[REPORT - 실습1]

-패스워드 크래킹-

과 목 명 : 컴퓨터보안 - 02
교 수 : 김영부 교수님
학번 : 2020112119
이름 : 강동희
제출일 : 2023.03.21

목 차

1 서 론

1.1 문제 정의

1. 실습 1 : John the Ripper 패스워드 크래킹
Kali Linux 운영체제에 내장되어 있는 패스워크 크래킹 도구인 John the Ripper를 사용하여 가상의 유저를 생성해 패스워크를 크래킹 및 분석한다.

2. 실습 2 : Hash Suite 패스워드 크래킹
Windows 운영체제 전용 패스워크 크래킹 도구인 Hash Suite을 사용하여 윈도우 계정의 패스워드를 크래킹 및 분석한다.

1.2 문제 분석

1. 패스워드 크래킹(Password Cracking)
[패스워드 크래킹 정의와 목적]
암호 해독 및 컴퓨터 보안에서 컴퓨터 시스템에 의해 변형된 형태로 전송된 데이터에서 암호를 복구하는 프로세스를 말한다.[1] 패스워드 크래킹은 사용자가 잊어버린 패스워드를 복구할 수 있도록 도와주며, 허가되지 않은 사용자(해커)가 시스템을 해킹하는 것을 방지하기 위해 패스워드 강도를 확인하기 위해서 패스워드 크래킹을 사용할 수 있다.

[패스워드 크래킹의 종류[2]]
패스워드 크래킹의 종류에는 Brute-Force, Dictionary, Rainbow table, Hybrid, Collision Attack 등이 있다.

1) Brute-Force Attack: 조합 가능한 모든 패스워드를 대입하여 시스템에 저장된 해시값과 비교하는 방법이다.
2) Dictionary Attack : 많은 유저가 사용한 패스워드를 모두 저장하여 대입하여 크래킹하는 방법이다. 사전에 저장된 패스워드만 대입하기 때문에 패스워드를 알아내는데 실패할 수도 있지만, Brute-Force Attack에 비해 저장된 패스워드만 대입하므로 상대적으로 짧은 시간내에 패스워드를 알아낼 수 있다.
3) Rainbow table Attack: 패스워드는 시스템에 '패스워드 + 해시'로 저장되어 있어 패스워드 별로 해시값을 미리 저장해 해시값을 통해 역으로 패스워드를 알아내는 기법이다. 해시함수가 다양하여 모든 데이터를 저장하는 것은 메모리적으로 많은 부담이 되고, 생성하는데 시간 소요가 크기도 하다.

---

[1] Password Cracking: https://en.wikipedia.org/wiki/Password_cracking
[2] Computer Security: Principles and Practice,3rd Ed (W.Stalling & L.Brown, Pearson) Chapter 3.2

4) Hybrid Attack: Brute-Force와 Dictionary Attack을 혼합한 기법이다. Dictionary에 저장된 패스워드를 기반으로 무작위로 대입하여 찾아낸다.
5) Collision Attack: 해시 함수의 특성을 이용한 기법으로, 같은 문자(패스워드)의 경우 같은 해시값을 가져 충돌이 발생한다는 점을 이용해 같은 해시값이 나올 때까지 반복하여 알아낸다.

2. 실습 1: John the Ripper 패스워드 크래킹

[Kali Linux[3]]
Kali Linux는 Offensive Security가 개발한 컴퓨터 운영체제로 '데비안'을 기반으로 만들었다. 다양한 해킹 도구와 라이브러리를 포함하여 보안 학습과 취약점 점검의 목적으로 사용한다. 설치 방법에 대해서는 2장 본론에서 작성한다.

[John the Ripper[4]]
가장 흔히 사용되는 유닉스 패스워드 크래킹 프로그램으로 Brute-Force와 Dictionary Attack의 크래킹 방법을 이용한다. Brute-Force Attack에서는 가능한 모든 일반 텍스트를 해시 테이블과 비교한다. 그 중에서도 문자 빈도표를 통해 자주 사용되는 문자가 포함된 일반 텍스트를 먼저 시도하고, Dictionary에 나오지 않는 패스워드를 크래킹하는데 유용하지만 시간이 오래걸리는 단점이 있다. Dictionary Attack에서는 문자열 샘플을 가져와 비교하는 방법이다. 또한 Dictionary에 저장된 문자열을 기반으로 변형을 통해 대입하기도 한다. Kali Liunx에서 설치한 방법은 2장 본론에서 작성한다.

[시사점]
실습 1을 진행하기 위해서 본인의 운영체제(Mac OS)에서 진행한 설치 방법을 정리하고, John the Ripper(이하 JTR)를 이용한 패스워드 크래킹 방법에 대해 원리와 시스템에 저장된 계정 정보와 암호화 형태에 대해 알아보고자 한다.

3. 실습 2: Hash Suite 패스워드 크래킹

[Hash Suite[5]]
Windows에서 암호 해시의 보안을 테스트하는 프로그램이며 개발사인 openwall이 소개하는 목표는 다음과 같다.
높은 성능을 가진 프로그램을 제공하며, GUI를 통해 단순하고 직관적으로 사용할 수 있다. 또한 통계를 통한 보고서와 최신 Dictionary를 이용해 취약한 암호를 쉽게 파악할 수 있다고 한다. Hash Suite는 13가지의 해시 유형을 지원한다.
설치 방법은 2장 본론에서 작성한다.

---

[3] 칼리 리눅스 : https://ko.wikipedia.org/wiki/칼리_리눅스

[4] 존 더 리퍼 : https://ko.wikipedia.org/wiki/존_더_리퍼

[5] Hash Suite: https://hashsuite.openwall.net (Home, Hash Suite main object)

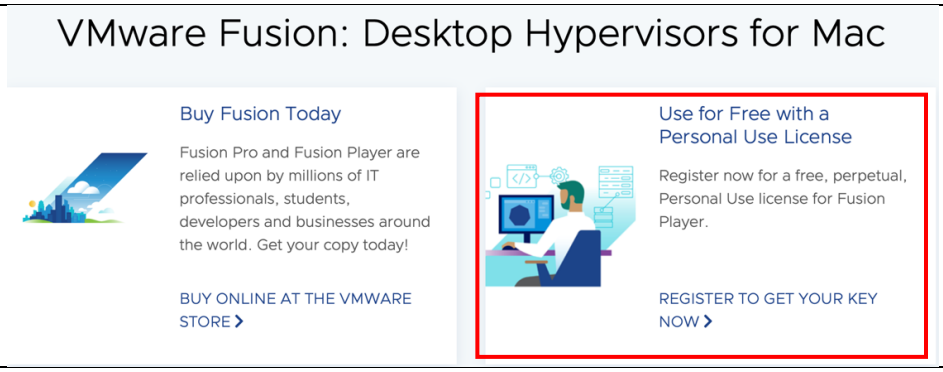
[시사점]
실습 2를 진행하면서 실습 1과 마찬가지로 운영체제(Windows)에서 진행한 설치 방법을 정리하고, 실습 결과에 대해 분석과 더불어 패스워드 길이에 따른 Time Complexity의 분석을 진행하고자 한다.

2 본 론

2.1 실습 1: John the Ripper 패스워드 크래킹

1. 실습환경 구축
   [실습환경 정보]
   Device: MacBook Pro (2019)
   OS: macOS Ventura (ver. 13.2.1)
   Processor: 2.6 GHz 6-core Intel Core i7
   Memory: 16GB 2667 MHz DDR4
   GPU: Intel UHD Graphics 630

1. 가상머신 설치하기
VMware 공식 홈페이지[6]에 접속해 로그인 및 우측 Free 버전 설치를 위해 Personal Use License를 발급받아 설치를 진행한다.

