump 파일 형식으로 바꾸어 NTML_hash.txt 파일로 저장

NTLM_hash.txt

```
test01:1001::7CE21F17C0AEE7FB9CEBA532D0546AD6:::  
test02:1002::0A640404B5C386AB12092587FE19CD02:::  
test03:1003::94107BA6DA0F65B39708D48313ADDB34:::  
test04:1004::224A7BCBAA261BDA875F8C58408ECA89:::  
test05:1005::32BD5774D802DAEAC8B90350D16FC18C:::  
test06:1006::2D36556640D811910751C1B0FE8D55BA:::
```

- 레인보우 테이블을 이용한 패스워드 크랙은 ophcrack 툴과 같은 웹사이트에서 ‘Vista Special’이라는 레인보우 테이블로 수행

윈도우 패스워드 크래킹하기

3. 레인보우 테이블을 이용한 패스워드 크랙하기

- ‘Vista Special’ 테이블은 ophcrack 툴을 다운로드한 동일 웹 사이트에서 다음과 같이 다운로드 할 수 있음

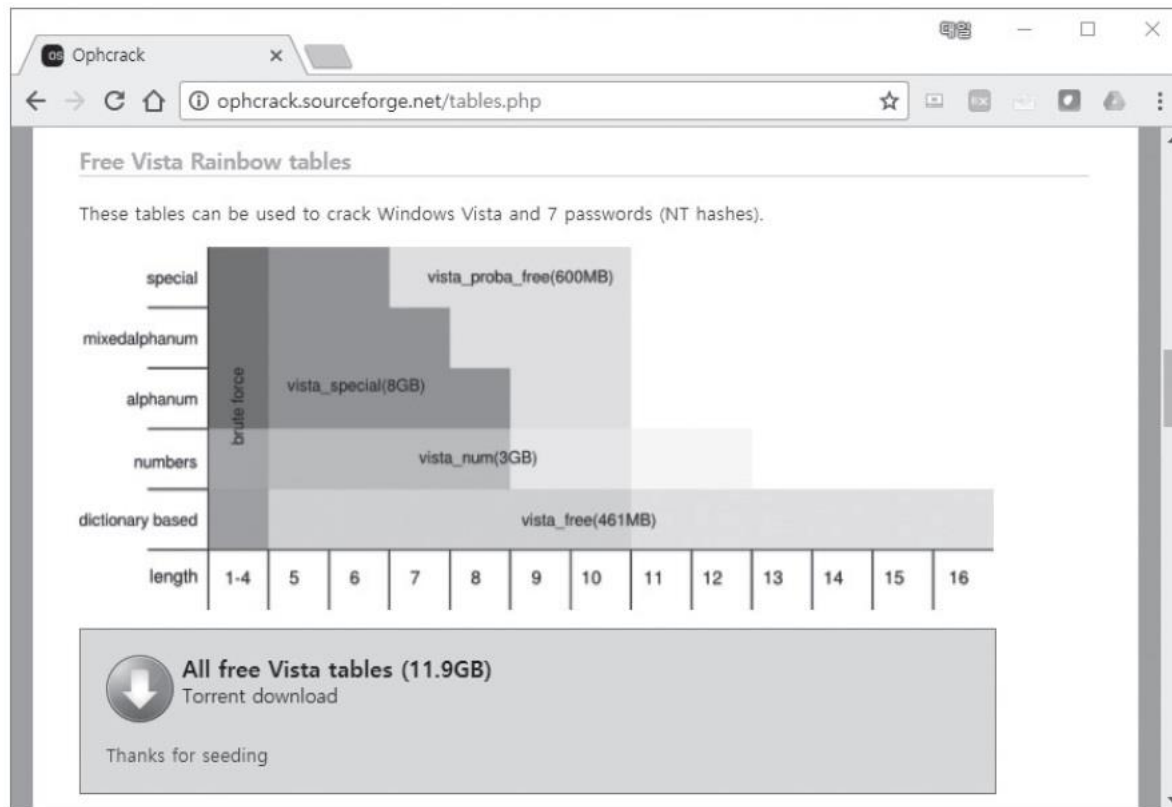


그림 4-21 레인보우 테이블 다운로드

윈도우 패스워드 크래킹하기

3. 레인보우 테이블을 이용한 패스워드 크랙하기

- ophcrack 툴에서 [Tables] 메뉴를 선택한 후 Table Selection 창에서 <Install> 버튼을 누름
- 'Vista Special' 테이블을 저장한 폴더를 선택하면 다운로드한 테이블을 다음과 같이 추가

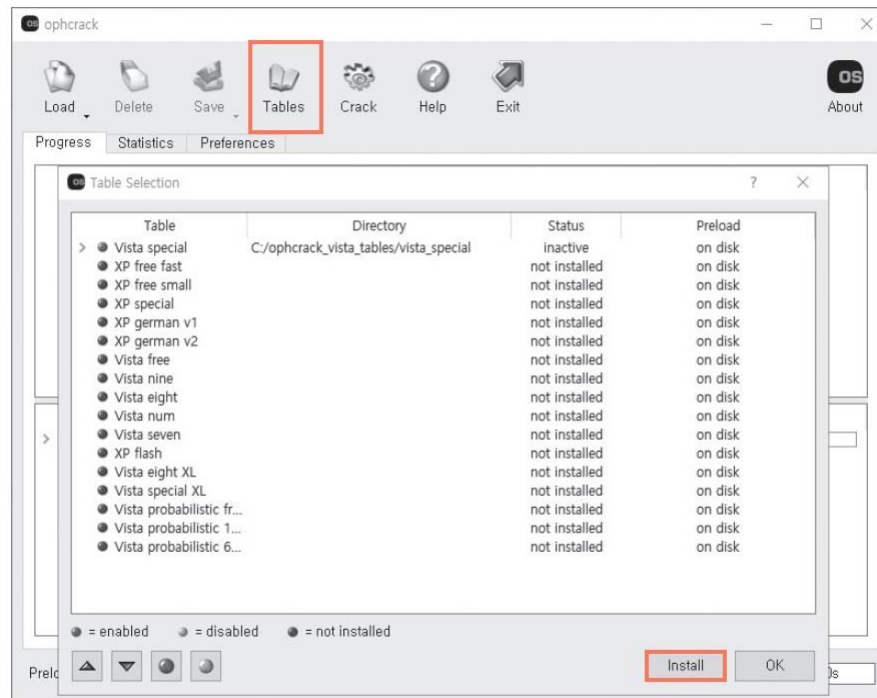


그림 4-22 레인보우 테이블 설치

윈도우 패스워드 크래킹하기

3. 레인보우 테이블을 이용한 패스워드 크랙하기

- [Load] - [PwDump File] 메뉴를 선택하고, 앞서 만든 NTLM_hash.txt 파일을 열어 패스워드 크래킹 대상을 읽어 들임
- [Crack] 메뉴를 선택하면 다음과 같이 'Vista Special' 테이블을 이용한 패스워드 크랙을 수행

