



4 패스워드 크래킹

IT CookBook, 정보 보안 개론과 실습 : 시스템 해킹과 보안(개정판)

❖ 학습목표

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

❖ 내용

- 패스워드 크래킹에 대한 이해
- 리눅스/유닉스 인증과 패스워드
- 서비스 데몬 패스워드 크래킹



❖ 패스워드 관리

- 보안 관리자의 첫번째 방어책
- 크래킹되기 쉬운 패스워드
 - 길이가 너무 짧거나 널(Null)인 패스워드
 - 사전에 나오는 단어나 이들의 조합으로 이루어진 패스워드
 - 키보드 자판을 일련된 순서로 나열한 패스워드 : asdf, qwert, 1234
 - 사용자 계정 정보에서 유추 가능한 단어들로 된 패스워드 : id wishfree, pw wishfree76
- 좋은 패스 워드 : 기억하기는 쉽고 크래킹하기 어려운 패스워드

인증의 유형	종류
알고 있는 것	패스워드, 주민등록번호, I-PIN 등
가지고 있는 것	신분증, 여권, 신용카드, 인증서, OTP, Key, 스마트카드
그 자체	홍채, 지문, 각막, 행동, 서명
위치해 있는 곳	지역, IP 주소



(참고) 패스워드 크래킹에 대한 이해

❖ 패스워드 설정의 취약점

- 패스워드 관리 애플리케이션 개발 회사인 SplashData는 사용자들이 사용하는 패스워드 중 해커들에게 쉽게 노출되는 최악의 패스워드 25개를 발표했다

순위	패스워드	순위	패스워드	순위	패스워드	순위	패스워드	순위	패스워드
1	password	6	monkey	11	baseball	16	ashley	21	654321
2	123456	7	1234567	12	111111	17	bailey	22	superman
3	12345678	8	letmein	13	iloveyou	18	passwOrd	23	qazwsx
4	qwerty	9	trustno1	14	master	19	shadow	24	michael
5	abc123	10	dragon	15	sunshine	20	123123	25	football



(참고) 패스워드 크래킹에 대한 이해

❖ 안전한 패스워드

속성	내용	예시
길이	세 가지 종류 이상의 문자 구성으로 8자리 이상	abc123!@
	두 가지 종류 이상의 문자 구성으로 10자리 이상	angel12345
속성	특정 명칭을 예측이 어렵도록 가공	'인터넷보안과해킹'의 경우 홀수번째 문자인 '인넷안해'를 영문으로 타이핑한 'dlssptdksgo' 사용
	노래 제목이나 명언, 속담, 가훈 등을 가공	'백설공주와 일곱 난장이'를 '백설+7난장'으로 줄인 후 영문으로 타이핑한 'qortjf+7skswkd' 사용
	패스워드 길이를 증가시키기 위해 알파벳 문자의 중간에 특수문자나 숫자를 삽입	Security10이 아닌 Se1cu@@nity
	자신의 기본 패스워드 문자열을 설정하고 사이트별 특정 규칙을 적용	기본문자열을 i486U로 하고 사이트 이름을 앞뒤에 추가하는 방법 da+i486U+um, nati486U+ver



패스워드 크래킹에 대한 이해

❖ 그리스 시대 암호화 방식 : 스키테일 (Scytale) 암호

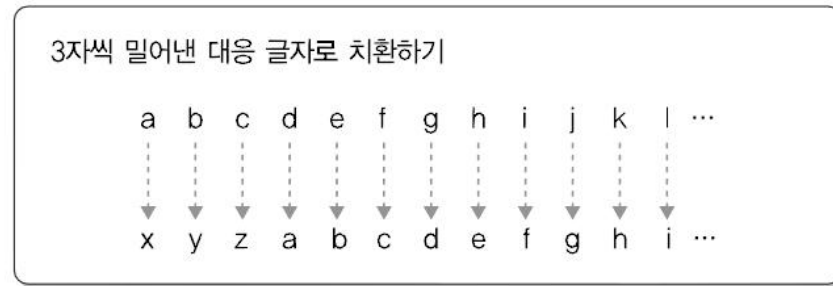
- 암호문이 쓰인 긴 양피지를 스키테일(나무 막대기)에 말아 가로로 읽어 해독하는 방식



❖ 로마시대 암호화 방식 : 기본적인 치환(Substitution) 암호



❖ 로마시대 암호화 방식 : 기본적인 치환(Substitution) 방식 (그림 4-2) 3자씩 알파벳을 밀어내 대응되는 글자로 치환



[그림 4-2] 밀어내기식 치환 방법

- 'Wish to be free from myself' 라는 문장의 암호화
- 암호화 알고리즘 : '알파벳을 밀어서 대응되는 글자로 치환'
- 암호화 키(Key) : 3

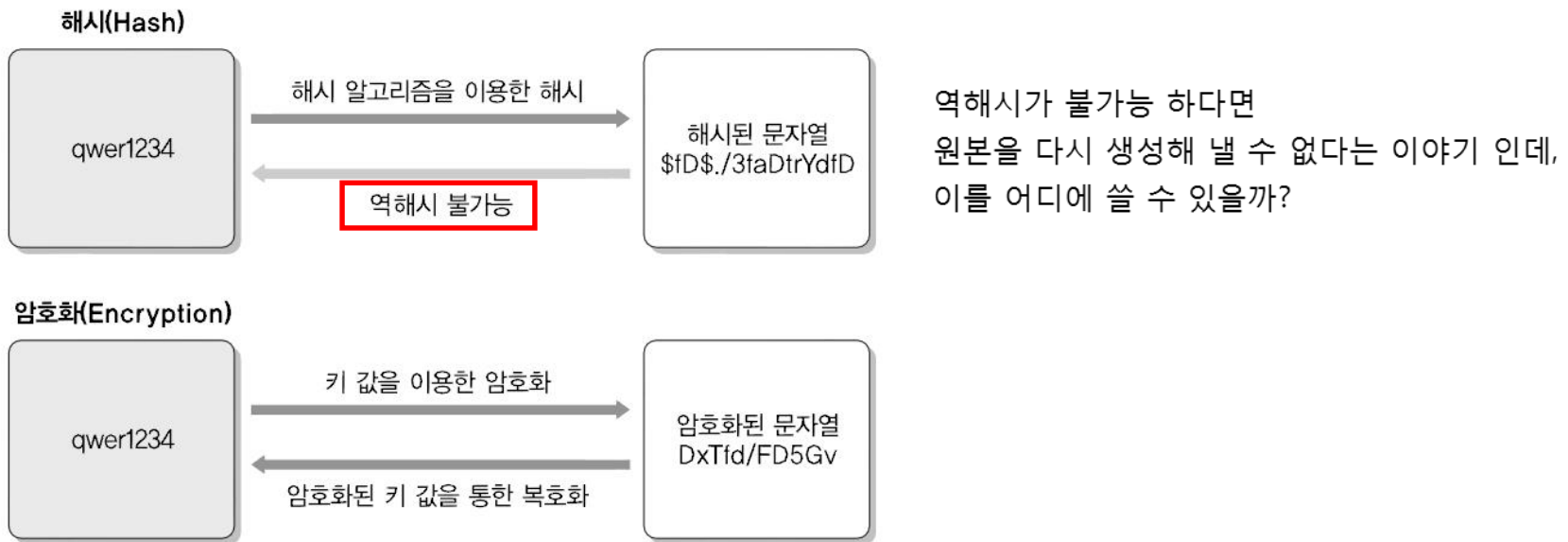
abcde fghij klmno pqrst uvwxy z
xyzab cdefg hijkl mnopq rstuv w
Wish to be free from myself → tfpk ql yb cobb colj jvpbic

[그림 4-3] 밀어내기식 치환 방법 예



❖ 해시의 암호화

- 해시(Hash) : 임의의 데이터로부터 일종의 짧은 '전자 지문'을 만들어 내는 방법
- 암호화(Encryption) : 특별한 알고리즘을 이용해 데이터를 전달하는 것



[그림 4-1] 해시와 암호화 알고리즘의 차이

해쉬와 암호화는 모두 패스워드를 효과적으로 숨기는 기술이나, 해쉬 알고리즘은 결과물의 복호화가 불가능하다.