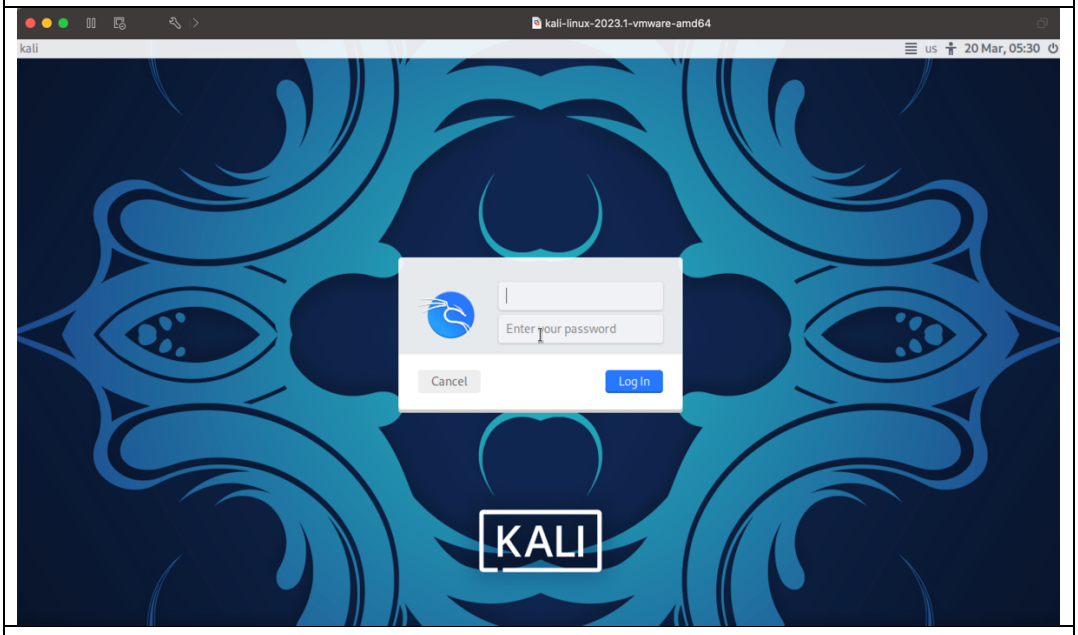


2. Kali Linux 설치하기[7]
칼리 리눅스 다운로드 페이지에서 VMware 전용 ISO 파일을 다운로드.

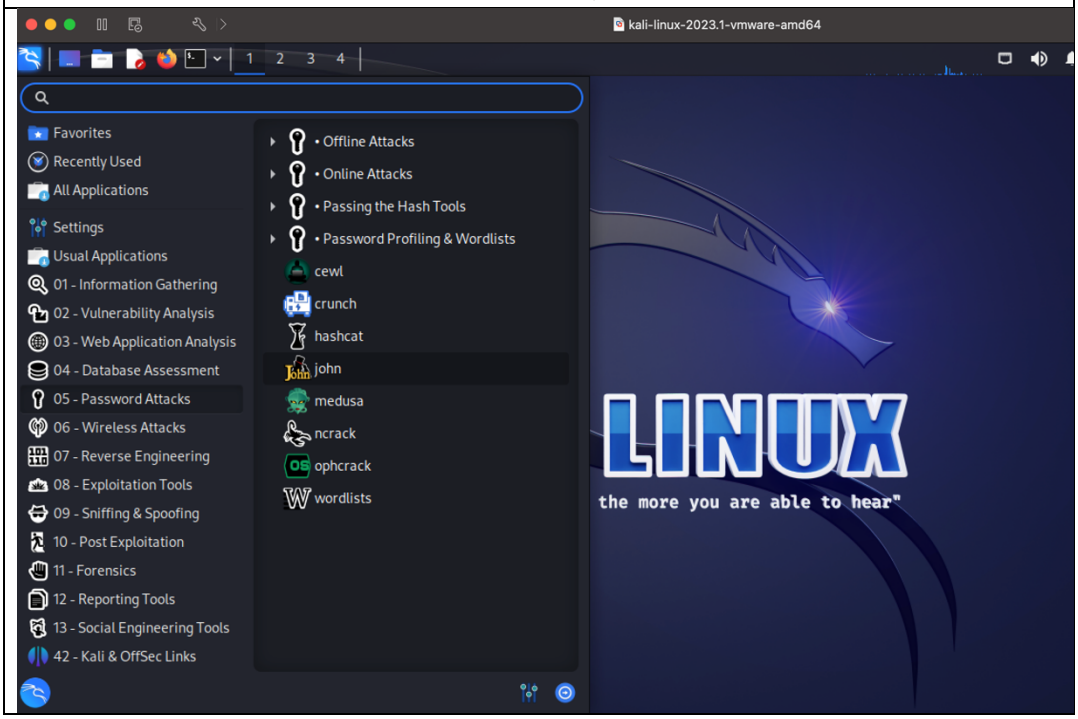
[6] VMware Official Homepage: https://www.vmware.com/products/fusion.html
[7] Kali Linux VM Download page: https://www.kali.org/get-kali/#kali-virtual-machines

3. ISO 파일 압축해제 후 실행하여 설치를 진행한다.

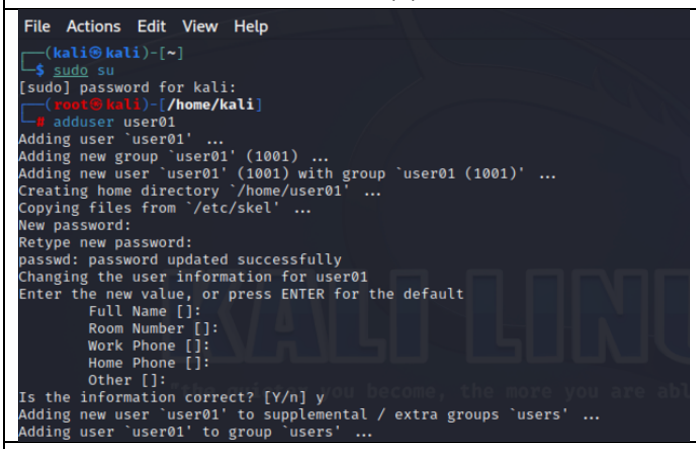
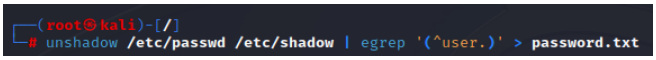
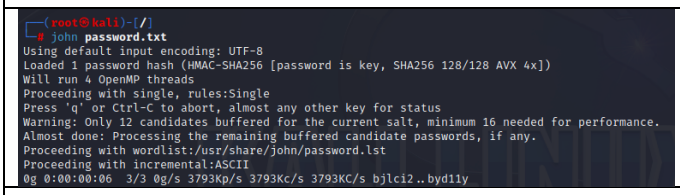
초기 부팅 후 로그인 정보는 ID : kali, PW : kali 로 로그인하여 접속한다.



4. 접속 후 GUI를 보면 JTR이 내장되어 있어 실습 환경 구축을 완료한다.