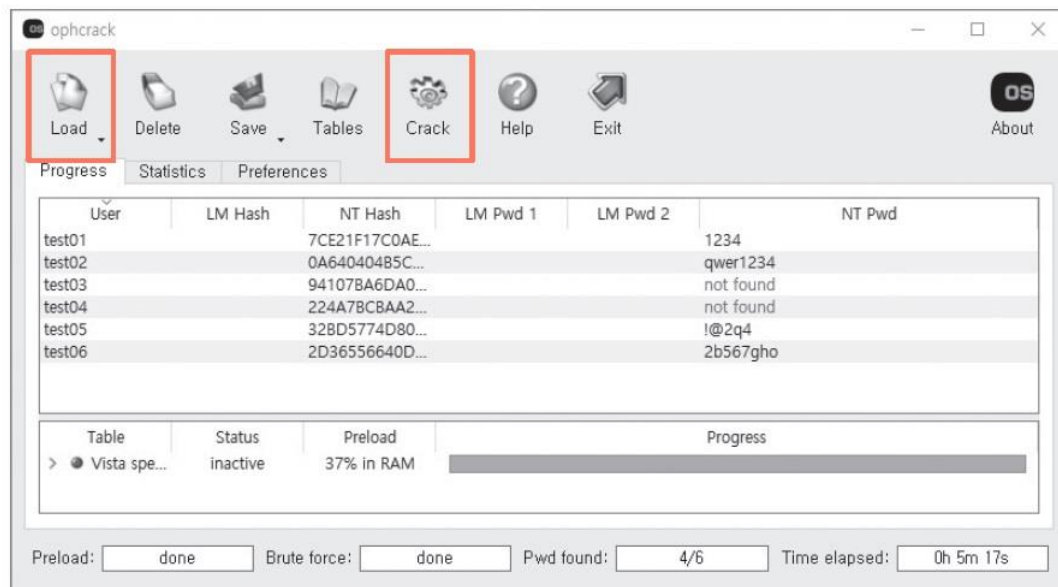


그림 4-23 레인보우 테이블을 이용한 패스워드 크랙

윈도우 패스워드 크래킹하기

4. 레인보우 테이블 생성하기

- 무료로 제공하는 레인보우 테이블은 보통 패스워드 범위가 제한되어 있음
- Cain & Abel에 포함된 Winrtgen 같은 툴로 직접 생성해도 됨

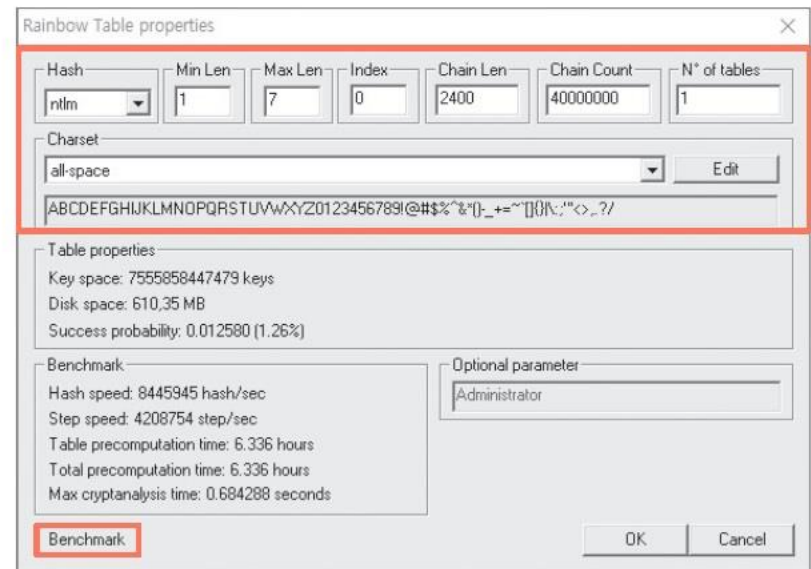
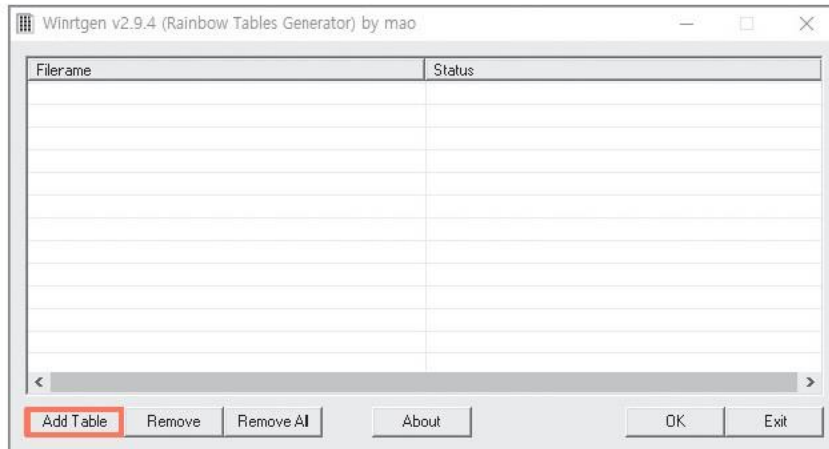


그림 4-24 레인보우 테이블 생성



윈도우 패스워드 크래킹하기

4. 레인보우 테이블 생성하기

- Winrtgen을 실행한 후 <Add Table> 버튼을 누르면 레인보우 테이블을 생성할 수 있는 창이 뜬다
- 생성하려는 목적에 따라 Hash를 선택하고, 패스워드 길이 등 옵션을 설정

표 4-9 해시 종류와 패스워드 설정 강도에 따른 레인보우 테이블 생성 조건

해시 종류	LM	NTLMv2	NTLMv2	NTLMv2
패스워드 최대 길이	7	7	10	10
구성 문자	알파벳(소문자), 숫자, 특수 문자(14가지)			알파벳, 숫자
레인보우 테이블 숫자	30	30	3,700,000	1,400,000
레인보우 테이블 용량	17.88GB	17.88GB	2,205,371.85GB	83,456.50GB
성공률	97%	97%	97.16%	97.19%
레인보우 테이블 생성 시간	20일	12일	5140년	189년

