PATH HOWTO

Esa Turtiainen *mailto: etu@dna.fi* <mailto:etu@dna.fi> v0.4, 15 November 1997

번역: 고영훈 *gogoonee@bbs.para.co.kr* <mailto:gogoonee@bbs.para.co.kr>

제 1 절 Introduction

이 문서는 unix 와 linux의 환경 변수, 특히 path에 관한 일반적인 요령과 문제점들을 설명하고 있다. path 는 명령어를 찾기위한 디렉토리들의 리스 트로 데비안 리눅스 1.3 배포판을 기준으로 설명한다. 한가지 알아둘 것은 이글은 베타판이라는 것이다. 전할말이나 교정사항을 알려주면 좋겠다.

제 2 절 Copyright

제 3 절 General

모든 유닉스의 프로세스들은 "환경"을 담고 있다. 이것은 변수들의 리스트로 name 이나 또는 특정 값 들이다.(둘다 단지 string 으로서 대부분의 character가 들어갈 수 있다). 모든 유닉스 프로세스들은 패런트 프로세스 (부모 프로세스)를 가지고 있다 (부모 프로세스란 현재의 프로세스를 (이전에) 생성한 프로세스로 이때 현재의 프로세스는 차일드 프로세스라 한다(자식 프로세스)). 자식 프로세스는 부모 프로세스로부터 환경을 물려받게 되고 자식 프로세스는 또 다시 자기 자식프로세스를 만들면서 환경을 바꿀수 있다.

그 중 중요한 환경변수 가운데 하나가 path 이다. path는 명령어를 찾기위해 검색해야될 디렉토리들의 리스트로서 콜론(:)으로 구분되어진다. 만약 당신 이 'foo' 라는 명령어를 쳤다면 path 에 정의된 순서대로 디렉토리들이 검색 되어지며 실행파일인(x 비트가 on 상태인 파일) 'foo' 파일을 찾게되고 발견 이 되면 실행이 된다.

이 howto 문서에서 나는 path 메카니즘에 의해 짧은 이름만으로(풀 패스 네임이 아닌:역자주) 호출되어지는 실행파일을 가리켜 '명령어'(command) 라 부르겠다.

리눅스에서는 프로세스를 시작하기 위한 저수준의 운영체제 시스템 호출도 (exec 패밀리 콜들) path 에 정해진 디렉토리들을 검색하여 실행되어진다: 또한 당신이 어디에서 명령을 실행하건 path 메카니즘에 의해 실행되어진다. 만약, exec 시스템 콜이 '/'이 포함되지 않은 파일네임(즉, 풀패스 네임이 아 닌것:역자 주)을 실행할 것을 요구받으면 path 변수를 참조하게 되고 PATH 가 설정되어있지 않더라도 적어도 /bin 과 /usr/bin 디렉토리가 적당한 명령 을 위해 검색되어지게 될 것이다.

환경변수 세팅은 sh 에서는 export 명령어를 사용하고 csh 에서는 setenv 명령어를 사용한다. 예를 들면:

sh: PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:.

csh: setenv PATH usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:.

C-프로그램에서는 setenv() 라이브러리콜을 사용하여 환경을 바꿀 수 있 다. Peal 에서는%ENV 라는 조합 행렬을 가지고 있어서 다음과 같이 path를 설정할 수 있다.

$ENV{PATH}="/bin".

env 명령어는 현재의 환경변수값들을 알아보는데 기본적인 명령어이며 또한 환경 변수를 바꾸는 데에도 쓰일 수 있다.

기본적인 환경(변수) 메카니즘에 대한 정보들은 매뉴얼 페이지의 'environ', 'execl', 'setenv'를 보거나 info 파일 'env' 그리고 쉘에 관한 문서를 보기 바란다.

리눅스가 부팅될 때 제일 처음 시작되는 것은 init 프로세스이다. 이것은 특별한 프로세스로서 모든 프로세스들의 조상이 되며 자신은 부모프로세스를 가지지 않는다. 즉 init 의 환경은 명백하게 바뀌지 않는한 다른 모든 프로세 스의 환경으로서 남아있게 된다.(사실 대 부분의 프로세스들은 환경을 수정 한다)

init 은 여러 프로세스 그룹들을 실행시키는데, /etc/inittab 파일이 어떤 프 로세스를 시작시킬것인지 알려준다. 이 프로세스들은 init 으로부터 받은 환 경을 가지고 수행되고, 'getty'(콘솔에 'login:'이라고 나타내주는 프로세스) 같은 것들이 있다. 만약 여기서(시스템 스타트 시에) ppp접속을 시작한다면 이 때는 init의 환경변수 아래서 수행되고 있음을 알아야 한다. 시스템 초기 화는 대게 여기서 시작되는 스크립트들에 의해 이루어 진다. 데비안 1.3의 초기화는 /etc/init.d/rc 이며 이 스크립트는 또 다시 다른 초기화 스크립트 들을 호출한다.