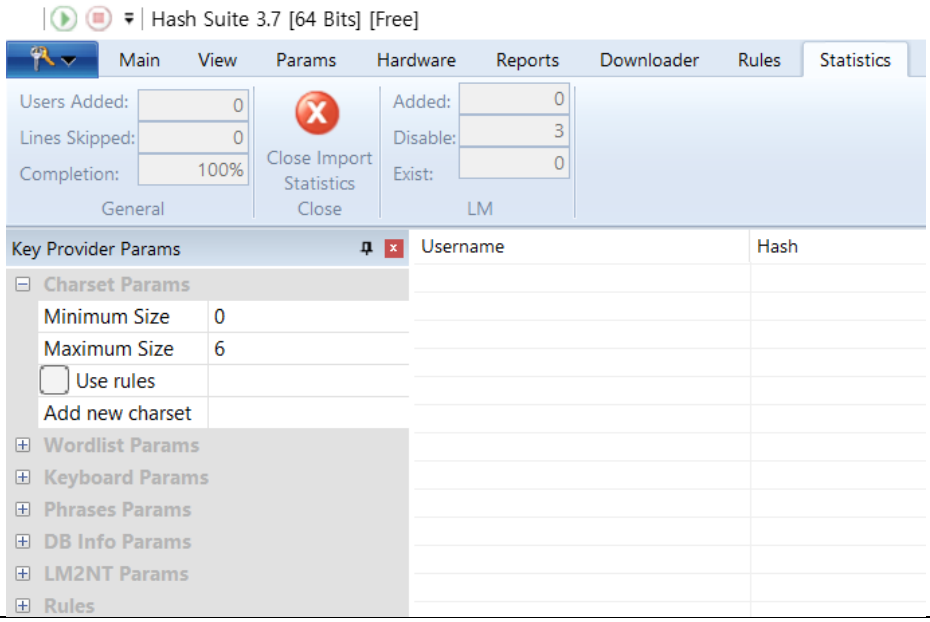
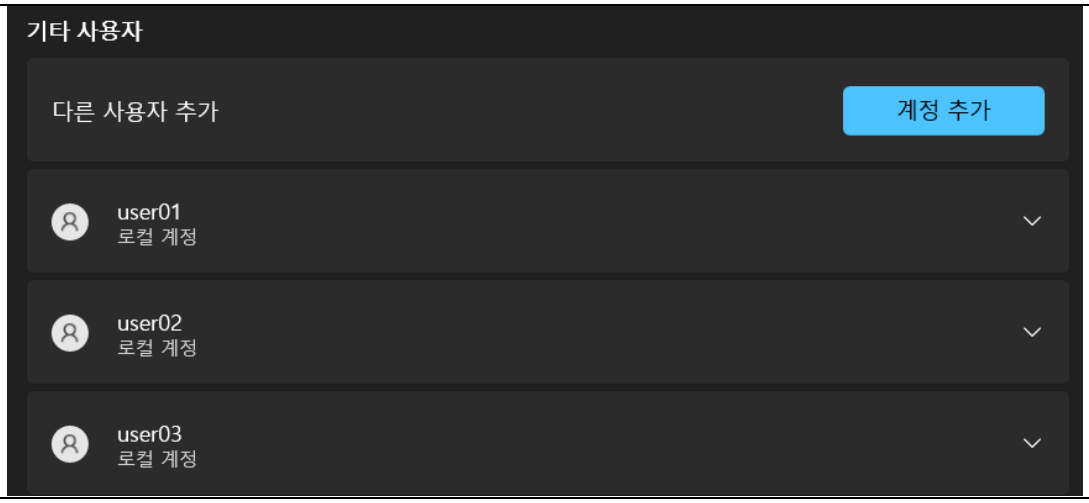


2. JTR Password Cracking 실습 진행

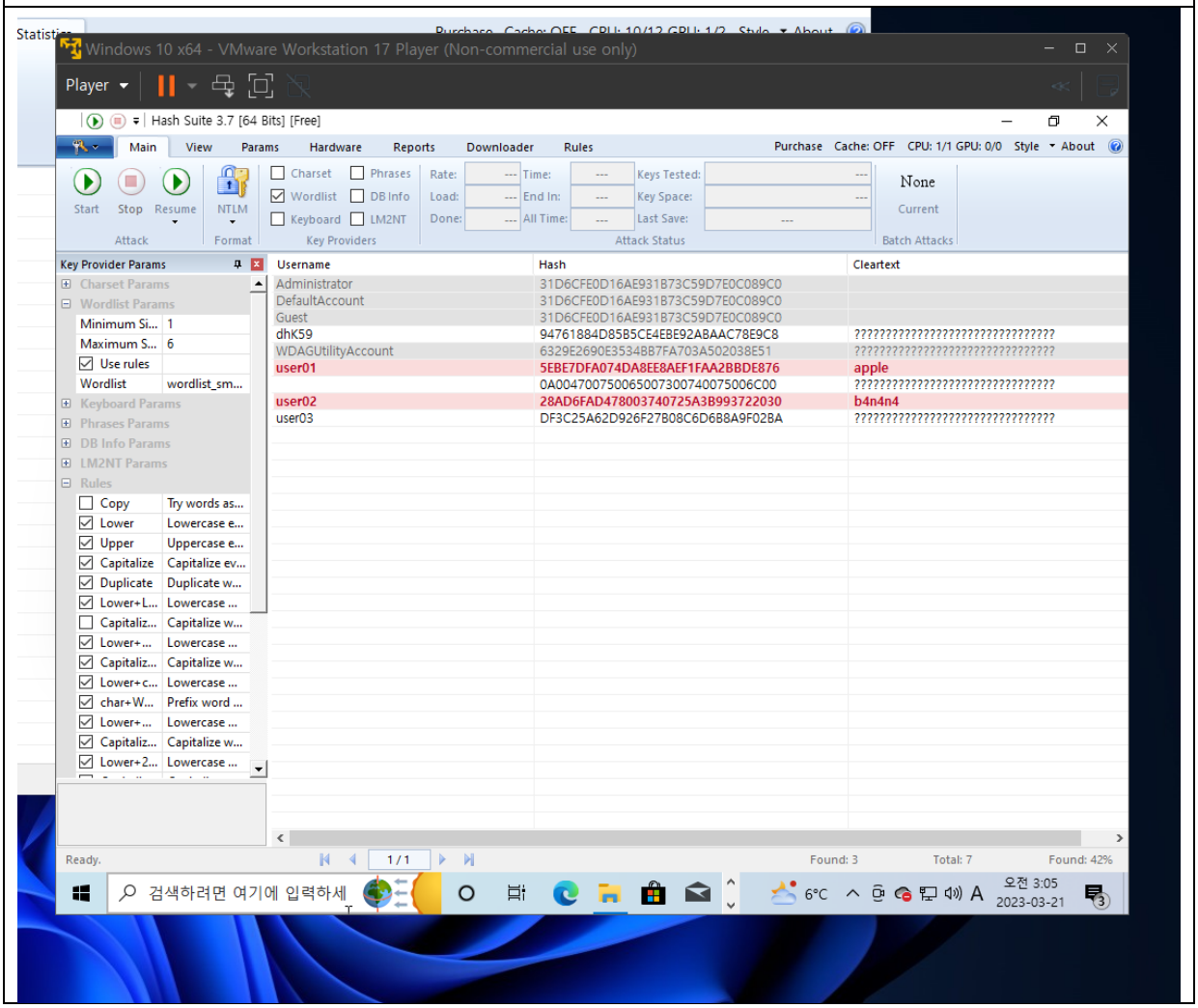
1. 터미널에서 관리자 권한 획득 후 가상 유저 생성(User 01, apple)	Terminal Command
	>sudo su sudo(SuperUser DO):루트 권한 su(Switch User):계정 전환 >adduser user01 'user01'의 계정을 추가
2. txt 파일에 생성한 유저의 passwd, shadow 정보를 저장한다.	unshadow /etc/passwd /etc/shadow | egrep '(^user.)' > password.txt
	password.txt 파일에 user(01-03)의 패스워드와 shadow 정보를 저장하는 명령어 **다른 계정도 여러 개 있어 따로 추출하여 해당 계정만 크래킹을 진행하기 위해서 진행하였다.*
3. JTR을 이용해 생성한 계정의 패스워드를 크래킹을 진행한다.	> john password.txt
	2에서 진행한 password.txt의 계정 정보를 크래킹한다.
4. user02(b4n4n4)와 user03(P@ssw0rD)로 추가하여 2~3번을 진행하여 결과를 확인한다.	-

2.2 실습 2: Hash Suite 패스워드 크래킹

1. 실습환경 구축
   [실습환경 정보]
   Device: DELL Inspiron 7590
   OS: Windows 11 Pro
   Processor: Intel i7-9750H
   Memory: 8GB
   GPU: NVIDIA GeForce GTX 1650

