2009 졸업연구 결과보고서

유닉스 계정&패스워드 관리체계 개선방안 연구

A study for UNIX ID & Password management system's improvement

팀 명 LEGEND

팀 원 김 동 규
김 민 수
민 경 훈

2009. 10. 21

중부대학교 정보보호학과

요 약 문

1. 연구제목

(국문)유닉스 계정&패스워드 관리체계 개선방안 연구

(영문)A study for UNIX ID & Password management system's improvement

2. 연구 목적 및 필요성

유닉스 운영체제에서는 로그인시 사용자의 계정과 패스워드를 입력받아 사용자를 인증하는데, 통상 사용자의 계정은 쉽게 노출됨에 따라 패스워드를 보호하는 것이 시스템의 안전에 중요한 요소가 된다.

따라서 다양한 크래킹 공격에도 안전한 패스워드를 사용하도록 운영체제에서 사용자가 패스워드를 입력하거나 변경할 때 패스워드의 취약점을 사전 검사하여 시스템의 보안에 기여할 수 있는 패스워드를 설정하도록 관리할 필요성이 제기되고 있다.

3. 연구 내용

유닉스 운영체제에서 사용자 계정 및 패스워드 입력, 패스워드 암호화(DES) 및 시스템 파일 갱신 절차 등을 분석하여 보안 취약요인을 도출한다.

패스워드 크래킹 방법 기술조사 및 응용방안을 연구하여 이를 기초로 유닉스 운영체제에서 보안에 견고한 패스워드 설정방안을 제안하고, 유닉스 로그인 프로세스의 시뮬레이터를 개발하여 연구내용을 시스템으로 구현한다.

이하 프로젝트(프로그램)는 Little login이라 표현함

4. 연구 결과

패스워드 보안 측정 및 평가 시스템을 구현해서 사용자 개인이 자신이 입력하는 계정과 패스워드가 어떻게 취약한지 그리고 어느 부분을 더 보완해서 사용할지를 알 수 있어, 개인 정보보호에 보템이 된다.

목 차

그림 목차

요약

유닉스 운영체제에서 사용자 계정 및 패스워드 입력 하였을 시 패스워드는 암호화(DES) 및 시스템파일(/etc/.passwd , /etc/.shadow) 갱신 절차 등을 분석하여 보안에 필요한 취약요인들을 도출한 내용을 나타낼 수 있다.
패스워드 크래킹 방법 등 기술조사 및 응용방안을 연구하여 이를 기초로 유닉스 운영체제에서 보안에 견고한 패스워드 설정방안을 제안한다.
유닉스 로그인 프로세스의 시뮬레이터를 개발하여 연구내용을 시스템으로 구현하는 것이다.
저희 졸업 작품은 기존에 사용되고 있는 유닉스의 운영체제를 더욱 더 안전한 관리체계를 운영하기 위한 프로그램을 만들어보았습니다.
이하 프로젝트(프로그램)는 Little login 이라 표현함

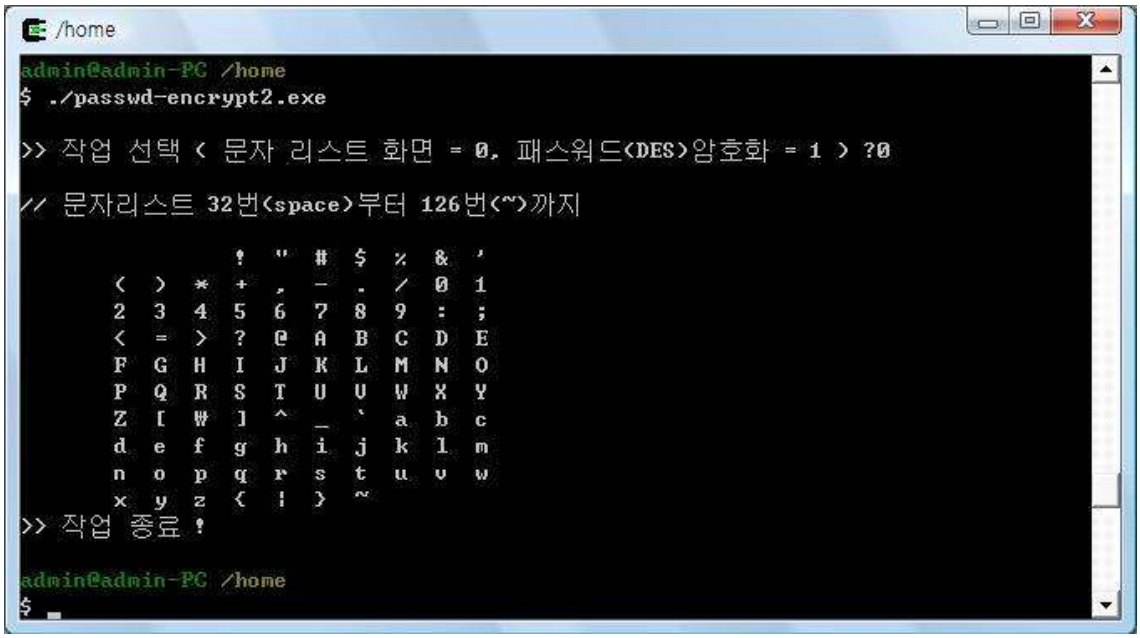
1. 연구의 목적

유닉스 운영체제에서는 로그인시 사용자의 계정과 패스워드를 입력받아 사용자를 인증하는데, 통상 사용자의 계정은 쉽게 노출됨에 다라 패스워드를 보호하는 것이 시스템의 안전에 중요한 요소가 된다.
따라서 다양한 크래킹 공격에도 안전한 패스워드를 사용하도록 운영체제에서 사용자가 패스워드를 입력하거나 변경할 때 패스워드의 취약점을 사전 검사하여 시스템의 보안에 기여할 수 있는 패스워드를 설정하도록 관리할 필요성이 있기 때문에 연구를 시작하였다.

2.연구방법

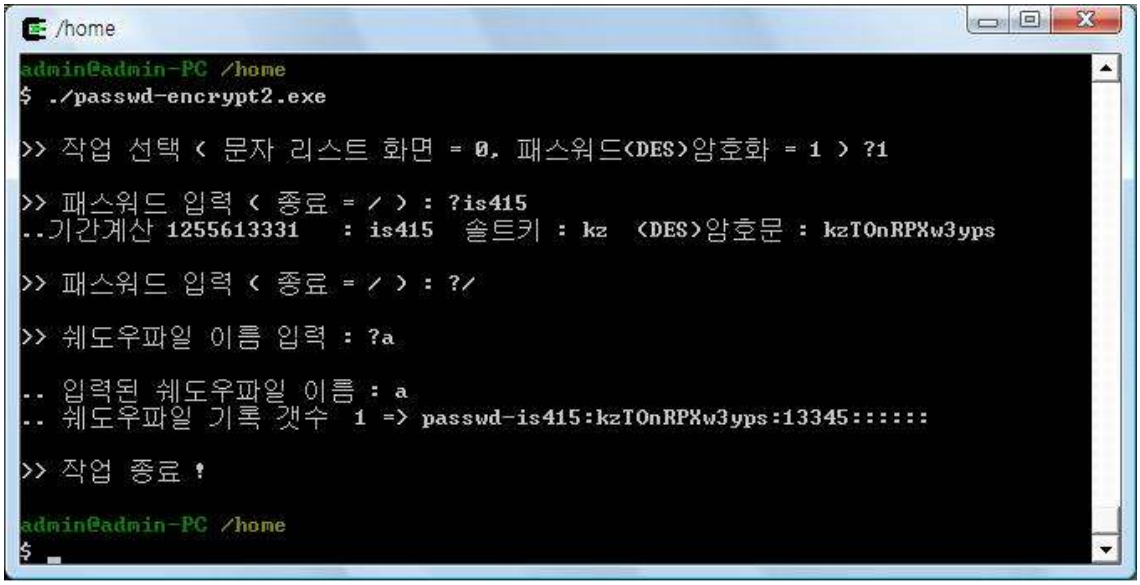
패스워드에 사용된 문자 리스트, 패스워드 암호화(DES) 및 기준일계산,
시스템파일 초기화 모듈, 로그인 모듈, 패스워드 수정 모듈, 시스템파일 디스플레이 및 업데이트 모듈, 개발모듈을 모두 통합 후 Little login 완성, 시스템 개발환경 등을 알아보겠다.

2.1 패스워드에 사용된 문자 리스트



(그림1-1) 암호화시 가용한 모든 캐릭터 수 화면

2.2 패스워드 암호화(DES) 및 기준일계산



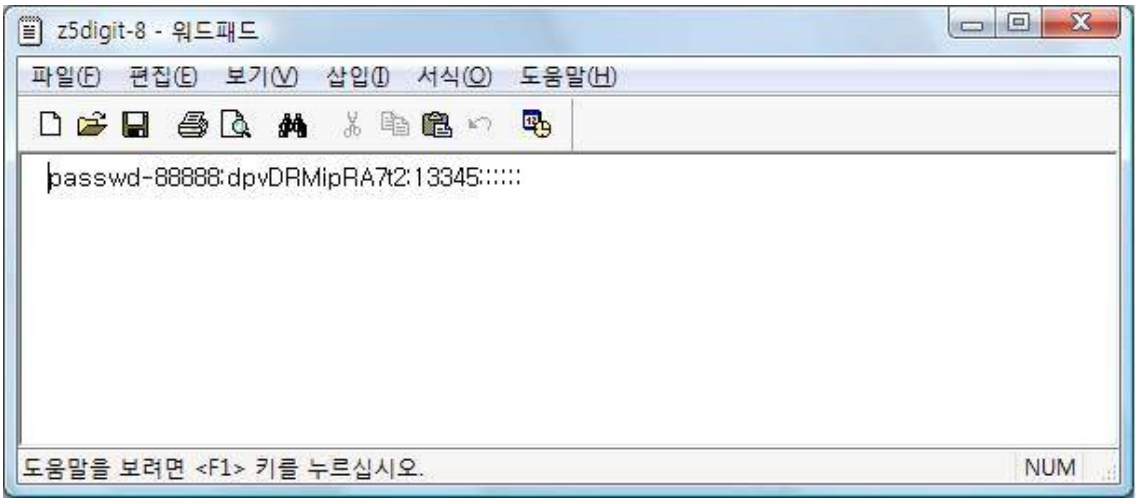
(그림1-2) 기준일계산과 솔트키생성 암호입력에대한 암호문생성

2.3 암호화된 shadow파일 및 shadow파일 크랙

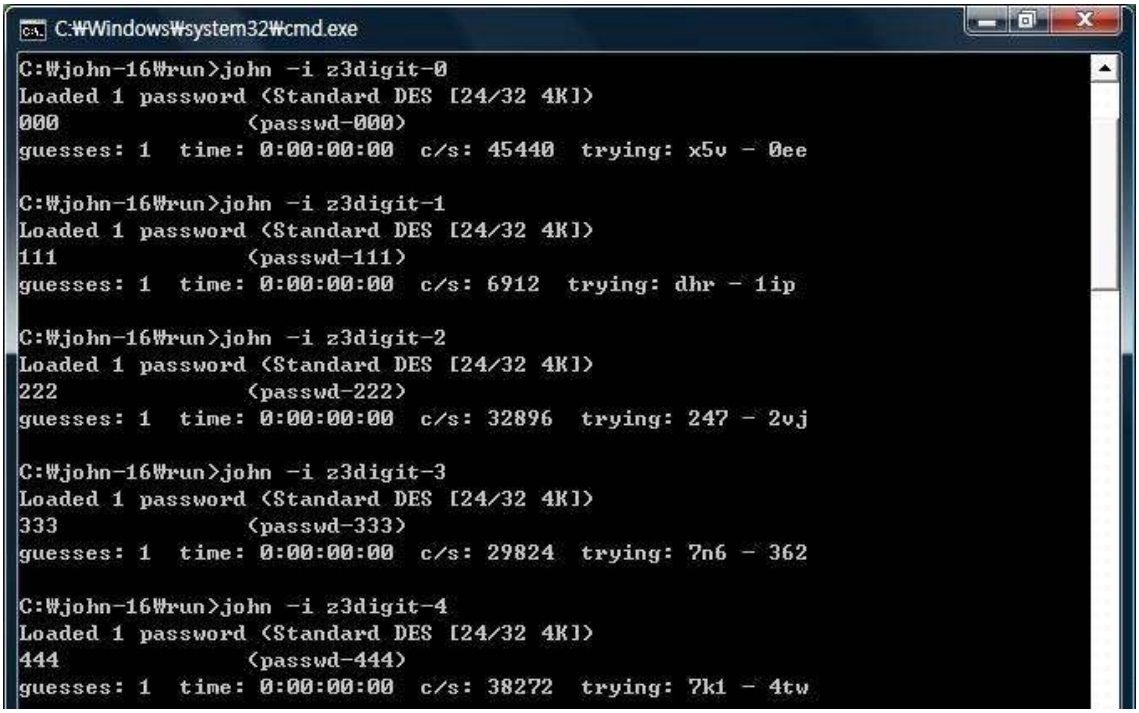
쉐도우파일			쉐도우파일 크랙		
규칙	개별	숫자,소문자,대문자,특수문자	규칙	개별	숫자,소문자,대문자,특수문자
	그룹	숫,소,대,특		그룹	숫,소,대,특
불규칙	개별	숫,소,대,특	불규칙	개별	숫,소,대,특
	그룹	숫,소,대,특		그룹	숫,소,대,특
종합	개별	3자리,4자리,5자리	종합	개별	3자리,4자리,5자리
	그룹	3,4,5자리		그룹	3,4,5자리

위와 같이 암호화된 shadow파일과 shadow파일크랙(John the Ripper활용)한 연구 결과를 공개한다.

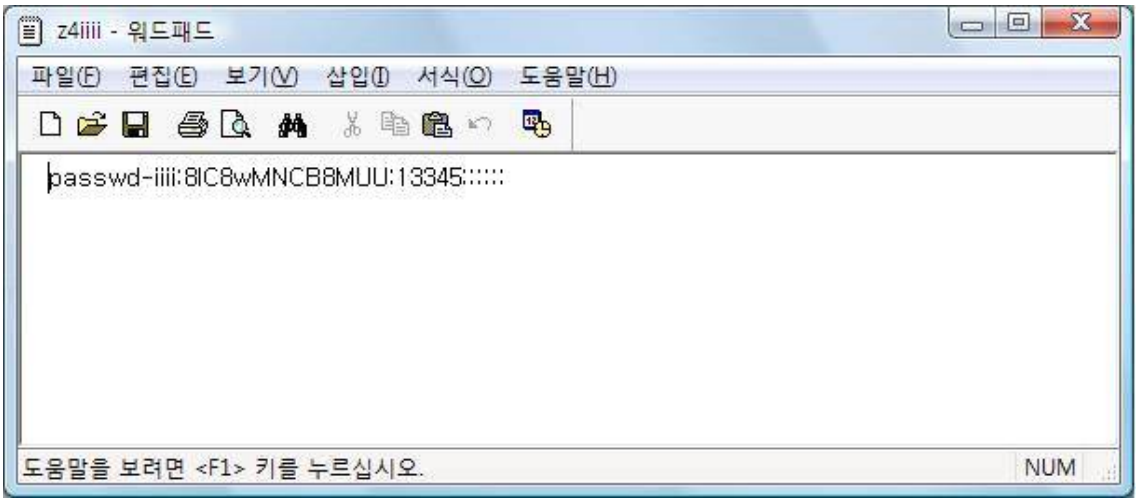
앞으로 공개할 그림 결과들은 개별적으로 된 입력된 패스워드, 그룹으로 입력된 패스워드, 각 숫자, 소문자, 대문자, 특수문자 또는 규칙, 불규칙, 종합적인 여부를 나눠서 보게 될 것인데 모두 공개하기엔 공간이 역부족이라 일부만을 공개한다.