- 위 표의 세 번째 NTMLv2 해시의 경우 패스워드 최대 길이를 10자리로 하고, 구성 문자를 알파벳(소문자), 숫자, 특수문자(14가지)로 설정하면 약 5140년의 시간이 소요됨을 알 수 있음 (실질적으로 패스워드 크래킹이 불가능)



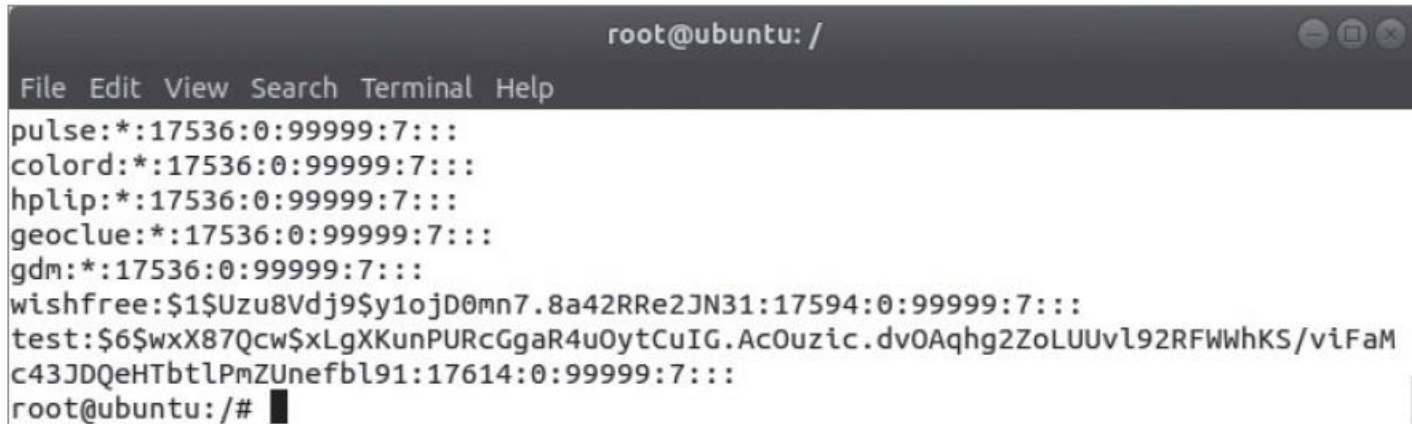
리눅스/유닉스 인증과 패스워드



리눅스/유닉스 인증과 비밀번호

- 유닉스 인증 방식
 - 윈도우 인증 방식보다 단순하지만 더 취약한 것은 아님
 - 유닉스에서 인증에 가장 중요한 역할을 하는 것은 비밀번호 파일과 shadow 파일
 - 비밀번호 파일의 구조는 앞장에서 살펴본 바 있음
 - 비밀번호는 shadow 파일에 암호화하여 저장

```
cat /etc/shadow
```



```
root@ubuntu: /  
File Edit View Search Terminal Help  
pulse*:17536:0:99999:7:::  
colord*:17536:0:99999:7:::  
hplip*:17536:0:99999:7:::  
geoclue*:17536:0:99999:7:::  
gdm*:17536:0:99999:7:::  
wishfree:$1$Uzu8Vdj9$y1ojD0mn7.8a42RRe2JN31:17594:0:99999:7:::  
test:$6$wxX87Qcw$LgXKunPURcGgaR4uOytCuIG.AcOuzic.dv0AQhg2ZoLUUv192RFWWhKS/viFaM  
c43JDQeHTbtlPmZUneftl91:17614:0:99999:7:::  
root@ubuntu: /#
```

그림 4-25 /etc/shadow 파일



- shadow 파일의 root 계정에서 다음 정보를 확인할 수 있음

root : \$6\$p4Hf~중략~4mUqpGaGzMxN0./ : 14923 : 0 : 99999 : 7 : ____ : ____ : ____ :

(1) (2) (3) (4) (5) (6) (7) (8) (9)

- ① 사용자 계정
- ② 암호화된 사용자의 패스워드를 저장, '\$1\$'로 시작하면 MD5를 가리키고 '\$5\$'와 '\$6\$'로 시작하면 각각 SHA256, SHA512를 가리킴
- ③ 1970년 1월 1일부터 마지막으로 패스워드를 변경한 날까지 계산한 값, 약 41년
- ④ 패스워드를 변경하기 전에 먼저 패스워드를 사용한 기간, 최초 설정 후 바꾸지 않음
- ⑤ 패스워드를 바꾸지 않고 최대한 사용할 수 있는 기간, 이 값은 보안 정책에 따라 달라질 수 있으며, 보통 패스워드의 최대 사용 기간은 60일로 권고함
- ⑥ 패스워드 최대 사용 기간에 가까워지면 미리 사용자에게 이 사실을 알려야 하며, 여기에 서 패스워드 사용 기한 며칠 전에 경고를 보낼지 지정함
- ⑦ 계정에 대한 사용 제한을 결정, 며칠 후에 완전히 사용을 정지할지 설정
- ⑧ 계정이 완전 사용 정지된 기간을 1970년 1월 1일부터 계산한 값을 기록
- ⑨ 관리자가 임의로 사용할 수 있는 부분



리눅스/유닉스 인증과 패스워드

- shadow 파일
 - 암호화된 패스워드를 저장하는 기능 외에 패스워드에 대한 보안 정책도 적용 가능
 - 시스템에 shadow 파일이 없고 passwd 파일에 암호화된 패스워드를 저장했다면, 시스템에 계정 보안 정책을 거의 적용하지 않았다고 볼 수 있음