1. Hash Suite 무료 버전 설치

2. Windows-계정관리-사용자 정보-가족 및 다른 사용자-PC에 다른 사용자에서 계정을 생성한다. [user01(apple), user02(b4n4n4), user03(P@ssw0rD)]


3. Key > Import > Local accounts, 실행



3 결 론

3.1 실습 결과

1. 실습 1: John the Ripper 패스워드 크래킹

1. 명령어를 통해 생성한 계정의 패스워드 크래킹을 진행한다.

```
┌──(root㉿kdhee)-[/home/kdh/Downloads/john-1.9.0/run]
└─# ./john -format=crypt /etc/shadow
Loaded 4 password hashes with 4 different salts (crypt, generic crypt(3) [?/6
4])
Press 'q' or Ctrl-C to abort, almost any other key for status
```

2. 실행 후 생성한 가상 유저의 계정 패스워드를 알아냈다 (user01, 02’38” 소요)

```
0g 0:00:02:38 100% 1/3 0g/s 73.83p/s 73.83c/s 73.83C/s 999991912..999991900
apple            (user01)
```

3. user02 가상 유저의 계정 패스워드 크래킹을 진행했지만 실패했다. (9시간 소요)

```
1g 0:00:02:45 0% 2/3 0.006059g/s 71.75p/s 74.39c/s 74.39C/s piglet..knight
1g 0:00:02:58 0% 2/3 0.005617g/s 68.67p/s 75.43c/s 75.43C/s brenda..keith
1g 0:00:03:55 1% 2/3 0.004239g/s 57.93p/s 75.65c/s 75.65C/s 1234qwer..babygirl
1g 0:05:16:20 3/3 0.000052g/s 33.06p/s 97.97c/s 97.97C/s 129986..129197
1g 0:05:16:32 3/3 0.000052g/s 33.06p/s 97.97c/s 97.97C/s 129414..129304
1g 0:05:16:44 3/3 0.000052g/s 33.06p/s 97.97c/s 97.97C/s 117856..117364
1g 0:08:03:45 3/3 0.000034g/s 33.01p/s 98.23c/s 98.23C/s saizun..shairy
1g 0:08:04:04 3/3 0.000034g/s 33.01p/s 98.23c/s 98.23C/s shrom2..shmi19
1g 0:08:04:07 3/3 0.000034g/s 33.01p/s 98.23c/s 98.23C/s shmis3..sh1019
1g 0:09:16:58 3/3 0.000029g/s 32.99p/s 98.28c/s 98.28C/s smiggi..smitch
```

4. user03 가상 유저의 계정 패스워드 크래킹을 진행했지만 실패했다. (13시간 소요)

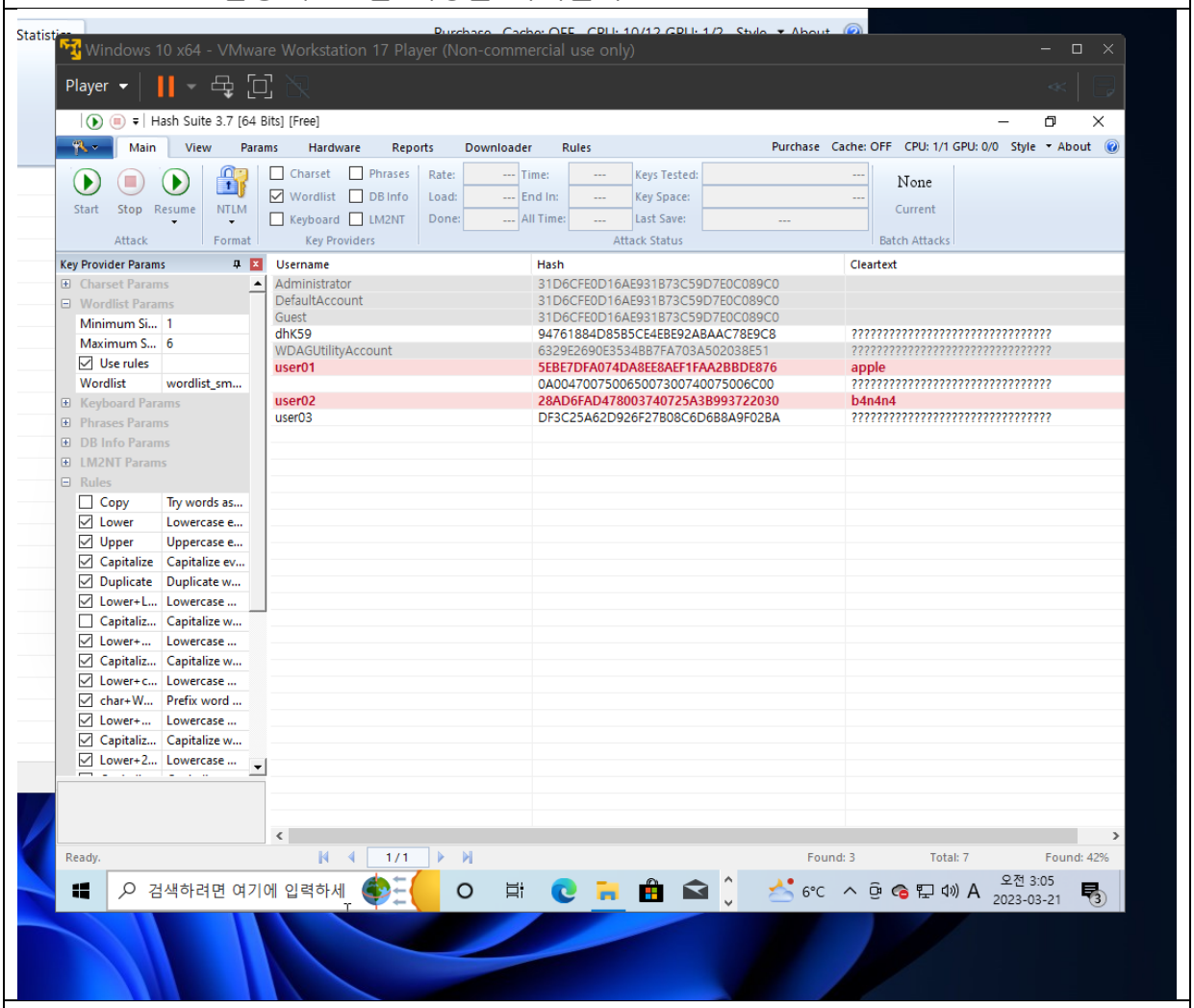
```
Proceeding with incremental:ASCII
0g 0:00:30:25  3/3 0g/s 90.32p/s 177.4c/s 177.4C/s 10120..asdan
0g 0:00:31:48  3/3 0g/s 90.11p/s 177.1c/s 177.1C/s juige..jight
0g 0:00:47:06  3/3 0g/s 89.41p/s 176.7c/s 176.7C/s 146485..146188
0g 0:00:48:44  3/3 0g/s 88.97p/s 175.9c/s 175.9C/s aresun..baisen
0g 0:00:54:01  3/3 0g/s 89.17p/s 176.5c/s 176.5C/s mykuma..mypols
0g 0:01:59:02  3/3 0g/s 89.36p/s 177.9c/s 177.9C/s 096322..096063
0g 0:02:06:47  3/3 0g/s 89.29p/s 177.7c/s 177.7C/s 065770..069846
0g 0:13:21:57  3/3 0g/s 89.42p/s 178.7c/s 178.7C/s maron147..marocore
0g 0:13:28:38  3/3 0g/s 89.42p/s 178.7c/s 178.7C/s brompier..brompy25
```

5. Mac에서 실패하여 windows 노트북으로 진행했지만 user02와 user03 계정 모두 크래킹하는데 실패했다. (37시간 소요)

```
Proceeding with incremental:ASCII
0g 0:10:57:41  3/3 0g/s 59.12p/s 176.9c/s 176.9C/s djamo1..djam2x
0g 0:13:56:41  3/3 0g/s 59.06p/s 176.8c/s 176.8C/s pslin..psluv
0g 0:14:14:41  3/3 0g/s 59.05p/s 176.8c/s 176.8C/s spc371..spcbcs
0g 1:03:57:47  3/3 0g/s 59.17p/s 177.3c/s 177.3C/s taiak1..tiggg4
0g 1:10:18:28  3/3 0g/s 59.20p/s 177.4c/s 177.4C/s 12211386..12212214
0g 1:12:55:18  3/3 0g/s 59.20p/s 177.4c/s 177.4C/s mutulo1..mutaca1
0g 1:13:01:11  3/3 0g/s 59.19p/s 177.4c/s 177.4C/s sepel05..sepaket
```

2. 실습 2: Hash Suite 패스워드 크래킹

1. Hash Suite 실행 후 로컬 계정을 가져온다.



2. 패스워드 크래킹을 수행한 결과이다.