❖ 기본적인 해쉬 알고리즘 사례

- 123456789수와123486789 가 있을 때, 두 수를 가운데를 기준으로 둘로 나누고 큰 수를 작은 수로 (가령) 나눈다 (나누기 이외의 어떠한 연산도 가능)
- 앞 6자리 숫자를 버리고(!) 나머지 값이 해시의 결과 값이라면 123456789의 해시 값은 2677861, 123486789의 해시 값은 4569155
- 로직을 알더라도 버려진 1.81838과 1.81882를 알 수 없으므로 두 해시 값만으로 해시 전의 원래 수를 알아내는 것은 불가능 (one-way)
- 해시는 로직을 알고 있을 경우 해시의 결과 값을 구하기가 쉽다 (검증 용이, 전자지문)
- 하지만, 해시 결과 값을 통해 해시를 생성하기 전의 원래 값은 알기 어렵다
- 또한, 값이 아주 조금만 다르더라도 (5 vs. 8) 해시 결과 값은 매우 상이하게 생성

$$\begin{array}{r} 12345 \\ \hline 6789 \end{array} = 1.81838 \text{ } 2677861 \dots$$
$$\begin{array}{r} 12348 \\ \hline 6789 \end{array} = 1.81882 \text{ } 4569155 \dots$$

자료 구조에서의 해쉬
블록 체인 등에서도 필수적 이해 요소 임

[그림 4-4] 나눗셈을 이용한 해시 값 획득



❖ Salt

- 해시나 암호화로 패스워드 저장 → **같은 패스워드는 항상 같은 해시 값, 같은 암호문으로 저장된다.**
- 같은 해시 결과나 암호문은 같은 결과만으로도 패스워드를 노출시키는 약점
 - 어랏! 저 친구의 암호문이 나와 같네? 나랑 같은 패스워드군!
- Salt(양념)는 이런 상황을 막기 위해 패스워드 해시와 암호화에 사용되는 첨가물의 일종
- 운영체제별로 다양한 알고리즘 존재.
- 동일한 암호 'eodlf@!11'을 Salt가 추가된 MD5 해시 알고리즘을 적용한 사례 (next slide)



패스워드 크래킹에 대한 이해

[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 패스워드 합한 'a2eodlf@!11'과 '4Feodlf@!11'을 각각 해시한 결과 값은 다름
- 이를 위해, 패스워드 파일로 저장할 경우 MD5 해시 결과 값만 저장하면 안됨
- 시스템이 패스워드와 어떤 salt와 합해 해시를 구한 것인지 알 수 없기 때문
- 패스워드 파일에 저장 시 간단한 인코딩을 통해 해시 결과 값 앞이나 뒤에 Salt 붙임

[표 4-2] 해시된 패스워드 값에 Salt 정보를 붙인 예

패스워드 파일에 salt를 보여 줌.

계정	Salt+MD5
root	a29EF83D5BEF4A7CBC6F7D4940D8447089
wishfree	4F6B796B0DD16C30CCF0B7F02E6457F024

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



❖ 패스워드 크래킹 방법에 대한 이해

■ 영화에서의 시나리오

- 패스워드를 한자리씩 해독해 감
- 8자리 패스워드라면 200개 정도의 문자 대입 * 8회 < 2000회 이하
- 실제로는 한 글자씩 풀리는 암호는 없음. 단지 영화에서의 극적 요소일 뿐임!
- 대부분의 패스워드는 전체 자리의 문자를 한번에 적용하지 못하면 풀리지 않는다.

■ 사전(dictionary) 대입공격

- 사용자가 설정하는 대부분의 패스워드에 특정 패턴이 있음에 착안
- 패스워드로 사용할 만한 것을 사전으로 만들어놓고 하나씩 대입하여 패스워드 일치 여부 확인

■ 무작위 대입 공격 (brute-force attack)

- 패스워드에 사용될 수 있는 문자열의 범위를 정하고,
- 그 범위 내에서 생성 가능한 모든 패스워드 생성하여 입력
- 패스워드가 그다지 복잡하지 않거나 짧은 경우 단시간에 크래킹 가능



❖ 패스워드 크래킹 방법에 대한 이해

소개되는 여러 크래킹 기법 : 클래식 기법

지금은 통하는 방식이 아니나, 왜 이러한 것들을 배워야 하는가?

■ 레인보우 테이블을 이용한 공격

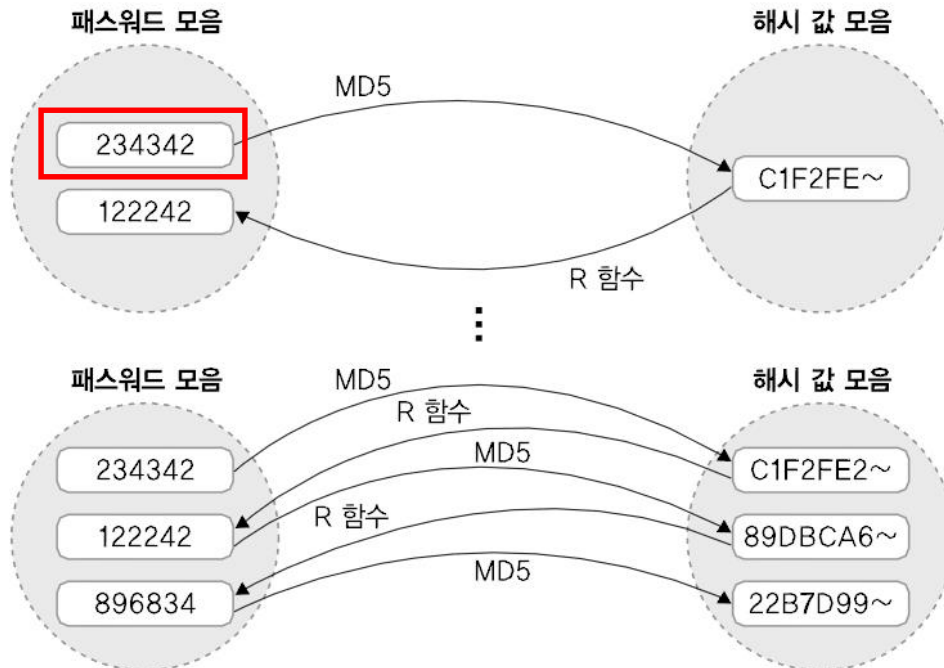
- 최초의 알고리즘은 1980년 마틴 헬만(Martin Hellman)에 의해 소개
- 2000년대 윈도우 LM 패스워드(로그인 패스워드)를 몇분만에 크래킹하면서 유명해 짐
- 레인보우 테이블은 하나의 패스워드에서 시작해 변이된 형태의 여러 패스워드 생성
- 변이된 각 패스워드의 해시를 고리처럼 연결하여 일정 수의 패스워드와 **해시로 이루어진 체인(Chain)**을 무수히 만들어 놓은 테이블
- 레인보우 테이블의 가장 기본적인 개념 : **패스워드별 해시 값 미리 생성**
- 크래킹하고자 하는 **해시 값을 테이블에서 검색하여 원래 패스워드를 찾는 것**
- 패스워드 '12qw', 해시 값 '123452323242'라면 각 패스워드별 해시 값을 미리 구해둔 해시 테이블에서 '123452323242'를 찾아 역으로 '12qw'의 패스워드 유추
- 일반적인 해시 테이블의 크기 : 수십 GB 이상



패스워드 크래킹에 대한 이해

■ 레인보우 테이블의 핵심 아이디어

- 대용량으로 생성될 수 있는 해시 테이블을 **R(Reduction) 함수**를 이용해 작은 크기로 줄이는 것
- 예를 들어 패스워드가 6자리 숫자로 이루어진 **'234342'** 라 하자.
- MD5 해시 값은 'C1F2FE224298D6E39EBA607D46F3D9CC'
- R 함수는 이 해시 값에서 또 다른 형태의 무작위 패스워드 추출
- **(가령) R 함수가 MD5 해시 값 중 문자를 제외한 앞의 6개 숫자만 뽑아낸다고 가정**
- R(C1F2FE224298D6E39EBA607D46F3D9CC)은 '122242'



[그림 4-6] MD5 해시와 R 함수의 동작

주어진 과정을 무수히 많이 반복하여
테이블 값의 조합이 수십~수백G 분량으로 준비
2000년대 초반 기준 굉장히 큰 데이터 사이즈임!