표 4-10 운영체제별 passwd와 shadow 파일 위치

운영체제	shadow 파일 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

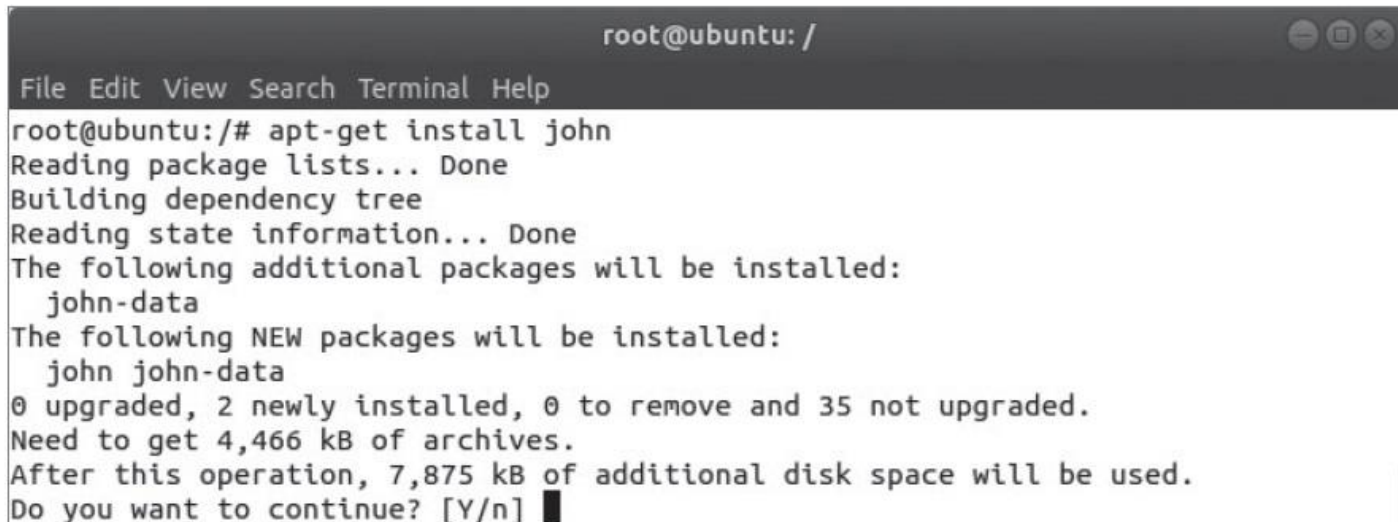


리눅스 패스워드 크래킹하기

1. John-the-ripper 설치

- 우분투 17 버전에서 John-the-ripper는 apt-get으로 간단히 설치

```
apt-get install john
```



```
root@ubuntu: /
File Edit View Search Terminal Help
root@ubuntu:/# apt-get install john
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  john-data
The following NEW packages will be installed:
  john john-data
0 upgraded, 2 newly installed, 0 to remove and 35 not upgraded.
Need to get 4,466 kB of archives.
After this operation, 7,875 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

그림 4-26 John-the-ripper 설치



리눅스 패스워드 크래킹하기

1. John-the-ripper 설치

- john the ripper 설치 후 john 명령으로 간단한 사용법 등을 확인

john

```
root@ubuntu: /
File Edit View Search Terminal Help
root@ubuntu:/# john
Created directory: /root/.john
John the Ripper password cracker, version 1.8.0
Copyright (c) 1996-2013 by Solar Designer
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single                "single crack" mode
--wordlist=FILE --stdin  wordlist mode, read words from FILE or stdin
--rules                 enable word mangling rules for wordlist mode
--incremental[=MODE]    "incremental" mode [using section MODE]
--external=MODE         external mode or word filter
--stdout[=LENGTH]       just output candidate passwords [cut at LENGTH]
--restore[=NAME]         restore an interrupted session [called NAME]
--session=NAME           give a new session the NAME
--status[=NAME]          print status of a session [called NAME]
--make-charset=FILE      make a charset, FILE will be overwritten
--show                  show cracked passwords
--test[=TIME]            run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,..] [do not] load this (these) user(s) only
--groups=[-]GID[,..]     load users [not] of this (these) group(s) only
--shells=[-]SHELL[,..]   load users with[out] this (these) shell(s) only
--salts=[-]N             load salts with[out] at least N passwords only
--save-memory=LEVEL      enable memory saving, at LEVEL 1..3
--node=MIN[-MAX]/TOTAL   this node's number range out of TOTAL count
--fork=N                 fork N processes
--format=NAME            force hash type NAME: descript/bsdictcrypt/md5crypt/
                        bcrypt/LM/AFS/tripcode/dummy/crypt

root@ubuntu:/#
```

그림 4-27 John-the-ripper 사용법 확인

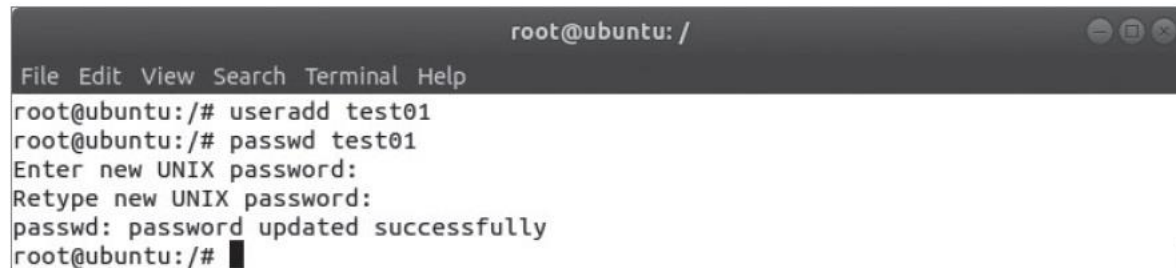


리눅스 패스워드 크래킹하기

2. 테스트 계정 생성 및 패스워드 설정하기

- 윈도우에서처럼 패스워드 크래킹에 사용할 계정을 몇 개 생성
- 다양한 난이도의 패스워드 크래킹 시도를 위해 숫자로만 된 패스워드, 짧은 패스워드, 영문자와 숫자로만 된 패스워드, 특수문자 등을 포함한 패스워드 등으로 설정
- 리눅스에서는 useradd 명령으로 계정 생성 가능