Administrator	31D6CFE0D16AE931B73C59D7E0C089C0	
DefaultAccount	31D6CFE0D16AE931B73C59D7E0C089C0	
Guest	31D6CFE0D16AE931B73C59D7E0C089C0	
user01	5EBE7DFA074DA8EE8AEF1FAA2BBDE876	apple
user02	28AD6FAD478003740725A3B993722030	b4n4n4

3.2 결과 분석

1. 실습 1: John the Ripper 패스워드 크래킹
   [첫번째 시도 : user01의 패스워드 크래킹 성공]
   3.1의 1과 같이 JTR을 이용한 실습에서는 user01의 계정만 패스워드 크래킹을 성공했다. 첫번째 시도할 때 모든 계정을 순차적으로 크래킹을 진행하는 것이 오래 소요될 것 같아 계정을 하나씩 생성하여 진행했다. 하지만 user02에서 진행하면서 약 9시간 진행한 결과 확인할 수 없었고, user02보다 패스워드

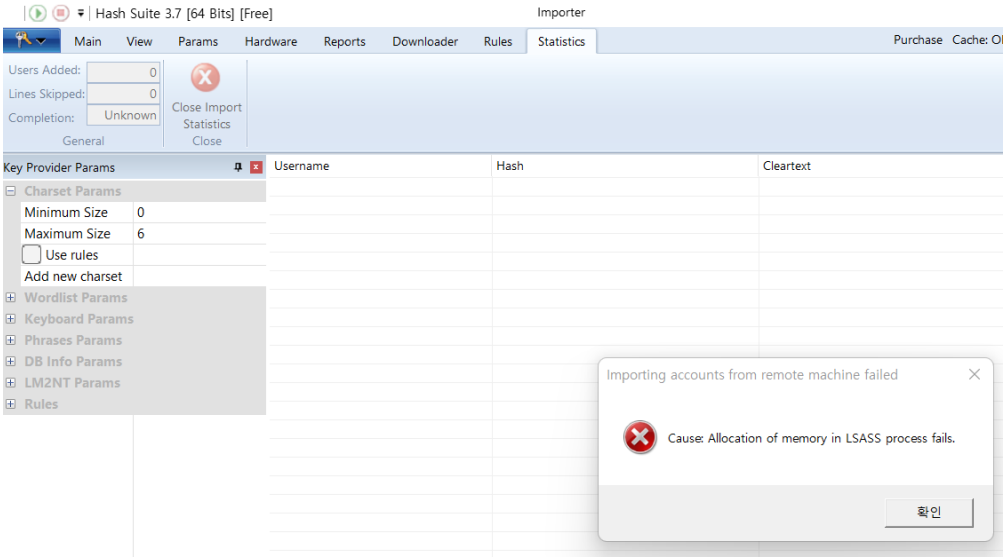
의 길이와 복잡한 패스워드인 user03도 역시 확인할 수 없었다.

[두번째 시도 : 운영체제 변경 후 패스워드 크래킹 진행]
다른 운영체제인 Windows 노트북(Dell Inspiron 7590, Intel i7-9750)으로 진행하여 user02와 03의 패스워드 크래킹을 약 37시간에 걸쳐 실습을 진행했지만 실패했다.

[JTR의 패스워드 크래킹 방법 분석]
실습에서 진행한 JTR의 작동방식을 알아보았을 때 두가지 모드로 패스워드 크래킹을 진행했다. 첫번째로 공식 제작사에서 권장한 모드로 'Single Crack mode'이다. 해당 모드는 유닉스 비밀번호와 shadow 파일안의 GECOS 필드에 있는 사용자의 전체 이름, 사용자명 등의 단서를 이용해 비밀번호를 추측해 무차별 대입하여 크래킹하는 방식으로 진행되었다. 만약 Single Crack mode에 실패했을 경우, 다음 단계인 'Wordlist mode'를 진행한다. Dictionary attack과 마찬가지로 여러 단어(패스워드)가 존재하는 파일을 이용해 대입하고 변형하여 적용하는 크래킹 방식으로 진행된다.
JTR은 사용자 계정 및 정보가 유닉스 내에서 /etc/passwd 파일에 저장되고 각 계정의 패스워드는 /etc/shadow 파일에 저장되는 점을 이용하여 shadow 파일에 접근하여 크래킹을 진행한다. 따라서 실습을 진행할 때 모든 계정에 접근해서 크래킹을 진행하는 것을 방지하고자 passwd와 shadow 파일에 있는 테스트 계정을 하나의 파일로 저장하여 실습을 진행하였다. 2장 2.1.1.2와 같이 password.txt의 파일로 합쳤다.

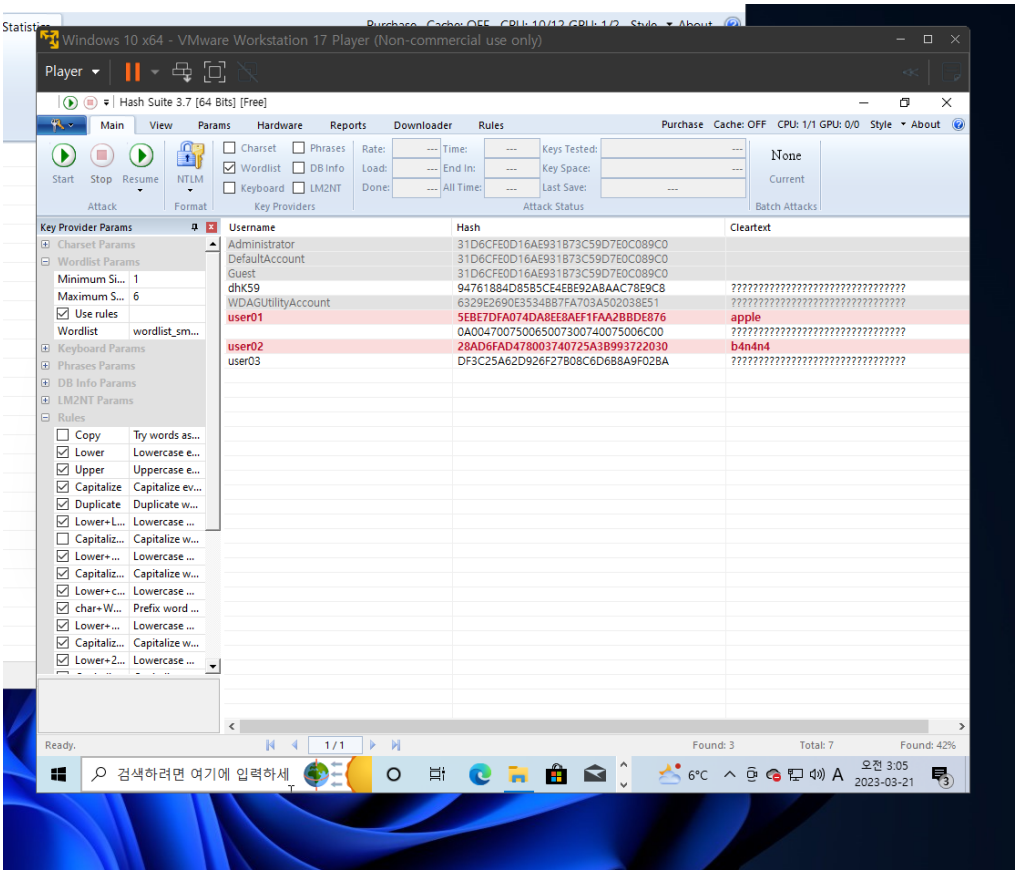
unshadow 명령어를 통해 사용자의 계정 정보가 담긴 /etc/passwd, 각 계정의 패스워드가 저장된 /etc/shadow 중에서 **egrep '(^가져올 계정 단어.)'**를 통해 user로 시작하는 계정을 가져와 password.txt 파일에 저장하였다.
하지만, 테스트 계정만 가져왔음에도 불구하고 실습환경의 사양으로 인해 무차별 대입 방법으로 알아내기 힘들었다고 판단했다.