시스템에는 돌아가고 있는 많은 서버(데몬이라고도 함)들이 있는데 이 것들 은 디폴트의 환경변수를 사용할 수도 그렇지 않을수도 있는데, 대부분의 서 버들은 초기화 스크립트에서 실행되어지며 그래서 init 의 환경을 가지고 있 다.

사용자가 시스템에 로그인 하게되면 환경변수들은 프로그램에 컴파일되어 들어간 환경변수 설정에 의해서, 또는 시스템 전반에 영향을 주는 초기화 스 키립트나 사용자 초기화 스크립트에 의해 변하게 된다. 이것은 꽤 복잡하고 여기서 설명하기에는 충분치 않다고 본다. 예를 들어 사용자가 콘솔로 로그 인을 하는 경우, 아니면 xdm 또는 network를 통해서 로그인을 하게되는 경 우가 완전히 다르게 된다.

제 4 절 Init

init 은 다른 모든 프로세스들의 부모 프로세스로서, 다른 프로세스들은 init 의 환경을 따르게 되고 다른 패스가 지정되지 않는한(드믄 경우이다) init의 path를 따르게 된다.

init의 path는 init 프로그램의 소스에 다음과 같이 정해져 들어가 있다.

/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin

init의 path에 /usr/local/bin이 들어가있지 않다는 것을 눈여겨 봐두기 바 란다. /etc/inittab 파일, 특별히 시스템 초기화 스크립트인 /etc/int.d로부터 실 행되는 모든 프로그램들 은 init의 환경에 따르게 된다. (debian 1.3)

시스템 초기화 스크립트에 의해 실행되는 모든 것들은 init 의 환경을 디폴 트환경으로 가지게 되는데 이러한것에는 syslogd, kerneld, pppd (시스템 부팅시 시작될 때) , gpm, 그리고 아주 중요한 lpd 와 inetd 가 그러하며 이 들은 디폴트 환경을 바꾸지 않는다.

부팅 스크립트에서 시작되는 몇 개의 프로그램중에는 부팅 스크립트에서 path 변수가 지정되기도 하는데 이러한 것에는 atd, sendmail, apache 그 리고 squid가 있다. 부팅스크립트에서 시작되는 프로그램중에는 path를 아예 바꿔버리는 것도 있는데 한 예가 cron 이다.

제 5 절 Login

텍스트 콘솔에는 getty 라는 프로그램이 사용자의 로그인을 기다린다. 이 프로그램은 'login:' 등의 메시지를 화면에 나타내며 init의 환경변수하에서 실행된다. 사용자가 로그인 하게되면 getty는 login 프로그램을 기동시키며 login은 사용자 환경을 세팅하고 shell을 기동한다.

login 프로그램은 /usr/include.path.h 에 정해진대로 path를 설정한다. 일 반적인 사용자를 위한 패스지정은 다음과 같다.(_PATH_DEFPATH)

/usr/local/bin:/usr/bin:/bin:.

root를 위해서는 다음이 지정되어있다. (_PATH_DEFPATH_ROOT)

/sbin:/bin:/usr/sbin:/usr/bin

일반사용자의 패스는 어떠한 sbin 디렉토리도 포함하고 있지않다. 그러나 현재디렉토리인 '.'을 포함하고 있는데 이 디렉토리를 path에 넣는 것은 root에게는 위험한 것으로 보는 경우가 많다. 심지어 root에게 /usr/local/bin 디렉토리도 path에는 없다.