```
useradd test01  
passwd test01
```



```
root@ubuntu: /  
File Edit View Search Terminal Help  
root@ubuntu:/# useradd test01  
root@ubuntu:/# passwd test01  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
root@ubuntu:/#
```

그림 4-28 패스워드 크래킹을 위한 테스트 계정 추가

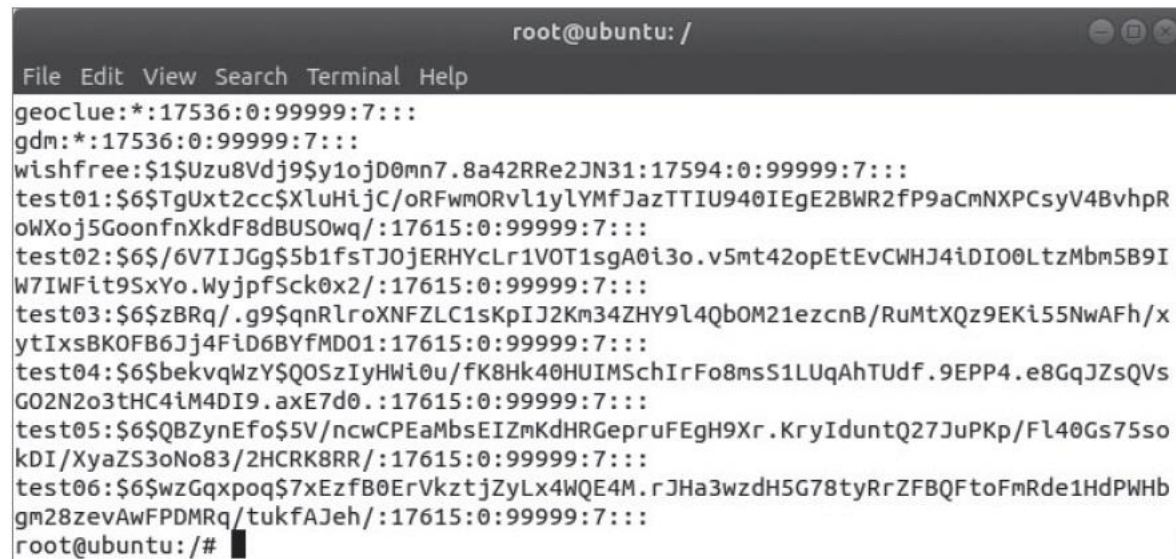


리눅스 패스워드 크래킹하기

2. 테스트 계정 생성 및 패스워드 설정하기

- 추가한 사용자의 패스워드 정보는 /etc/shadow 파일에서 확인 가능

```
cat /etc/shadow
```



```
root@ubuntu: /
File Edit View Search Terminal Help
geoclue:*:17536:0:99999:7:::
gdm:*:17536:0:99999:7:::
wishfree:$1$Uzu8Vdj9$y1ojD0mn7.8a42RRe2JN31:17594:0:99999:7:::
test01:$6$TgUxt2cc$XluHiJC/oRFwmORv11yLYMFJazTTIU940IEgE2BWR2fP9aCmNXPCsyV4BvhpR
oWXoj5GoonfnXkdF8dBUS0wq/:17615:0:99999:7:::
test02:$6$/6V7IJGg$5b1fsTJOjERHYcLr1VOT1sgA0i3o.v5mt42opEtEvCWHJ4iDI00LtzMbm5B9I
W7IWFit9SxYo.WyjpgfSck0x2/:17615:0:99999:7:::
test03:$6$zBRq/.g9$qnRlroXNFZLC1sKpIJ2Km34ZHY9l4QbOM21ezcnB/RuMtXQz9EKi55NwAfH/x
ytIxsBKOFB6Jj4FiD6BYfMDO1:17615:0:99999:7:::
test04:$6$bekvqWzY$QOSzIyHWi0u/fK8Hk40HUIMSchiRfFo8msS1LUqAhTUdf.9EPP4.e8GqJZsQVs
G02N2o3tHC4iM4DI9.axE7d0.:17615:0:99999:7:::
test05:$6$QBZynEfo$5V/ncwCPEaMbsEIZmKdHRGepuFEgH9Xr.KryIduntQ27JuPKp/FL40Gs75so
kDI/XyaZS3oNo83/2HCRK8RR/:17615:0:99999:7:::
test06:$6$swzGqxpq$7xEzfB0ErVktztjZyLx4WQE4M.rJHa3wzdH5G78tyRrZFBQFtoFmRde1HdPWHb
gm28zevAwFPDMRq/tukfAJeh/:17615:0:99999:7:::
root@ubuntu: /#
```

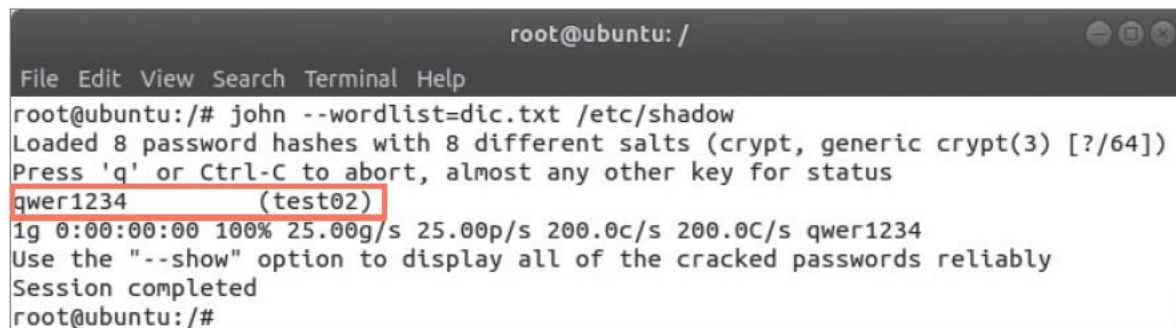
그림 4-29 추가한 계정에서 SHA512로 해시된 패스워드 확인

리눅스 패스워드 크래킹하기

3. 패스워드 크래킹

- 리눅스에서 패스워드 크래킹은 우선 패스워드로 사용할 수 있는 사전 파일을 미리 만들어 두고, 이 사전 파일에 있는 패스워드를 대입하여 수행함
- 사전 파일에 'qwer1234'라는 패스워드를 미리 넣어 둔 상태에서 John-the-ripper로 패스워드 크래킹을 시도

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