패스워드 크래킹에 대한 이해

- 최초 패스워드 '234342'에서 MD5 해시 값을 3번 구하고, R 함수가 2번 동작
[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

- [표 4-5] 또 다른 숫자 암호 346343의 MD5와 R 함수의 반복 실행 결과

패스워드		MD5 해시
최초 패스워드	346343	A62798B2BFCE406BD76FCBC7A3678876
첫 번째 R 함수 동작 결과	627982	570727EE4270E0C1A4D8FBB741926DB8
두 번째 R 함수 동작 결과	570727	86AB6B3355F33F7CD62658FDDA5AF7D6



패스워드 크래킹에 대한 이해

[표 4-6] 또, 또 다른 숫자 898232의 MD5와 R 함수의 반복 실행 결과

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

실제로는 6자리 모든 숫자에 대해
이와 같은 과정을 2000번 이상 반복

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

패스워드	MD5 해시
234342	22B7D9922C994737D0D9DFCCF6B415B6
346343	86AB6B3355F33F7CD62658FDDA5AF7D6
898232	E2038DD2A8315D9BF7F72AE5C07530F8



패스워드 크래킹에 대한 이해

- 레인보우 테이블을 이용한 패스워드 크래킹 과정
- 패스워드 해시 값 '570727EE4270E0C1A4D8FBB741926DB8'
- ① 레인보우 테이블에 크래킹하려는 해시 값과 같은 MD5 해시 값이 있는지 확인
→ [표 4-7]에는 '570727EE4270E0C1A4D8FBB741926DB8' 해시 값이 테이블에 없다(일반적으로 없음)
- ② 레인보우 테이블에 크래킹하려는 해시 값이 없으면 크래킹할 해시 값에 R 함수 적용 패스워드 구하고 다시 해시 값 구한다.
→ '570727EE4270E0C1A4D8FBB741926DB8'에 R 함수 적용 패스워드 '570727' 구한다
'570727' 해시 값을 구하면 '86AB6B3355F33F7CD62658FDDA5AF7D6'
- ③ ②에서 구한 해시 값 '86AB6B3355F33F7CD62658FDDA5AF7D6' 레인보우 테이블에 있는지 확인
→ [표 4-7]에서 생성한 레인보우 테이블에 '86AB6B3355F33F7CD62658FDDA5AF7D6' 값이 있다
(값이 없다면 같은 해시 값이 나올 때까지 ②와 ③ 과정을 반복) 없으면 다시 반복
- ④ 레인보우 테이블에서 확인한 해시 값 발견한 뒤 그 해시 값에 해당하는 최초 패스워드 구한다
→ [표 4-7]에서 '86AB6B3355F33F7CD62658FDDA5AF7D6'에 해당하는 패스워드는 '346343'
- ⑤ 확인한 최초 패스워드에서 다시 패스워드와 일치하는 해시 값이 나올 때까지 MD5 해시와 R 함수 반복 수행. 해당 해시 값이 확인되면 찾는 패스워드는 해당 해시 값을 생성한 문자열 (표 4-5 활용, 그 경로상의 627982가 최종 해독된 패스워드)
- 체인을 2,000개 사용하는 레인보우 테이블에서 해시 값을 10,000개 저장하고 있다면, 레인보우 테이블에서 확인할 수 있는 패스워드의 종류는 20,000,000 (2000*10000)개

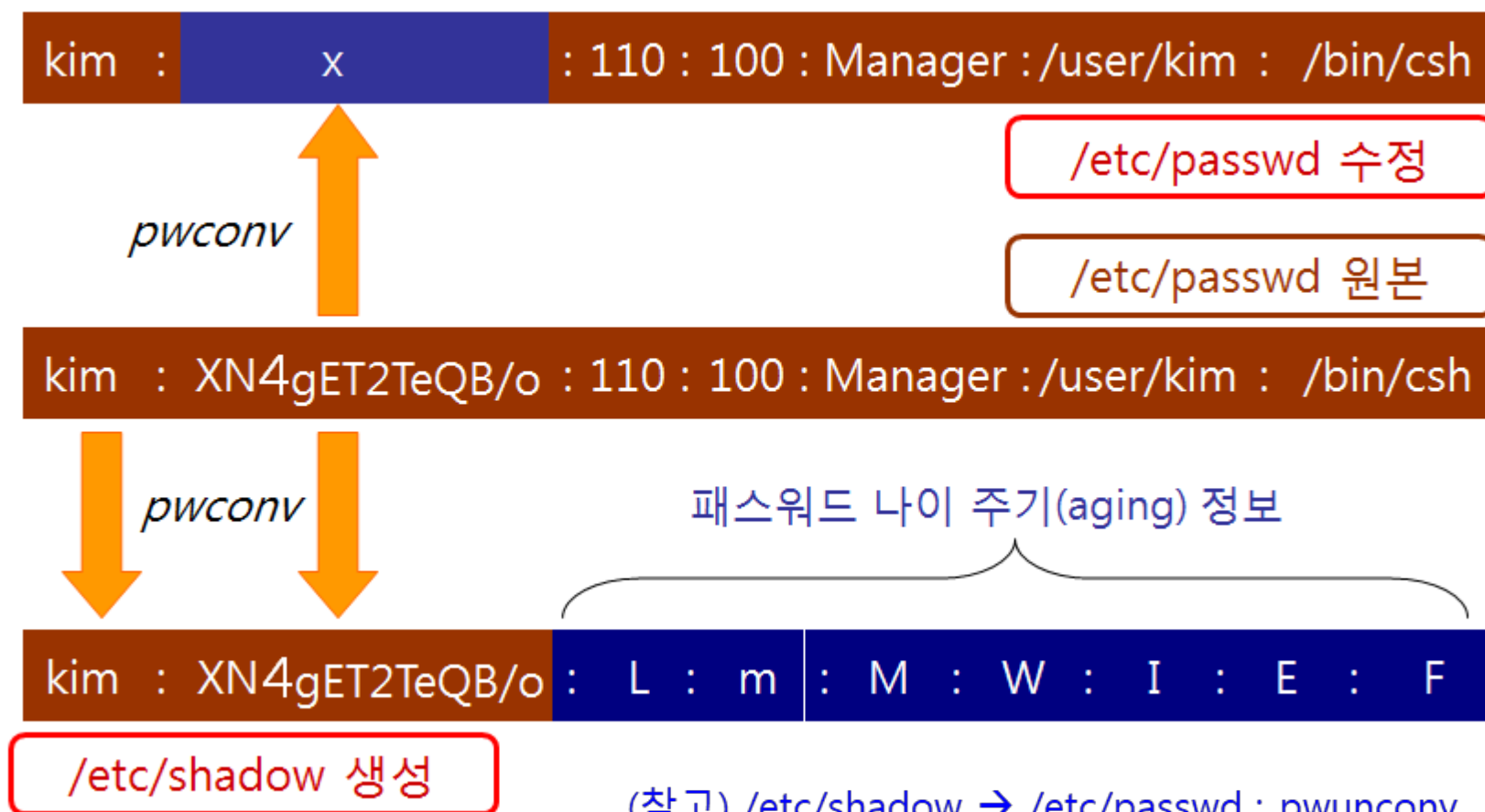
값이 있을 확률이 큰 이유가 R 함수의 특성 상,

숫자 6개로 이루어진 000000 ~ 999999 까지의 모든 값은 해시 테이블에 일단 있다.