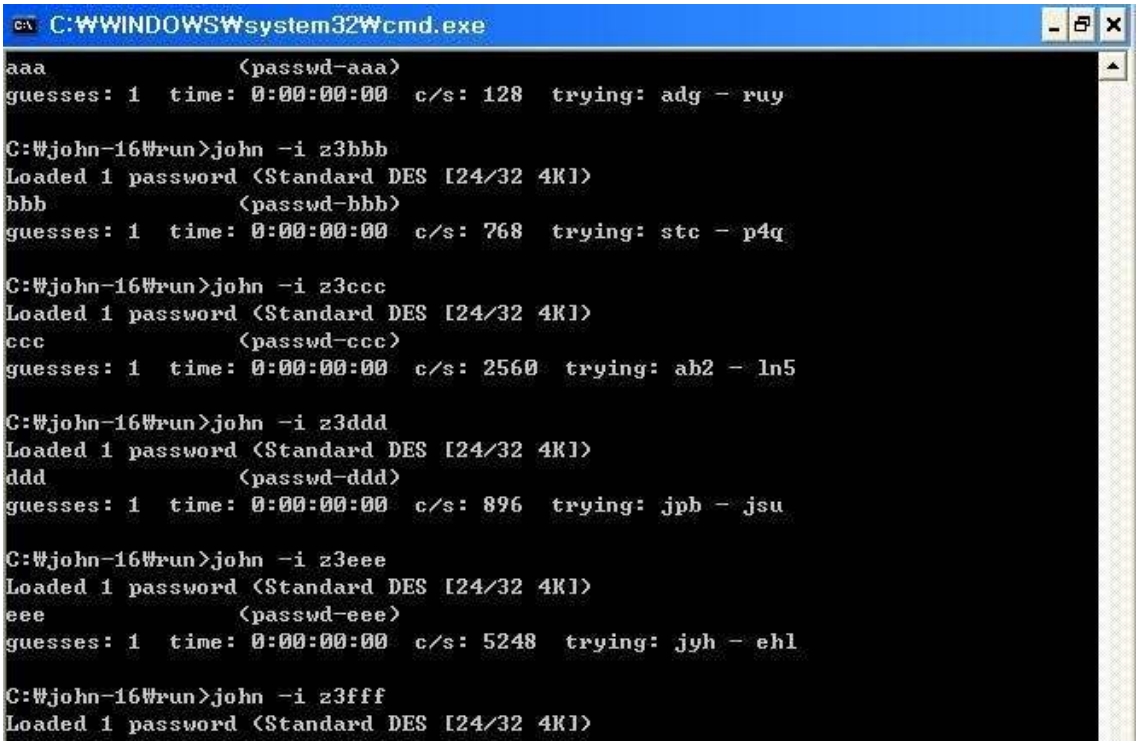
(그림2-1) 규칙적으로 입력된 개별적인 숫자 패스워드



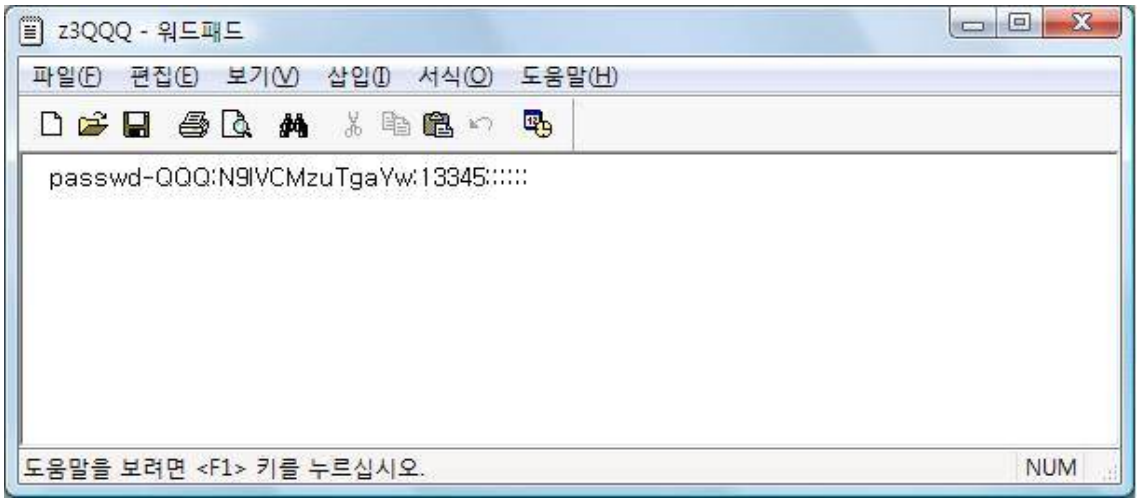
(그림2-1) 규칙숫자 쉐도우파일 크랙화면



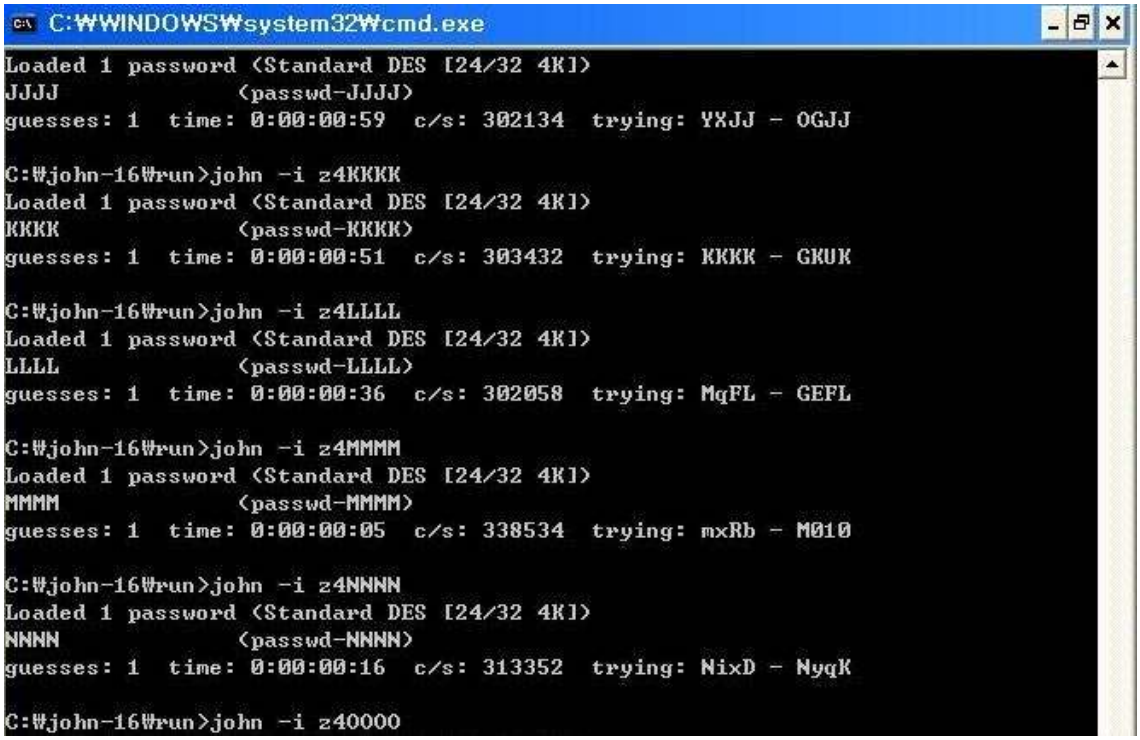
(그림2-2) 규칙적으로 입력된 개별적인 소문자 패스워드



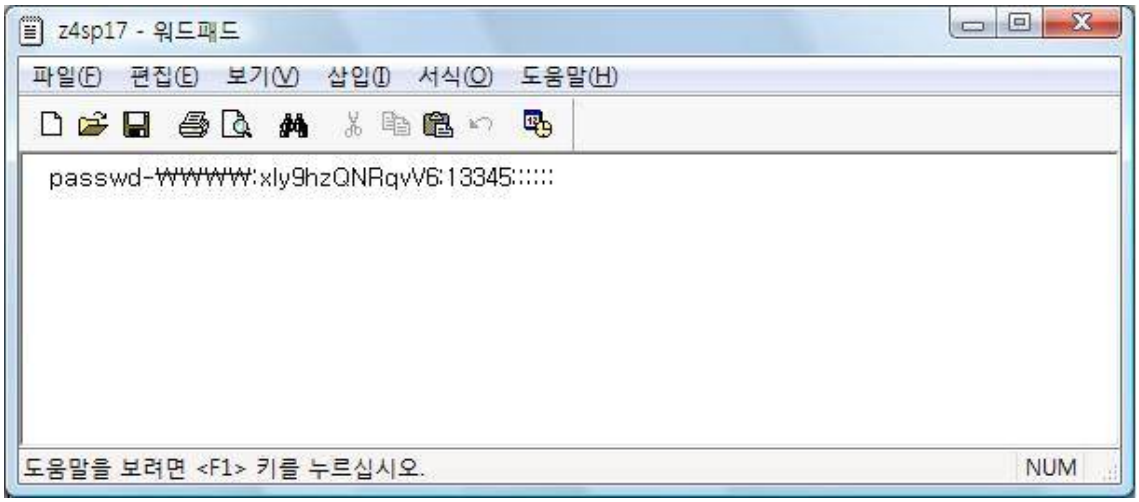
(그림2-2) 규칙소문자 쉐도우파일 크랙화면



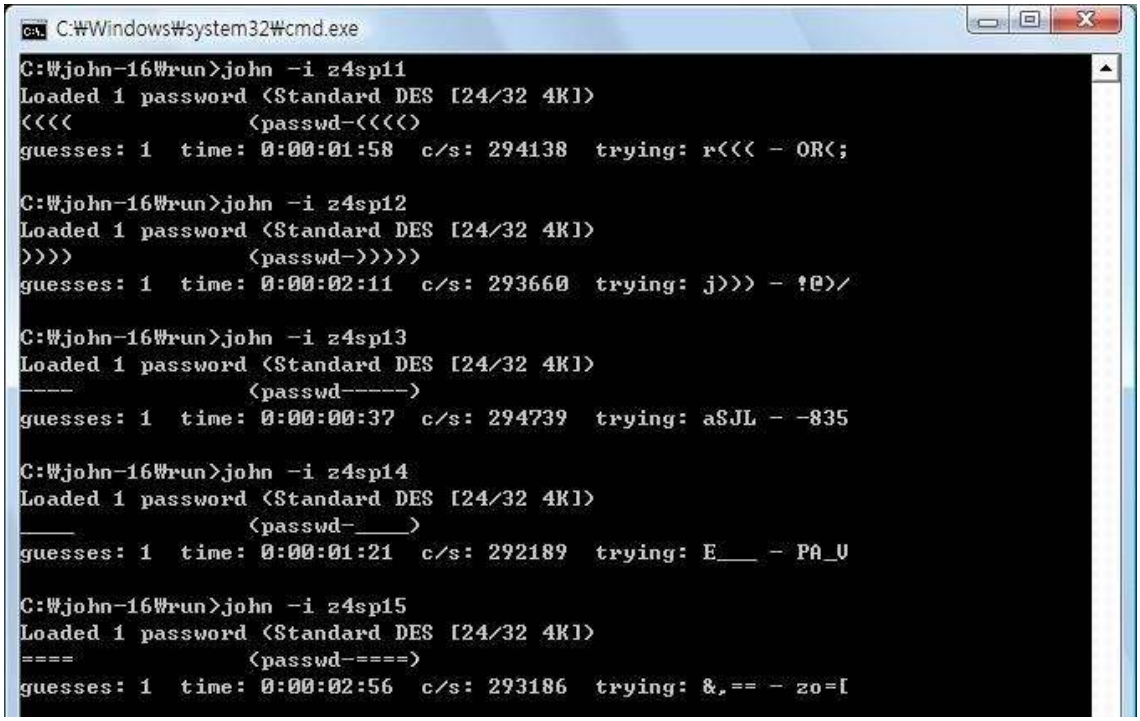
(그림2-3) 규칙적으로 입력된 개별적인 대문자 패스워드



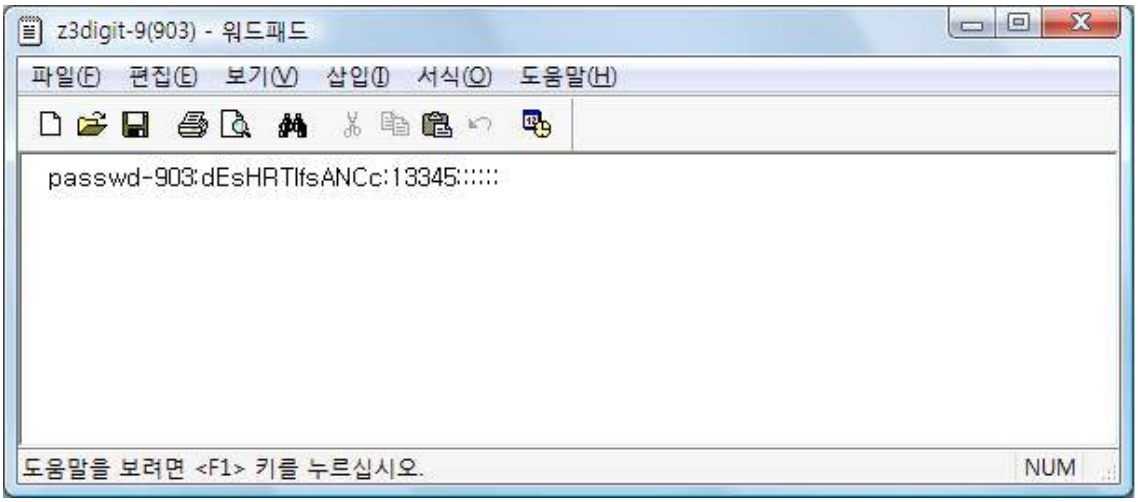
(그림2-3) 규칙대문자 쉐도우파일 크랙화면



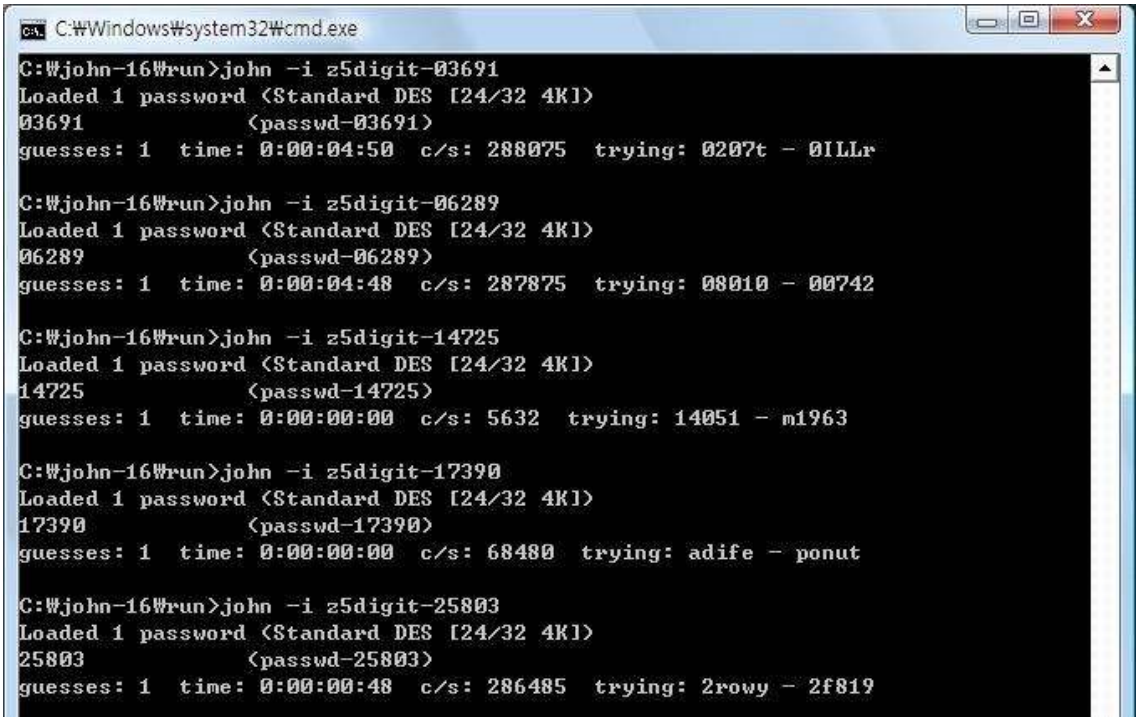
(그림2-4) 규칙적으로 입력된 개별적인 특수문자 패스워드



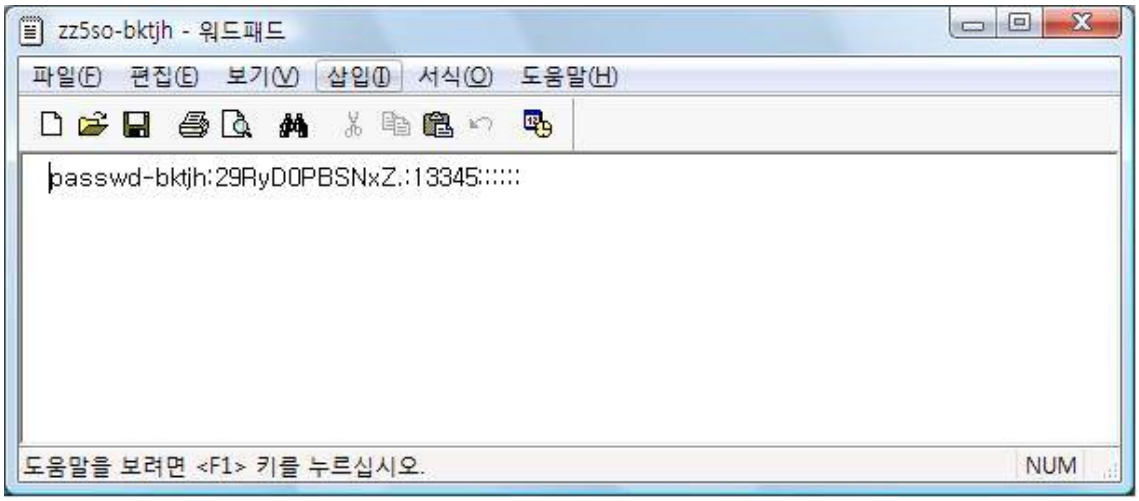
(그림2-4) 규칙특수문자 쉐도우파일 크랙화면



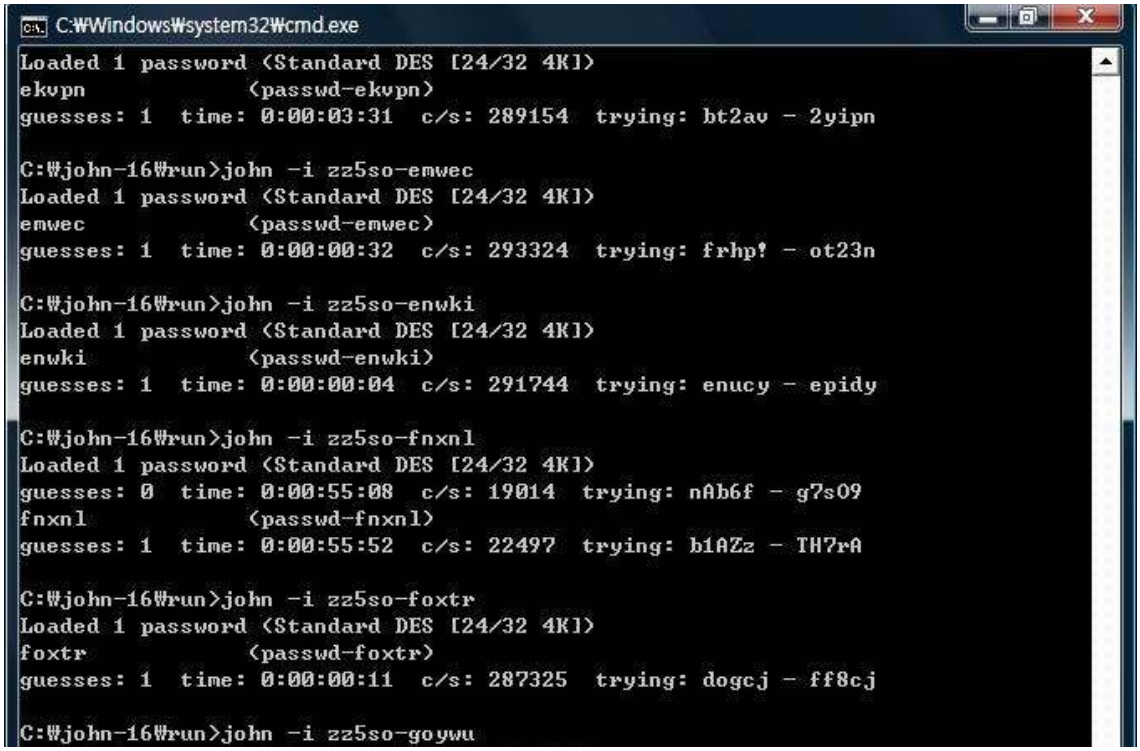
(그림2-5) 불규칙적으로 입력된 개별적인 숫자 패스워드



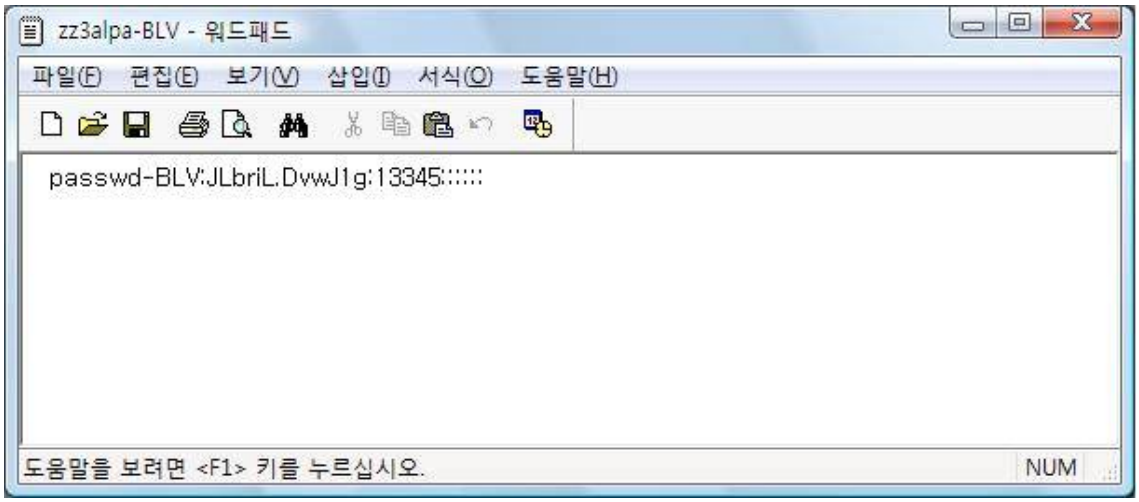
(그림2-5) 불규칙숫자 쉐도우파일 크랙화면



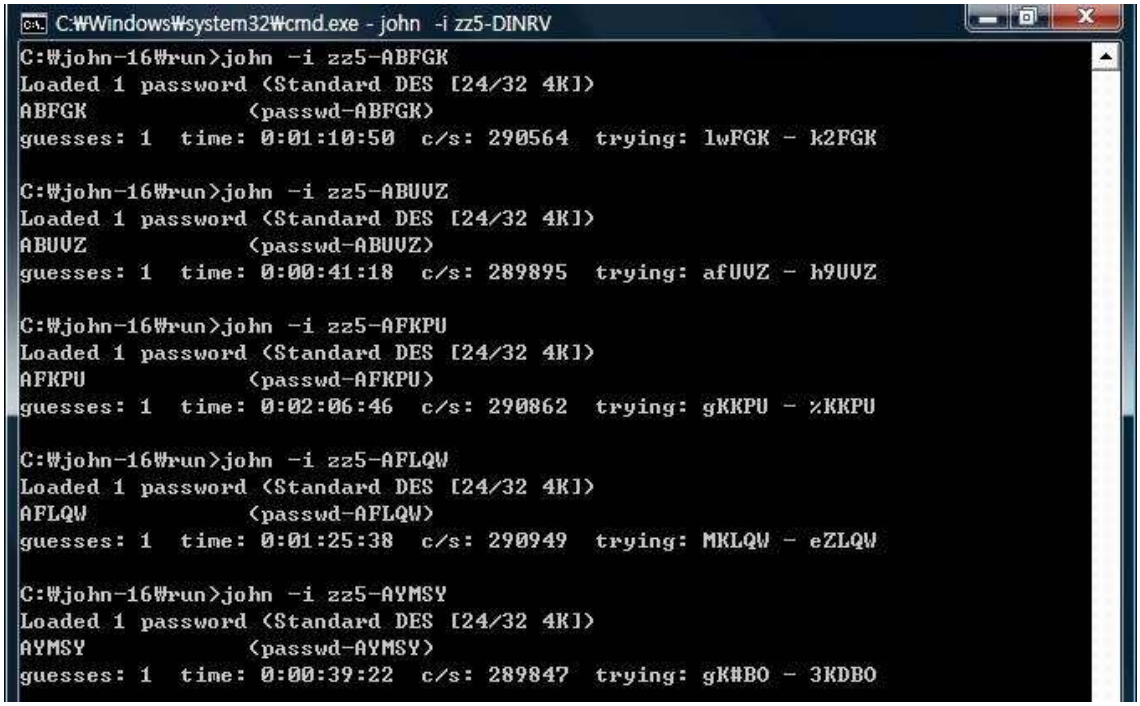
(그림2-6) 불규칙적으로 입력된 개별적인 소문자 패스워드



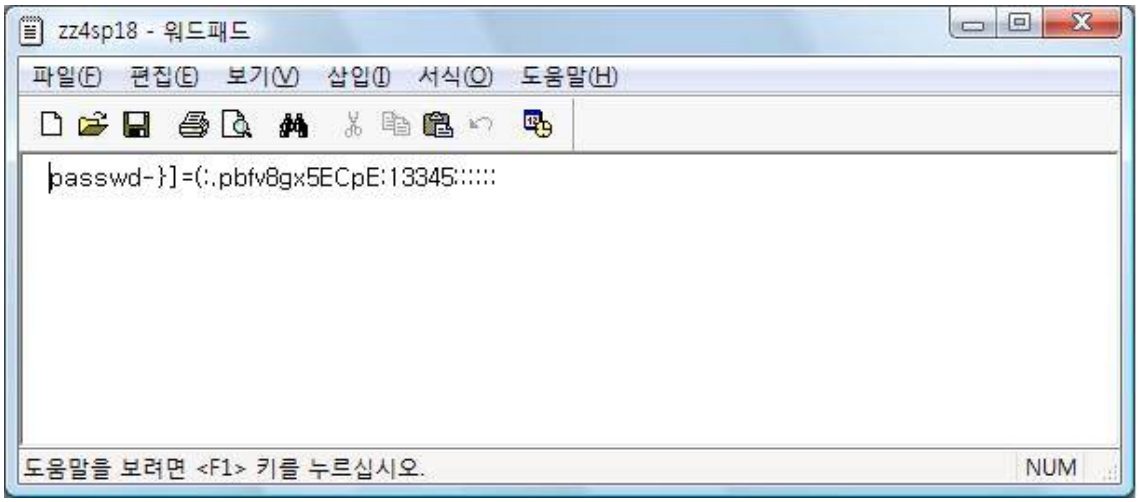
(그림2-6) 불규칙소문자 쉐도우파일 크랙화면



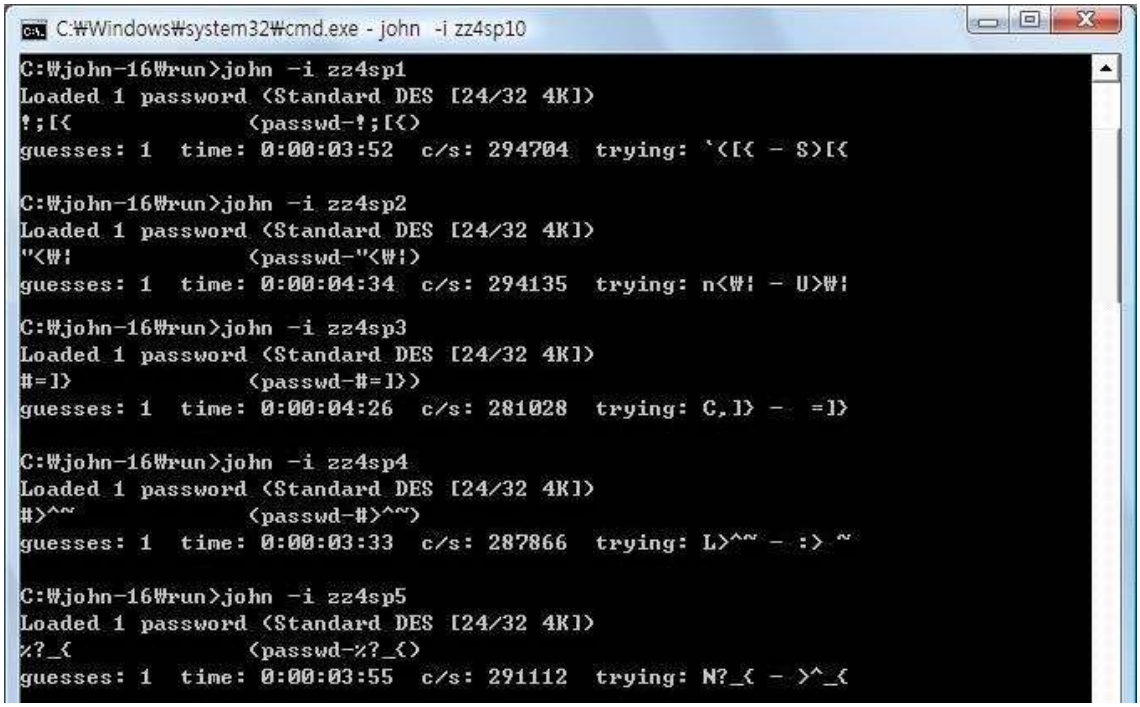
(그림2-7) 불규칙적으로 입력된 개별적인 대문자 패스워드



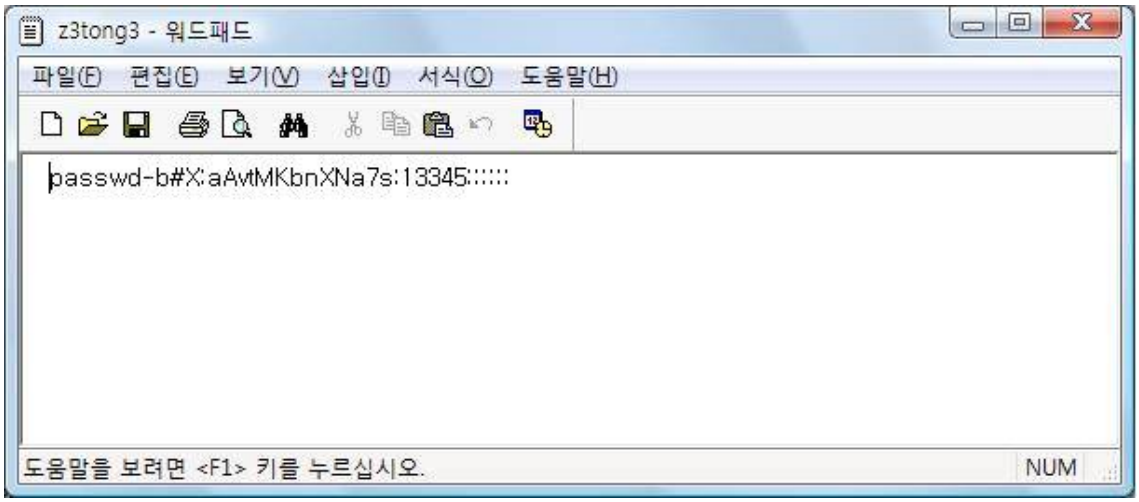