```
root@ubuntu: /
File Edit View Search Terminal Help
root@ubuntu:/# john --wordlist=dic.txt /etc/shadow
Loaded 8 password hashes with 8 different salts (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
qwer1234 (test02)
1g 0:00:00:00 100% 25.00g/s 25.00p/s 200.00c/s 200.00C/s qwer1234
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@ubuntu:/#
```

그림 4-30 사전 대입법을 이용한 패스워드 크래킹



리눅스 패스워드 크래킹하기

3. 패스워드 크래킹

- 사전 대입 공격에 실패했을 때는 무작위 대입법으로 패스워드 크래킹 시도
- SHA512 해시를 이용한 shadow 파일을 무작위 대입법을 써서 패스워드 크래킹하기는 매우 어려움
 - SHA512 알고리즘을 사용한 해시를 생성하는데 시간이 오래 걸려 많은 경우의 수를 모두 대입하기가 어렵기 때문
- SHA512 같은 알고리즘으로 해시된 shadow 파일을 크래킹할 때는 레인보우 테이블을 이용하여 크래킹하는 것이 효율적



서비스 데몬 패스워드



서비스 데몬 패스워드

- 서비스 데몬 패스워드
 - HTTP, FTP, 텔넷, SMB(NetBIOS) 데몬처럼 서버에서 제공하는 서비스 프로그램도 패스워드 크래킹 가능
 - 대부분의 서비스 데몬은 운영체제와 동일한 ID와 패스워드를 가지고 있음
 - 서비스 데몬을 이용한 패스워드 크래킹 시도는 운영체제의 다른 서비스에도 접근할 수 있는 권한을 얻을 가능성이 높음
 - 윈도우의 파일 공유 서비스(SMB(NetBIOS))나 리눅스의 텔넷 서비스 등이 대표적인 예
 - 서비스 데몬에서 패스워드 크래킹을 하여 획득한 계정이 반드시 운영체제에 존재한다는 보장은 없음
 - 일부 윈도우 서비스에서는 서비스 데몬 계정과 패스워드를 별도로 생성하여 관리하는 경우도 있음

서비스 데몬 패스워드

- Bruter 툴
 - 원격 데몬의 패스워드를 크랙할 수 있는 다음 기능이 포함되어 있음
 - FTP, SSH2, telnet
 - HTTP(Basic), HTTP(Form)
 - SMTP, IMAP, POP3
 - MSSQL, MySQL
 - SMB-NT
 - SNMP

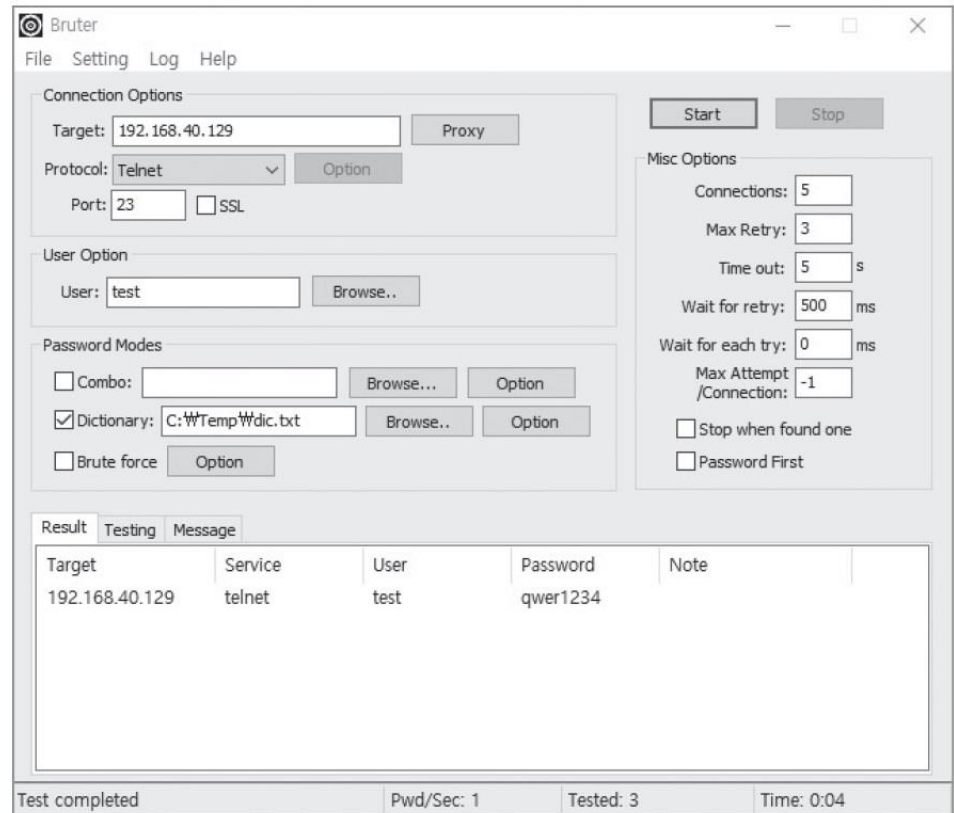


그림 4-31 telnet 데몬에서 Bruter 툴을 사용한 결과



실습 FTZ Level 8. 리눅스 패스워드 파일 크랙



문제 파악

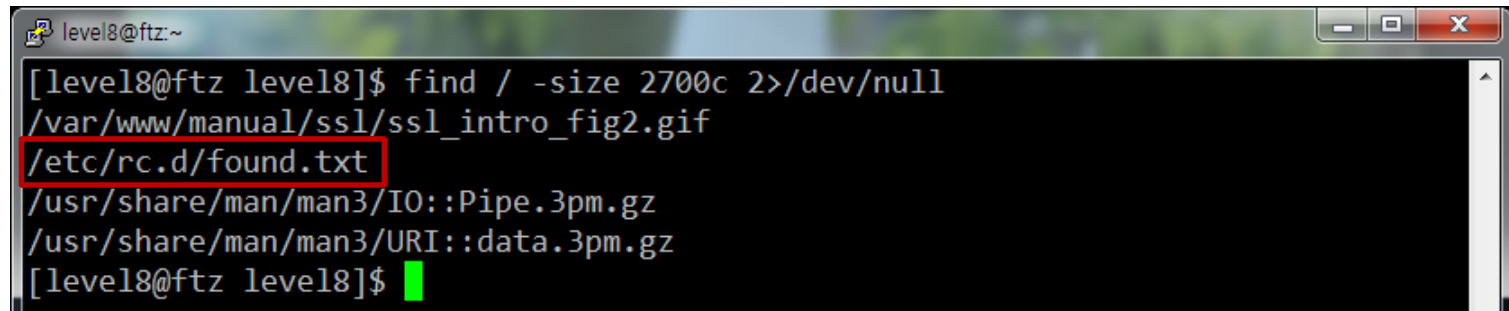
- level8 계정으로 로그인 → 힌트 확인 (암호: break the world)
 - Level9의 shadow 파일이 서버 어딘가에 숨어있다.
 - 그 파일에 대해 알려진 것은 용량이 “2700”이라는 것 뿐이다.

```
level8@ftz:~  
login as: level8  
level8@192.168.232.131's password:  
[level8@ftz level8]$ ls -l  
total 12  
-rw-r----- 1 root    level8    109 Jan 14  2010 hint  
drwxr-xr-x  2 root    level8   4096 Feb 24  2002 public_html  
drwxrwxr-x  2 root    level8   4096 Jan 14  2009 tmp  
[level8@ftz level8]$ cat hint  
  
level9의 shadow 파일이 서버 어딘가에 숨어있다.  
그 파일에 대해 알려진 것은 용량이 "2700"이라는 것 뿐이다.  
  
[level8@ftz level8]$
```



find 명령 활용

- 용량이 2700 바이트인 파일 찾기
 - 파일 크기를 기준으로 파일 찾기
 - -size [크기] [단위]
 - 단위
 - b: 512 바이트
 - c: 바이트
 - k: 킬로바이트
 - Permission denied 오류 메시지를 나오지 않게 하려면?
 - 2> /dev/null
 - 결합하면?



