1. **Introduction**

해당 과제는 과목37의 중간고사 대체과제로서, zap (http://cd.textfiles.com/hackersencyc/HACKING/UNIX/ZAP.C) 이 가지고 있는 문제점을 해결하는 과제이다. 구체적으로, zap은 wtmp, utmp 파일 내부에 EOF 가 아닌 nullbyte가 발생하도록 하고, 이는 궁극적으로 rootkit detector인 ckhwtmp로부터 들키게 된다.

나아가 lastlog는 모두 제거(nullbyte 로 채움)해버리는데, 이 또한 rootkit detector 인 chklastlog로부터 들키게 된다. 이 보고서에서는 이를 해결하기 위해 어떤 방법을 택했는지, 또 실제로 적용해 봄으로서 그 방법의 타당성 또한 증명할 것이다. 모든 소스코드는 깃헙에서 확인할 수 있다. (https://github.com/cloneofsimo/zap)

1. **Method**

간략하게 요약하자면, 문제 자체는 파일 내부에 hole이 존재한다는 것이기에 제거 대상이 아닌 데이터를 모아 hole 이 없도록 압축하면 된다. 한편, wtmp, utmp는 조건에 해당하는 데이터를 삭제, 변형하면 되지만 (utmp 는 wtmp에 대응적으로 존재하기에 같은 방법을 써도 된다), 문제점이 있다면 이에 상응하게 lastlog 또한 바꿔줘야 한다는 것이다. 가령 다음과 같은 상황을 보자.

“A 유저가 12월 2일에 접속하였다. B 유저가 12월 3일에 접속하였다 A 유저가 12월 4일에 접속하였다”

이때 “A 유저가 12월 4일에 접속한 기록을 날려라” 라는 쿼리에 대해 wtmp와 utmp는 앞서 서술한 방법으로 삭제할 수 있어도 lastlog에서 12월 4일의 접속을 제거하면 “Never logged in”, 즉 한번도 접속한 기록이 없는 상태가 된다. 때문에 lastlog 에서는 제거가 아닌 업데이트를 해줘야 함을 알 수 있다.

하지만 업데이트 자체를 동적으로 하는 것은(즉 쿼리문에 해당하는 업데이트를 explicit 하게 구하여 업데이트 하는 것) 구현적으로 복잡하다. 때문에 wtmp에서의 정보와 일치시키는 함수 (resolve\_lastlog)를 구현하여 매번 wtmp 의 정보와 일치시키는 방향으로 프로그램을 구현하였다.

**2.1 Algorithm**

알고리즘은 다음과 같다.

1. **kill\_tmp**

void kill\_tmp(char \*fname, char \*who, char \*host, char \*mdy, int condition)

utmp, wtmp에서 사용자가 입력한 쿼리의 조건에 부합한 기록을 전부 제거하는 함수로서,

파일을 오픈하고, utmp 들을 동적할당하여 조건에 부합하지 않은 utmp들을 모으며, 마지막으로 기존 파일에 오버라이딩 한다. who, host, mdy는 각각 사용자 이름, 서버 호스트/ip 주소, mmddyy 형식의 날짜이다. kill\_tmp는 함수 check\_utmp\_condition를 사용해 utmp가 조건에 부합한지 확인한다.

1. **rep\_tmp**

void rep\_tmp(char \*fname, char \*who, char \*host, char \*mdy, char \*who\_att, char \*host\_att, char \*mdy\_att)

비슷하게 utmp, wtmp에서 사용자가 입력한 쿼리의 조건과 부합하게 기록을 조작하는 함수이다.

이번엔 파일의 nullbyte 가 생길 것을 굳이 걱정하지 않아도 되므로, 동적할당은 하지 않는다. who, host, mdy는 kill\_tmp와 같고, who\_att, host\_att, mdy\_att는 바뀌어질 조건들이다.

1. **resolve\_lastlog**

void resolve\_lastlog()

앞서 서술한 바와 같이, wtmp 의 데이터를 조회하면서 lastlog 파일을 업데이트 해주는 함수이다. lastlog 파일을 RDWR으로 열어 모든 uid 마다 열람하여, wtmp 에서 가장 최근에 접속한 기록을 가져와 이로 업데이트 해준다.

* 1. **Compile**

본 프로그램에서는 Ubuntu – Linux 환경에서 컴파일 하였으며, 이를 기준으로 작성하였다. 특이한 점이 있다면 strptime 이 둘 다 필요한데 이는 XOPEN\_SOURCE 옵션이 추가적으로 필요하다. 컴파일 옵션에 추가해도 되고 매크로로 선언할 수도 있다.