로그인 패스는 자주 셀 초기화 과정에서 수정되어진다. 그러나 사용자 쉘로 서 /etc/passwd 에 보통 쓰이는 쉘말고 다른 프로그램도 사용될수 있다. 예 로 나는 특별히 만든 유저네임으로 로그인 하는 경우에는 ppp를 스타트 시 키기위해 다음과 같은 방법을 이용한다.(이 경우 pppd의 패스는 login 프 로그램의 패스와 같다)

etu-ppp:viYabVlxPwzDl:1000:1000:Esa Turtiainen,PPP:/:/usr/sbin/pppd

제 6 절 Shells

사용자 프로세스들은 /etc/passwd 에 정해진 쉘의 자식프로세스인 경우가 많으며 대게 쉘의 초기화 파일에 의해서 path는 수정되어지는 경우가 많다. login 프로그램에서는 쉘의 이름앞에 '-' 가 붙는다. 예를들어 bash는 '-bash'라고 불려진다. 이 표시는 그것이 로그인 쉘임을 불려지는 쉘에게 알린다. 이 경우에 쉘은 로그인 초기화 파일을 수행하게 된다. 로그인 쉘이 아닐경우에는 간단한 초기화가 이루어진다.

추가적으로, 쉘이 상호 대화식쉘인지 즉, 명령어가 파일로부터 입력이 되는 지 tty에서 입력이 되는지 체크를 하는데 이것은 쉘의 초기화를 수정하게 되 며 그래서 대화식 쉘이 아니고 로그인 쉘도 아니라면 초기화는 아주 간단하게 이루어진다.(bash는 이경우에는 아무런 초기화도 하지 않는다!)

6.1 bash

전형적인 로그인 쉘로서 bash는 시스템전반에 걸쳐 쓰이는 /etc/profile 파 일을 참조한다. 여기에는 시스템 환경과 path가 bash 사용자를 위해 설정되 어 질 수 있다.

그러나 시스템이 대화식 쉘이 아니라고 판단하게 되면 이 파일은 읽혀지지 않는다. 중요한 경우로 리모트에서 명령어가 실행되는 rsh 가 있는데 이 경우 /etc/profile은 읽혀지지 않고 path는 rsh 데몬에게서 상속되어진다.

bash는 명령 인수 -login과 -i 로써 로그인 쉘인지 아닌지 와 대화형 쉘인 지 아닌지를 정할 수 있다. 사용자는 홈디렉토리에 /.bash_profile, /.bash_login 또는 /.profile 파일을 만들어서 /etc/profile 에 정해진 값을 수정할 수 있다. 주의할 것은 csh의 초기화 과정과는 다르게 이 파일중 처음것만 실행된다는 것이다. /.bash_login 은 로그인 쉘일 경우 실행되지 않으며 /.bash_profile 이 있다면 /.bash_login은 전혀 실행안된다!!

만약 bash가 sh 란 이름으로 사용되어진다면 bash는 오리지날 본쉘의 초 기화를 에뮬레이트한다. 즉, /etc/profile, /.profile 과 로그인 쉘의 초기 화만 따른다.

6.2 tcsh

tcsh 가 로그인 쉘이면 tcsh 는 다음 순서로 파일들을 수행한다.

```
/etc/csh.cshrc
/etc/csh.login
~/.tcshrc
~/.cshrc (.tcshrc 가 없는경우)
~/.history
~/.login
~/.cshdirs
```

tcsh는 cshrc 스크립트 전에 login 스크립트를 먼저 수행하도록 컴파일할수 도 있다. 알아두어라!

대화형 쉘이 아닐경우에는 단지 *cshrc스크립트만 실행된다. *login 스크 립트는 path를 지정하기 위해 로그인 시에 한번 사용된다.

제 7 절 Changing user ID

7.1 su

su 명령어를 사용할때는 사용하려는 새로운 아이디를 적어주는데 아무것도 적어주지 않으면 root 가 사용된다. 대게의 경우 su 는 다른아이디로 서브쉘을 기동하는데, 아규먼트로 '-'(최 근에는 -l 또는 -login)를 붙일 경우 su 는 로그인쉘을 기동한다. 어쨌거나 su는 이런 기능을 수행하기위해 login프로그램을 쓰지는않는다. 대신 로그인 시뮬레이션(소스 코드에 이 용어를 쓰더군요)을 위한 다른 내장된(built in) path가 사용된다.

그 path는 일반사용자에게는 /usr/local/bin:/usr/bin:/bin:/usr/bin/X11:.

root 에게는

/sbin:/bin:/usr/sbin:/usr/bin:/usr/bin/X11:/usr/local/sbin:/usr/local/bin 이다.

뿐만 아니라 su는 꽤나 미세한 환경설정의 변화를 가져온다.