```
level8@ftz:~  
[level8@ftz level8]$ find / -size 2700c 2>/dev/null  
/var/www/manual/ssl/ssl_intro_fig2.gif  
/etc/rc.d/found.txt  
/usr/share/man/man3/IO::Pipe.3pm.gz  
/usr/share/man/man3/URI::data.3pm.gz  
[level8@ftz level8]$
```



파일 내용 확인 - 암호화된 비밀번호

```
level8@ftz:~  
[level8@ftz level8]$ ls -al /etc/rc.d/found.txt  
-r--r----- 1 root      level8      2700 Sep 10  2011 /etc/rc.d/found.txt  
[level8@ftz level8]$ cat /etc/rc.d/found.txt  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524  
level9:$1$vkY6sSlG$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524
```



사용자 유지 및 관리

- /etc/passwd : 사용자 계정 정보가 저장되는 기본 파일
- 파일의 구성

로그인 ID	:	x	:	UID	:	GID	:	설명	:	홈 디렉토리	:	로그인 셸
①		②	③	④	⑤		⑥			⑦		

- 로그인ID : 사용자 계정 이름
- x : 암호 부분으로 실제 암호는 /etc/shadow 파일에 저장
- UID : 사용자 번호
- GID : 기본 그룹 번호
- 설명 : 사용자에 대한 부가적 설명
- 홈 디렉토리 : 사용자 계정의 홈 디렉토리 (절대경로)
- 로그인 셸 : 사용자의 기본 셸



사용자 유지 및 관리 파일

- /etc/shadow 파일 : 사용자 패스워드 정보를 별도로 관리하는 파일
 - /etc/passwd 파일은 누구나 읽을 수 있음
 - /etc/shadow 파일은 root 사용자만 읽기 가능