리눅스/유닉스의 인증과 패스워드

- 유닉스에서 인증에 가장 중요한 역할을 하는 것은 passwd 파일과 shadow 파일
- 패스워드는 shadow 파일에 암호화되어 저장
- shadow 파일에서 root 계정에 대한 정보 확인(각 정보는 :로 나뉘어 있다)



리눅스/유닉스의 인증과 패스워드

root : \$6\$LL489S99Pyh6~중략~PazruKuAkuFTlk0 : 14923 : 0 : 99999 : 7 : : :
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

- ① 사용자 계정 (login name)
- ② 암호화된 사용자의 패스워드 저장, 시스템마다 조금씩 다르며 페도라 14 버전에서는 SHA512 형식을 기본으로 저장, MD5, SHA256 등의 해시를 선택, '\$1\$'로 시작하면 MD5, '\$5\$'와 '\$6\$'로 시작하면 각각 SHA256, SHA512를 나타냄 (각각 서로다른 암호 알고리즘을 사용) (encrypted password)
- ③ **패스워드 최종 수정일**과 1970년 1월 1일 사이의 날 수 (클수록 좋음), 14923일은 약 41년 (Last changed)
- ④ 패스워드 변경하기 전 패스워드 사용한 기간, **패스워드 변경 사이의 최소 일수** 가령, 3개월에 한번씩 패스워드 변경 (Minimum), 너무 자주 변경되는 것을 방지
- ⑤ 현재의 패스워드가 유효한 최대 일 수, 패스워드 바꾸지 않고 최대한 사용할 수 있는 유효 기간, 보통 **패스워드의 최대 사용 기간**을 60일로 권고 (Maximum)



리눅스/유닉스의 인증과 패스워드

root : \$6\$LL489S99Pyh6~중략~Pazr/uKuAkuFT0/ : 14923 : 0 : 99999 : 7 : : :
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

- ⑥ 패스워드 최대 사용 기간에 가까워질 경우, 사용자에게 패스워드 **사용 기한 며칠 전에 경고**를 보낼지 지정 (Warning), 해당 일부터 매일 경고 메시지가 뜨게 됨
 - ⑦ 패스워드 사용 일 수 (비활성화 설정), 계정에 대한 사용 제한 설정, **비밀 번호가 만료된 후, 계정이 잠기기 전까지의 기간** (Inactive), 빈 필드는 계정이 잠기지 않음을 의미
 - ⑧ **패스워드 사용 만기일(계정 사용 만기일)**, 1970년 1월 1일부터 해당 날짜가 지나면 계정 사용 불가(Expire), 빈 필드는 계정이 만료되지 않음을 의미
 - ⑨ 관리자 임의 사용 부분 (Reserved)
- shadow 파일 : 암호화된 패스워드 저장 기능, 패스워드에 대한 보안 정책을 적용, 시스템에 shadow 파일 존재하지 않고 **passwd 파일에 암호화된 패스워드 저장되면, 시스템에 계정에 대한 보안 정책 적용 안된 것으로 판단 가능**



리눅스/유닉스의 인증과 패스워드

시스템 파일 /etc/passwd

```
kim : XN4gET2TeQB/o : 110 : 100 : Manager : /user/kim : /bin/csh
```

계정이름
(username)

UID

GID
(기본)

사용자
정보

홈디렉토리
경로

Shell
종류

암호화된
패스워드

빈칸이면 패스워드 설정이
안되어 있는 계정이므로 주의

/usr/sbin/pwconv 명령을 사용하여 /etc/shadow
를 생성하면 암호화된
패스워드는 옮겨지고 이 부분은 x로 처리됨



리눅스/유닉스의 인증과 패스워드

시스템 파일 /etc/shadow

kim : XN4gET2TeQB/o : L : m : M : W : I : E : F

계정 이름
(username)

암호화된
패스워드

LK : 사용 중단 계정
NP: 패스워드가 없는 계정

- Lastchg: 패스워드의 최종 수정일과 1970년 1월 1일 사이의 일수
- min: 패스워드 변경 사이의 최소 일수
- Max: 현재의 패스워드가 유효한 최대 일수
- Warn: 패스워드 수정 경고가 보여지는 일수
- Inactive: 패스워드를 사용해야 하는 최소한의 일수(비활성화 일수)
- Expire: 패스워드 사용 만기일
- Flag: 나중에 사용할 예비용 필드

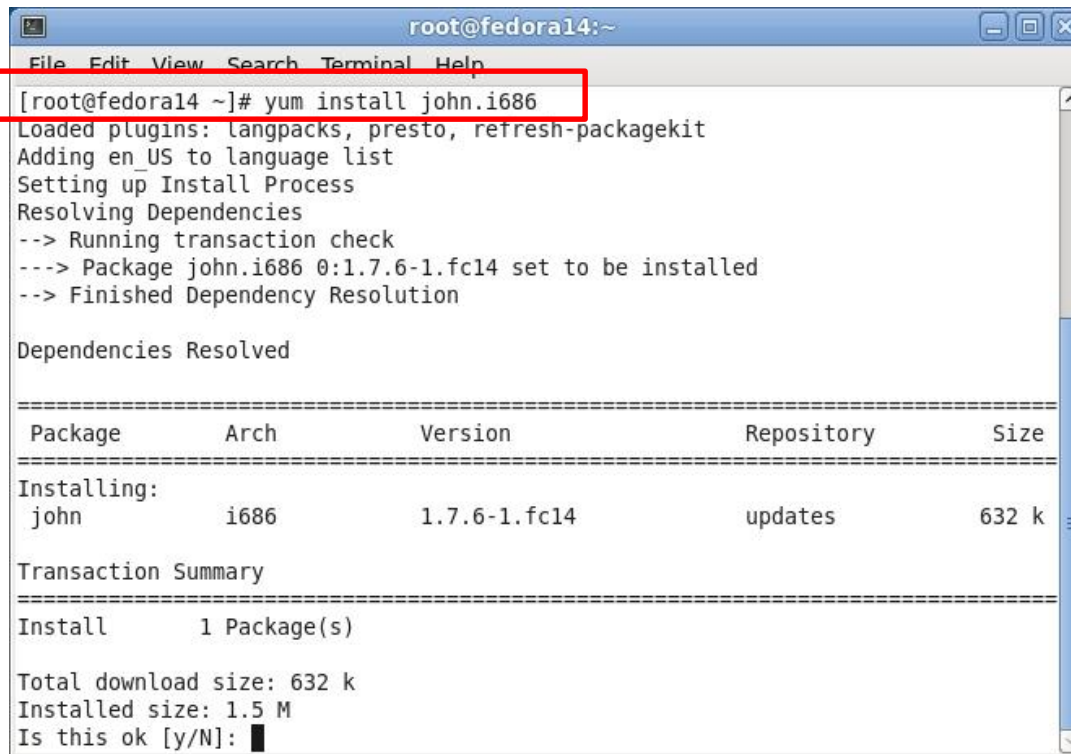


실습 4-2 리눅스 패스워드 크래킹하기

1 John-the-ripper 설치

- 페도라 14에서 Johnthe-ripper를 yum을 이용해 설치

```
yum install john.i686
```



```
root@fedora14:~  
File Edit View Search Terminal Help  
[root@fedora14 ~]# yum install john.i686  
Loaded plugins: langpacks, presto, refresh-packagekit  
Adding en_US to language list  
Setting up Install Process  
Resolving Dependencies  
--> Running transaction check  
---> Package john.i686 0:1.7.6-1.fc14 set to be installed  
--> Finished Dependency Resolution  
  
Dependencies Resolved  
  
=====
```

Package	Arch	Version	Repository	Size
Installing: john	i686	1.7.6-1.fc14	updates	632 k

```
=====
```