7.2 sudo

수퍼유저 명령어들을 보다 안전하게 쓸수있도록 하는 명령어들이 있는데 이것들은 보다 나은 로깅과 유저별 제한과 개별적인 패스워드 사용을 가능 하게 해준다. 가장 널리쓰이는 것은 sudo 이다.

$ sudo env

이 명령은 env 명령을 수퍼유저 모드로 실행하게 한다. (만약 그것을 허용 하게끔 설정이 되어있다면) sudo 명령어는 path를 다루는데 있어서 또다른 접근방식을 갖고 있는데, path를 바꿔서 항상 현재 디렉토리가 path의 마지막에 있게 한다. 그러나 PATH 변수를 바꾸지는 않으며 단지 SUDO_USER 같은 몇 개의 환경변수 들만을 바꾼다.

제 8 절 Network servers

대부분의 네트워크 서버들은 어떤종류의 서브 프로세스도 가동하지 않는게 좋다. 보안을 생각한다면 그 프로세스들의 path 는 최소한 이여야 한다. 한가지 중요한 예외가 있다면 네트워크로부터 로깅을 허용하는 서비스들이 다. 이 장에서는 이 경우에 어떤 환경이 쓰이는지 설명하고 있다. 만약 rsh를 통해 어떤 명령어

가 리모트 머신에서 수행되면 path 는 ssh를 사용하는것 과 다르다. 비슷하게 rlogin으로 telnet 또는 ssh로 로그인 했을때도 각각 다르 다.

8.1 inetd

대부분의 네트워크 서버들은 리퀘스트를 기다리며 대기하고 있는 서브 프 로세스들을 가지고 있지 않다. 이 일은 inetd라는 인터넷 수퍼 서버가 대신 하게 되는데, inetd는 설정된 모든 네트워크 포트들을 듣고 있다가 리퀘스트 가 오면 해당하는 서버를 기동한다.

inetd 의 설정파일은 /etc/inetd.conf 이다. inetd는 시스템 시작 스크립트에서 시작되어 진다. init 프로세스로부터 path를 상속받고 수정하지는 않는다. 그래서 inetd 로부터 시작되는 모든 서버들은 init의 path를 가지고 있다. 예로서 IMAP post office 프로토콜인 imapd을 들수있다 다른 inetd프로세스는 telnet, rlogin, talkd, ftp, popd, 여러 http서버 등이 있다.

리얼 서버를 구동하기위해 따로 분리된(독립적인?:역자 주) tcpd 프로그램 을 사용한다면 inetd 의 사용법은 여전히 복잡하다. tcpd는 리얼 어플리케 이션이 시작되기 전에 추가적인 보안 점검을 하게되는데 path를 바꾸지는 않는다.(확인해보지 않았음)

8.2 rsh

rsh 데몬은 일반사용자를 위해 로그인 프로그램이 설정하는것과 같은 _PATH_DEFPATH (/usr/include/paths.h) 로부터 path를 설정한다. root 역시 일반유저와 같은 path를 갖는다. 사실, rshd 는 commandline에서 얻은 파라메터(예제에서 command-line:역자 주)로 다음과 같은 명령어를 실행하는 것이다.

shell -c command-line

이경우 로그인 쉘은아니다. /etc/passwd 에 기록된 모든 쉘들이 -c 옵션을 지원하는지 확인하기 바란다.

8.3 rlogin

Rlogin 은 real login 과정을 수행하기위해 login을 기동한다. rlogin을 통해 로그인 한다면 login 에서와 같은 path를 갖게 될 것이다. 다른방식의 리눅스 로그인은 대부분 login을 사용하지 않는다. rsh 와 다르다는 것에 주의하기 바란다.

실제 login 다음과 같은 명령을 사용하는데

login -p -h host-name user-name

-p 옵션은 HOME, PATH, SHELL, TERM, MAIL, LOGNAME을 제외한 환경변수들을 전에 있던 그대로 보존하고, -h 옵션은 로그인할 리모트 호스 트 명을 알려주게 된다.

8.4 telnet

telnet은 rlogin 과 비슷하다. login 프로그램을 사용하며 명령실행도 rlogin과 비슷한 방식으로 이루어진다.

8.5 ssh

ssh 는 자신만의 path 설정치를 갖고 있다. ssh는 고정돤 path 변수를 가 지며 ssh 가 놓여져있는 path를 추가한 것이다. 때로 이렇게 됨으로써 /usr/bin 디렉토리가 패스에 두 번 나타나게 되기도 한다.