(그림2-7) 불규칙대문자 쉐도우파일 크랙화면



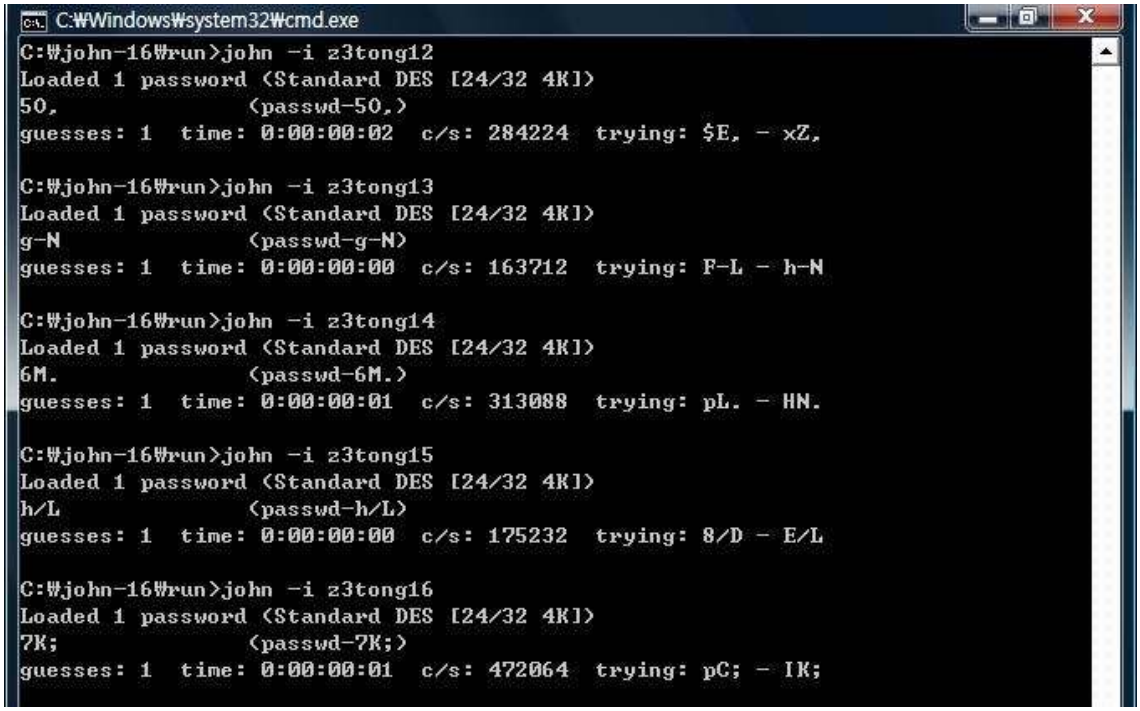
(그림2-8) 불규칙적으로 입력된 개별적인 특수문자 패스워드



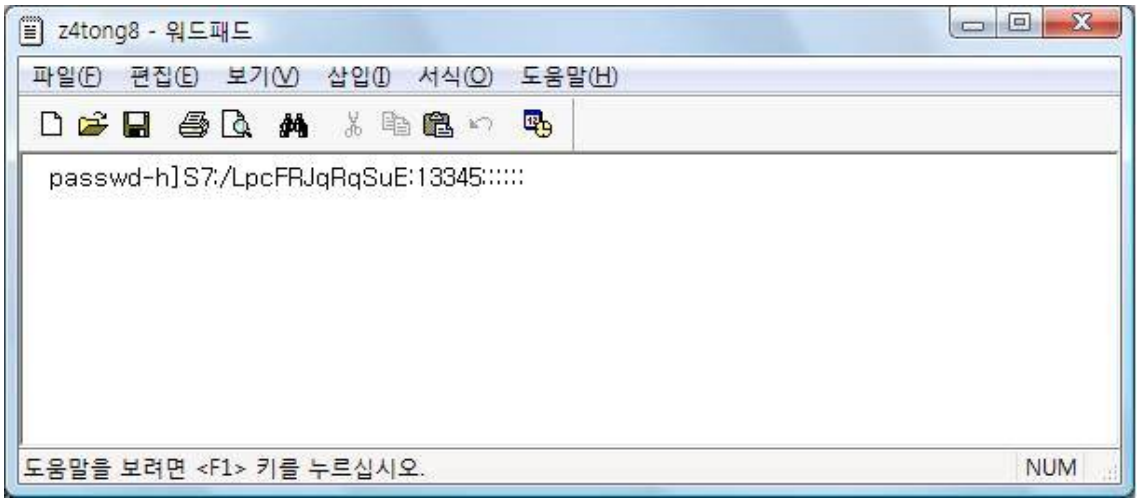
(그림2-8) 불규칙특수문자 쉐도우파일 크랙화면



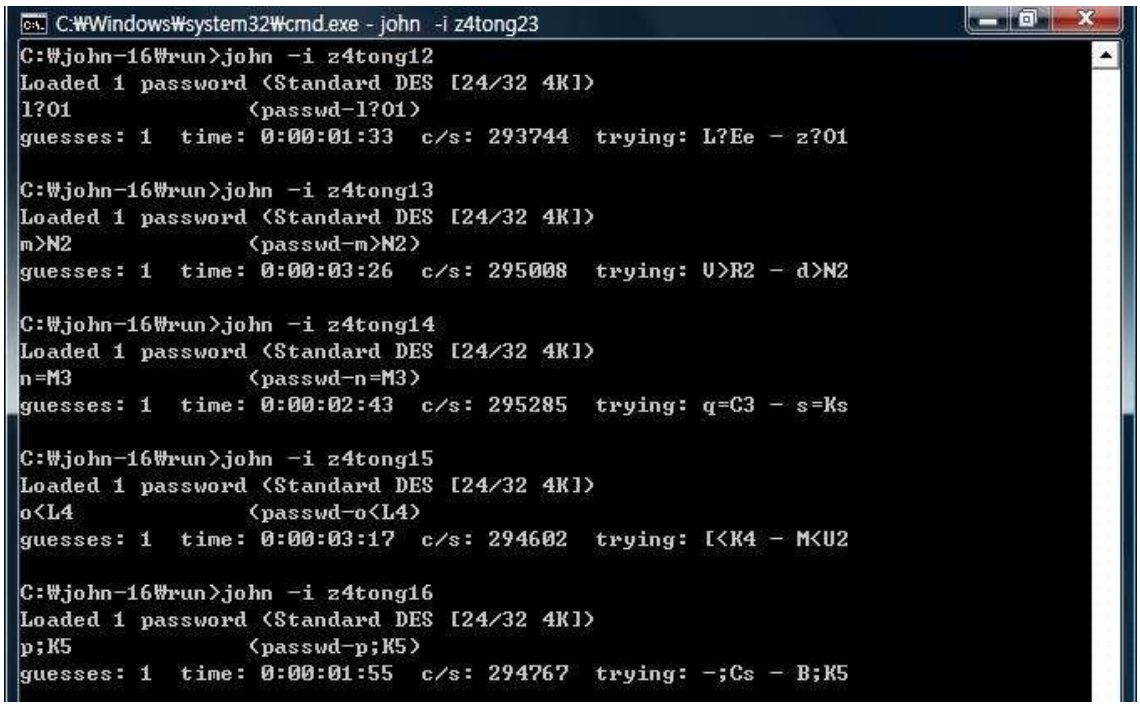
(그림2-9) 3자리로 구성된 개별적 종합 패스워드



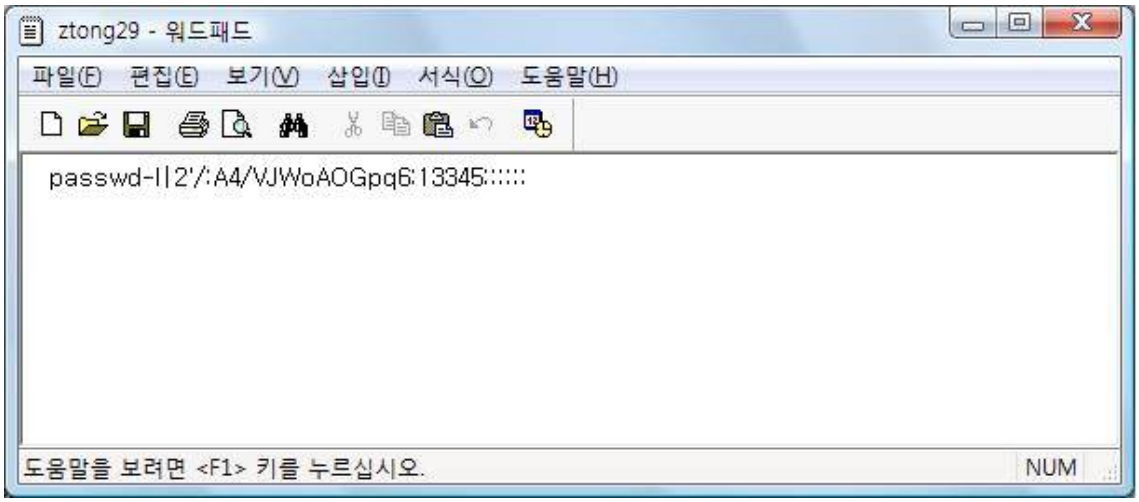
(그림2-9) 3자리 쉐도우파일 크랙화면



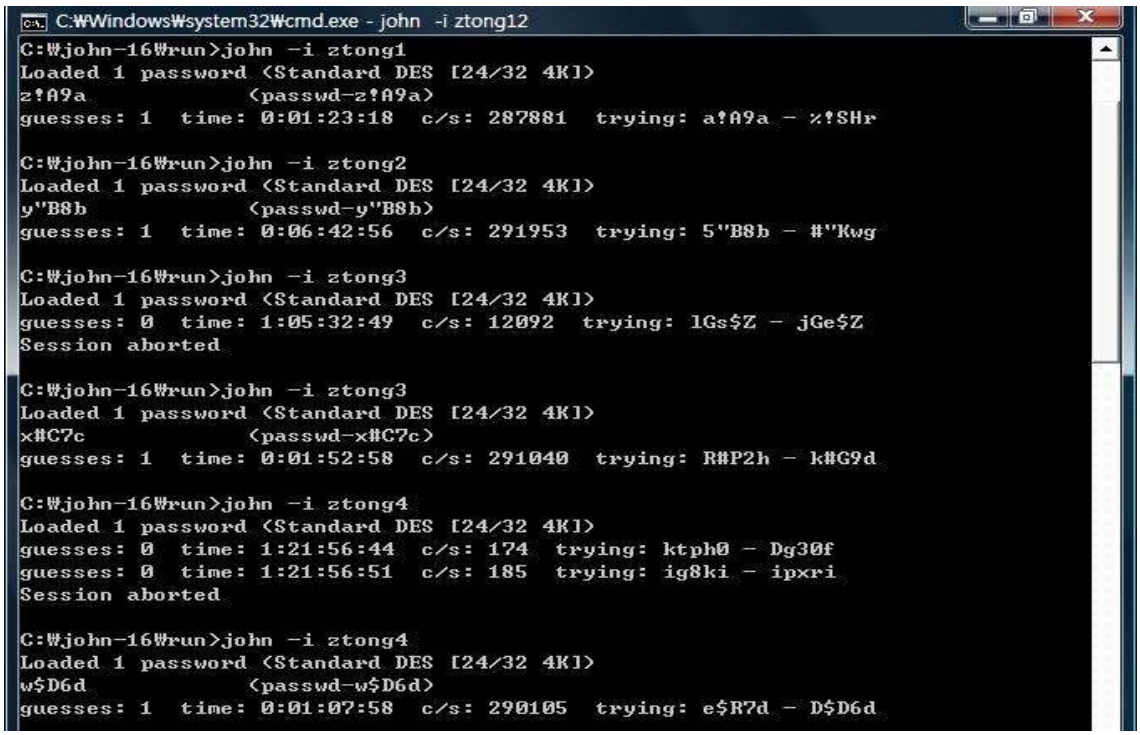
(그림2-10) 4자리로 구성된 개별적 종합 패스워드



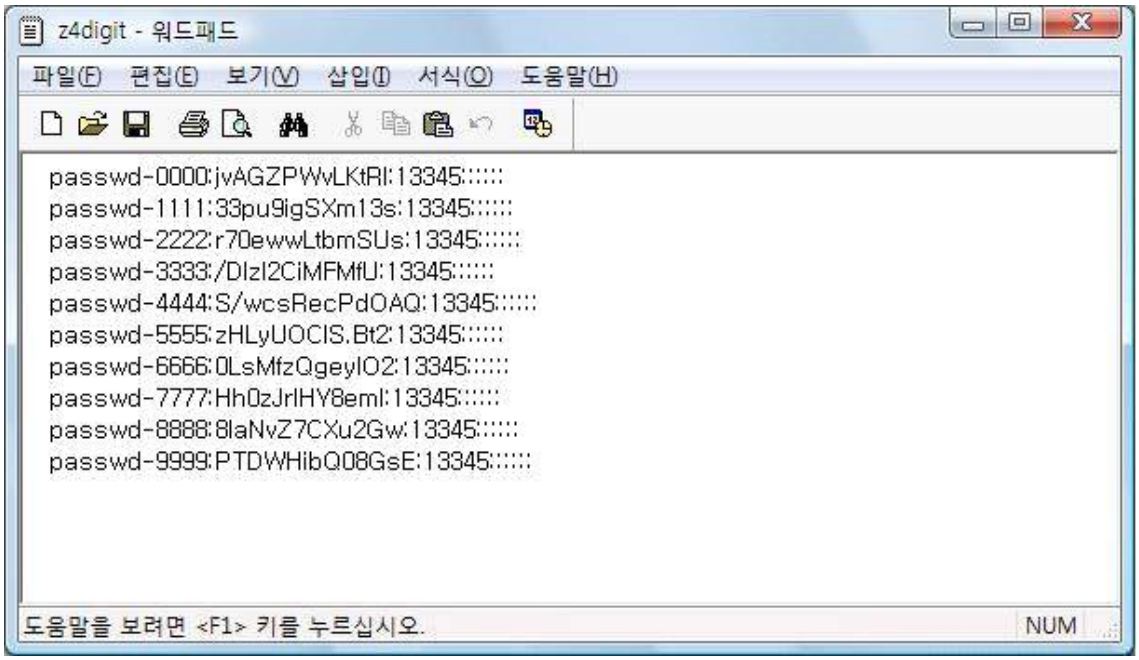
(그림2-10) 4자리 쉐도우파일 크랙화면



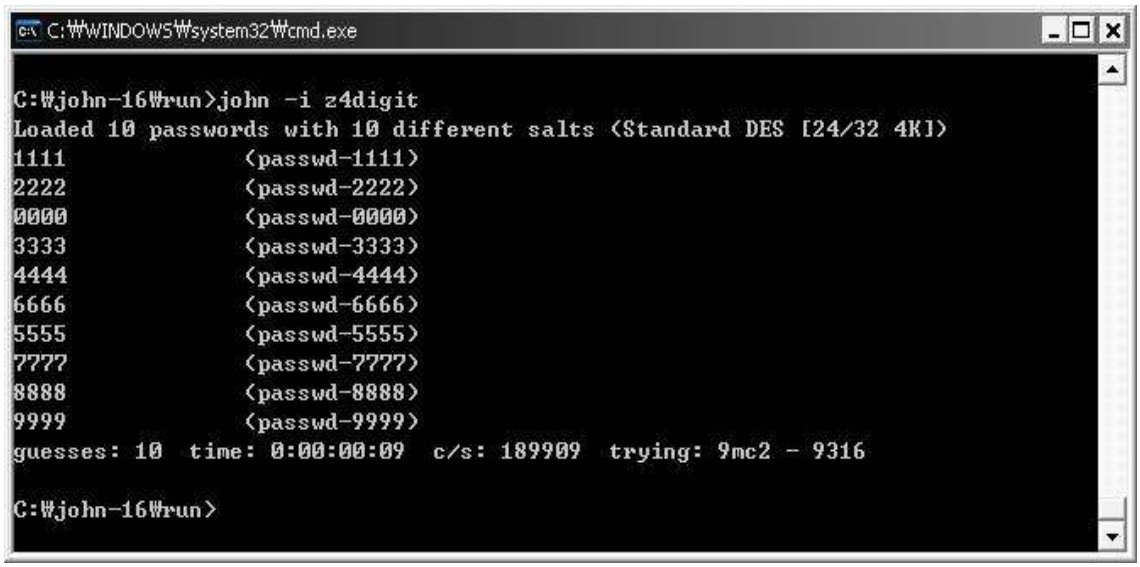
(그림2-11) 5자리로 구성된 개별적 종합 패스워드



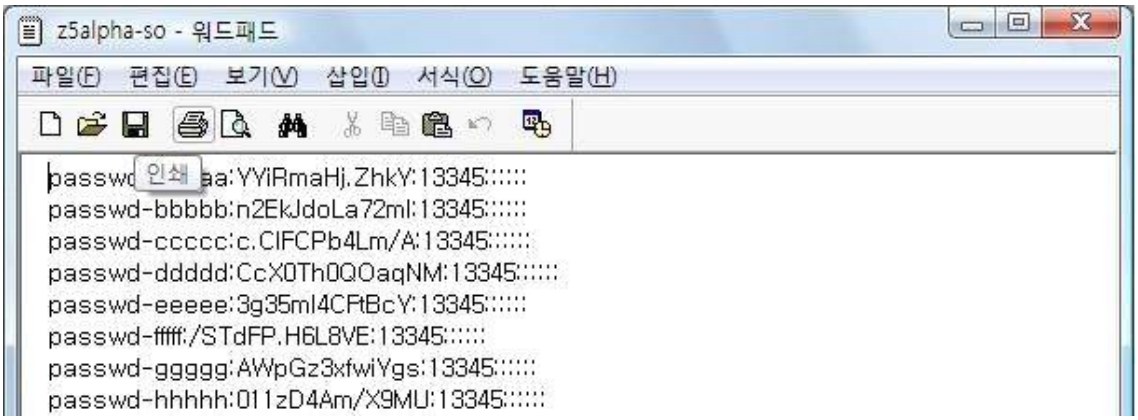
(그림2-11) 5자리 쉐도우파일 크랙화면



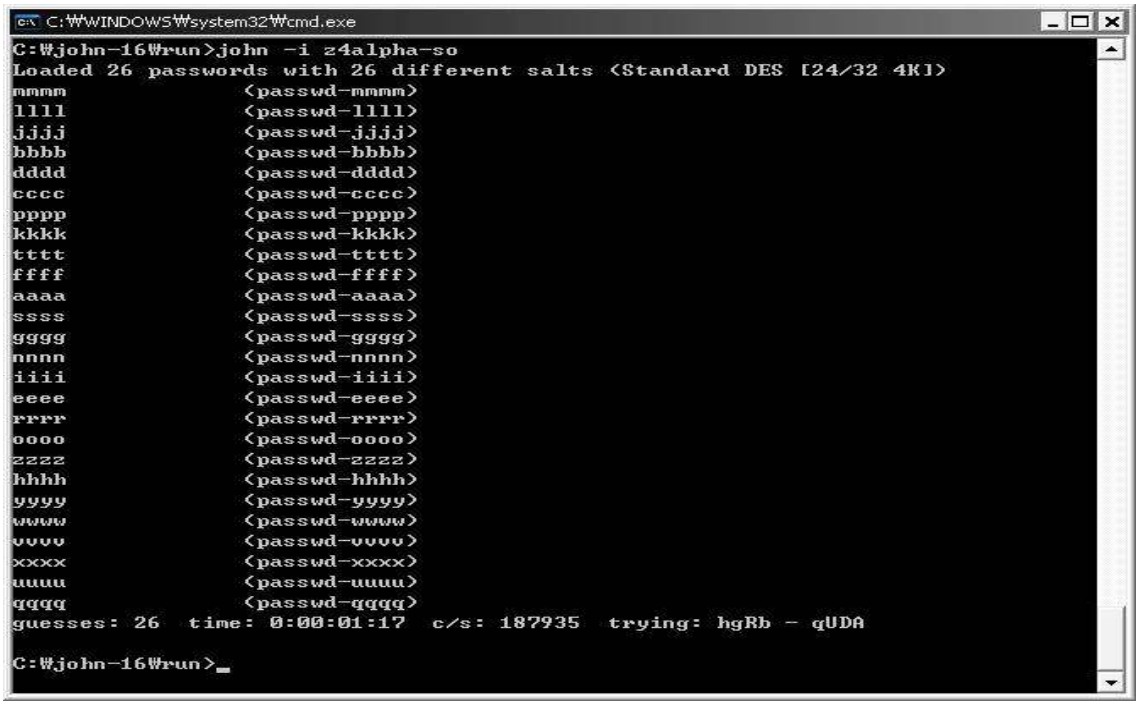
(그림2-12) 규칙적으로 입력된 그룹 숫자 패스워드



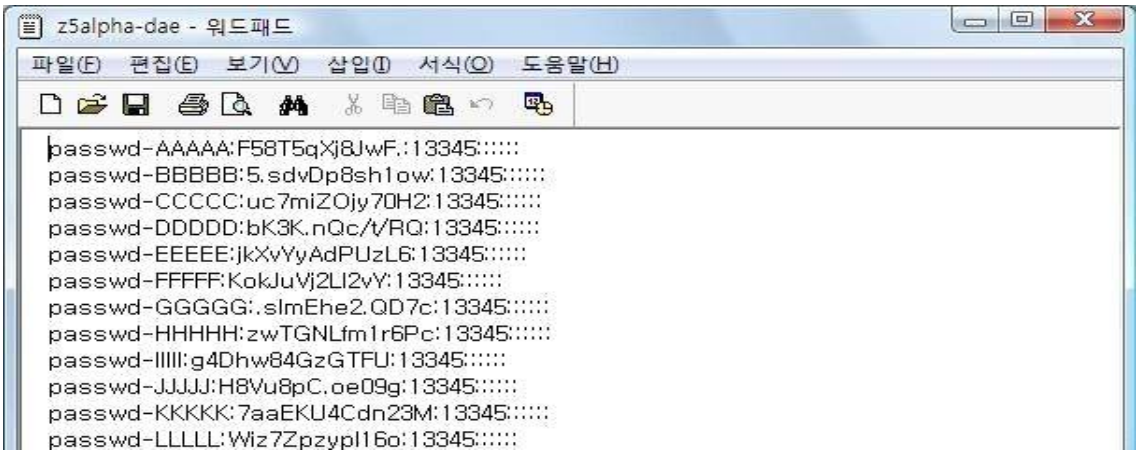
(그림2-12) 규칙그룹 숫자 쉐도우파일 크랙화면



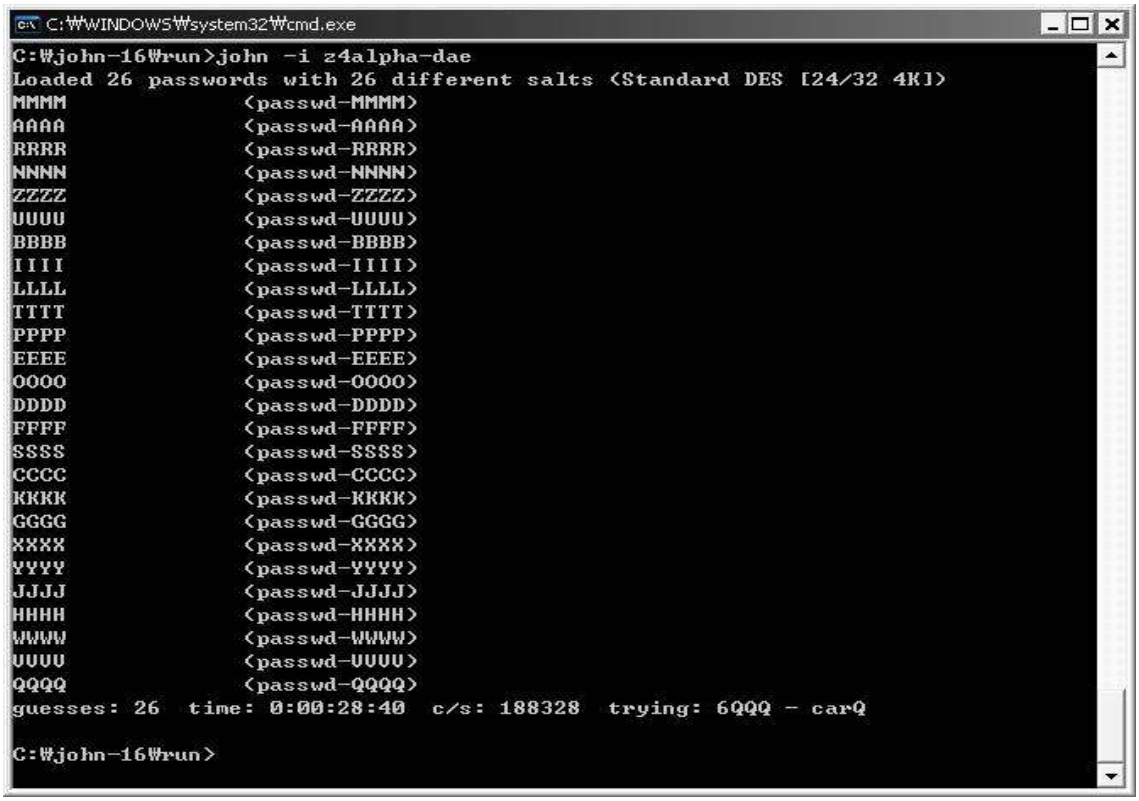
(그림2-13) 규칙적으로 입력된 그룹 소문자 패스워드



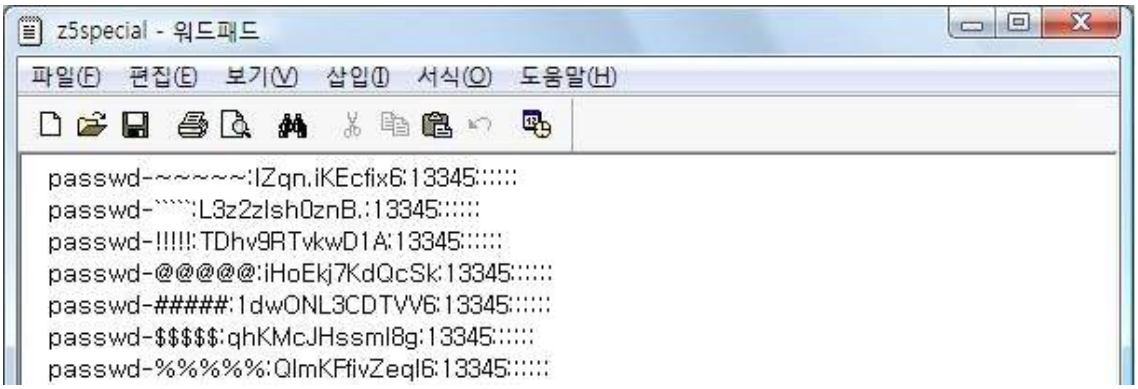
(그림2-13) 규칙그룹 소문자 쉐도우파일 크랙화면



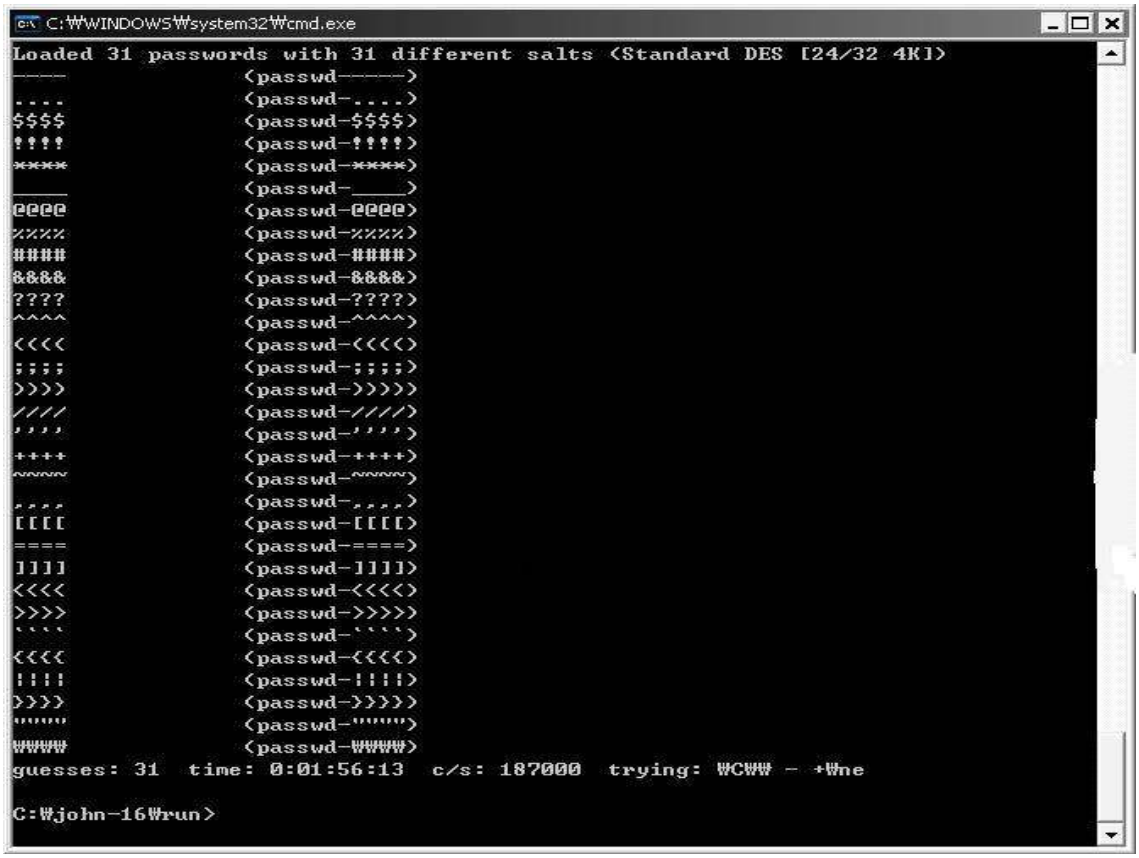
(그림2-14) 규칙적으로 입력된 그룹 대문자 패스워드