Transaction Summary

Transaction Summary	
Install	1 Package(s)

```
=====
```

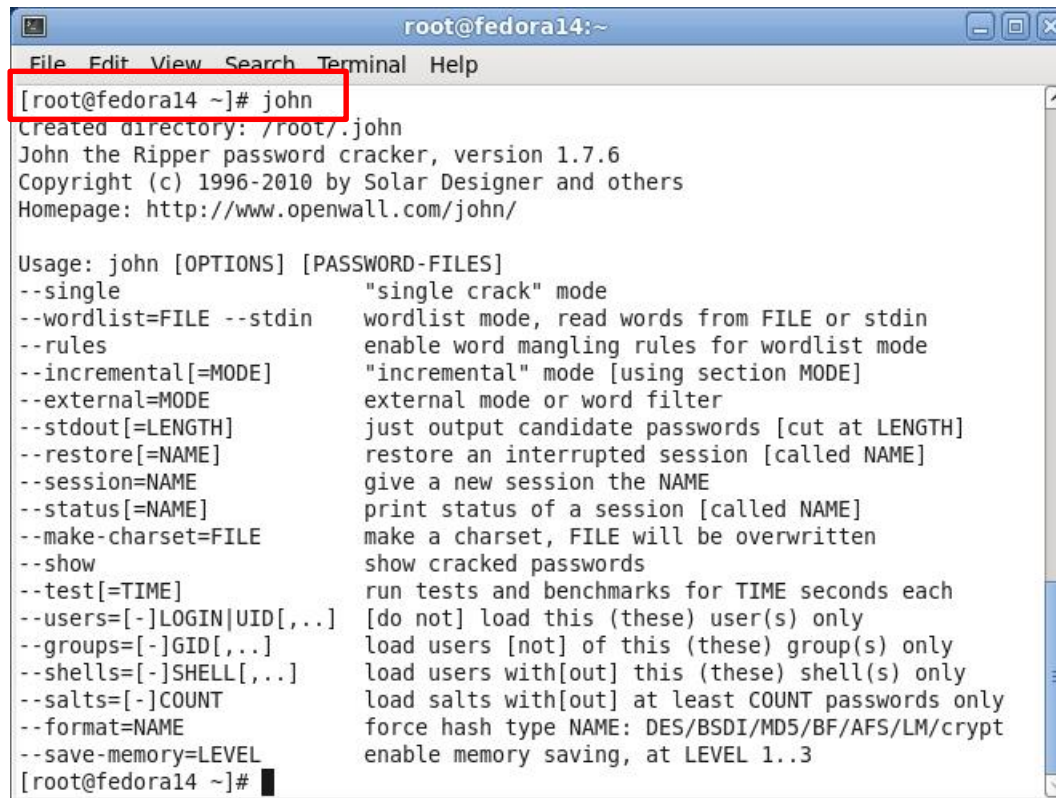
Total download size: 632 k
Installed size: 1.5 M
Is this ok [y/N]:

[그림 4-35] john the ripper의 설치



1 John-the-ripper 설치

- john the ripper 설치 후 john 명령(help 역할을 하는 명령)으로 간단한 사용법 등을 확인



```
root@fedora14:~  
File Edit View Search Terminal Help  
[root@fedora14 ~]# john  
Created directory: /root/.john  
John the Ripper password cracker, version 1.7.6  
Copyright (c) 1996-2010 by Solar Designer and others  
Homepage: http://www.openwall.com/john/  
  
Usage: john [OPTIONS] [PASSWORD-FILES]  
--single "single crack" mode  
--wordlist=FILE --stdin 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=[-]COUNT load salts with[out] at least COUNT passwords only  
--format=NAME force hash type NAME: DES/BSDI/MD5/BF/AFS/LM/crypt  
--save-memory=LEVEL enable memory saving, at LEVEL 1..3  
[root@fedora14 ~]#
```

[그림 4-36] john the ripper의 사용법 확인



실습 4-2 리눅스 패스워드 크래킹하기

2 테스트 계정 생성 및 패스워드 설정

- 패스워드 크래킹에 사용할 계정 생성
- 다양한 난이도의 패스워드 크래킹 시도를 위해 숫자만으로 이뤄진 패스워드, 짧은 패스워드, 영문자와 숫자만으로 이루어진 패스워드, 특수문자 등을 포함한 패스워드 등으로 설정
- useradd 명령으로 계정 생성

```
useradd user  
passwd user
```



```
root@fedora14:/  
File Edit View Search Terminal Help  
[root@fedora14 /]# useradd user  
[root@fedora14 /]# passwd user  
Changing password for user user.  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.  
[root@fedora14 /]#
```

[그림 4-37] 패스워드 크래킹을 위한 테스트 계정 추가



실습 4-2 리눅스 패스워드 크래킹하기

2 테스트 계정 생성 및 패스워드 설정

- 추가한 사용자의 패스워드 정보는 /etc/shadow 파일에서 확인

```
cat /etc/shadow
```



```
root@fedora14:/  
File Edit View Search Terminal Help  
ntp:!!:14976:::::::  
nm-openconnect:!!:14976:::::::  
mailnull:!!:14976:::::::  
smmsp:!!:14976:::::::  
sshd:!!:14976:::::::  
smolt:!!:14976:::::::  
pulse:!!:14976:::::::  
gdm:!!:14976:::::::  
wishfree:$6$/dvTFbouH.nzNafZ$i58DTsWLF79qmFX4ITSwschsvz1Rof/n.3J8c9ZBLH7Wxk0lWKS  
G8Emx1BX2K/Z6TiAxkoKL8deXL7nLYD7To0:14976:0:99999:7:::  
user:$6$j5f/AtZ0$lEVdsUG5V3snWSsUgjxHWCKcEH60N50tU/7HFQawyRiMiXivq3pq4ijWX1oERIA  
Hm8k8iMNez8c5InHIbGJc/1:14976:0:99999:7:::  
user0:$6$cSdHzvUl$XrupNSYCSe495p51hIyhLNFEQy56thexVuB/SpfjEw7uwcYB12.tQwI3B0cZ.P  
4b1Kw64tfON7B0Mk/TgTbK/:14976:0:99999:7:::  
user1:$6$tqGAdFwx$AFsiCDYgF.cLKncLDxFbwYlld8ZbPShRDswacxAbLjfxGVKtEv4Zr1fd.7QeSM  
6arq8.GkhXsCWvtyTt8N5mw/:14976:0:99999:7:::  
user2:$6$yAxcLLl/$Vkc51IhTAPiSFHXDKHR046Un0gm0sVk6spPdQA9ZBgMyycgwxEtCzZEM/0cTv3  
xZDV5I/qYclwTZKxlQuJaLo/:14976:0:99999:7:::  
[root@fedora14 /]#
```

[그림 4-38] 추가한 계정의 SHA512로 해시된 패스워드 확인

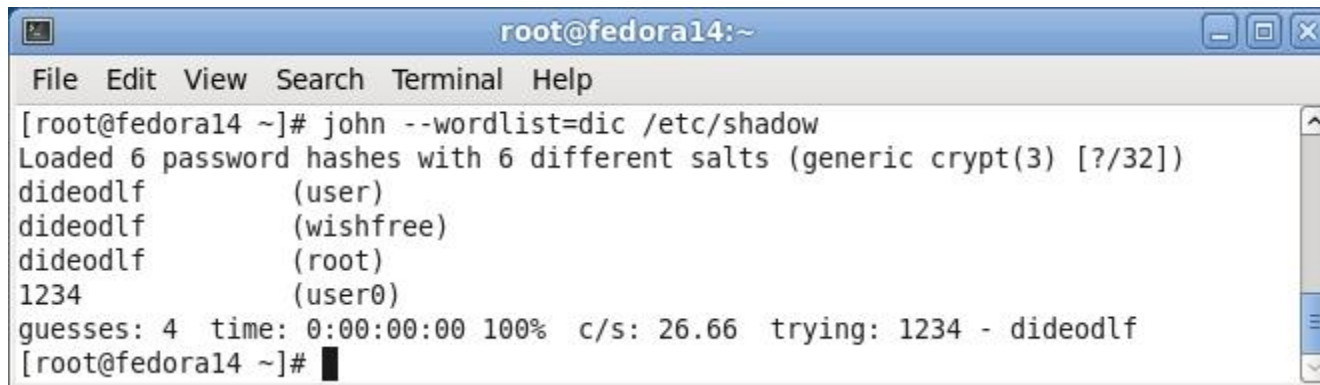


실습 4-2 리눅스 패스워드 크래킹하기

3 패스워드 크래킹

- 패스워드 크래킹은 사전(dictionary) 파일을 미리 작성하여 활용
- 이 사전(dictionary) 파일에 있는 패스워드 대입
- 사전 파일에 'dideodlf(양대일)'이라는 패스워드를 미리 넣어 두자.
- john the ripper로 패스워드 크래킹을 시도

john --wordlist=dic /etc/shadow



```
root@fedora14:~  
File Edit View Search Terminal Help  
[root@fedora14 ~]# john --wordlist=dic /etc/shadow  
Loaded 6 password hashes with 6 different salts (generic crypt(3) [?/32])  
dideodlf (user)  
dideodlf (wishfree)  
dideodlf (root)  
1234 (user0)  
guesses: 4 time: 0:00:00:00 100% c/s: 26.66 trying: 1234 - dideodlf  
[root@fedora14 ~]#
```