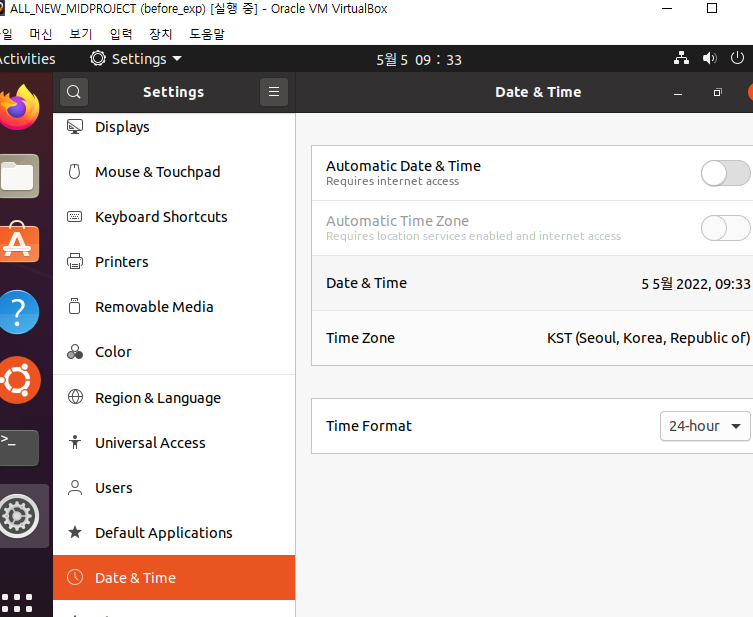
#define \_XOPEN\_SOURCE 700

1. **Experiment / Testing**

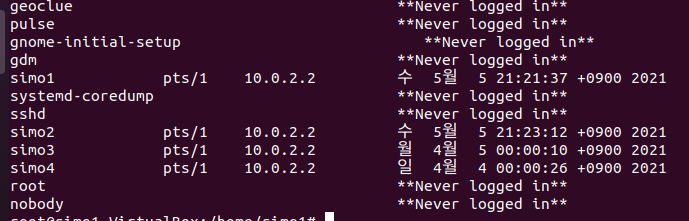
실제로 잘 작동하는지 확인하기 위해 실험 환경을 다음과 같이 구성했다.

|  |  |  |
| --- | --- | --- |
| 접속자 | 접속 terminal / IP address | 접속 시간 |
| simo3 | 10.0.2.2 | 2021년 3월 3일 |
| simo4 | 10.0.2.2 | 2021년 4월 4일 |
| simo3 | 10.0.2.2 | 2021년 4월 5일 |
| simo1 | 10.0.2.2 | 2021년 5월 5일 |
| simo2 | 10.0.2.2 | 2021년 5월 5일 |

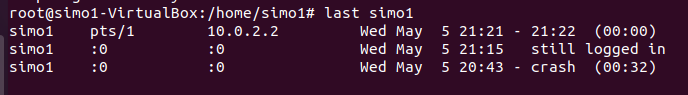
이러한 접속 기록을 만들기 위해, date --set “20 March 2022” 등과 같이, 날짜를 바꿔준다. 물론 이 과정에서 아래와 같이 인터넷 업데이트를 방지해야 한다.



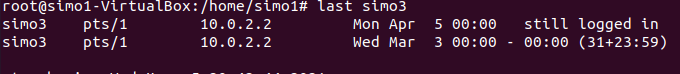
환경을 다 구성하고 나면 다음과 같은 모습이다. (lastlog)



last simoi 의 결과는 다음과 같다





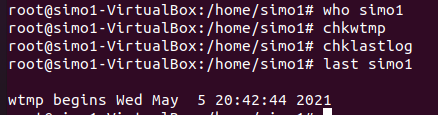


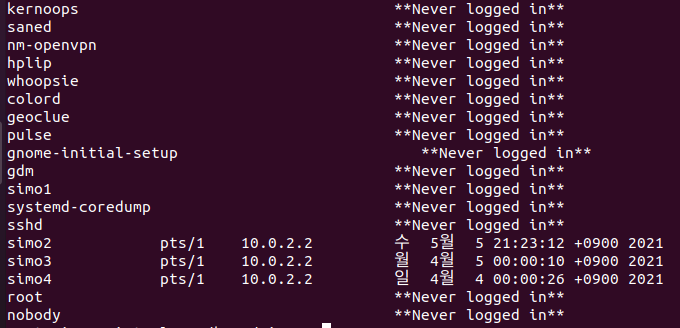


이제 다음과 같은 테스트를 할 것이다.