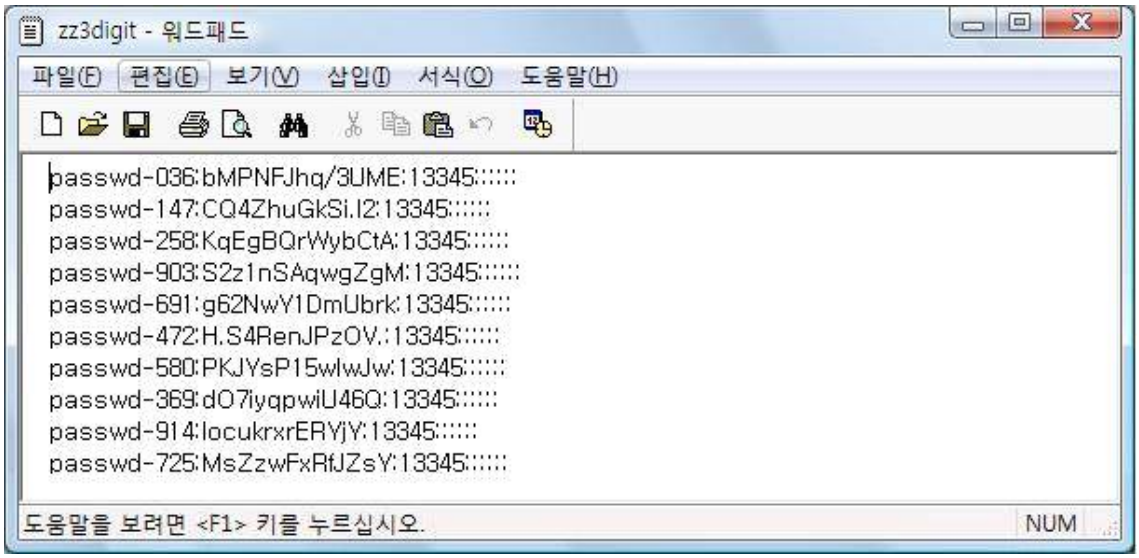
(그림2-14) 규칙그룹 대문자 쉐도우파일 크랙화면



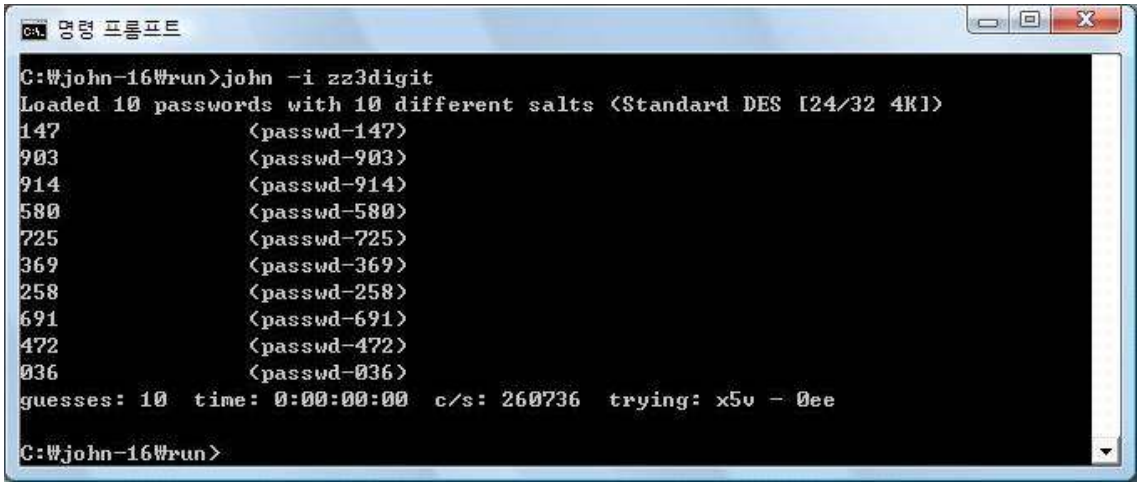
(그림2-15) 규칙적으로 입력된 그룹 특수문자 패스워드



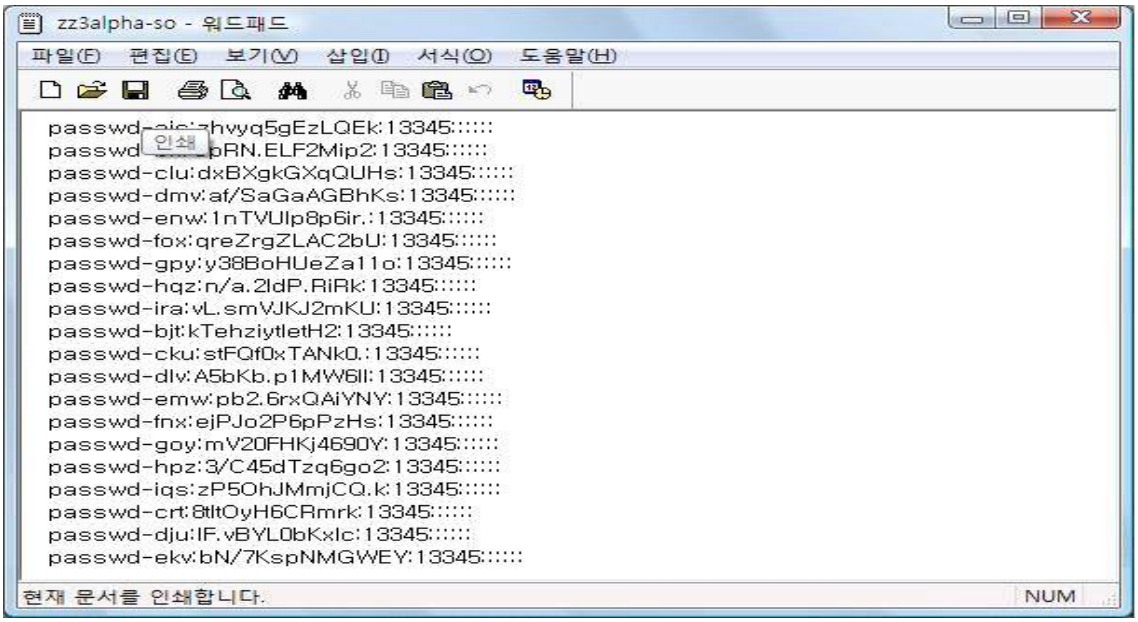
(그림2-15) 규칙그룹 특수문자 쉐도우파일 크랙화면



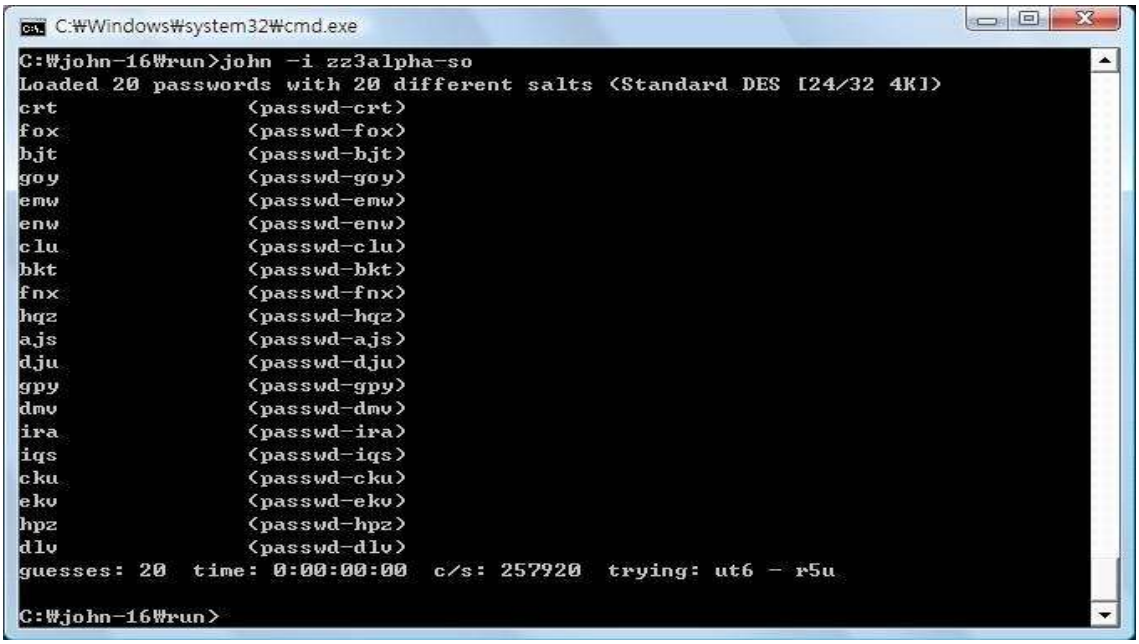
(그림2-16) 불규칙적으로 입력된 그룹 숫자 패스워드



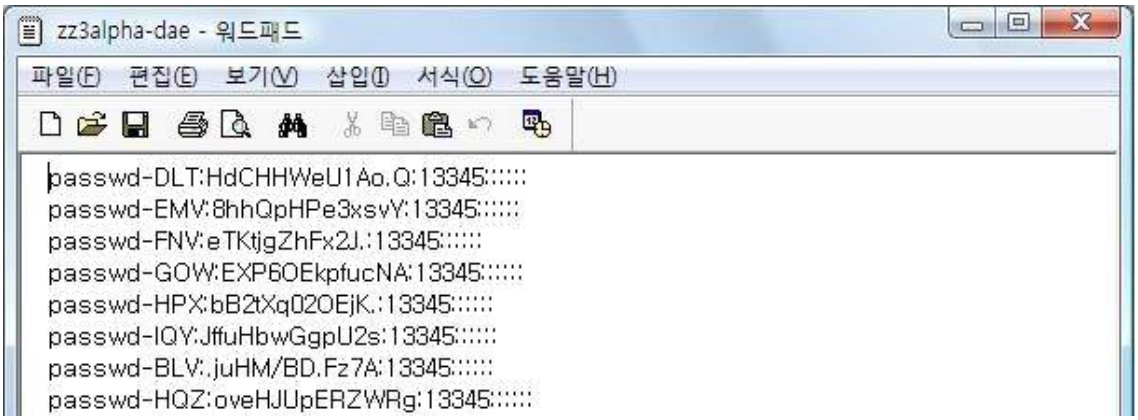
(그림2-16) 불규칙그룹 숫자 쉐도우파일 크랙화면



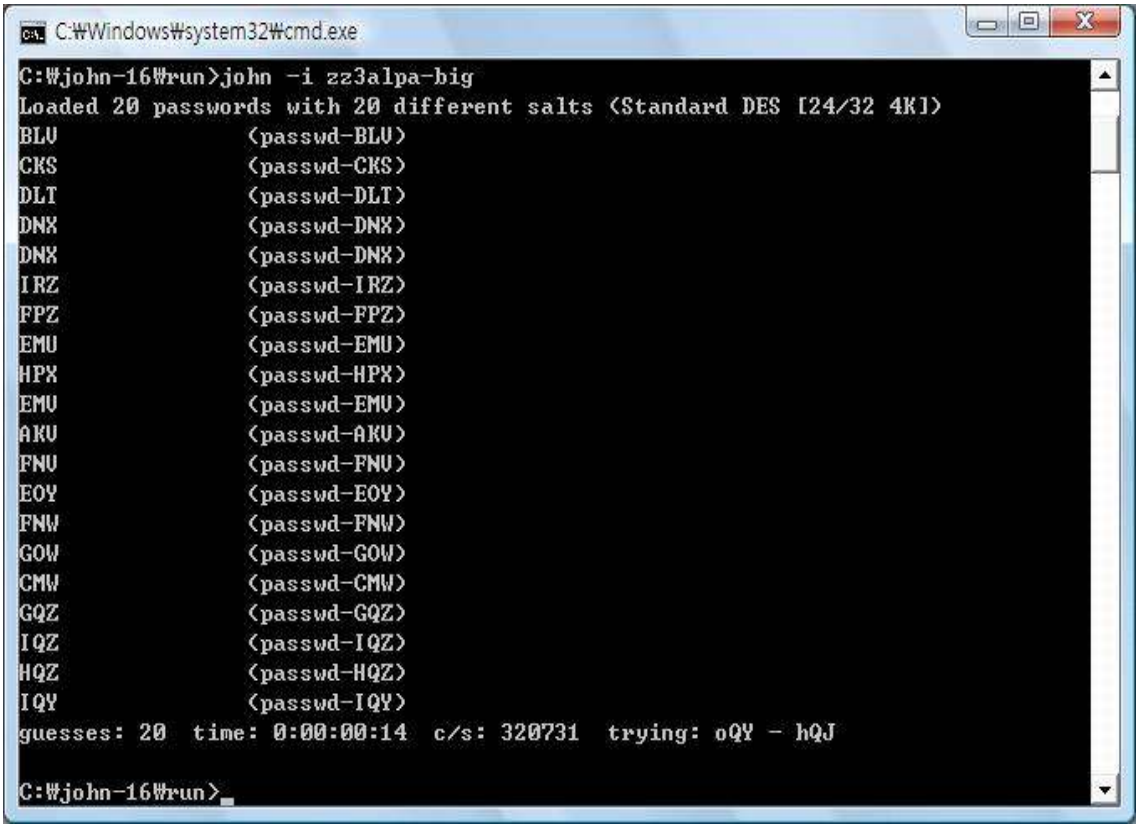
(그림2-17) 불규칙적으로 입력된 그룹 소문자 패스워드



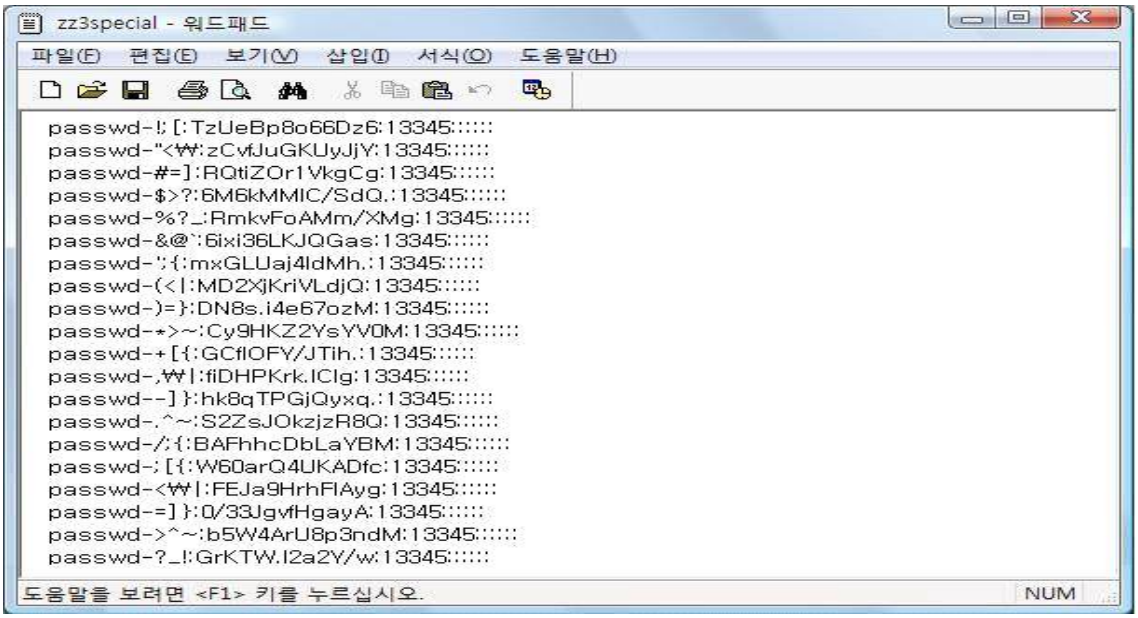
(그림2-17) 불규칙그룹 소문자 쉐도우파일 크랙화면



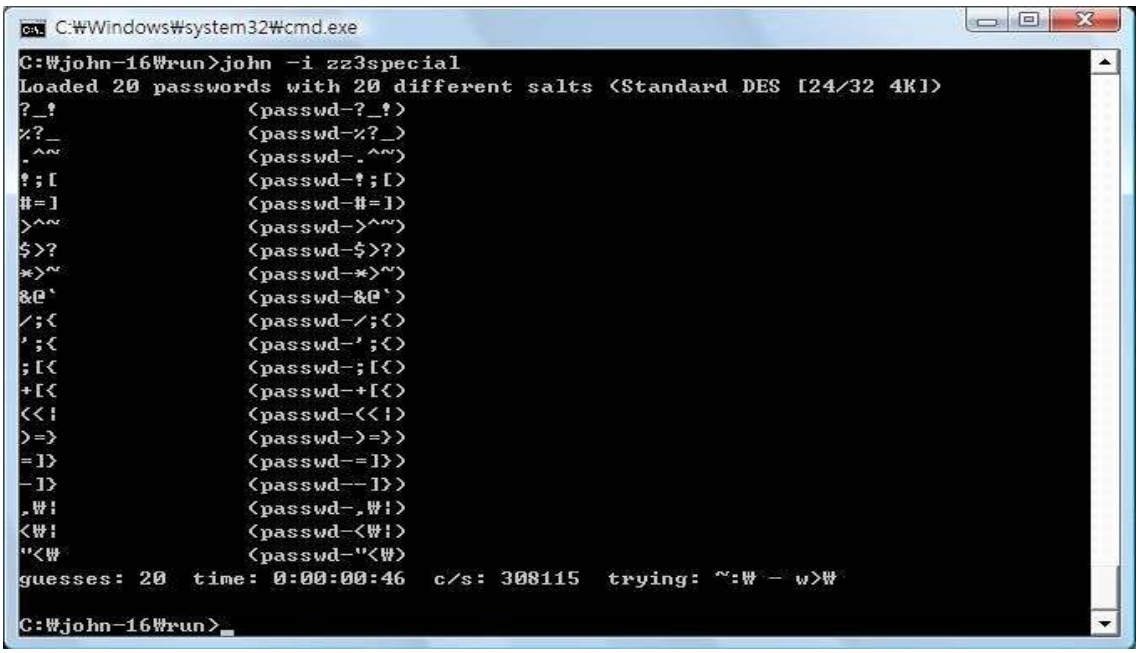
(그림2-18) 불규칙적으로 입력된 그룹 대문자 패스워드



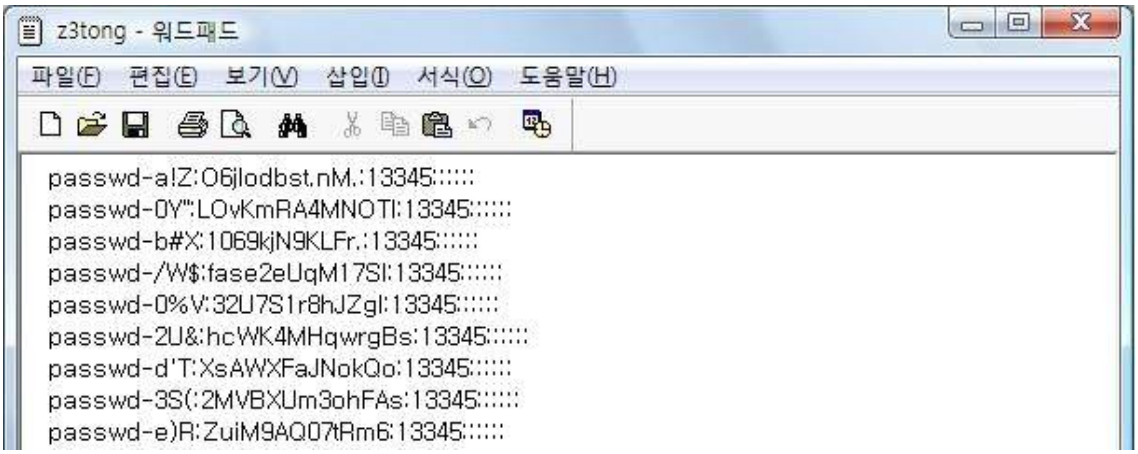
(그림2-18) 불규칙그룹 대문자 쉐도우파일 크랙화면



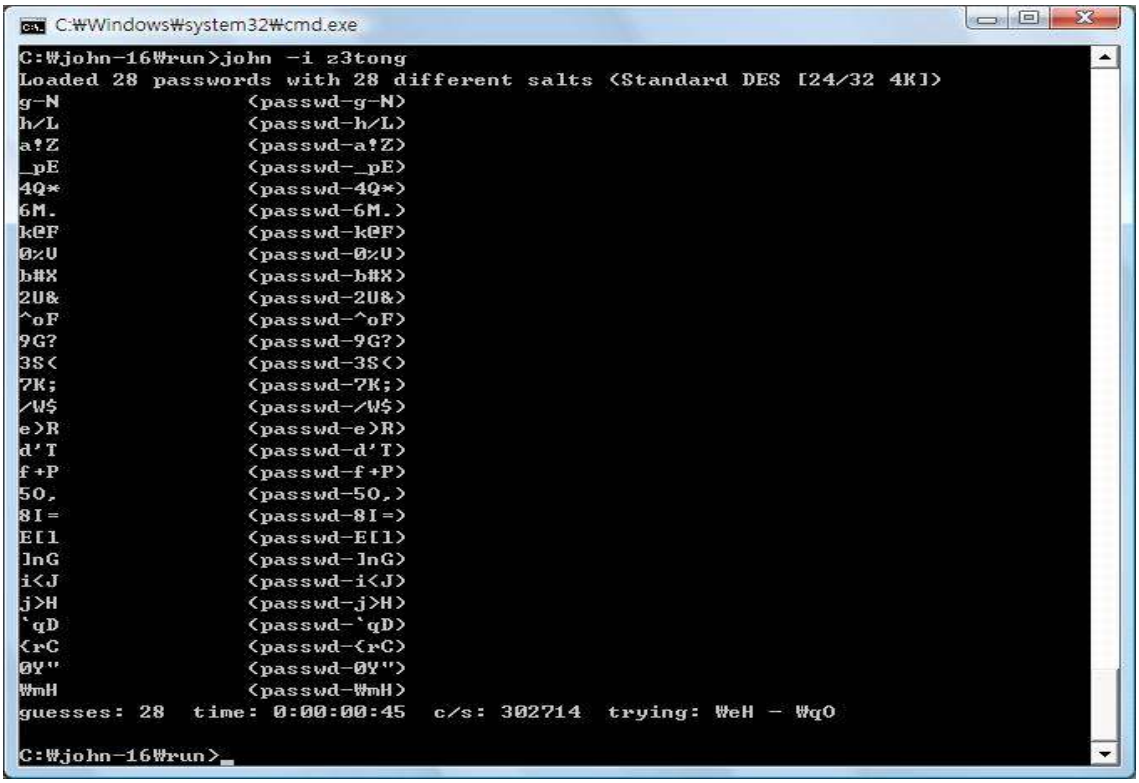
(그림2-19) 불규칙적으로 입력된 그룹 특수문자 패스워드



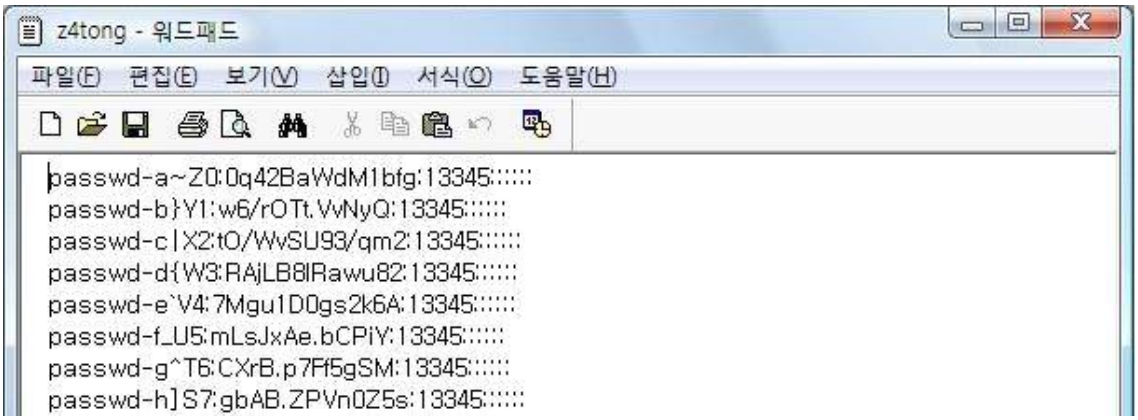
(그림2-19) 불규칙그룹 특수문자 쉐도우파일 크랙화면



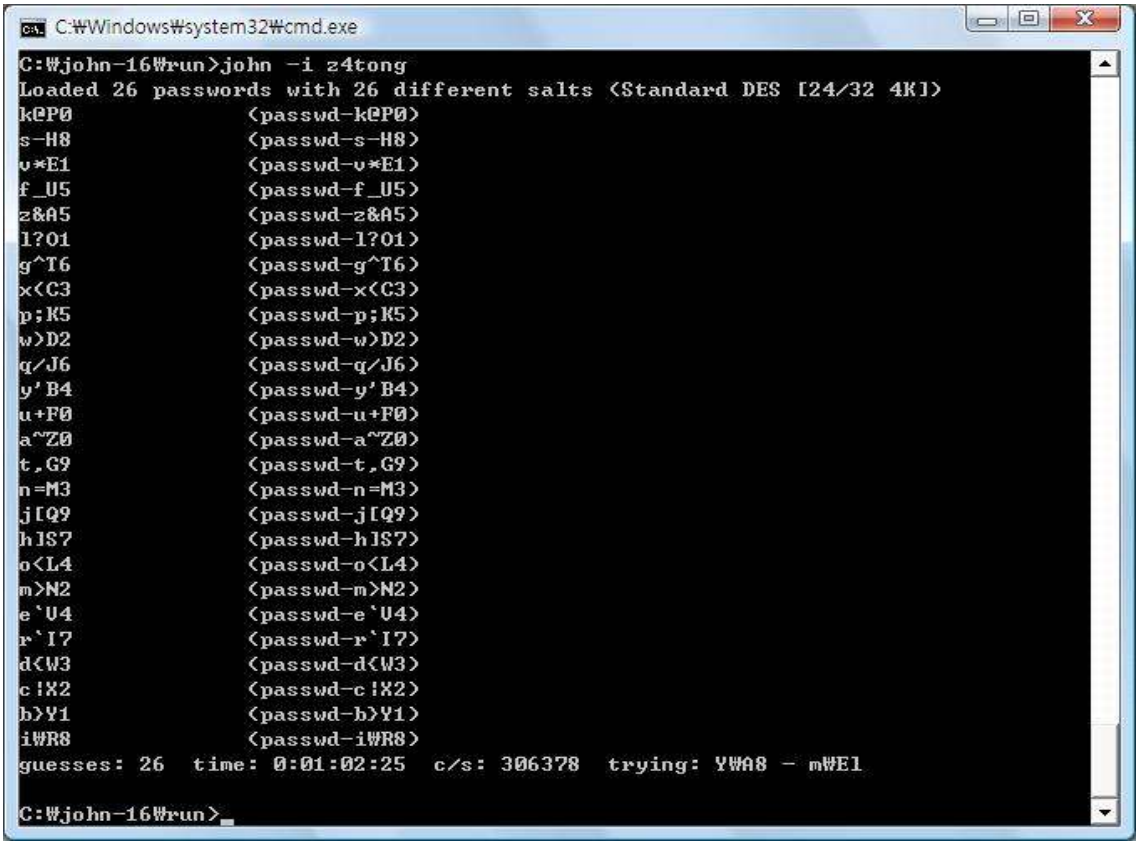
(그림2-20) 3자리로 구성된 그룹 종합 패스워드



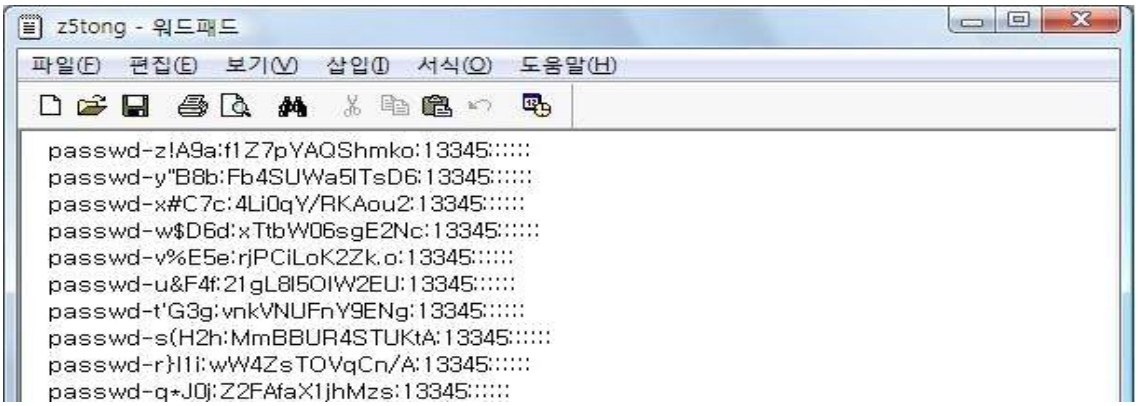
(그림2-20) 3자리 쉐도우파일 크랙화면