2. 실습 2: Hash Suite 패스워드 크래킹
[로컬 환경에서 실행 시 오류 발생]
Hash Suite 실습을 진행하면서 방화벽 해제, 백신 프로그램 종료 후 설치하여 실습을 진행하였다. 테스트 유저 User01~03을 생성 후 Local Accounts를 Import 하는 과정에서 아래 이미지와 같은 오류가 발생했다. 운영체제 메모리에 접근할 수 없다는 오류로 판단하여 가상머신을 통해 방화벽, 백신에 영향없도록 윈도우를 설치하여 실습을 진행하여 해결하고자 했다.



<로컬 계정을 가져오는 과정에서의 에러>

VMware를 설치 후 Windows 운영체제를 설치하고 실습 과정처럼 Hash Suite을 설치, 가상 유저 01~03을 생성하여 실습을 진행하였다.



보안 프로그램에 의한 제어가 없어 로컬 계정을 가져올 수 있었고, 패스워드 크래킹도 진행할 수 있었다. Hash Suite Free 버전을 사용하여 비밀번호가 최대 6자리밖에 되지 않아 user03의 패스워드인 P@ssw0rd, 8자리의 비밀번호를 진행하지 못했다.

3.3 고찰(Discussion)

1. 리눅스 운영체제에서의 사용자 정보와 접근 권한에 관하여

실습 1을 원활하게 진행하고자 했던 unshadow를 사용하면서 사용자 계정 정보와 계정의 비밀번호가 따로 저장된 것을 알게 되어 password.txt로 합쳐서 크래킹을 진행했었다. 이때 사용자의 정보와 관련된 필드에 대해 궁금하여 알아보았다.

먼저 사용자 정보가 담긴 /etc/passwd 파일에 대해 조사하였다. passwd 파일을 통해 사용자의 계정과 인증을 관리하게 된다. Root 계정을 예시로 각 필드의 의미하는 바를 알아보았다. (colon을 기준으로 필드와 설명을 정리했다[8])

Field	Description
계정 이름	사용자의 계정 이름을 나타낸다.
패스워드	x는 /etc/shadow에 암호화되어 저장된 것을 의미한다.
UID	사용자의 user ID를 나타낸 것으로, 0은 관리자 계정을 의미한다.
GID	사용자의 group ID를 나타낸 것으로, 0은 관리자 그룹을 의미한다.
Comment	유저에 대한 정보로 계정명으로 나타낸다.
Home Dir	유저의 홈 디렉토리를 나타낸다. 관리가 계정은 root이며, 다른 유저의 계정은 보통 home으로 위치한다.
Login Shell	유저가 로그인 할 때 사용할 shell을 의미한다. 보통 bash shell을 사용하며, 로그인이 불필요한 계정 같은 경우 /nologin, /false로 표기(deamon 계정 참고).

[8] Linux UID Meaning: https://www.fosslinux.com/39230/what-is-uid-in-linux-and-how-to-find-and-change-it.html

다음으로 사용자 계정의 패스워드가 담긴 /etc/shadow 파일이다. 해당 파일에는 암호화된 패스워드와 패스워드의 암호화 정책이 담겨있다. 관리자 계정/그룹일 경우 해당 파일에 접근할 수 있다. Shadow 파일을 확인하여 소유자 및 권한 설정 필드를 알아보았다.[9]

```
┌──(root㉿kali)-[~]
└─# cat /etc/shadow
root:*:19426:0:99999:7:::
```

계정 이름	계정 이름을 나타낸다.
암호화된 패스워드	*은 패스워드가 잠긴 상태이며, 설정하지 않은 생태이다. (kali linux를 설치 후 default 계정을 사용하기 때문)
마지막 수정	패스워드가 마지막으로 변경한 날이다. (default는 1970.01.01로 19,426일이 지났다는 의미이다.)
Min 사용기간	패스워드 변경 후 해당 기간만큼 변경하지 못한다는 의미
Max 사용기간	패스워드 변경 후 만료일을 의미한다
경고, 비활성화, 만료일	7은 패스워드 만료일 7일 전에 경고한다는 의미이다.

Shadow 파일에서 2주차 이론 수업에 배운 접근 권한에 대해서도 확인할 수 있는지 알아보았다. ls -l 명령어를 통해서 확인할 수 있었다.
(ls : list의 약어로 디렉토리, 파일 목록을 출력하는 명령어)

```
┌──(root㉿kali)-[~]
└─# ls -l /etc/shadow
-rw-r----- 1 root shadow 1510 Mar 20 06:12 /etc/shadow
```

Permissions Examples (Regular Files)

-rw-r—r--	read/write for owner, read-only for everyone else
-rw-r-----	read/write for owner, read-only for group, forbidden to others
-rwx------	read/write/execute for owner, forbidden to everyone else
-r--r--r--	read-only to everyone, including owner
-rwxrwxrwx	read/write/execute to everyone