```
ubuntu@arm:~$ ls -l /etc/passwd /etc/shadow
```

-rw-r--r--	1	root	sys	1216	04월	11일	05:40	/etc/passwd
-r-----	1	root	root	708	04월	11일	05:40	/etc/shadow



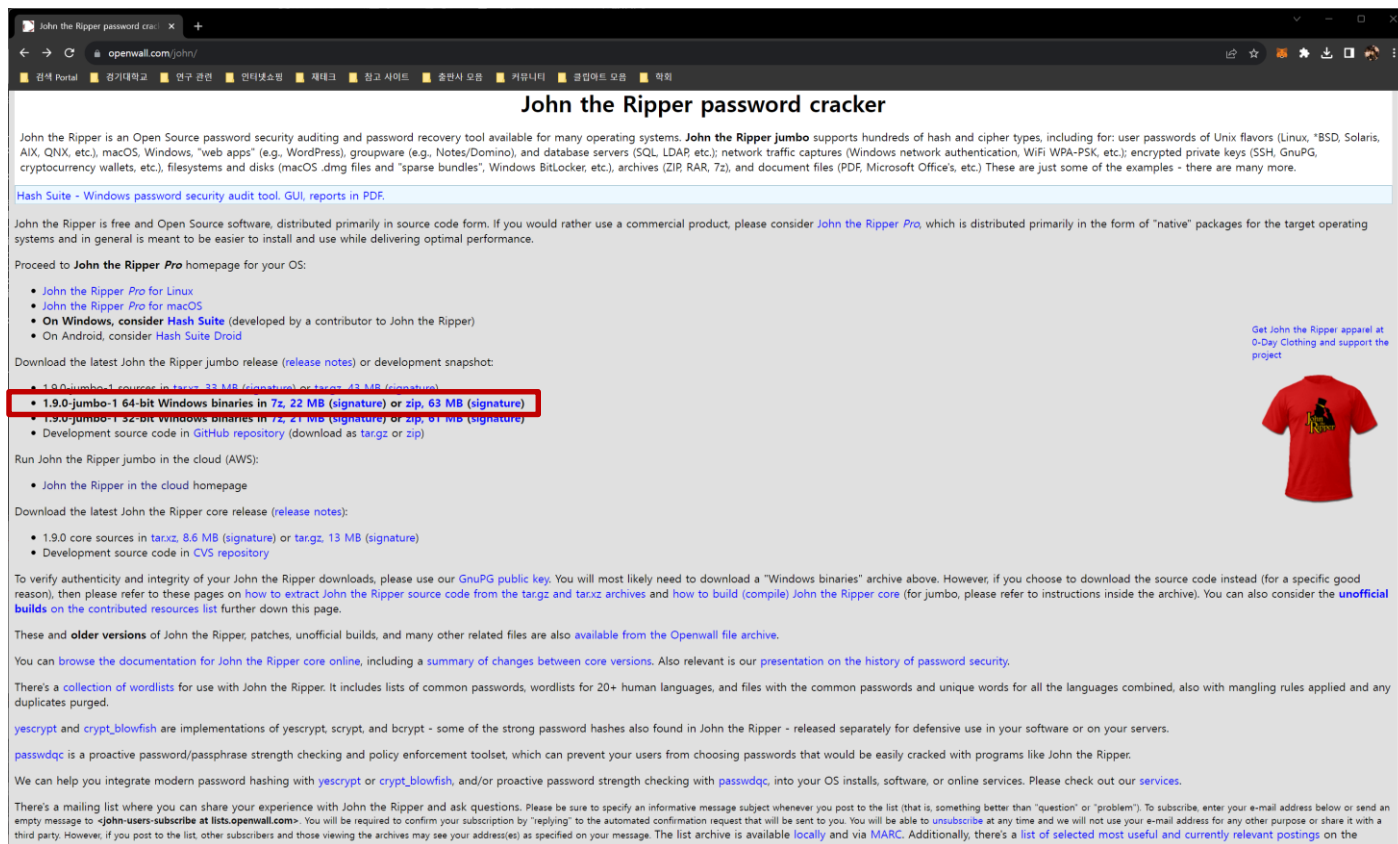
/etc/shadow 파일 구조

열1	열2	열3	열4	열5	열6	열7	열8	열9
level8	\$111 \$jdjdDD##@djddjd0	15534	0	9999	7	-1	-1	134540332
구분				설명				
열1				아이디				
열2				암호화된 패스워드				
열3				마지막 패스워드 변경일(1970. 1. 1부터 계산)				
열4				패스워드 변경 주기 일수("0일"이므로 설정 안 함)				
열5				현재 패스워드의 유효 기간("99999일"까지만 사용 가능)				
열6				패스워드 사용 만료 전, 경고 표시 일 수("99993일" ~ "99999일" 경고 표시)				
열7				패스워드 만료 후, 계정을 비활성화하는 일 수 (" -1"이므로 비활성화됨)				
열8				계정 만료(Expire) 일 수(1970. 1. 1부터 계산) (" -1"이므로 만료됨)				
열9				추가 사용을 위해 예약된 필드				



암호 해독 툴 다운로드

- John the Ripper
 - 패스워드 크랙 툴
 - <http://www.openwall.com/john>



The screenshot shows the homepage of John the Ripper, a password security auditing and recovery tool. The page is titled "John the Ripper password cracker" and provides information about the tool's capabilities, including support for various operating systems and file formats. It also offers download links for different versions and platforms, with a red box highlighting the "1.9.0-jumbo-1 64-bit Windows binaries in 7z, 22 MB (signature) or zip, 63 MB (signature)" option. The page includes a sidebar with a red t-shirt image and a footer with a mailing list subscription link.

John the Ripper password cracker

John the Ripper is an Open Source password security auditing and password recovery tool available for many operating systems. **John the Ripper jumbo** supports hundreds of hash and cipher types, including for: user passwords of Unix flavors (Linux, *BSD, Solaris, AIX, QNX, etc.), macOS, Windows, "web apps" (e.g., WordPress), groupware (e.g., Notes/Domino), and database servers (SQL, LDAP, etc.); network traffic captures (Windows network authentication, WiFi WPA-PSK, etc.); encrypted private keys (SSH, GnuPG, cryptocurrency wallets, etc.); filesystems and disks (macOS .dmg files and "sparse bundles", Windows BitLocker, etc.); archives (ZIP, RAR, 7z), and document files (PDF, Microsoft Office's, etc.) These are just some of the examples - there are many more.

[Hash Suite](#) - Windows password security audit tool. GUI, reports in PDF.

John the Ripper is free and Open Source software, distributed primarily in source code form. If you would rather use a commercial product, please consider [John the Ripper Pro](#), which is distributed primarily in the form of "native" packages for the target operating systems and in general is meant to be easier to install and use while delivering optimal performance.

Proceed to [John the Ripper Pro](#) homepage for your OS:

- [John the Ripper Pro for Linux](#)
- [John the Ripper Pro for macOS](#)
- [On Windows, consider Hash Suite](#) (developed by a contributor to John the Ripper)
- [On Android, consider Hash Suite Droid](#)

Download the latest John the Ripper jumbo release ([release notes](#)) or development snapshot:

- [1.9.0-jumbo-1 source in tar.gz, 33 MB \(signature\) or tar.gz, 43 MB \(signature\)](#)
- **[1.9.0-jumbo-1 64-bit Windows binaries in 7z, 22 MB \(signature\) or zip, 63 MB \(signature\)](#)**
- [1.9.0-jumbo-1 32-bit Windows binaries in 7z, 21 MB \(signature\) or zip, 61 MB \(signature\)](#)
- [Development source code in GitHub repository](#) (download as tar.gz or zip)

Run John the Ripper jumbo in the cloud (AWS):

- [John the Ripper in the cloud homepage](#)

Download the latest John the Ripper core release ([release notes](#)):

- [1.9.0 core sources in tar.gz, 8.6 MB \(signature\) or tar.gz, 13 MB \(signature\)](#)
- [Development source code in CVS repository](#)

To verify authenticity and integrity of your John the Ripper downloads, please use our [GnuPG public key](#). You will most likely need to download a "Windows binaries" archive above. However, if you choose to download the source code instead (for a specific good reason), then please refer to these pages on [how to extract John the Ripper source code from the tar.gz and tar.gz archives](#) and [how to build \(compile\) John the Ripper core](#) (for jumbo, please refer to instructions inside the archive). You can also consider the [unofficial builds on the contributed resources list](#) further down this page.

These and **older versions** of John the Ripper, patches, unofficial builds, and many other related files are also [available from the Openwall file archive](#).

You can [browse the documentation for John the Ripper core online](#), including a [summary of changes between core versions](#). Also relevant is our [presentation on the history of password security](#).

There's a [collection of wordlists](#) for use with John the Ripper. It includes lists of common passwords, wordlists for 20+ human languages, and files with the common passwords and unique words for all the languages combined, also with mangling rules applied and any duplicates purged.

[yescrypt](#) and [crypt_blowfish](#) are implementations of yescrypt, crypt, and bcrypt - some of the strong password hashes also found in John the Ripper - released separately for defensive use in your software or on your servers.

[passwdqc](#) is a proactive password/passphrase strength checking and policy enforcement toolset, which can prevent your users from choosing passwords that would be easily cracked with programs like John the Ripper.

We can help you integrate modern password hashing with [yescrypt](#) or [crypt_blowfish](#), and/or proactive password strength checking with [passwdqc](#), into your OS installs, software, or online services. Please check out our [services](#).

There's a mailing list where you can share your experience with John the Ripper and ask questions. Please be sure to specify an informative message subject whenever you post to the list (that is, something better than "question" or "problem"). To subscribe, enter your e-mail address below or send an empty message to john-users-subscribe@lists.openwall.com. You will be required to confirm your subscription by "replying" to the automated confirmation request that will be sent to you. You will be able to [unsubscribe](#) at any time and we will not use your e-mail address for any other purpose or share it with a third party. However, if you post to the list, other subscribers and those viewing the archives may see your address(es) as specified on your message. The list archive is available [locally](#) and via [MARC](#). Additionally, there's a [list of selected most useful and currently relevant postings](#) on the



암호화된 패스워드 복사

- John the Ripper 설치 디렉토리 아래 run 폴더에 저장

A screenshot of a Notepad window titled "hp.txt - 메모장". The menu bar includes "파일(F)", "편집(E)", "서식(O)", "보기(V)", and "도움말(H)". The text area contains a single line of a hashed password: "level9:\$1\$vkY6sSIG\$6RyUXtNMEVGsfY7Xf0wps.:11040:0:99999:7:-1:-1:134549524".



John the Ripper 실행

```
명령 프롬프트
C:\john-1.9.0-jumbo-1-win64\run>john hp.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-opencl"
Use the "--format=md5crypt-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8x3])
Will run 20 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 474 candidates buffered for the current salt, minimum 480 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
apple (level9)
1g 0:00:00:00 DONE 2/3 (2024-03-24 12:19) 32.25g/s 169935p/s 169935c/s 169935C/s 123456..121212
Use the "--show" option to display all of the cracked passwords reliably
Session completed

C:\john-1.9.0-jumbo-1-win64\run>
```



THANK YOU!