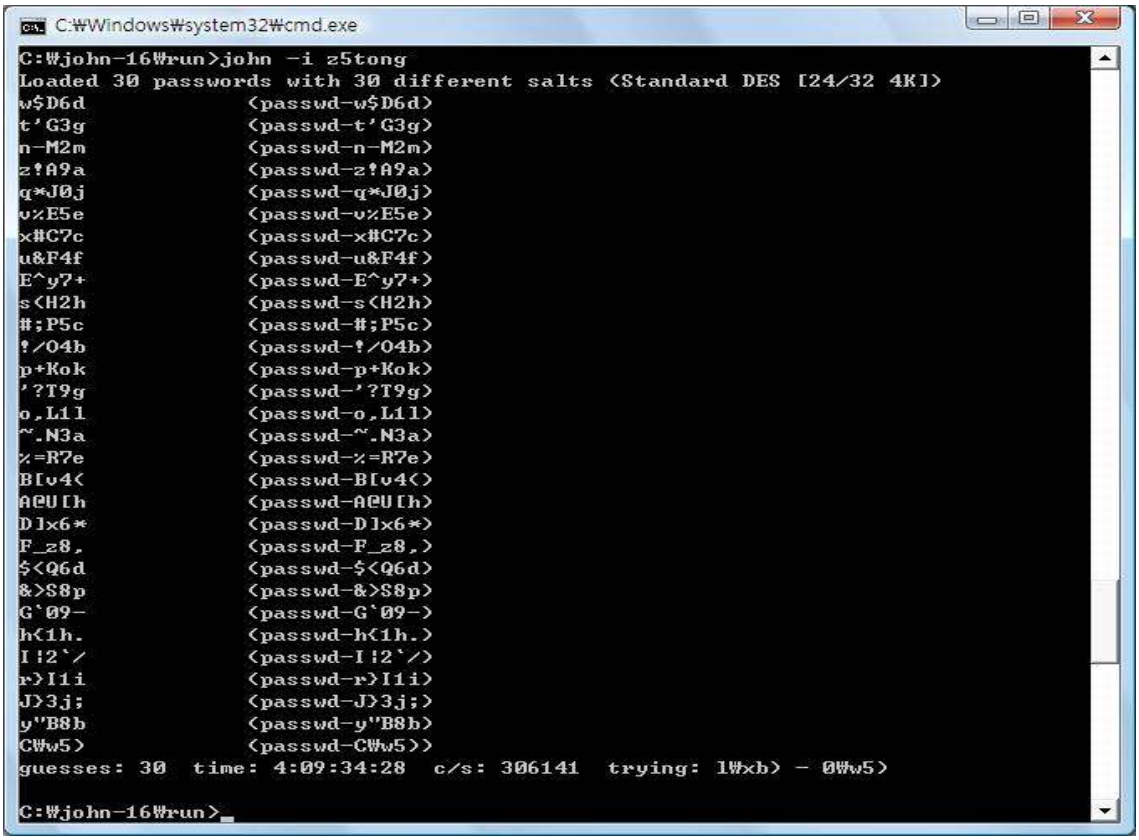
(그림2-21) 4자리로 구성된 그룹 종합 패스워드



(그림2-21) 3자리 쉐도우파일 크랙화면

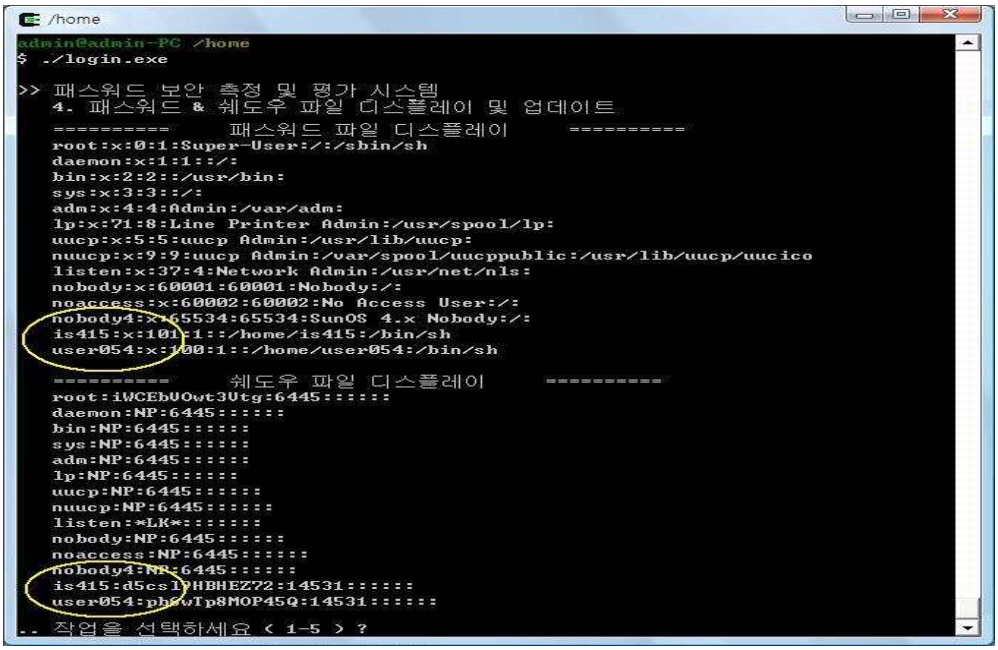


(그림2-22) 5자리로 구성된 그룹 종합 패스워드

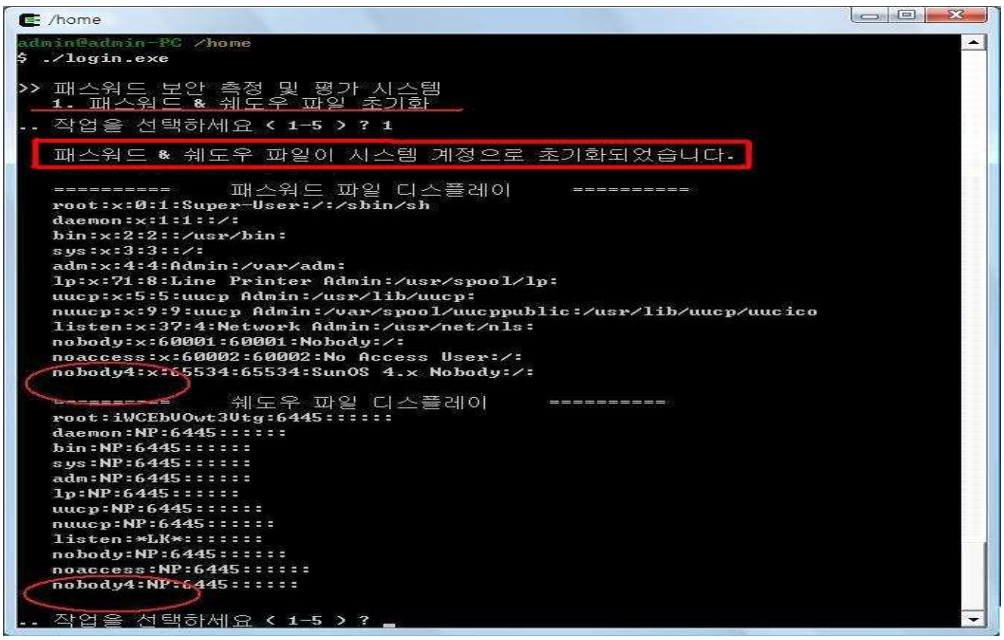


(그림2-22) 5자리 쉐도우파일 크랙화면

2.4 시스템파일 초기화 모듈

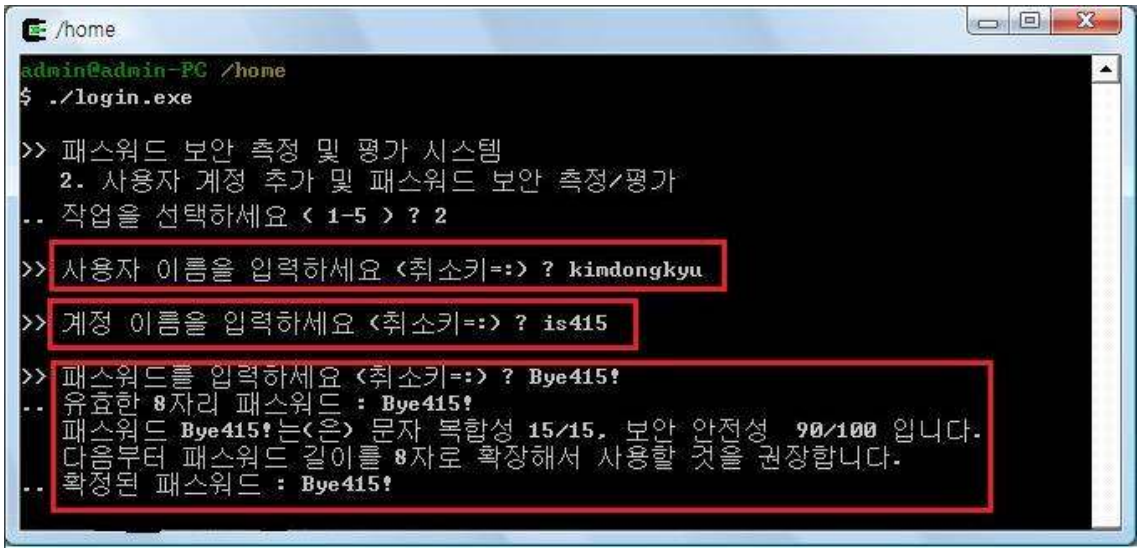


(그림1-3) 시스템파일 초기화 전



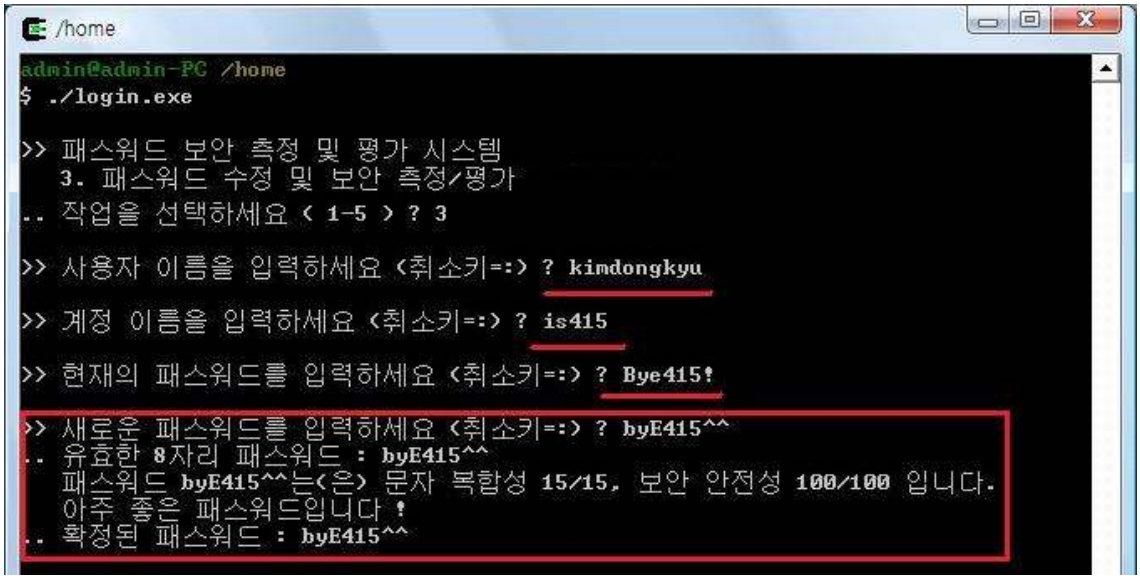
(그림1-4) 시스템파일 초기화 후

2.5 사용자 계정&패스워드 추가모듈



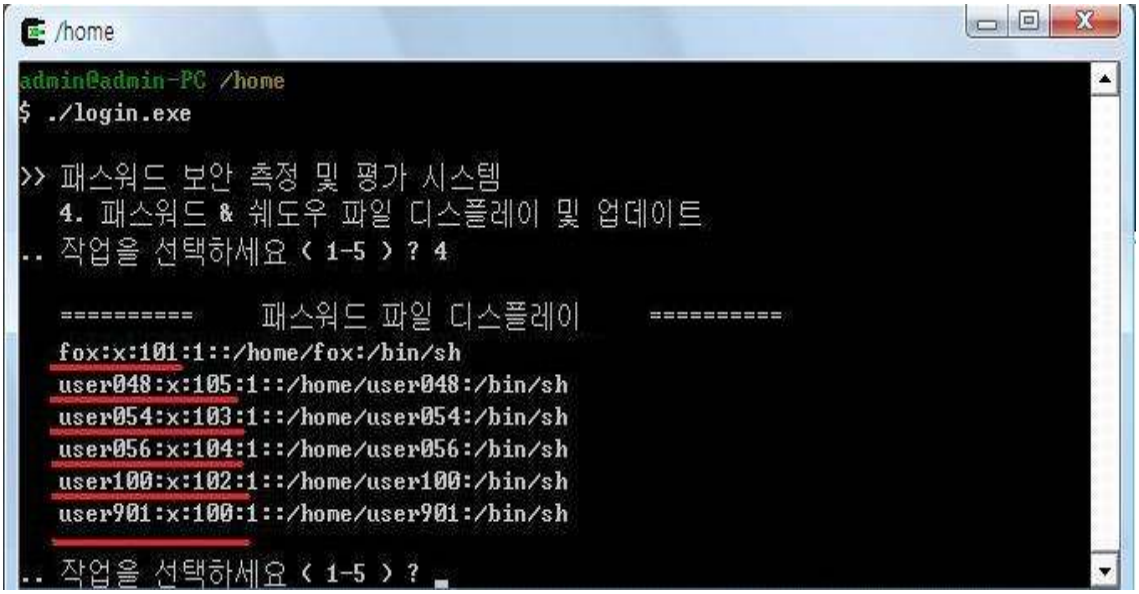
(그림1-5) 2번모듈은 계정과 패스워드 안전성평가

2.6 패스워드 수정 모듈

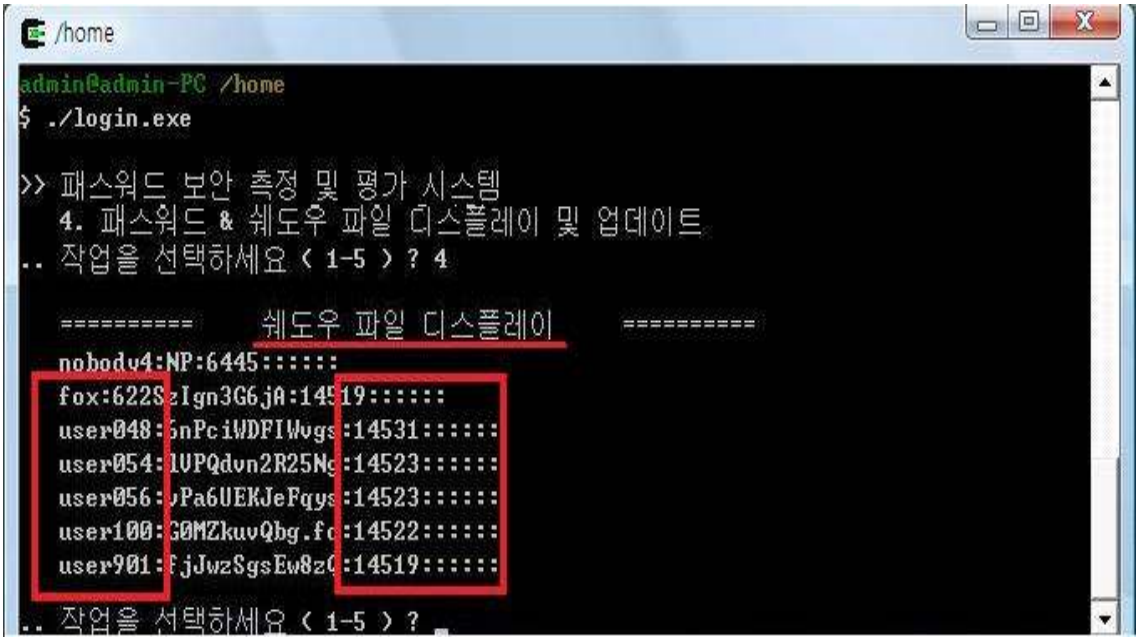


(그림1-6) 3번모듈은 패스워드 교체와 안전성평가

2.7 시스템파일 디스플레이 및 업데이트 모듈

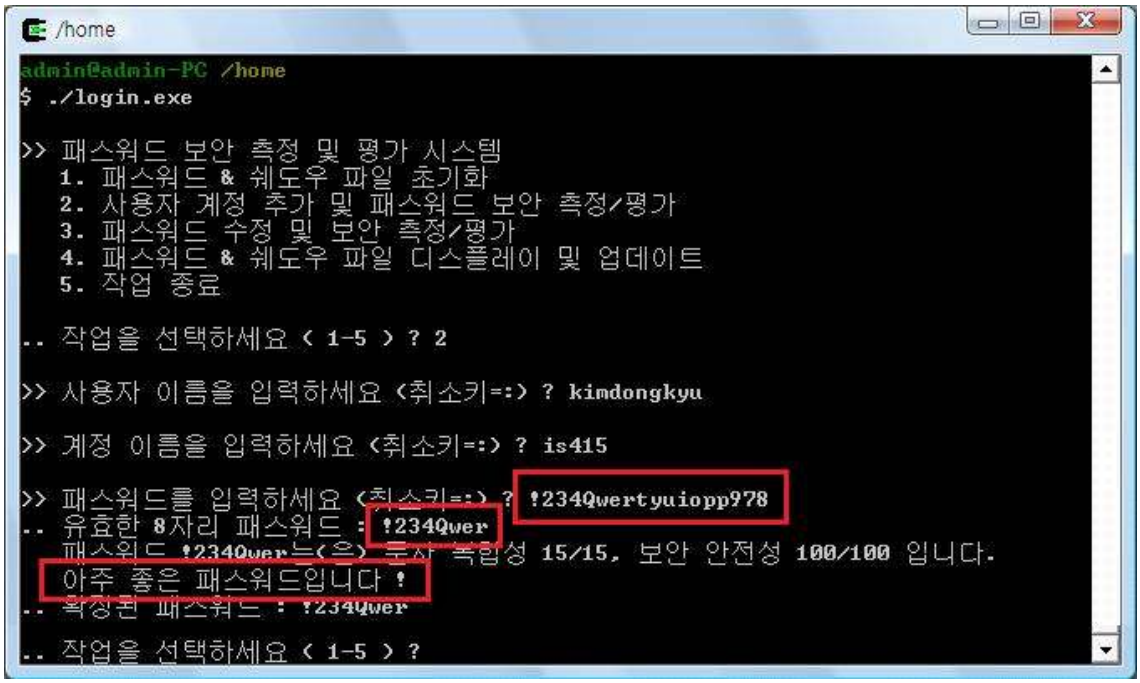


(그림1-7) 패스워드파일 디스플레이



(그림1-8) 쉐도우 파일 디스플레이와 기준일로 계산된 업데이트

2.8 개발모듈을 모두 통합 후 Little login 완성



(그림1-9) 모든 모듈을 통합한 "Little login" 모습

2.9 시스템 개발환경

운영체제	윈도우 비스타 , 윈도우7 ultmate 7100(Korean)
개발언어	Cygwin(gcc) 및 Vlsual C++
개발방법	-윈도우 비스타 Vlsual C++로 1차 프로그래밍 및 디버깅
	- DES 암호화 등은 Cygwin(gcc) 에서 구현
	- 디버깅 후 Cygwin(gcc)에 이식 및 운영환경 구축

3. UNIX란

3.1 UNIX의 역사