[그림 4-39] 사전 대입법을 이용한 패스워드 크래킹

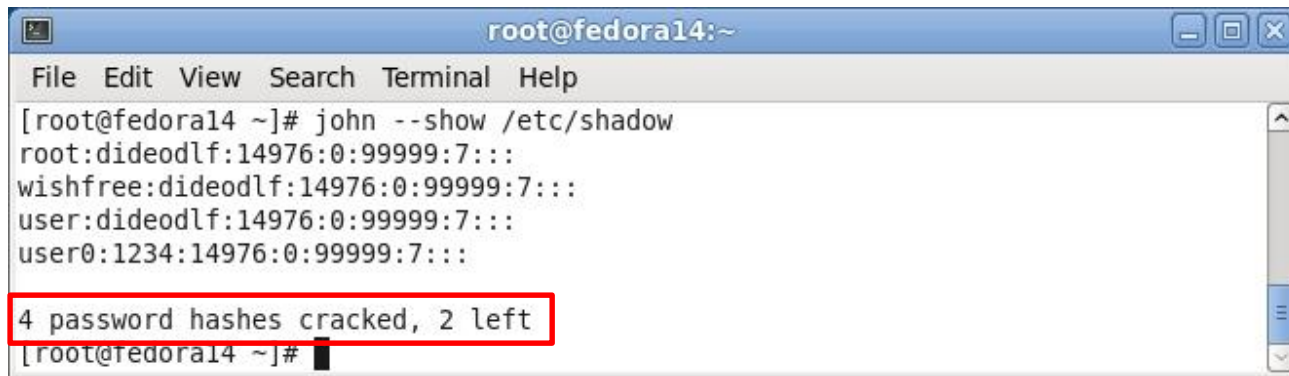


실습 4-2 리눅스 패스워드 크래킹하기

3 패스워드 크래킹

- 사전 대입 공격에 실패한 경우, 무작위 대입법(brute-force attack) 크래킹 시도
- 무작위 대입법을 이용한 패스워드 크래킹은 '--wordlist' 옵션 없이 바로 실행
- 이때 패스워드가 크래킹되면 바로 확인할 수 있도록 '--show' 옵션 사용

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



```
root@fedora14:~  
File Edit View Search Terminal Help  
[root@fedora14 ~]# john --show /etc/shadow  
root:dideodlf:14976:0:99999:7:::  
wishfree:dideodlf:14976:0:99999:7:::  
user:dideodlf:14976:0:99999:7:::  
user0:1234:14976:0:99999:7:::  
4 password hashes cracked, 2 left  
[root@fedora14 ~]#
```

[그림 4-40] 무작위 대입법을 이용한 패스워드 크래킹

- SHA512해시를 사용한 shadow 파일을 무작위 대입법을 사용해 패스워드 크래킹 하기는 매우 어렵다.



실습 4-2 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

이하 실습 4-2 슬라이드는 3판 기준 자료



실습 4-2 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: descrypt/bsdicrypt/md5crypt/
                        bcrypt/LM/AFS/tripcode/dummy/crypt

root@ubuntu:/#
```

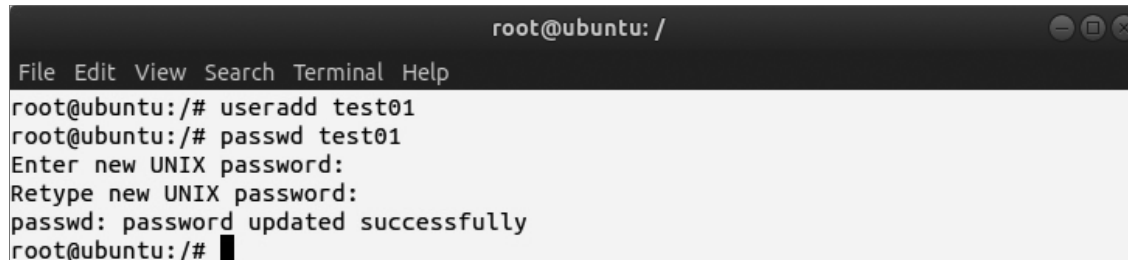
그림 4-27 john-the-ripper 사용법 확인



2 테스트 계정 생성 및 패스워드 설정

- 윈도우에서처럼 패스워드 크래킹에 사용할 계정 생성
- 다양한 난이도의 패스워드 크래킹 시도를 위해 숫자만으로 이뤄진 패스워드, 짧은 패스워드, 영문자와 숫자만으로 이루어진 패스워드, 특수문자 등을 포함한 패스워드 등으로 설정
- 리눅스에서는 `useradd` 명령으로 계정 생성

