

Project Proposal for Comp527

Resource Overbooking

Anupam Chanda(*anupamc@rice.edu*),
Amit Kumar Saha (*amsaha@rice.edu*)

October 21, 2002

Problem description

It is trivial to write malicious code that hogs up a particular resource, for example, memory, disk, etc. As an effect of this the other applications running on the same machine suffer and their performance degrades. As a pathological test we wrote a simple C program which just *mallocs* 4K pages (in an infinite loop) and never frees them. The code is attached in the appendix. Following were our observations on running the code on a FreeBSD 4.3 machine:

1. The system started swapping and hence hit the disk as the program *mallocd* more and more memory.
2. We were running KDE and all windows as well as the mouse took forever to respond.
3. Eventually the system ran out of swap space and all other programs were killed. This included the KDE server itself.

Proposed Solution

We propose to address this problem in the following way:

1. To exactly identify the reason why random processes are being killed by the operating system.
2. Try to identify a pattern in memory access which leads to this problem.
3. Once we identify that a process is behaving maliciously with respect to memory access, we need to decide what evasive action the kernel should take. These evasive actions might depend on the pattern that the kernel is able to identify for a particular process.

4. Such a malicious process, if identified, should not be allowed to hamper the performance of other unrelated processes. Even if the process is not identified early enough and the system does degrade in performance, the other unrelated processes should not be killed.

Roadmap

Date	Target
10/21/02	Submit project proposal
10/28/02	Identify why random processes are being killed by the kernel
11/04/02	Design solution
11/06/02	Status report
11/11/02	Implementation and debugging
11/18/02	Implementation and debugging
11/25/02	Final Results
11/27/02	In class presentation
12/06/02	Final report

Appendix

```
#include <stdio.h>

int main()
{
    int i = 0 ;
    while(1)
    {
        char *p;
        i++;
        if(i%1024 == 0)
            printf("Memory (MB) : %d\n",4*(i/(1024)));
        p = (char*)malloc(sizeof(char)*4096);
        memset(p,0,4096);
    }
}
```