최초의 UNIX는 1969년 미국 AT&T사의 Bell연구소의 CSRG(Computer Science Research Group)팀에 있었던 Ken Thompson라는 한 사람에 의해 개발되엇다. 그 당시 Ken Thompson은 태양계에서 행성이 어떻게 이동하는가를 시뮬레이션하기 위해 우주여행(Space Traval)이라고 불리우는 프로그램을 개발하고 있었다. 이 프로그램은 General Electric(GE) 회사에 의해서 만들어진 GE645라는 거대한 컴퓨터에서 실행되고 있었는데 이 컴퓨터의 운영체제로서 MULTICS를 사용하고 있었다. MULTICS는 MIT에서 개발된 다중 사용자가 동시에 여러 가지 작업을 할 수 있는 멀티유저 운영체제였다. 하지만 MULTICS는 운영하는 비용이 비싸고 다루기도 곤란한 운영체제였다.
따라서 Thompson은 DEC(Digital Equipment Corporation)회사에서 만든 PDP-7이라는 미니 컴퓨터가 유지비용이 덜 들어간다는 것을 발견하고 MULTICS에서 실행되던 우주여행 프로그램을 PDP-7에서 보다 편리하게 실행하기 위해 새로운 운영체제를 구현하였다. 이것이 바로 UNIX의 효시였다. Brian Kemighan은 이 시스템을 UNICS라 불렀다. 이것은 MULTICS(MUL Tiplexed Information & Computing System)에 비하여 아주 작은 시스템이라는 점에서 “UNiplexed” 로 명명을 바꾸어 붙였다.
PDP-7에서 다시 PDP-11/40과 /45에 이식하게 되었고, 나중에는 Bell 연구소에서 널리 사용된 PDP-11/70에서 사용되었다. 그와 동시에 AT&T는 PDP-11/70 컴퓨터를 사용하고 있던 각 대학에 싼 값으로 UNIX를 보급하게 되었다. 이러한 AT&T의 움직임으로 UNIX는 미국 내에 있는 전체 대학 전산소의 80% 이상이 UNIX를 사용하게 되었고, 매년 수천 명의 학생이 UNIX를 배워 사회로 배출되었다.