```
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

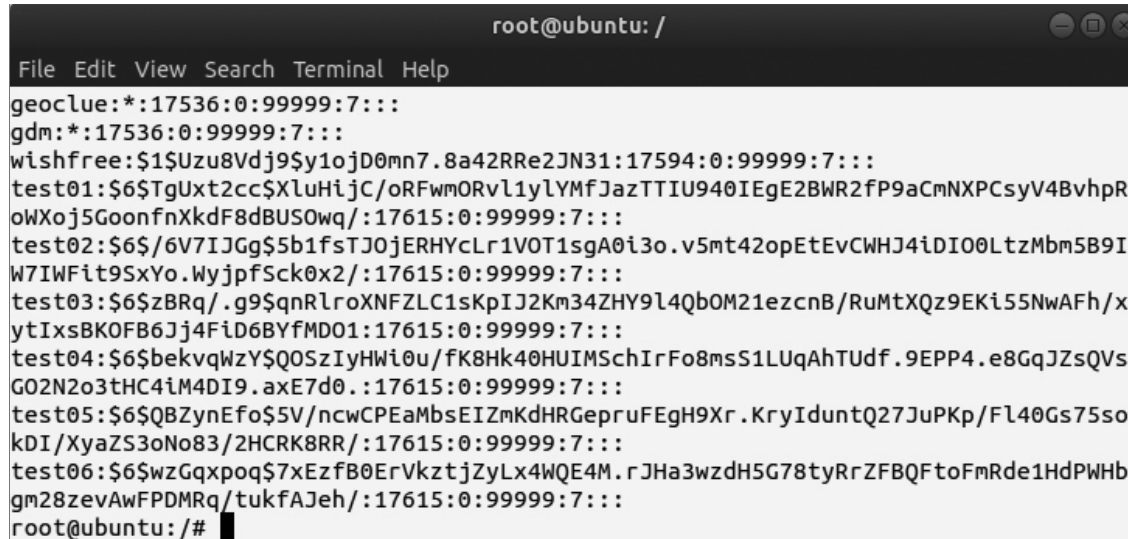


실습 4-2 리눅스 패스워드 크래킹하기

2 테스트 계정 생성 및 패스워드 설정

- 추가한 사용자의 패스워드 정보는 /etc/shadow 파일에서 확인

```
cat /etc/shadow
```

A terminal window titled 'root@ubuntu: /' showing the output of the 'cat /etc/shadow' command. The output lists system and test users with their hashed passwords. The test users are test01 through test06, each with a unique SHA512 hash.

```
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/oRFwmORv1yLYMfJazTTIU940IEgE2BWR2fP9aCmNXPCsyV4BvhpR
oWXoj5GoonfnXkdF8dBUS0wq/:17615:0:99999:7:::
test02:$6$/6V7IJGg$5b1fsTJOjERHYcLr1VOT1sgA0i3o.v5mt42opEtEvCWHJ4iDI00LtzMbm5B9I
W7IWFit9SxYo.WyjpgfSck0x2/:17615:0:99999:7:::
test03:$6$zBRq/.g9$qnRlroXNFZLC1sKpIJ2Km34ZHY9l4Qb0M21ezcnB/RuMtXQz9EKi55NwAFh/x
ytIxsBK0FB6Jj4FiD6BYfMD01:17615:0:99999:7:::
test04:$6$bekvqWzY$QOSzIyHWi0u/fK8Hk40HUIMSchIrFo8msS1LUqAhTUdf.9EPP4.e8GqJZsQVs
G02N2o3tHC4iM4DI9.axE7d0.:17615:0:99999:7:::
test05:$6$QBZynEfo$5V/ncwCPEaMbsEIZmKdHRGepuFEgH9Xr.KryIduntQ27JuPKp/Fl40Gs75so
kDI/XyaZS3oNo83/2HCRK8RR/:17615:0:99999:7:::
test06:$6$wzGqxpoq$7xEzfB0ErVktztjZyLx4WQE4M.rJHa3wzdH5G78tyRrZFBQFtoFmRde1HdPWHb
gm28zevAwFPDMRq/tukfAJeh/:17615:0:99999:7:::
root@ubuntu: /#
```

그림 4-29 추가한 계정에서 SHA512로 해시된 패스워드 확인

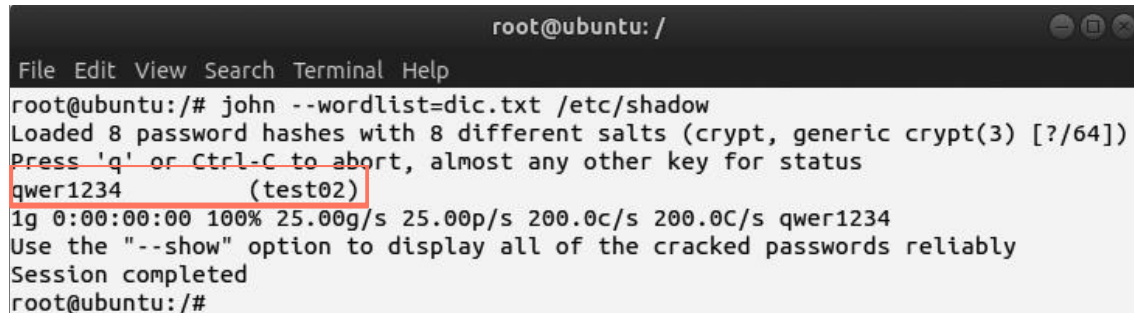


실습 4-2 리눅스 패스워드 크래킹하기

3 패스워드 크래킹

- 리눅스에서의 패스워드 크래킹은 패스워드로 사용될 수 있는 사전 파일 미리 작성
- 이 사전 파일에 있는 패스워드 대입하여 수행
- 사전 파일에 'qwer1234'라는 패스워드를 미리 넣어둔 상태에서 john-the-ripper로 패스워드 크래킹을 시도

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

A terminal window titled 'root@ubuntu: /' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'john --wordlist=dic.txt /etc/shadow' has been executed. The output shows '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', and a red box highlighting 'qwer1234 (test02)'. Below this, it shows '1g 0:00:00:00 100% 25.00g/s 25.00p/s 200.0c/s 200.0C/s qwer1234', 'Use the "--show" option to display all of the cracked passwords reliably', and 'Session completed'. The prompt returns to 'root@ubuntu: /#'.

```
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 200.0c/s 200.0C/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

3 패스워드 크래킹

- 사전 대입 공격에 실패한 경우 무작위 대입법으로 패스워드 크래킹 시도
- SHA512해시를 사용한 shadow 파일을 무작위 대입법을 사용해 패스워드 크래킹 하기는 매우 어렵다.
- SHA512 알고리즘을 사용한 해시 생성에 시간 소요 많아 경우의 수 모두 대입 어려움
- SHA512와 같은 알고리즘으로 해시된 shadow 파일을 크래킹할 때는 레인보우 테이블을 이용하여 크래킹하는 것이 효율적



John the Ripper 최신화



전제조건

- ❖ apt에 접근 가능해야 하므로, 실습 사용자 본인이 관리자 권한, 혹은 sudoers 권한이 있는 PC에서 실습하여야 한다.
- ❖ 아래 설명은 별도의 보안 설정을 추가하지 않은 순정 우분투 22.04 LTS 기준으로 설명되었으므로, 다른 버전의 리눅스 배포판이나 별도의 보안 설정 수행 시 달라질 수 있음



1. apt-get의 업데이트

```
> sudo apt-get update && sudo apt-get upgrade
Hit:1 http://mirror.kakao.com/ubuntu jammy InRelease
Hit:2 http://mirror.kakao.com/ubuntu jammy-updates InRelease
Hit:3 http://mirror.kakao.com/ubuntu jammy-backports InRelease
Hit:4 http://mirror.kakao.com/ubuntu jammy-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  gjs libgjs0g libnvidia-cfg1-470 libnvidia-compute-470 libnvidia-compute-470:i386 libnvidia-decode-470 libnvidia-decode-470:i386
  libnvidia-encode-470 libnvidia-encode-470:i386 libnvidia-extra-470 libnvidia-fbc1-470 libnvidia-fbc1-470:i386 libnvidia-gl-470
  libnvidia-gl-470:i386 libnvidia-ifrl-470 libnvidia-ifrl-470:i386 linux-generic-hwe-22.04 linux-headers-generic-hwe-22.04
  linux-image-generic-hwe-22.04 linux-modules-nvidia-470-generic-hwe-22.04 nvidia-compute-utils-470 nvidia-driver-470
  nvidia-kernel-common-470 nvidia-kernel-source-470 nvidia-utils-470 python3-update-manager ubuntu-advantage-tools update-manager
  update-manager-core xserver-xorg-video-nvidia-470
The following packages will be upgraded:
  libc-bin libc-dev-bin libc-devtools libc6 libc6:i386 libc6-dbg libc6-dev libc6-dev-i386 libc6-dev-x32 libc6-i386 libc6-x32
  linux-libc-dev locales
13 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
Need to get 37.3 MB of archives.
After this operation, 17.4 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

- ❖ 레포지토리에서 최신화된 업데이트 정보를 가지고 오고, 패키지를 업그레이드한다.
- ❖ 여기서, 특히 바로 ubuntu를 설치한 환경에서는 apt-get update에 의한 레포지토리 최신화가 가장 중요



2. john 설치

```
> sudo apt-get install john
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
john is already the newest version (1.8.0-4ubuntu3).
0 upgraded, 0 newly installed, 0 to remove and 43 not upgraded.
```

❖ 위 과정에서 설치가 진행되지 않을 경우

1. 리눅스 배포판의 버전을 확인
2. 1번 apt-get update가 적절하게 수행되었는지 확인



3. unshadow

```
> sudo unshadow /etc/passwd /etc/shadow > unshadow_passwd
> head unshadow_passwd -n 20
root!:0:0:root:/root:/usr/bin/zsh
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:*:2:2:bin:/bin:/usr/sbin/nologin
sys:*:3:3:sys:/dev:/usr/sbin/nologin
sync:*:4:65534:sync:/bin:/bin/sync
games:*:5:60:games:/usr/games:/usr/sbin/nologin
man:*:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:*:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:*:8:8:mail:/var/mail:/usr/sbin/nologin
news:*:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:*:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:*:13:13:proxy:/bin:/usr/sbin/nologin
www-data:*:33:33:www-data:/var/www:/usr/sbin/nologin
backup:*:34:34:backup:/var/backups:/usr/sbin/nologin
list:*:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

- ❖ 반드시 필요한 과정은 아니나, 추천되는 과정
- ❖ /etc/passwd(계정 정보 저장)과 /etc/shadow(비밀번호 저장) 파일을 병합하는 과정
- ❖ 병합하여 별도의 파일로 출력



4. john the ripper 사용