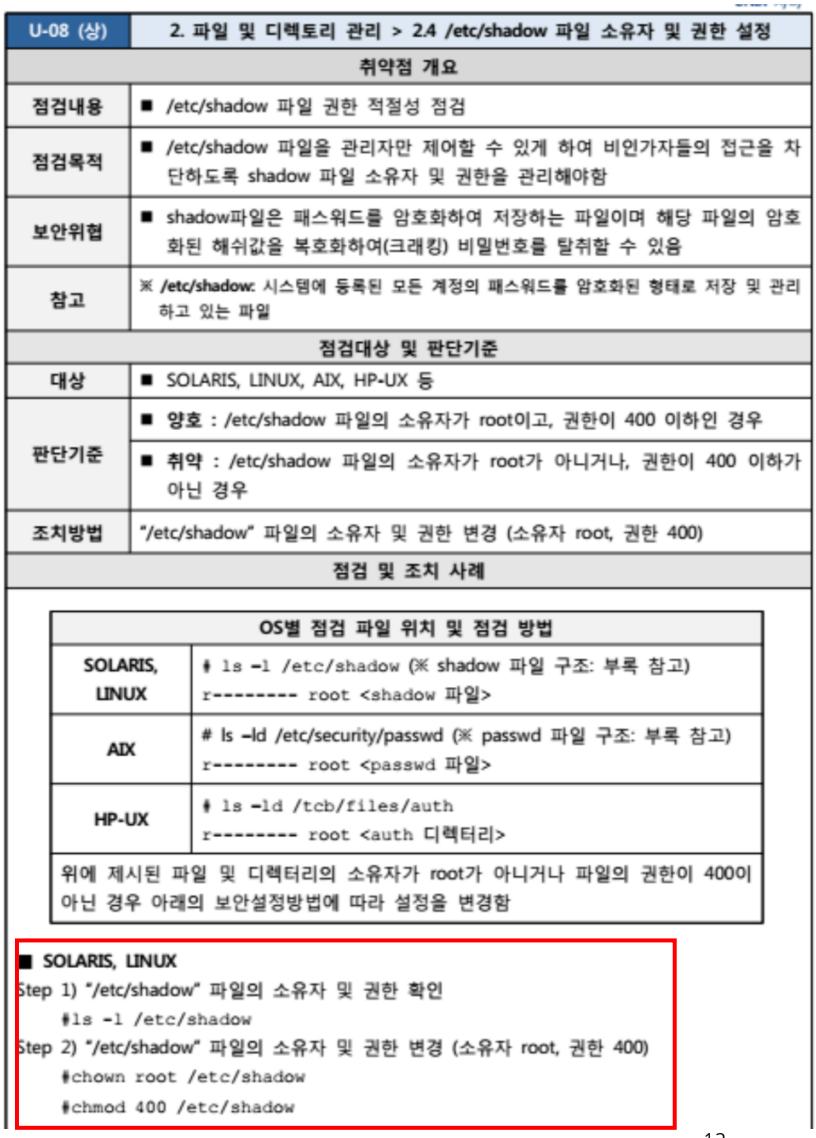
<컴퓨터보안-3장 OS Security P.42[10]>

[9] Linux Shadow field: https://www.cyberciti.biz/faq/understanding-etcshadow-file/

[10] 동국대학교 컴퓨터보안 강좌(김영부 교수) 교안 참고(03-OS Security)

Root 계정의 접근 권한으로는 owner는 Read/Write할 수 있으며, Group은 Read만 가능한 상태인 것을 확인할 수 있다.
(Read/write for owner, read for group, forbidden to others)

Linux 계정 권한에 대해 조사하던 중 '주요정보통신기반시설 취약점 Unix 점검 항목[11]'에 대해서도 알게 되었는데 /etc/shadow 권한에 대한 적절성을 점검하는 절차가 필요하다고 한다. 중요 기관 및 시설은 신규로 지정된 후 6개월 그리고 매년 취약점 분석/평가를 실시한다고 한다. 총 453개의 관리/물리/기술적 점검항목에 대한 취약여부를 점검 및 대응을 위한 개선 과정이라고 한다. 해당 절차에 따라 현재 내 운영체제의 관리자 권한도 점검 및 판단해보았다.



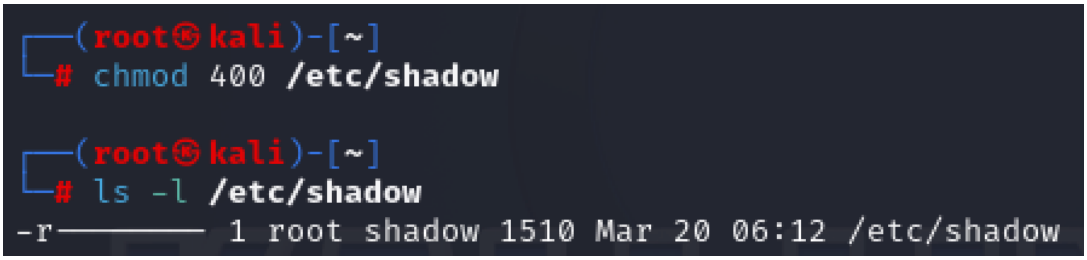
U-08 (상)	2. 파일 및 디렉토리 관리 > 2.4 /etc/shadow 파일 소유자 및 권한 설정
취약점 개요	
점검내용	■ /etc/shadow 파일 권한 적절성 점검
점검목적	■ /etc/shadow 파일을 관리자만 제어할 수 있게 하여 비인가자들의 접근을 차단하도록 shadow 파일 소유자 및 권한을 관리해야함
보안위협	■ shadow파일은 패스워드를 암호화하여 저장하는 파일이며 해당 파일의 암호화된 해쉬값을 복호화하여(크래킹) 비밀번호를 탈취할 수 있음
참고	※ /etc/shadow: 시스템에 등록된 모든 계정의 패스워드를 암호화된 형태로 저장 및 관리하고 있는 파일
점검대상 및 판단기준	
대상	■ SOLARIS, LINUX, AIX, HP-UX 등
판단기준	■ 양호 : /etc/shadow 파일의 소유자가 root이고, 권한이 400 이하인 경우 ■ 취약 : /etc/shadow 파일의 소유자가 root가 아니거나, 권한이 400 이하가 아닌 경우
조치방법	"/etc/shadow" 파일의 소유자 및 권한 변경 (소유자 root, 권한 400)
점검 및 조치 사례	

OS별 점검 파일 위치 및 점검 방법	
SOLARIS, LINUX	# ls -l /etc/shadow (※ shadow 파일 구조: 부록 참고) r-------- root <shadow 파일>
AIX	# ls -ld /etc/security/passwd (※ passwd 파일 구조: 부록 참고) r-------- root <passwd 파일>
HP-UX	# ls -ld /tcb/files/auth r-------- root <auth 디렉터리>
위에 제시된 파일 및 디렉터리의 소유자가 root가 아니거나 파일의 권한이 400이 아닌 경우 아래의 보안설정방법에 따라 설정을 변경함	

■ SOLARIS, LINUX
Step 1) "/etc/shadow" 파일의 소유자 및 권한 확인
#ls -l /etc/shadow
Step 2) "/etc/shadow" 파일의 소유자 및 권한 변경 (소유자 root, 권한 400)
#chown root /etc/shadow
#chmod 400 /etc/shadow

〈주요정보통신기반시설 기술적 취약점 분석,평가 방법 상세가이드[12](2021.3)〉

[11] https://www.law.go.kr/행정규칙/주요정보통신기반시설%20취약점%20분석·평가%20기준
[12] https://www.kisa.or.kr/2060204/form?postSeq=12&lang_type=KO&page=1#fndoDocumentPreview



Step 1은 위에서 확인했으므로 Step2 진행 및 권한 변경 확인을 진행해보았다.

2. 암호화 강도의 수학적 접근

진행한 실습의 경우 모든 경우의 수를 Brute-Force로 대입하여 패스워드를 크래킹하여 실습 환경 사양에 따라 차이가 있겠지만, 시간 소요가 오래 걸린다는 단점이 있으며, 이론적으로 얼마나 오래 걸리는지, 암호 조합의 중요성을 확인하고자 알고리즘적으로 접근해보고자 한다.

(1) 길이 r의 패스워드를 중복없이 n개의 문자 리스트에서 생성할 경우
해당 암호의 경우 Permutation이다. 문자열 리스트에서 순서대로 뽑아 나열하는 것이 순열이므로 경우의 수도 ${}_nP_r$ = n x (n-1) x ⋯ x n-r+1이 된다. Permutation의 대표적인 문제로 Traveling Salesman Problem(TSP)가 있다. 시간 복잡도는 $O(n!)$ 이다.

(2) 길이 r의 패스워드를 중복을 허용한 n개의 문자에서 생성할 경우
(1)과 달리 차이점을 중복을 허용한다는 점에서 Combination이다. 따라서 경우의 수는 ${}_nC_r = \frac{n!}{(n-r)!r!}$이 된다. 시간 복잡도는 $O(r^n)$.

패스워드를 생성할 수 있는 경우는 Permutation과 Combination 정도로 알아보고, 다음으로 생각할 수 있는 점은 문자 리스트에서 살펴볼 수 있다.

(3) 알파벳 소문자로 생성할 경우
(4) 알파벳 대,소문자로 생성할 경우

(1)~(4)의 시간 복잡도를 확인하기 위해 간단한 코드를 구현하여 측정 및 분석을 진행해보았다.