/usr/local/bin:/usr/bin:/bin:.:/usr/bin

path에 /usr/X11/bin 이 들어있지 않으며, ssh가 기동한 쉘은 로그인 쉘이 아니다. 그러므로

ssh remotehost xterm

이런 명령은 실행되지 않고 /etc/profile이나 /etc/csh.cshrc값을 바꿔준다 해도 소용없다. 사용하려면 풀 패스를 적어줘야한다 (/usr/bin/X11/xterm) ssh는 /etc/environment 파일안의 VAR=VALUE, 이런 형식의 환경변수 값을 읽어들인다. 그래서 XFree86 실행시 문제를 일으킨다. (뒤에 추가적으 로 설명됨:역자 주)

제 9 절 XFree86

9.1 XDM

XDM은 그래픽 환경의 터미널로 로그인 시에 많이 사용되는 방법이다. login 과 비슷하게 보일지 모르나 내부적으로는 완전히 다른것이다. 전과는 다른 로그인 과정을 실행하기 위한 설정파일들이 /etc/X11/xdm 디 렉토리에 있다. Xstartup (screen 0 는 Xstartup_0) 은 유저가 로그인한후 실행되는 명령들이 있다.(명령은 root 권한으로 실행되어진다) 유저들을 위한 path 는 /etc/X11/xdm/xdm-config 파일에 있다. 다음과 같은 라인들인데

DisplayManager*userPath: /usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games

DisplayManager*systemPath: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11

당연히 일반 사용자와 root를 위한 기본 path 가 될것이며 /usr/bin/X11 디 렉토리가 path에 걸려있다는 것에 주목하기 바란다. 만약에 X 유저가 다른 머신에 들어가 X 클라이언트 프로그램을 실행하기 위해서는 (X terminal로 부터 직접 로그인 하지 않았더라도..) /usr/bin/X11 이 path에 걸려있어야 한다.

Xstartup을 실행한후 XDM은 최종사용자로서 /etc/X11/Xsession을 실행 한다. /etc/environment 파일이 존재한다면 Xsession은 이 파일을 불러들 이고 로칼 설정은 이 파일에서 끝나게 된다.(Xsesion은 /bin/sh 에 의해 실 행되므로 /etc/environment 는 sh파일이어야 한다) 이 때문에 ssh를 사용할 경우 문제가 발생한다. ssh는 /etc/environment가 VAR=VALUE, 이런 형 식의 라인들이 들어가 있는 파일일 것이라 생각하기 때문이다.

9.2 xterm -ls

디폴트로 X window메니져 메뉴로부터 실행된 명령들을 위한 path는 XDM 으로부터 상속받은 것이다. 다른 path를 원한다면 직접 세팅해줘야 한다. 몇몇 기본적인 패스를 가지고 터미널 에뮬레이터를 시작하려면 특별한 옵션 들이 사용되어져야 한다. xterm에서는 -ls(login shell) 옵션을 사용하여 shell 로그인 초기화 파일의 path를 지닌 로그인 쉘을 기동할 수 있다.

9.3 Window manager menus and buttons

윈도우 메니져는 XDM으로부터 환경변수를 물려받는다. 윈도우 메니져로 부터 실행된 모든 프로그램들은 윈도우 메니져의 환경을 물려받는다. 사용자 쉘의 환경변수는 윈도우 메니져의 버튼이나 메뉴로부터 실행되는 프로그램에는 영향을 미치지 않는다. 예를들어, 'xterm -ls'에서 프로그램이 스타트 된다면 로그인 쉘의 기본 환경을 갖게 되나, 메뉴에서 실행된다면, 단지 윈도우 메니져의 환경만을 가질뿐이다.

제 10 절 Delayed commands cron and at

10.1 cron

Cron 은 주기적으로 /etc/crontab과 유저가 정의한 crontabs안의 명령을 실행하는 명령어이다. 데비안 1.3 에는 /etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly 파일들로써 표준적인 기능을 수행하게 된다. Cron 은 부트 스트립트에서 시작되나 자신의 path를 조금은 생소한 것으로 바꾼다. 즉,

/usr/bin:/binn:/sbin:/bin:/usr/sbin:/usr/bin 이건 cron 의 버그인 것 같다!!! init 의 path에다 앞부분에 /usr/bin:/bin을 덮어 써버린 것이다! 이 버그는 다른 모든 시스템에 존재하는 것은 아니다. crontab 에는 path 정의를 할 수 있는데, 데비안 1.3에는 /etc/crontab 의 앞부분에 다음의 기본 설정 라인이 있다.