1. zap -A simo1 : simo1

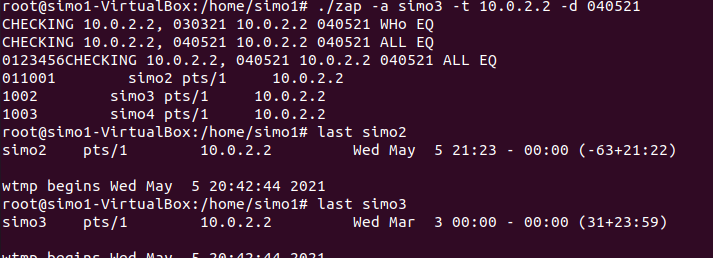
현재 접속하고 있는 계정이 simo1 이다. 위의 명령어를 하면 who simo1, chkwtmp, chklastlog, last simo1 전부 아무것도 뜨지 않아야 한다. 실험을 통해 이를 확인할 수 있다.



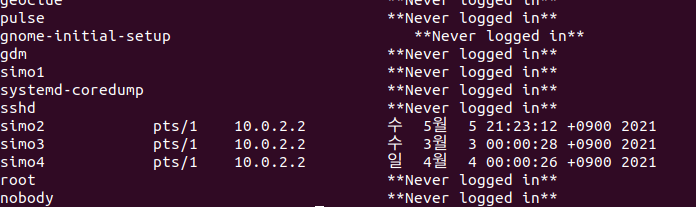
또한 lastllog 또한 다른 계정의 정보는 남아있으나, simo1 의 접속 기록만 사라졌음을 확인할 수 있다. 

1. zap -a simo3 -t 10.0.2.2 -d 040521

여기서 핵심적으로 보아야 할 사항은 두 사항이다. wtmp, utmp 가 올바르게 제거되었는지와 (즉, simo3 가 4월 5일 접속한 기록 만 제거되었는가) lastlog가 아예 사라진 것이 아닌, 가장 최근 접속이 3월 3일으로 업데이트 되는가이다. 이 또한 역시 확인할 수 있다. 또한, 이후에 접속한 사람의 기록이 사라진 것은 아닌 것을 확인할 수 있다 (다른 파일은 건드리지 않고 제거되었다는 것을 뜻한다)

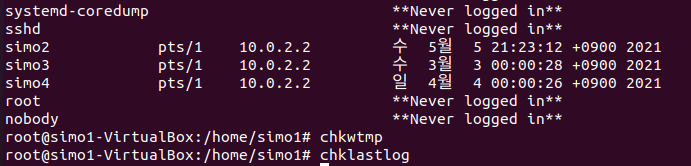


<last simo3 가 3월 3일의 접속 기록으로 업데이트 됨을 확인할 수 있다>



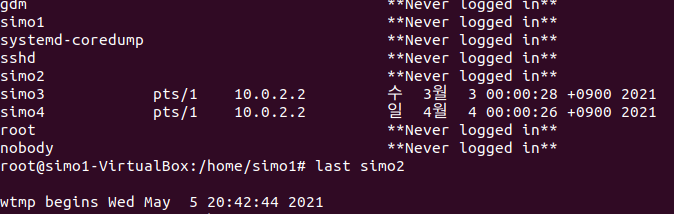
<lastlog 가 다른 데이터와 부합하게 업데이트 됨을 확인할 수 있다.>

마찬가지로 chkwtmp, chklastlog에서 들키지 않음 또한 확인된다.



1. zap -R simo2 simo4 -t 10.0.2.2 abcdefgh -d 050521 052019

마지막으로 replace 쿼리이다. 우린 확인 포인트가 있다. 첫 번째는 simo2 가 접속한 기록이 사실상 사라져야 한다 (simo2 는 10.0.2.2 로 5월 5일 접속한 기록이 전부이다). 또한 simo4는 작년의 접속 기록이 생긴 것이므로 lastlog에서 접속 기록이 바뀌지 않아야한다. 이를 확인할 수 있었다.



<simo4 또한 바뀌지 않은 모습을 확인할 수 있다.>

마지막으로 simo4 에서 추가적인 기록이 생긴 것과, 접속 호스트 명이 바뀐 것과, chklastlog 와 chkwtmp가 반응하지 않는 것 까지 확인할 수 있었다.