*기존 계획은 숫자 및 특수기호를 활용하여 소문자/대+소문자/소문자+숫자/대소문자+숫자/소문자+특수기호/대소문자+특수기호/대소문자+숫자+특수기호 등 조합할 수 있는 모든 경우의 수를 비교해보고자 했지만, 실습과 병행하기 힘들어 모두 진행하지 못했다.

(1) 실험을 위한 1,2의 기본 함수 생성 및 문자열 리스트 생성

```python
#combination의 패스워드 생성 코드
#실험을 위해 문자열과 패스워드의 길이를 parameter로 설정하였다.
def passwd_combinations(arr, n):
    result =[]

    if n == 0:
        return [[]]

    for i in range(0, len(arr)):
        elem = arr[i]
        rest_arr = arr[i + 1:]
        for C in passwd_combinations(rest_arr, n-1):
            result.append([elem]+C)

    return result
```

```python
#permutation의 패스워드 생성코드
#실험을 위해 문자열과 패스워드의 길이를 parameter로 설정하였다.
def passwd_permutations(arr, n):
    result = []

    if n == 0:
        return [[]]

    for i, elem in enumerate(arr):
        for P in passwd_permutations(arr[:i] + arr[i+1:], n-1):
            result += [[elem]+P]

    return result
```

```python
from string import ascii_lowercase, ascii_letters

passwd_lower = list(ascii_lowercase) #알파벳 소문자 리스트

passwd_letters = list(ascii_letters) #알파벳 대소문자 리스트
```

(2) 패스워드 생성 함수 및 결과 확인

Permutation의 경우 문자열 길이에 따른 시간 분석

```
import time

length = [3, 5]

for t in length:
  start = time.time()
  passwd_permutations(passwd_lower, t)
  end = time.time()
  print(f"패스워드 길이 {t} | 소요시간 {end - start:.5f} 초")
```

```
패스워드 길이 3 | 소요시간 0.02471 초
패스워드 길이 5 | 소요시간 23.35069 초
```

Combination의 경우 문자열 길이에 따른 시간 분석

```
[5]  import time

     length = [0, 3, 5, 8, 10]

     for t in length:
       start = time.time()
       passwd_combinations(passwd_lower, t)
       end = time.time()
       print(f"패스워드 길이 {t} | 소요시간 {end - start:.5f} 초")
```

```
패스워드 길이 0 | 소요시간 0.00000 초
패스워드 길이 3 | 소요시간 0.00563 초
패스워드 길이 5 | 소요시간 0.13344 초
패스워드 길이 8 | 소요시간 6.60741 초
패스워드 길이 10 | 소요시간 30.61861 초
```

(3) 실험 결과 분석

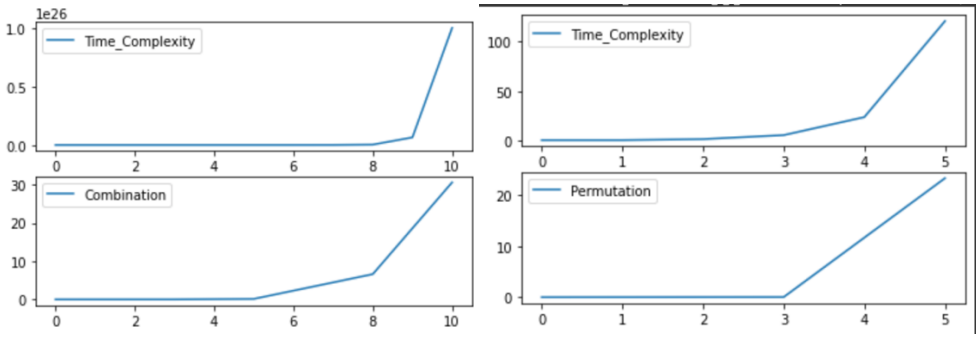
실습은 Colab을 통해 파이썬 라이브러리를 활용하여 Permutation과 Combination 함수를 구현하여 문자열의 길이와 패스워드를 생성할 수 있는 문자(소문자/대+소문자)를 변경하여 파라미터로 입력 받을 수 있게 설계하였다.

첫번째 분석은 소문자의 조합으로 Permutation과 Combination으로 패스워드를 생성했을 때의 시간 소요를 비교해보았다.

Permutation (lower)		Combination (lower)	
Length	Time (sec)	Length	Time (sec)
0	0	0	0
3	0.02471	3	0.00563
5	23.35069	5	0.13344
-	-	8	6.60741
-	-	10	30.61861

Permutation의 경우 Colab의 RAM이 약 13GB밖에 되지 않아 셀을 돌리던 중 멈추게 되어 중단했다. 예상한 실험 결과는 중복을 허용하지 않은 Permutation의 코드가 중복을 허용하는 Combination의 코드 실행시간보다 짧게 소요되어야 패스워드 크래킹 중 Brute-Force의 걸리는 시간과 암호의 강도를 확인할 수 있을 것이라고 생각했다. 하지만 Permutaion 코드의 실행시간이 Combination 코드보다 오래 실행된 것을 확인할 수 있었다. 원인을 보니 각각의 코드가 동일한 형태(반복문, 리스트 형태 등)가 아니라 실행 시간이 다르게 나왔을 것으로 판단했다.

두번째 분석은 순열, 조합으로 실행한 코드와 해당 함수의 시간복잡도와 유사한지 시각화를 통해 비교해보았다.



적은 수의 데이터이지만 그래프의 형태는 유사하여 시간 복잡도 분석이 올바르게 되었다는 것을 검증할 수 있었다.

3. 느낀 점

컴퓨터 보안의 첫번째 실습을 진행해보면서 CS분야 중에서 새롭게 공부하는 보안에 대해 재미를 느낄 수 있었다. 보안에 대한 이론강의를 수강하고 실습을 진행해 보면서 파일 시스템의 권한부터 패스워드 크래킹 등 직접 확인하고 경험한 부분이 좋았다. 실습을 진행하는 과정이 시간도 오래 걸리고 실습 환경을 세팅하는 부분에 있어서 많은 어려움이 있었지만, 진행하기 위해 다양한 문서들을 읽어보면서 해결하는 과정도 보고서를 작성하면서 정리되면서 도움이 되었던 것 같다. 지금까지 오랜 시간에 걸쳐 실습을 한 과목이 처음이고, 계속 고민하고 찾아보면서 접근한 적은 처음이었다. 교안부터 컴퓨터보안 원서도 찾아보고 실습의 결과가 나온 부분에 대해서 올바르게 판단했는지 검증하기 위해서 정말 다양한 자료를 찾아본 것 같았다. 다만, 시간 분배를 잘 하지 못해 3.3절 토의부분에 해당하는 실험에 대해 꼼꼼하고 깊이 있게 접근하지 못해 아쉬웠다. 기회가 된다면 해당 부분에 대해서 시간투자를 많이 해서 패스워드의 자리 수에 따른 패스워드 강도에 대해서 심도 있게 실험해보고 싶다.

4. 참고자료

1 Password Cracking: https://en.wikipedia.org/wiki/Password_cracking

2 Computer Security: Principles and Practice,3rd Ed (W.Stalling & L.Brown, Pearson)

3 https://ko.wikipedia.org/wiki/칼리_리눅스

4 https://ko.wikipedia.org/wiki/존_더_리퍼

5 https://hashsuite.openwall.net (Home, Hash Suite main object)

6 https://www.vmware.com/products/fusion.html

7 https://www.kali.org/get-kali/#kali-virtual-machines

8 https://www.fosslinux.com/39230/what-is-uid-in-linux-and-how-to-find-and-change-it.html

9 https://www.cyberciti.biz/faq/understanding-etcshadow-file/

10 동국대학교 컴퓨터보안 강좌(김영부 교수) 교안 (03-OS Security)

11 https://www.law.go.kr/행정규칙/주요정보통신기반시설%20취약점%20분석·평가%20기준

12 https://www.kisa.or.kr/2060204/form?postSeq=12&lang_type=KO&page=1#fndoDocumentPreview