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin 이 것으로인해, crond 프로그램의 PATH는 유저 프로그램에서는 적용되지 않으며, /etc.cron* 디렉토리의 모든 스트립트들은 이 패스를 기본으로 사 용한다. 심지어 프로그램의 실행자가 root가 아니라 일반사용자일 경우에도 crond 프로그램은 이 패스를 사용한다.

10.2 at

at 는 특정한 시간에 프로그램을 한번 실행하기 위해 사용하는 명령어이다.

atd 는 init 의 패스를 사용하나, 유저들의 프로그램은 sh 명령어로 항상 유 저의 환경하에서 수행되어진다. 그러므로 shell 에 의해 환경변경이 적용되 어진다. bash 에 관한 글을 읽어보라.

제 11 절 Some examples

11.1 magicfilter

magicfilter 는 프린트할 파일을 다루기 위해 주로 사용되는 툴이다. 프린트 될 파일을 분석하여 적절한 필터 스크립트를 기동한다. 이러한 스크립트들은 init이 /etc/init.d/lpd에서 시작시킨 lpd 가 수행하게 되며 그래서 init의 패 스를 가지고 있다. 그러므로 /usr/bin/X11은 path 에 들어가 있지 않다!! 메직필터에 PDF 파일을 넣어서 출력하고 싶을 것이다. 이것은 /usr/bin/xpdf를 사용하면 가능하다. 이때 당신은 풀패스 파일네임을 적어 줘야한다. 그렇지 않으면 메직필터는 그 파일을 찾을수없기 때문이다. (그러 나) 메직필터가 사용하는 대부분의 프로그램들은 /bin또는 /usr/bin에 있기 때문에 앞의 경우 외에는 풀패스를 적을필요는 없다.

11.2 Printing from X applications

지금 사용하는 프린터가 뭔지 나타내기 위해서 PRINTER 환경변수를 사용할 수 있다. 그러나 어떤 경우에는 작동하지 않을수도 있다는걸 알아두기 바란다.

만약에 XDM에서 X session이 시작되었다면 윈도우 메니져는 shell 로그인 스크립트를 실행하지 않는다는 것을 기억해야된다. xterm에서 실행된 모든 X application들은 PRINTER값을 가지고 있지만 메뉴나 윈도우 메니져 버튼 에서 실행된다면 PRINTER값이 없을것이다. 어떤 경우에는 하위계층으로 상속되는 경우도 있다. 예를들자면 모질라 help 프로그램은(플러그 인 프로그램:역자 주- 이 후 프러그인 이라 부른다) PRINTER 변수값을 가질 수도 있고 그렇지 않을 수도 있다.

제 12 절 Security concerns

패스는 때로 중대한 보안문제를 야기하기도 한다. 잘못된 패스설정을 악용 하여 시스템을 해킹하는 것은 흔한경우이다. 과거(?) 패스설정에서 흔한 실수는 root의 패스에 '.'이 들어가 있는경우이다. 악의있는 해커는 자신의 홈디렉토리에 'ls' 프로그램을 두어 root가 다음처럼

cd ~hacker # ls

해커 자신의 디렉토리에서 ls를 실행하게 되면 해커가 만든 프로그램이 실행 되도록 한다. 이러한원리는 root 권한으로 실행되는 모든 프로그램에 간접적으로 적용이 된다. 모든 데몬 프로세스들은 일반 유저가 쓰기가능한 것이면 어떠한 것도 실행해서는 안된다. 몇몇 시스템에서는 /usr/local/bin 디렉토리에 프로그 램들이 있고 그리 엄격하지 않은 보안 상태에 있다.(단지 이 디렉토리가 root의 패스에 없을뿐..) 어쨌거나,

만약에 어떤 데몬이 'foo' 라는 명령을 실행하는데 /usr/local/bin 디렉토리가 패스에 걸려있다면 '/bin/foo' 대신에 '/usr/local/bin' 이 실행될수도있을 것이다. 그렇다면 /usr/local/bin 에 쓰기 권한이 있는 사용자는 누구나 시스템에 침입해 들어갈 수 있게 된다. 패스에 설정된 디렉토리들의 '순서'가 어떤지 주의깊게 살펴보는것도 상당히 중요하다. 만약 /usr/local/bin이 /bin 보다 앞에 패스가 걸려있다면 ,음.. 그건 보안문제가 있는 것이다. 반대로 /bin 이 앞에있고 /usr/local/bin 이 뒤에있다면 로컬라이즈된 /usr/local/bin에 있는 프로그램이 /bin 에 있는 것을 대신하여 수행될 가능성은 없다.