이후 이기종간의 이식성을 높이기 위하여 1973년 Dennis Ritche에 만들어진 C언어로 재 작성하게 되었다. C언어로 UNIX를 작성하게 됨에 따라 유닉스를 서로 다른 컴퓨터에 이식하는데 드는 노력이 훨씬 줄어들게 되었다. 뿐만아니라 범용성 언어로서 현대적인 명령어들을 갖춘 C언어는 어셈블리 언어보다 훨씬 이해하기 쉬웠다. 실행 속도면에서는 어셈블리 언어보다 뛰어날 수 없지만 C언어는 훨씬 더 편리했고, 사용자들이 UNIX를 자유로이 개정하고 개선시켜 나갈수 있도록 자연스럽게 유도 하였다. UNIX는 90%가 C언어로 작성되어 있고 나머지 10%가 어셈블리 언어로 작성되어 있다.

현재 UNIX의 표준화를 주도하고 있는 단체는 OSF(Open Software Foundation)와 UI(Unix International)라는 단체가 있다. OSF는 IBM, DEC를 비롯한 몇 개의 업체들이 모여서 결성한 비영리 연구 단체이다. OSF는 현재 'OSF/1'이라는 유닉스를 가지고 있으며 그래픽 사용자 인터페이스로서 OSF/Morif를 가지고 있다. UI는 AT&T가 주축이 된 단체로서 'UNIX System V'라는 유닉스를 가지고 있다.

3.2 UNIX의 특징

3.2.1 대화식 운영체제

사용자에게 명령어를 입력 받기위해서 UNIX는 Shell 프롬프트를 화면에 나타낸다. 프롬프트가 나타난 상태에서 사용자가 명령을 입력하면 시스템은 명령을 수행하고 결과를 알려준다.

3.2.2 다중 작업(Multi Tasking)

UNIX에서 실행중인 명령어를 프로세스(Process)라고 하는데 유닉스는 동시에 여러개의 프로세스를 실행할 수 있는 Multi Tasking 운영체제이다.

3.2.3 다중 사용자(Multi User)

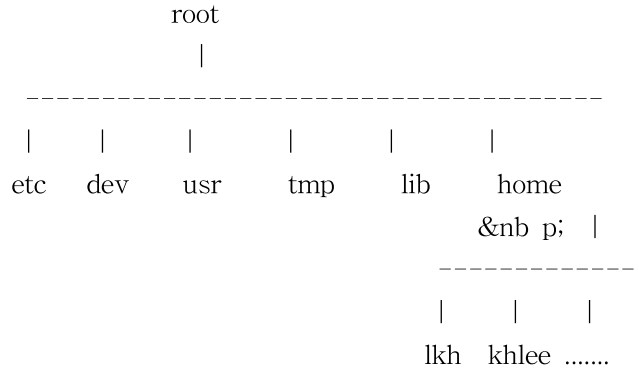
UNIX에서는 동시에 여러 명의 사용자가 작업할 수 있다. 이러한 Multi User 환경은 다중 사용자간에 프로그램 및 데이터를 공유할 수 있도록 해준다. 이러한 자원의 공유는 사용자에게 작업하기 편리한 환경을 제공하며 개발 단계에서의 효율을 극대화 시킬 수 있도록 해준다.

3.2.4 높은 이식성(Portability)

이식성이란 하드웨어의 종류에 관계없이 운영되는 특성을 말한다.
UNIX는 개인용 PC에서부터 슈퍼컴퓨터까지 자유로이 설치 운영되어진다.

3.2.5 계층적 파일 시스템

UNIX 파일 시스템은 계층적 파일 구조를 가지고 있다.



3.3 UNIX의 구성

복잡하고 어렵게 보이는 UNIX 시스템도 크게 커널(Kernel), 쉘(Shell), 유틸리티(Utility) 및 명령어의 세부분으로 나누어져 있다.

3.3.1 커널(Kernel)

커널은 하드웨어와 직접적으로 연결되어 있는 UNIX 시스템의 핵심 부분이다. 도스와 비교하면 도스의 IO.SYS와 MSDOS.SYS 가 바로 UNIX의 커널에 해당되는 부분이라고 할 수 있다. 커널은 컴퓨터를 부팅하였을 때 보조기억 장치로부터 주기억 장치로 옮겨지며, 프로세스 스케줄링, 기억장치 관리, 입.출력 장치 관리 및 쉘(Shell)과 응용 프로그램에 필요한 서비스를 제공 등을 수행하는 부분으로써, 엄밀히 말했을 때의 UNIX 운영체제는 바로 이 커널 부분을 말하고 쉘(Shell)이나 유틸리티는 일반적인 프로그램이라고 볼 수 있다.

3.3.2 쉘(Shell)

쉘은 응용 프로그램과 Kernel 사이에서 인터페이스(Interface) 역할을 해주는 일종의 명령어 해독기이면서 동시에 사용자가 프로그램을 할 수 있는 프로그래밍 언어이기도 하다. 즉, 사용자와 순수한 운영체제 사이를 연결시켜주고 실행하는 프로그램으로 작업관리 기능을 제공하고, 입.출력의 흐름을 제어하며, 쉘 스크립트(일종의 배치화일)를 쓰기 위한 쉘 명령어를 제공한다.
UNIX는 Bourne Shell, C Shell, Korn Shell, Extended C Shell, Bourne Again Shell 등 여러 종류의 쉘을 지원하며 사용자가 어떤 쉘을 사용하는가에 따라 각기 다른 시스템 사용 환경을 만들 수 있다.

3.3.3 유틸리티(Utility) 및 명령어들

UNIX는 각종 프로그래밍 언어들을 운영체제 안에서 지원한다. 사용할 수 있는 언어에는 FORTRAN, PASCAL, BASIC, C, COBOL, LISP 등이 있고, 수많은 유틸리티와 명령어들이 존재한다.
명령어는 사용자 작업(Task)를 수행하는 프로그램이다. 명령어는 사용자가 시스템에 작업을 수행하기 위한 일종의 도구 역할을 수행한다.
유닉스는 새로운 명령어를 생성하기 위해 복합될 수 있는 수백개의 명령어를 제공한다. MAPPER와 Oracle같은 어플리케이션 프로그램은 쉘을 통하여 엑세스될 수 있다.

4. Cracking툴 및 패스워드 안전성 기준

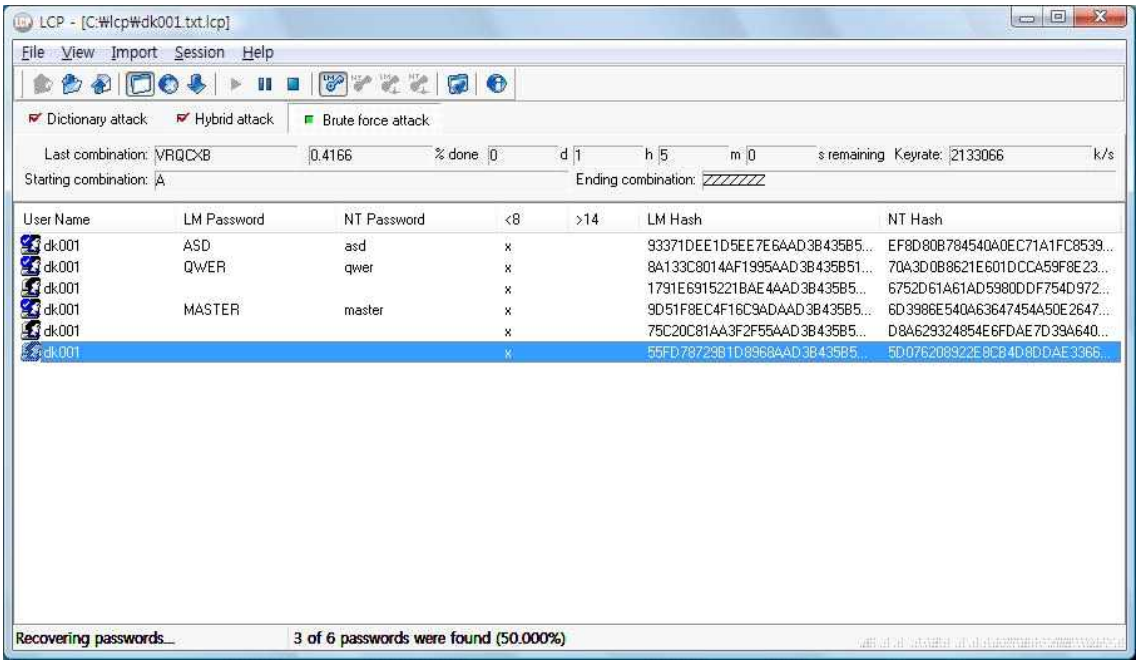
4.1 John the Ripper란 무엇인가

John the Ripper는 Solar Designer가 개발한 Unix계열 password crack tool 이다. 무료 도구이며, UNIX계열 크래킹도구이지만DOS, Win9x, NT, 2000 등의 플랫폼도 지원한다. 속도를 높이기 위해 Intel MMX기술이나 AMD K6프로세서의 특수 기능들을 이용한 최적화된 코드를 집어넣기도 하였다
http://www.openwall.com/john/ 에서 개발버전, 안정버전, linux용, window용 등을 다운가능하다.

4.2 LCP란 무엇인가

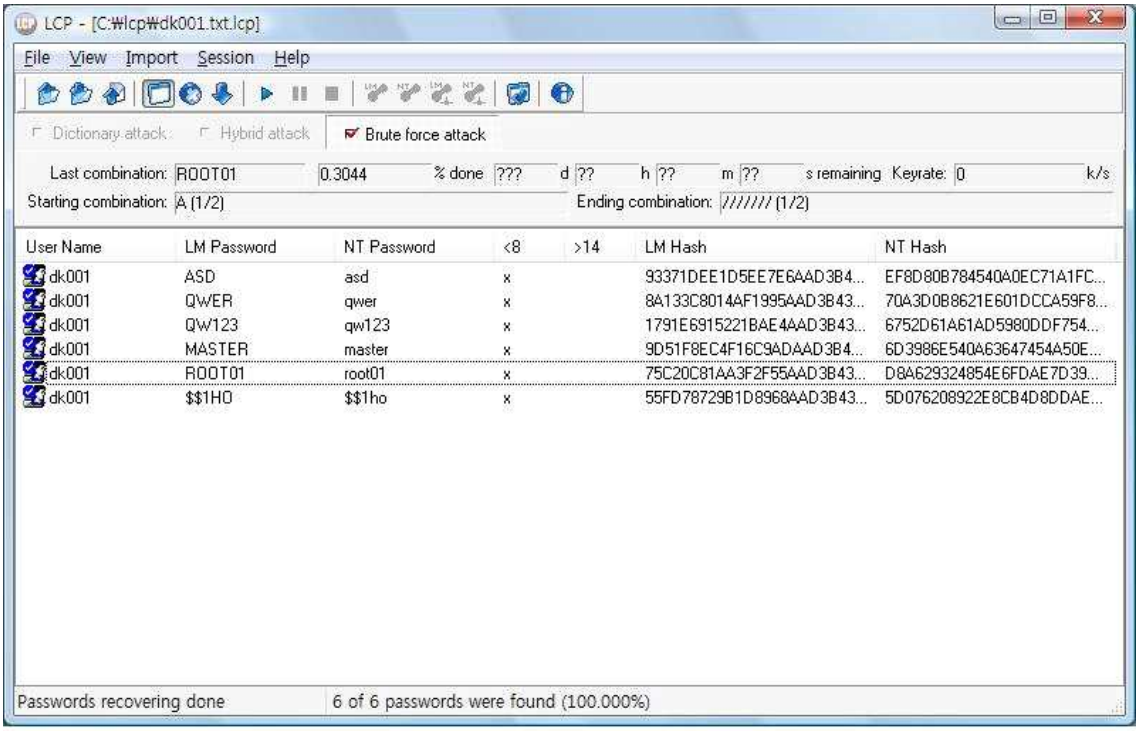
패스워드 크래킹 툴(윈도우기반으로 사용)
LCP는 SID방식으로 사용자 프로그램으로 무료 도구이며 윈도우계열에서 사용이 가능 윈도우 패스워드 크래킹에 탁월함
http://www.lcpsoft.com/english/index.html에서 English vision, Russian vision등 두 가지 언어로 사용자의 편의를 제공한다.

크래킹이 되는 과정



(그림1-10) LCP로 크랙하는 과정

크래킹이 완료된 모습



(그림1-11) LCP로 크랙완료한모습

4.3 안전한 패스워드기준

안전하다. 안전하지않다 는 기준은 해커들이 개인PC를 사용한다는 전제하에 아래와 같은 계산법을 이용해서 7자리나 8자리같은 경우는 적어도 몇 달, 몇 년 까지 의 크랙시간이 걸리기 때문에 관리자는 크랙시간이 걸리는 만큼 사용자들에게 위험경고를 하고 교체 할 것을 알려준다.

4.4 수학적 계량화 계산법

이 식을 이용하여 자릿수가 하나씩 추가 될 때마다 키보드 자판의 패스워드로 이용할 수 있는 95가지를 한번 식 더 곱해서 95의 제곱 순으로 풀어 나간다.
t(3)=45초인 이유는 우리조 가 John the Ripper를 활용해 세 자리 수의 평균 시간을 기준으로 한 시간 이다.

t(n)=n자리수 크래킹 소요시간, t(n)=95*t(t-1), 8≧n≧4
t(2), t(1)=초 단위이내 이기에 계산 및 정의가불가능 하다.

t(3)=45초
t(4)=95*t(3)= 95*45초
t(5)=95*t(4)= 95*95*45초
t(6)=95*t(5)= 95^3*45초
t(7)=95*t(6)= 95^4*45초
t(8)=95*t(7)= 95^5*45초
--> ∴ $t(n)=95^{n-3}*t(3)$, 4≦n≦8

5. 결론

사이버해킹은 날이 갈수록 발전해 가고 있다 그로인해 개인정보의 유출이 증가추세를 보이고 있다. 이를 방지하기 위하여 가능한 안전한 패스워드를 사용하게 하기 위해 시뮬레이터를 개발하였다.
우리가 지금까지 해온 것들은 솔직히 내부사용자와 외부사용자가 모두 사용할수 있으며, 어떤 사람들이 들어왔는지 확인하고 관리할 수 없는 것이 현실이었다. 크래킹을 해본 결과 우리가 주로 사용한 패스워드 중에는 안전하지 않은 패스워드가 많다는 것을 알 수있었고, 이제라도 더 낳은 더 안전한 패스워드를 사용하는 정보보호인이 되어야겠다고 생각했다.
시작부터 끝까지 졸업작품을 하면서 더 좋은 의견을 찾다가 팀원간 많이 다투기도 했지만 지금 이 시점에 돌이켜서 생각해 보면 아쉬움도 많이 남고, 대학생활중 가장 멋진 추억으로 간직할 것 같아서 뜻 깊다.
끝으로 부족한 실력으로 여기까지 오는데 도움을 준 지도 교수님외 같은 고생한 LEGEND팀원에게 감사의 말씀을 전하고 연구 발표를 마치겠다.

5.1 향후과제

애초에 연구목적에 세웠던 사전식 크래킹은 이번 연구에서는 도입하지 못하여서 큰 아쉬움을 남겼고, 후배들이 이를 기반으로 사전식 크래킹도입을 대학생의 젊은 패기로 도전해보는 것도 좋을 것 같다.

부록[1]. 최종모듈소스

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <time.h>

#define N 503          // prime number
#define NC 129       // number of character
#define NEC 95     // effective number of character
#define NGR 5     // number of character's category
#define GR  16   // number of password's pontential grade
time_t now_time;

//    passwd file data of system's accounts
char passwd[]="root:x:0:1:Super-User:/:/sbin/sh\n"
 "daemon:x:1:1::/:\n"   "bin:x:2:2::/usr/bin:\n"
 "sys:x:3:3::/:\n"   "adm:x:4:4:Admin:/var/adm:\n"
 "lp:x:71:8:Line Printer Admin:/usr/spool/lp:\n"
 "uucp:x:5:5:uucp Admin:/usr/lib/uucp:\n"
 "nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico\n"
 "listen:x:37:4:Network Admin:/usr/net/nls:\n"
 "nobody:x:60001:60001:Nobody:/:\n"
 "noaccess:x:60002:60002:No Access User:/:\n"
 "nobody4:x:65534:65534:SunOS 4.x Nobody:/:\n";

//    shadow file data of system's accounts
char shadow[]="root:iWCEbVOwt3Vtg:6445::::::\n"
 "daemon:NP:6445::::::\n"
 "bin:NP:6445::::::\n"
```

```
 "sys:NP:6445::::::\n"
 "adm:NP:6445::::::\n"
 "lp:NP:6445::::::\n"
 "uucp:NP:6445::::::\n"
 "nuucp:NP:6445::::::\n"
 "listen:*LK*:::::::\n"
 "nobody:NP:6445::::::\n"
 "noaccess:NP:6445::::::\n"
 "nobody4:NP:6445::::::\n";

char buff[100], *passwd_item[N][7], *shadow_item[N][9], *user_item[N][5],
*calloc_ptr[1000];

char *name_ptr, *passwd_ptr, *buff_ptr, *saltkey_ptr, *user_name, *tmp_ptr;
int  i, j, k, l, sel, no_account, no_account_save, sys_account, idx, ps_index[N];
int  current_date, max_acc, jj1, jj2, calcnt;
int  passwd_value, max_user, current_user;
int                 category[NC]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1,1,1,1,1,1,1,1,1,//
sp  !  "  #  $  %  &  '
    1,1,1,1,1,1,1,1,2,2,
//  (  )  *  +  ,  -  .  /  0  1
    2,2,2,2,2,2,2,2,1,1,
//  2  3  4  5  6  7  8  9  :  ;
    1,1,1,1,1,3,3,3,3,3,
//  <  =  >  ?  @  A  B  C  D  E
    3,3,3,3,3,3,3,3,3,3,
//  F  G  H  I  J  K  L  M  N  O
    3,3,3,3,3,3,3,3,3,3,
//  P  Q  R  S  T  U  V  W  X  Y
    3,1,1,1,1,1,1,4,4,4,
//  Z  [  \  ]  ^  _  `  a  b  c
    4,4,4,4,4,4,4,4,4,4,
//  d  e  f  g  h  i  j  k  l  m
    4,4,4,4,4,4,4,4,4,4,
```

```
//   n  o  p  q  r  s  t  u  v  w
   4,4,4,1,1,1,1,0};
//   x  y  z  {  |  }  ~
//       특, 수, 대, 소
int weight[NGR]={0,  8,  1,  4,  2};
int  complex[GR]={0, 1, 2,  5,  3,  6,  7,  11,  4,  8,  9,  12,
10,  13,  14,  15};
int  grade[GR]=  {0, 6, 8, 30,  10, 32,  34,  60,  12,  36,  38,  64,
40,  68,  72,  100}
//수,소,수+소,대,수+대,소+대,수+소+대,특,수+특,소+특,수+소+특,대+특,수+대+특,  소
+대+특,수+소+대+특
int  length_value[9]={0, 6, 12, 18, 28, 40, 60, 80, 100};
char *test_ptr;
FILE *fpp, *fps, *fpu;


//     ==================== password key-in and check subroutine
=====================================
int password_check(char *ptr)
{int count, idx, char_ctg;
unsigned char passwd_state[9], total_state;
char *tm_ptr;

count=strlen((char *)ptr);

//  user-profiles comparision procedure
j1=0;
for (i=0; i<=4; i++) {
l=strlen(user_item[current_user][i]);
for (j=1; j<=(l-3);j++) {
tm_ptr=calloc(l-j+2, sizeof(char)); calloc_ptr[calcnt++]=tm_ptr;
strcpy((char *)tm_ptr,(char *)user_item[current_user][i]+j-1);
for (k=strlen(tm_ptr); k>=4; k--) {
buff_ptr=calloc(k+1, sizeof(char)); calloc_ptr[calcnt++]=buff_ptr;
strncpy((char *)buff_ptr, (char *)tm_ptr, k);
```

```
if(strstr((char *)ptr, (char *)buff_ptr)!=0) goto conform;
}
}
continue;
conform:
        if (i==0) printf("\n.. 패스워드(%s)가 성명(%s)과 ", ptr,
user_item[current_user][i]);
        if (i==1) printf("\n.. 패스워드(%s)가 계정(%s)과 ", ptr,
user_item[current_user][i]);
        if (i==2) printf("\n.. 패스워드(%s)가 생년월일(%s)과 ", ptr,
user_item[current_user][i]);
        if (i==3) printf("\n.. 패스워드(%s)가 전화번호(%s)와 ", ptr,
user_item[current_user][i]);
        if (i==4) printf("\n.. 패스워드(%s)가 학번(%s)과 ", ptr,
user_item[current_user][i]);
printf("%d자리 일치(%s)합니다.", k, buff_ptr);
jj1++;
}
if (jj1!=0) goto recommend;