```
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: descrypt/bsdicrypt/md5crypt/
                   bcrypt/LM/AFS/tripcode/dummy/crypt
```

❖ 주요 파라미터

- **--wordlist: dictionary attack**
- **--incremental: brute-force attack**
- **--format: shadow 파일의 암호화를 수동 지정.**



4. john the ripper – format

- --format=NAME
 - 1.8.0 기준 descript, bsdictcrypt, md5crypt, bccrypt, LM, AFS, tripcode, dummy 및 crypt
 - 크래킹 세션을 시작할 때 또는 "--test", "--show", "--make-charset" 중 하나와 함께 이 옵션을 사용할 수 있음.
 - John은 동시에 다양한 유형의 해시를 해독할 수 없다. 둘 이상의 해시 유형을 사용하는 비밀번호 파일을 얻은 경우 각 해시 유형에 대해 John을 한 번 호출해야 하며 이 옵션을 사용하여 John이 기본적으로 자동 감지하는 유형이 아닌 다른 유형의 해시를 크랙하도록 해야한다
 - "--format=crypt"는 John의 특정 빌드에서 지원될 수도 있고 지원되지 않을 수도 있다.
 - 이 옵션을 지정하고 지원하면 John은 시스템의 crypt(3) 또는 crypt_r(3) 기능을 사용하게 됨.
 - 이는 시스템에서 지원하지만 아직 John의 최적화된 암호화 루틴에서는 지원되지 않는 비밀번호 해시를 감사하는 데 필요할 수 있다.
 - 현재 이는 최신 버전의 Fedora 및 Ubuntu에서 사용되는 glibc 2.7+ SHA-crypt 해시와 최신 버전의 Solaris에서 지원되는(기본적으로 사용되지는 않는) SunMD5 해시의 경우 사용하여야 함.
- 주어진 실습에서는, crypt로 설정하여 수행
- format이 정해지지 않았을 경우(혹은, 적절하지 않은 타입)

```
> john --incremental unshadow_passwd  
No password hashes loaded (see FAQ)
```



5. john the ripper 전체 명령줄 예시

- Brute-Force Attack

```
> sudo john --incremental --format=crypt unshadow_passwd
Loaded 53 password hashes with 53 different salts (crypt, generic crypt(3) [?/64])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
1234 (tester)
1g 0:00:00:13 0.07513g/s 0p/s 194.7c/s 194.7C/s 123456..misa
1g 0:00:00:14 0.07012g/s 0p/s 195.2c/s 195.2C/s 123456..misa
```

- Dictionary Attack

```
> sudo john --wordlist=dict.txt --format=crypt unshadow_passwd
Loaded 53 password hashes with 53 different salts (crypt, generic crypt(3) [?/64])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
1234 (tester)
1g 0:00:00:01 100% 0.5319g/s 1.595p/s 84.57c/s 84.57C/s 1234..password
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

```
> cat dict.txt
1234
1q2w3e4r
password
```



❖ 데몬(daemon)이란?

- 서버에서 제공되는 서비스 프로그램
 - HTTP, FTP 데몬처럼 서버에서 제공하는 서비스 프로그램도 패스워드 크래킹 가능
 - 서비스 데몬 대부분은 운영체제와 동일한 아이디와 패스워드 소유
 - 서비스 데몬을 통한 패스워드 크래킹 시도로 운영체제의 다른 서비스에 대한 접근 권한을 얻을 수도 있음
- 데몬 크래킹은 해킹 대상 시스템에 **처음 비집고 들어갈 지점을 찾기 위한 공격**



실습 4-3 서비스 데몬 패스워드 크래킹하기

- 실습에 사용할 Bruter 툴에는 원격 데몬의 패스워드를 크랙할 수 있는 다음 기능이 포함

- FTP, SSH2, telnet
- HTTP
- SMTP, IMAP, POP3
- MSSQL, MySQL
- SMB-NT
- SNMP

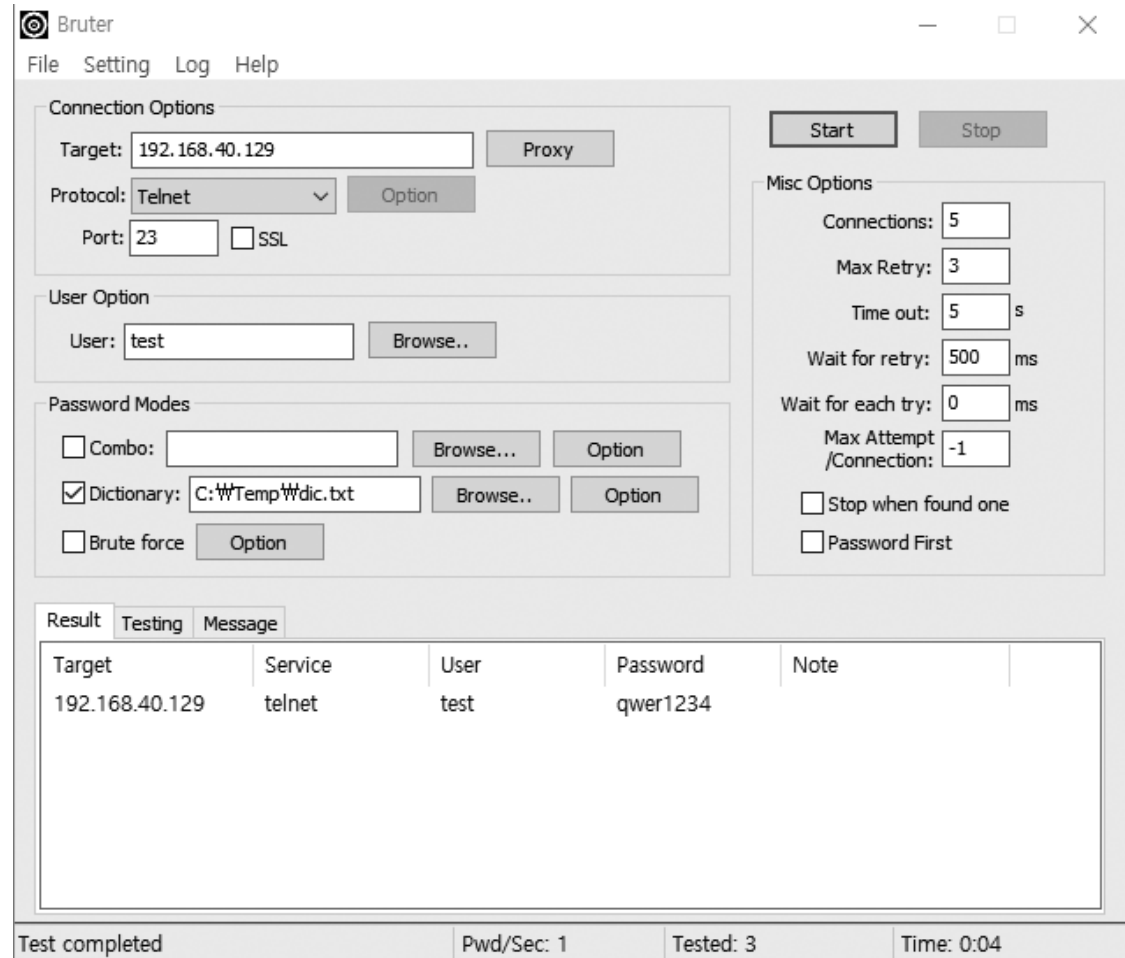


그림 4-31 telnet 데몬에서 Bruter 툴을 사용한 결과



Thank You !

IT CookBook, 정보 보안 개론과 실습 : 시스템 해킹과 보안(개정판)