리눅스에서 패스를 검토, 적용하는 것은 운영체제 시스템 콜 레벨에서 이루 어진다는 것을 기억해야한다. 어느곳에서나 명령어를 입력하면 설정된 디렉 토리가 검색되어지고 적어도 /bin 과 /usr/bin 이 검색되어진다.

제 13 절 How to debug problems?

환경변수를 읽어드리는 기본명령어는 /usr/bin/env 이다. /proc 디렉토리를 이용하면 어떤 프로그램의 패스를 알아낼수 있다. 먼저 프로세스 번호를 알아야 한다. (ps 명령어를 사용하면된다) 예를들면, xterm의 프로세스 번호가 1088 이면 다음 명령어로 프로그램의 환경변수를 알 수 있다.

more /proc/1088/environ

이 방법은 xdm과 같은 데몬 프로그램에게는 적용되지 않는다. 시스템 프로 세스의 환경변수, 또는 다른 사용자 프로세스의 환경변수를 알아내기 위해 서는 root 권한이 필요하다. 모질라를 디버그 해보자. /tmp/test 스크립트를 만든다:

```
$ cat > /tmp/test
  #!/bin/sh
  /usr/bin/env > /tmp/env
  ^d
  $ chmod +x /tmp/test
```

그리고 나서 아무 프러그인 프로그램을 설정한다. 예를들어 리얼오디오, audio/x-pn-realaudio를 '/tmp/test' 프로그램을 호출하도록한다. 그러고 나서 브라우저로 리얼오디오 링크가 있는데로가면 (예를들어 *www.realaudio.com/showcase* `<http://www.realaudio.com/showcase>`), 모질라는 /tmp/env 에 환경변수들을 저장하는 더미 프로그램(방금 만든 스크립트:역자주)을 호출할것이다.

제 14 절 Some strategies to get the same path for all the users

가장 중요한 설정들은 포괄적인 로그인 쉘의 쉘 초기화에서 할수있다. tcsh 인 경우는 /etc/csh.login bash인 경우는 /etc/profile. rsh , ssh 명령, 로그인 쉘을 명시적으로 시작하지 않는 X window메니져의 메뉴아이템들, inittab로부터 기동된 명령들, cron job들이나 lprd로 부터 시작된 메직 필터같은 데몬 job들,그리고 WWW CGI 스크립트 등등은 이러 한 패스가 적용되지 않는 예외인 경우다.

만약에 패스가 /etc/csh.cshrc 파일에 설정되있다면 rsh 나 ssh 가 리모트 머신에서 명령어를 수행했드라도 tcsh나 csh를 사용하는 계정에서라면 패 스는 유효하게 된다. 그러나 bash나 sh를 사용한다면 패스는 유효하지 않다. 패스 설정을 예를들어 /etc/environment-common 같은 하나의 파일에 넣 어서 사용할 수 있는데, 이 파일에 다음과 같이 적어두고

${EXPORT}PATH ${EQ}/bin:/usr/bin:/sbin:/usr/sbin:/usr/bin/X11:/usr/local/bin:/usr/games:.

이 것을 /etc/csh.login(tcsh 나 csh 인 경우) set EQ=" " set EXPORT="setenv " source /etc/environment-common

또는 /etc/profile (bash 인 경우, 원래의 sh는 안됨) EQ='=' EXPORT="export " . /etc/environment-common

또한 /etc/environment (XDM 인 경우) EQ="=" EXPORT="export " . /etc/environment-common 파일에 이와 같이 사용하면 적용시킬 수 있다.

이 방법은 ssh 인 경우에는 /etc/environment (그리고 EQ 와 EXPORT 변 수에 대하여)안의 라인들에 대해서 불평을할것이나 그 외,대게의 경우에는 잘 적용이 된다. (여전히, bash를 사용하는 rsh 명령에는 적용이 되지 않는다)

제 15 절 Acknowledgements

Ari Mujunen의 좌절이 이글은 쓰게 된 동기가 되었고, Juha Takala은 여러 조언을 해주었다.