6주차 3차시. 레이스 컨디션 공격에 대한 대응책

【학습목표】

1. 레이스 컨디션 공격의 다양한 사례들을 학습하고, 수행 및 대응책에 대해 설명할 수 있다.

학습내용1 : 레이스 컨디션 공격 사례

1. SunOS sendmail(/bin/mail)

* SunOS sendmail의 메일 저장 위치
/var/spool/mail/$USER

저장되는 메일은 각 사용자의 권한으로 저장됨

관리자 권한을 사용해서 각 사용자 권한으로 메일을 저장
* /var/spool/mail에 존재하지 않는 사용자에게 메일을 보내 레이스 컨디션 공격을 수행

2. Solaris 2.x ps(/usr/bin/ps)

* Solaris 2.x에 있는 ps는 /tmp 에 임시파일을 생성
임시파일 생성
임시파일에 내용 작성
임시파일 권한 변경
임시파일명을 /tmp/ps_data로 변경

* 레이스 컨디션 공격 수행

임시파일명을 알기 어려움
/tmp의 파일을 계속 검사하면서 파일마다 레이스 컨디션 공격을 수행
/tmp에 파일이 생길 때 마다 unlink() 하고, 원하는 파일로 링크를 걸어야 함

학습내용2 : 레이스 컨디션 공격 대응책

1. 공격 대응책

* SetUID를 이용하여 실행되는 프로그램 중 임시 파일을 만드는 실행 프로그램의 SetUID 를 제거
chmod -s 실행파일
* Root 관리자만 레이스 컨디션 조건을 갖춘 프로그램을 구동할 수 있도록 권한을 수정

chmod 4700 실행파일

* 가능한 임시파일 생성을 하지 않음
시스템의 처리량이 증가하면서 임시파일을 만들지 않고 메모리내에서 처리가 가능해짐
* 레이스 컨디션 공격을 위한 unlink() 를 못하도록 만듬
unlink()를 못하게 되면 심볼릭 링크 생성이 불가능해짐
> Solaris의 ps 버그는 /tmp가 0777 모드로 다른 사용자가 /tmp 파일을 변경 가능함
> /tmp를 1777로 변경하여 unlink() 수행을 불가능 하도록 막음
*create()와 open()을 구분
/bin/mail의 경우 공격 받을 경우 심볼릭 링크이거나 존재하지 않거나 두 가지 경우의 수가 생김
lstat()에서 파일이 존재하지 않는 경우 open() 할 때 옵션을 조정
open("file", O_CREATE | O_EXCL)과 같이 파일이 존재하는 경우에만 실행하도록 소스코드를 작성

* umask를 최하 022로 유지
파일 권한이 666이면 누구나 쓰기 가능
umask를 이용하여 다른 사용자의 쓰기 권한을 없애줌

학습내용3 : 레이스 컨디션 공격 대응을 위한 프로그램 로직

1. 심볼릭 링크 설정 여부와 권한에 대한 검사 과정 추가

*함수 내용 분석 (safeopen.c)
① if (lstat (filename, &st) != 0) : 심볼릭 링크의 유무에 대한 정보 반환

```
int safeopen(char *filename){
struct stat st, st2;
int fd;
        ① if (lstat (filename, &st) != 0)
                        return -1;
```

* 함수 내용 분석 (safeopen.c)
② if (!S_ISREG(st.st_mode))

```
                        return -1;
```

if (!S_ISREG(st.st_mode)) : 구조체 st에 대한 st_mode 값으로 파일의 종류에 대해 확인
> S_ISBLK - 블록 파일 테스트
> S_ISCHR - 문자 파일 테스트
> S_ISDIR - 디렉터리 테스트
> S_ISFIFO - FIFO 테스트
> S_ISREG - 일반적인 파일 테스트
> S_ISLNK - 기호 링크 테스트

③ if (st.st_uid != 0)

```
                return -1;
        fd = open (filename, O_RDWR, 0);
        if (fd <0 )
                return -1;
```

if (st.st_uid != 0)
> 생성된 파일의 소유자가 root가 아닌 경우 검사
> 공격자가 자신이 생성한 파일 삭제
> 접근하고자 하는 파일이 일반 계정 소유의 파일인지 확인

④ if (fstat (fd, &st2) != 0){

```
        close (fd);
        return -1;
                }
```

if (fstat (fd, &st2) != 0)
> 파일 포인터에 의해 열린 파일 정보 모아 st2 구조체에 전달
if (fstat (fd, &st2) != 0)
> 전달되는 데이터 -장치(device), I-노드, 링크 개수, 파일 소유자의 사용자 ID, 소유자의 그룹 ID, 바이트 단위 크기,

마지막 접근 시간, 마지막 수정된 시간, 마지막 바뀐 시간, 파일 시스템 입출력(I/O) 데이터 블록의 크기, 할당된 데이터 블록의 수 등
if (fstat (fd, &st2) != 0)
> 전달되는 데이터 -장치(device), I-노드, 링크 개수, 파일 소유자의 사용자 ID, 소유자의 그룹 ID, 바이트 단위 크기, 마지막 접근 시간, 마지막 수정된 시간, 마지막 바뀐 시간, 파일 시스템 입출력(I/O) 데이터 블록의 크기, 할당된 데이터 블록의 수 등
⑤ if (st.st_ino != st2.st_ino) || st.st_dev != st2.st_dev){
close (fd);
return -1;
}
return fd;
}
if (st.st_ino != st2.st_ino) || st.st_dev != st2.st_dev)
> 최초 파일에 대한 정보를 저장하고 있는 st와 파일을 연 후 st2에 저장된 I-노드 값, 장치(device) 값의 변경 여부 확인

【학습정리】

1. 임시 파일을 이용할 때 안전한 파일 열기 과정은 파일 링크 상태 확인, 파일의 종류 확인, 파일의 사용자 계정 검사, 최초 파일의 정보와 연 후의 파이르 정보 비교가 있다.