//  password is independent of user's personal data
if (count>8) count=8;
*(ptr+count)='\0'; total_state=0x00;
for (i=1; i<=8; i++) passwd_state[i]=0x00;
printf(".. 유효한 8자리 패스워드 : %s", ptr);

//  weight   8 : 특수 문자 4 : 영문 대문자  1 : 숫자   2 : 영문 소문자
for (i=1; i<=count; i++) {
idx=*(ptr+i-1);
char_ctg=category[idx];
        if (char_ctg==0) { printf("\n   패스워드에 허용되지 않는 문자가 포함되어 있습
니다.\n"); exit(0); }
passwd_state[i]=weight[char_ctg];
total_state=total_state | passwd_state[i];
}
```

```
passwd_value=(grade[total_state]+length_value[count])/2;
i=complex[total_state];
printf("\n    패스워드 %s는(은) 문자 복합성 %2d/15, 보안 안전성 %3d/100 입니다.",ptr,i, passwd_value);
switch (count) {
default: case 1: case 2: case 3: case 4:
printf("\n    패스워드의 길이가 너무 짧아 안전성에 문제가 많아요.");
if (total_state!=0xf) printf("\n    문자 배합도 고려하세요.");
goto recommend;
case 5: case 6:
printf("\n    패스워드를 7자 이상으로 확장해 주세요.");
if (total_state!=0xf) { printf("\n    문자 배합도 고려하세요."); goto recommend;}
printf("\n    문자 배합은 잘됐습니다."); goto recommend;
case 7:
            if (total_state!=0xf) {printf("\n    문자 배합을 고려하여 패스워드를 다시 입력해 주세요."); goto recommend;}
            printf("\n    다음부터 패스워드 길이를 8자로 확장해서 사용할 것을 권장합니다."); return(0);
case 8:
if (total_state==0xf) { printf("\n    아주 좋은 패스워드입니다 !"); return(0);}
printf("\n    문자 배합에 유의하세요.");
recommend:
            printf("\n\n>> 아래의 패스워드 구성요건에 맞게 패스워드를 다시 입력해 주세요.");
 · printf("\n    1. 영문 소문자 및 대문자, 숫자, 특수문자를 최소 1자 이상 배합");
printf("\n    2. 패스워드 길이는 가능한 8개의 문자로 구성");
printf("\n    3. 동일한 문자의 반복 사용이나 규칙성있는 사용을 금지");
 printf("\n    4. 본인 이름/생일/전화번호/학번 등과 관계가 없는 문자로 구성\n");
return(1);
}
}

//  = saltkey generation module (generated by the current time)
get_salt(char *seed)
```

```
{char gen_seed[3];
     char *c_set="./abcdefghijklmnopqrstuvwxyz"
                 "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                 "0123456789";
     memset(gen_seed, '\0', 3);
     time(&now_time);
     srand((unsigned int)now_time);
     gen_seed[0] = c_set[rand() % 64]; gen_seed[1] = c_set[rand() % 64];
     memcpy(seed, gen_seed, 3);
}


main()
{

//   process selection
start:
calcnt=0;
     printf("\n>> 패스워드 보안 측정 및 평가 시스템\n");
printf("   1. 패스워드 & 쉐도우 파일 초기화 \n   2. 사용자 계정 추가 및 패스워드 보안 측정/평가\n");
printf("   3. 패스워드 수정 및 보안 측정/평가 \n   4. 패스워드 & 쉐도우 파일 디스플레이 및 업데이트\n");
printf("   5. 작업 종료 \n\n.. 작업을 선택하세요 ( 1-5 ) ? ");
mp_ptr=calloc(5, sizeof(char));
scanf("%s", tmp_ptr);
if (strlen(tmp_ptr)>1) {free(tmp_ptr); goto abnormal;}
sel=*tmp_ptr-48; free(tmp_ptr);
if (sel==1) goto init; else if (sel==2 || sel==3 || sel==4) goto new_account; else if (sel==5) goto end;
abnormal:
printf("   1번부터 5번까지의 작업만 선택해 주세요 !\n"); goto start;

//   system's accounts initialization process
init:
```

```
//  passwd file initialization
if((fpp=fopen("passwd","w"))==NULL)
{printf("\n    패스워드 파일을 찾을 수 없습니다.\n"); goto end;}
fprintf(fpp,"%s",passwd); fclose(fpp);

//  shadow file initialization
if((fps=fopen("shadow","w"))==NULL)
{printf("\n    쉐도우 파일을 찾을 수 없습니다.\n"); goto end;}
fprintf(fps,"%s",shadow); fclose(fps);

 printf("\n    패스워드 & 쉐도우 파일이 시스템 계정으로 초기화되었습니다.\n");
goto freecalloc;

//      user accounts addition, new password process, passwd/shadow file
update
new_account:

//  shadow file item_table construction
if((fps=fopen("shadow","r"))==NULL)
{printf("\n    쉐도우 파일을 읽을 수 없습니다.\n"); goto end;}

for (i=0; i<N; i++){
l=fscanf(fps,"%s", buff); if (l==-1) goto shadow_last;
ps_index[i]=i+1;
buff_ptr=&buff[0];
for (j=0; j<8; j++) {
k=strcspn(buff_ptr,":");
            shadow_item[i][j]=calloc(k+1,                    sizeof(char));
calloc_ptr[calcnt++]=shadow_item[i][j];
strncat((char *)shadow_item[i][j], (char *)buff_ptr, k);
strcpy((char *)buff_ptr, (char *)(buff_ptr+k+1));
}
shadow_item[i][j]="\0";
}
```

```
shadow_last:
fclose(fps); ps_index[i-1]=0; no_account=i; no_account_save=no_account;
sys_account=12;    // number of system's accounts

//  passwd file item_table construction
if((fpp=fopen("passwd","r"))==NULL)
{printf("\n    패스워드 파일을 읽을 수 없습니다.\n"); goto end;}

max_acc=99; jj1=0;
for (i=0; i<N; i++){
l=fscanf(fpp,"%[^\n]", buff); if (l==-1) goto passwd_last;
buff_ptr=&buff[0]; l=strlen(buff);
for (j=0; j<6; j++) {
k=strcspn(buff_ptr,":");
            passwd_item[i][j]=calloc(k+1,                    sizeof(char));
calloc_ptr[calcnt++]=passwd_item[i][j];
strncat((char *)passwd_item[i][j], (char *)buff_ptr, k);
strcpy((char *)buff_ptr, (char *)(buff_ptr+k+1));
}
passwd_item[i][j]=calloc(strlen(buff_ptr)+1, sizeof(char));
strcpy((char *)passwd_item[i][j], (char *)buff_ptr);
calloc_ptr[calcnt++]=passwd_item[i][j];
k=ftell(fpp); fseek(fpp, k+1, 0);
if (i>11) sscanf(passwd_item[i][2],"%d",&jj1);
if (jj1>max_acc) max_acc=jj1;
}
passwd_last:
fclose(fpp);
if (sel==4) goto display;// go to passwd & shadow file display routine

//  user-profiles file item_table construction
if((fpu=fopen("user-profiles.txt","r"))==NULL)
{printf("\n    사용자 정보 파일을 읽을 수 없습니다.\n"); goto end;}
```

```
for (i=1; i<N; i++){
l=fscanf(fpu,"%s", buff); if (l==-1) goto user_last;
buff_ptr=&buff[0]; l=strlen(buff);
for (j=0; j<5; j++) {
k=strcspn(buff_ptr,":");
user_item[i][j]=calloc(k+1, sizeof(char)); calloc_ptr[calcnt++]=user_item[i][j];
strncat((char *)user_item[i][j], (char *)buff_ptr, k);
strcpy((char *)buff_ptr, (char *)(buff_ptr+k+1));
}
max_user=i-1;
}
user_last:
max_user=i-1;
fclose(fpu);
//  user's name key-in & compare user name's list
jj1=0;
user_name_again:
user_name=calloc(21, sizeof(char)); calloc_ptr[calcnt++]=user_name;
printf("\n>>  사용자  이름을  입력하세요  (취소키=:)  ?  "); scanf("%s",
user_name);
if (strcmp((char *)user_name,":")==0) goto end_acc;
for (i=1; i<=max_user; i++) {
if (strcmp((char *)user_item[i][0],(char *)user_name)==0) goto found;
}
printf("   사용자(%s)를 찾을 수 없습니다.\n", user_name);
jj1++; if (jj1>2) {printf("    더 이상 작업을 진행할 수 없습니다.\n"); goto
freecalloc;}
printf("     사용자 이름을 %d회 수정 입력할 수 있습니다.\n", 3-jj1); goto
user_name_again;
found:
current_user=i;

//  account's name key-in process
name_ptr=calloc(15, sizeof(char)); calloc_ptr[calcnt++]=name_ptr;
    printf("\n>>  계정  이름을  입력하세요  (취소키=:)  ?  "); scanf("%s",
```

```
name_ptr);
if (strcmp((char *)name_ptr,":")==0) goto end_acc;

//  sorting by account's name
j=ps_index[sys_account-1]; k=strlen(name_ptr);
jj1=11; jj2=no_account;
for (i=sys_account; i<no_account; i++) {
l=strncmp((char *)shadow_item[j][0], (char *)name_ptr, k);
if (l>0) goto change; else if (l<0) goto no_op;
if ((strlen(shadow_item[j][0])-k)!=0) goto change;

if (sel==3) goto update;    // goto new password process routine

 printf("    계정(%s)이 등록되어 있어 작업을 계속할 수 없습니다.\n", name_ptr);
goto freecalloc;

no_op:
jj1=j; j=ps_index[j];
}
change:
ps_index[jj2]=ps_index[jj1];
ps_index[jj1]=jj2;if (sel==3) {
            printf("   계정(%s)을 찾을 수 없습니다. 먼저 계정을 등록해 주세요.\n",
name_ptr);
goto freecalloc;
}

//  password key-in process
passwd_ptr=calloc(100, sizeof(char)); calloc_ptr[calcnt++]=passwd_ptr;
passwd_again:
printf("\n>> 패스워드를 입력하세요 (취소키=:) ? ");
scanf("%s", passwd_ptr);     // new account and its password key-in
if (strcmp((char *)passwd_ptr,":")==0) goto freecalloc;
if (strlen((char *)passwd_ptr)>20) {printf("      패스워드의 길이가 너무 깁니
다.\n"); goto passwd_again;}
```

```
jj1=password_check(passwd_ptr);
if (jj1==1) goto passwd_again;
printf("\n.. 확정된 패스워드 : %s\n", passwd_ptr);

//  saltkey generation & password encryption
saltkey_ptr=calloc(3,sizeof(char)); calloc_ptr[calcnt++]=saltkey_ptr;
get_salt(saltkey_ptr);  // saltkey generation
buff_ptr=calloc(14, sizeof(char)); calloc_ptr[calcnt++]=buff_ptr;
 strcpy(buff_ptr,(char *)crypt(passwd_ptr, saltkey_ptr)); // password encryption

//  new shadow item_table creation
for (i=0; i<9; i++) shadow_item[no_account][i]="";    // item_table clear
shadow_item[no_account][0]=name_ptr;      // account name
shadow_item[no_account][1]=buff_ptr;      // encrypted password
current_date=time(&now_time)/86400;       // current date from 19700101
passwd_ptr=calloc(5,sizeof(char)); calloc_ptr[calcnt++]=passwd_ptr;
sprintf(passwd_ptr,"%d",current_date);
shadow_item[no_account][2]=passwd_ptr; // current date

//  new passwd item table creation
passwd_item[no_account][0]=name_ptr;
passwd_item[no_account][1]="x";
passwd_item[no_account][2]=calloc(3,sizeof(char));
calloc_ptr[calcnt++]=passwd_item[no_account][2];
sprintf(passwd_item[no_account][2],"%d",++max_acc);
passwd_item[no_account][3]="1";
passwd_item[no_account][4]="";
passwd_item[no_account][5]=calloc(7+strlen(name_ptr),sizeof(char));
calloc_ptr[calcnt++]=passwd_item[no_account][5];
strcpy((char *)passwd_item[no_account][5],"/home/");
strcpy((char *)(passwd_item[no_account][5]+6),(char *)name_ptr);
passwd_item[no_account][6]="/bin/sh";
no_account++;

end_acc:
```

```
if (no_account==no_account_save) goto freecalloc;
goto display;

//  new password update process
update:
idx=i;

//  old_password key-in process
tmp_ptr=calloc(100, sizeof(char)); calloc_ptr[calcnt++]=tmp_ptr;
old_passwd_retry:
printf("\n>> 현재의 패스워드를 입력하세요 (취소키=:) ? ");
scanf("%s", tmp_ptr);    // old_password key-in
if (strcmp((char *)tmp_ptr,":")==0) goto freecalloc;
if (strlen((char *)tmp_ptr)>20) {printf("    패스워드의 길이가 너무 깁니다.\n");
goto old_passwd_retry; }

//  saltkey extraction & old_password encryption
saltkey_ptr=calloc(3,sizeof(char)); calloc_ptr[calcnt++]=saltkey_ptr;
strncpy(saltkey_ptr,(char *)shadow_item[idx][1], 2);
buff_ptr=calloc(14, sizeof(char)); calloc_ptr[calcnt++]=buff_ptr;
 strcpy(buff_ptr,(char *)crypt(tmp_ptr, saltkey_ptr)); // old_password encryption

//  check if encrypted old_password and encrypted password in shadow_item
are equal
if(strcmp((char *)buff_ptr, (char *)shadow_item[idx][1])==0) goto matched;
printf("    패스워드가 일치하지 않아 작업을 계속할 수 없습니다.\n");
goto freecalloc;

//  new_password key-in process
matched:
passwd_ptr=calloc(100, sizeof(char)); calloc_ptr[calcnt++]=passwd_ptr;
passwd_retry:
printf("\n>> 새로운 패스워드를 입력하세요 (취소키=:) ? ");
scanf("%s", passwd_ptr);    // new_password key-in
if (strcmp((char *)passwd_ptr,":")==0) goto freecalloc;
```

```
if (strlen((char *)passwd_ptr)>20) {printf("      패스워드의 길이가 너무 깁니다.\n"); goto passwd_retry; }

//  check if new & old_passwords are equal
if (strcmp((char *)tmp_ptr, (char *)passwd_ptr)!=0) goto no_match;
printf("          새 패스워드가 현재의 패스워드 '%s'과(와) 같습니다.\n", passwd_ptr);
goto passwd_retry;// if new & old_password are equal, key-in a new_password again

no_match:
jj1=password_check(passwd_ptr);
if (jj1==1) goto passwd_retry;
printf("\n.. 확정된 패스워드 : %s\n", passwd_ptr);

//  saltkey generation & password encryption
saltkey_ptr=calloc(3,sizeof(char)); calloc_ptr[calcnt++]=saltkey_ptr;
get_salt(saltkey_ptr);  // saltkey generation
buff_ptr=calloc(14, sizeof(char)); calloc_ptr[calcnt++]=buff_ptr;
 strcpy(buff_ptr,(char *)crypt(passwd_ptr, saltkey_ptr)); // password encryption

//  new encrypted password & current date insert process
shadow_item[idx][1]=buff_ptr;      // encrypted password
current_date=time(&now_time)/86400;        // current date from 19700101
passwd_ptr=calloc(5,sizeof(char)); calloc_ptr[calcnt++]=passwd_ptr;
sprintf(passwd_ptr,"%d",current_date);
shadow_item[idx][2]=passwd_ptr; // current date
goto display;// go to the file update routine

//  allocated memory spaces are returned free spaces
freecalloc:
if (calcnt<1) goto start;
for (i=calcnt-1; i>=0; i--) free(calloc_ptr[i]);
goto start;
```

```
//    shadow file & passwd file display and update
display:

//   passwd file display
if (sel==2 || sel==3) goto no_display1;   // process no. 2 or 3 : skip display
process
printf("\n            패스워드 파일 디스플레이        =\n");
k=0;
for (i=0; i<no_account; i++) {
printf("    "); if (i!=0) k=ps_index[k];
for (j=0; j<6; j++) printf("%s:",passwd_item[k][j]);
printf("%s\n",passwd_item[k][j]);
}

//   passwd file update
no_display1:if((fpp=fopen("passwd","w"))==NULL)
{printf("\n    패스워드 파일 쓰기 오류가 발생했습니다.\n"); goto end;}
k=0;
for (i=0; i<no_account; i++) {
if (i!=0) k=ps_index[k];
for (j=0; j<6; j++) fprintf(fps,"%s:",passwd_item[k][j]);
fprintf(fpp,"%s\n",passwd_item[k][j]);
}
fclose(fpp);

//   shadow file display
if (sel==2 || sel==3) goto no_display2;   // process no. 2 or 3 : skip displat
process
printf("\n            쉐도우 파일 디스플레이        \n");
k=0;
for (i=0; i<no_account; i++) {
printf("    "); if (i!=0) k=ps_index[k];
for (j=0; j<8; j++) printf("%s:",shadow_item[k][j]);
printf("%s\n",shadow_item[k][j]);
}
```

```
//   shadow file update
no_display2:
if((fps=fopen("shadow","w"))==NULL)
{printf("\n    쉐도우 파일 쓰기 오류가 발생했습니다.\n"); goto end;}
k=0;
for (i=0; i<no_account; i++) {
if (i!=0) k=ps_index[k];
for (j=0; j<8; j++) fprintf(fps,"%s:",shadow_item[k][j]);
fprintf(fps,"%s\n",shadow_item[k][j]);
}
fclose(fps);
goto freecalloc;
//    process termination
end:
printf("\n\n>> 모든 작업이 종료되었습니다 !\n");
}
```

부록[2]. 패스워드 암호화(DES)소스

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#define N 100 // number of password to be processed
char *passwd[N], *saltkey[N], *encrypted[N];
char *passwd_ptr, *file_ptr, *buff_ptr, *saltkey_ptr, *encrypted_ptr, *tst;
char head[]="passwd-", dlm[]=":", tail[]=":13345::::::";
time_t now_time;
FILE *fp;

// saltkey generation module (generated by the current time)
get_salt(char *seed)
{                       char gen_seed[3];
char *c_set="./abcdefghijklmnopqrstuvwxyz"
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
"0123456789";
memset(gen_seed, '₩0', 3);
time(&now_time);
srand((unsigned int)now_time);
gen_seed[0] = c_set[rand() % 64]; gen_seed[1] = c_set[rand() % 64];
memcpy(seed, gen_seed, 3);
}

main()
{                       int i, k, count;
loop:
printf("₩n>> select process ( char listing = 0, password encryption = 1 )
?");
scanf("%d", &i); if (i==0) goto lst; else if (i==1) goto ncr; else goto loop;
ncr:
                        count=0;
                        while(1) {
                        passwd_ptr=calloc(21, sizeof(char));
```

```
                    saltkey_ptr=calloc(3, sizeof(char));
                    encrypted_ptr=calloc(14, sizeof(char));
agn:                printf("₩n>> key-in password ( end = / ) : ?");
                    scanf("%s", passwd_ptr);
                    k=strlen((char *)passwd_ptr);
                    if (k>20) { printf("₩n.. password is too long"); goto
agn; }
                    if ((strcmp((char *)passwd_ptr, "/"))==0) goto nxtstp;
                    i=strcspn((char *)passwd_ptr,":");
                    if (i!=k) { printf("₩n.. colon is not permitted. try again
!"); goto agn; }

                    passwd[++count]=passwd_ptr;
                    get_salt(saltkey_ptr); saltkey[count]=saltkey_ptr;
                    strcpy((char *)encrypted_ptr,(char *)crypt(passwd_ptr,
saltkey_ptr));
                    encrypted[count]=encrypted_ptr;
                    printf(".. time %d  password : %s  ", now_time,
passwd[count]);
                    printf("saltkey : %s  encrypted : %s₩n", saltkey[count],
encrypted[count]);
                    }
nxtstp:
                    file_ptr=calloc(20, sizeof(char));
                    printf("₩n>> shadow file name : ?"); scanf("%s",
file_ptr);
                    printf("₩n.. shadow file name : %s₩n", file_ptr);
                    if((fp=fopen(file_ptr,"w"))==NULL)
                                        {printf("₩n.. Shadow File Write
Error !₩n"); exit(0);}
                    for (i=1; i<=count; i++) {
                    fprintf(fp, "%s%s%s%s%s₩n", head, passwd[i], dlm,
encrypted[i], tail);
                    printf(".. shadow record %2d => %s%s%s%s%s₩n", i,
head, passwd[i], dlm, encrypted[i], tail);
```

```
                    }
                    fclose(fp); goto end;
lst:
                    printf("₩n// char listing  from 32(space) to
126(~)₩n₩n              ");
    for (i=32; i<=126; i++) {
                    tst=(char *)i;
                    if ((i%10)==0) printf("₩n          %c", tst); else printf("
%c", tst);
                    }
end:
                    printf("₩n>> job terminated !₩n");
}
```

참고문헌

[1] 유닉스 시스템 프로그래밍 / 정재은 지음.
[2] 루트 권한 감시를 통한 유닉스의 보안 강화 / 文漢龜