SPF BASED SELINUX OPERATING SYSTEM FOR MULTIMEDIA APPLICATIONS

1NITISH PATHAK, 2NEELAM SHARMA
1BVICAM, GGSIPU, Delhi, India
2Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India
E-mail 1 nitish_pathak2004@yahoo.com, 2sharmaneelam2@gmail.com

ABSTRACT

Trusted Operating Systems, offer a number of security mechanisms that can help protect information, make a system difficult to break into, and confine attacks far better than traditional operating systems. However, this security will come at a cost, since it can degrade the performance of an operating system. This performance loss is one of the reasons why Trusted Operating Systems have not become popular. While Trusted Operating Systems offer an incredible amount of security, observations about computing workloads suggest that only some parts of the operating system security are actually necessary. Web servers are the best example. For many web servers, the majority of the information on the server is publicly readable and available on the Internet. Therefore, if a Trusted Operating System is used on a web server, any security used to secure the confidentiality of the server’s information is not necessary. Any security used to protect the confidentiality of web server data can be considered a waste of computational resources. The security needed in web servers is the security to protect the integrity of data, not the confidentiality of data. Other workloads such as multimedia or database workloads may also only need parts of the operating system security.


1. INTRODUCTION

This research paper proposes the Security Performance Flexibility (SPF) Framework for Trusted Operating Systems. SPF recognizes that not all computing workloads require all the security in Trusted Operating Systems. SPF allows system administrators to selectively disable parts of the security in Trusted Operating Systems. By disabling parts of the Trusted Operating System security, performance of the system can potentially be increased. The SPF framework allows system administrators to balance the security and performance needs in their particular computing environment.

2. LITERATURE SURVEY

In particular, security developed and placed in the application layer of a system is the most inadequate to provide high security. Cryptography and authentication techniques are probably the most popular form of security implemented and researched today. While it is extremely difficult to break, it has been and always will be subject to brute force or man in the middle attacks [1]. Firewalls and application layer access control can also help towards preventing malicious attacks. Most of today’s computer security is placed in the application layer of an operating system and at network entry points. However, there is little to no security placed inside an operating system kernel. All of the security problems that exist at the application layer suggest that operating system security mechanisms might protect a system far better [5]. Vendors such as Argus-Systems Group, Hewlett-Packard, and Sun Microsystems are developing highly secure operating systems with these security technologies [13]. Several open source operating systems are also being developed with high security in the kernel. The NSA has released a secure version of Linux called Security-Enhanced Linux (SELinux) [3] [22]. Linux distributions such as Rule Set Based Access Control Linux (RSBAC) and Linux Intrusion Detection System (LIDS) are also available.
3. TRUSTED EVALUATION CRITERIA

These highly secure operating systems are typically called Trusted Operating Systems (TOS), Trusted Systems, or Secure Systems. However, there seems to be four key features in all Trusted Operating Systems. They are Discretionary Access Control (DAC), Mandatory Access Control (MAC), Least Privilege, and auditing.

In very general terms, class B systems are Trusted Operating Systems. Argus System’s Pitbull, Hewlett-Packard’s Virtual Vault, and Sun Microsystems Trusted Solaris are all class B systems. Due to the strict criteria of the TCSEC, the NSA’s Security-Enhanced Linux is not considered a class B system. This is despite the fact that it offers most of the requirements to be considered a class B system [6].

4. PROBLEM DESCRIPTION

The additional security checks in the kernel will cause Trusted Operating Systems to be slower than traditional operating systems. This is one reason why Trusted Operating Systems have not become popular in the commercial sector.

Trusted Operating System security may also affect multimedia and video streaming services. Typically, when video is played, each frame is read off of disk one by one and displayed on a screen. In the case of video streaming, the frame is sent out through a socket onto the network. Every single read from disk and every write out to a network socket is now slowed down by repeated security checks. This may have a significant effect on the quality and frame rate of video, especially if the system is heavily loaded. Much like the example with web servers, this video quality degradation may not be entirely necessary. For example, a Video On Demand server may care more about the quality of the video stream than the security of read accesses to the server [7].

5. PROBLEM SOLUTION

As was stated in the previous section, there are several types of system workloads that repeatedly do security checks in a Trusted Operating System. For these workloads, the security checks may be undesired or completely unnecessary. The information in these workloads could be public, integrity of the system may be the primary concern, or quality of the workload is more desirable than the security of certain operations.

This suggests that performance in Trusted Operating Systems can be increased if security can be disabled in some parts of the operating system. In order to provide better performance for specific system workloads, this paper proposes the security performance flexibility (SPF) framework for Trusted Operating Systems [22][19]. The SPF framework allows system administrators to disable certain security checks in a Trusted Operating System. This gives system administrators the ability to balance their security needs with their performance needs.

![Figure 1: Security Performance Flexibility Framework Architecture.](image)

The architectural idea behind the SPF framework is illustrated in Figure 1. The SPF framework gives system administrators the option of disabling security checks for specific system calls. By skipping security checks in the kernel, performance for a system can be increased. For example, a system administrator can use SPF to turn off all read security checks in a web server. By turning off the read security checks of a web server, it is possible the web server’s throughput can be increased [22].

There are different levels at which the SPF framework can be implemented in a Trusted Operating System. Three different levels have been identified for evaluation in this paper:

1. System Wide Security Performance Flexibility (System-SPF).
6. SYSTEM WIDE SECURITY PERFORMANCE FLEXIBILITY (SYSTEM-SPF).

System-SPF provides the ability to disable all trusted security checks in particular operations of the system. For example, all read security checks can be disabled in the system. By disabling the read security checks in the entire system, performance of the system as a whole can improve. This may be particularly useful for dedicated web servers, because web servers generally have pure public information. Disabling read security checks for the entire web server may help increase web server throughput [9].

7. PROCESSES-SPF ARCHITECTURE

Process-SPF provides the ability to disable security checks in specific applications or processes. For example, a system administrator may disable read security for a MPEG video player. Therefore, every time a video frame is read from disk, the security check of this read operation will be skipped. By skipping the read security checks, we may be able to improve the quality of the MPEG video being played. In this example, Process-SPF only disables the read security on the MPEG video player. Different security checks can be disabled in other applications or processes [11] [14].

8. OBJECTS-SPF ARCHITECTURE

Object-SPF provides system administrators with a different fine-grained technique to manage their system. Object-SPF provides system administrators the ability to disable security checks on individual objects in the system [14]. Objects can refer to files, network connections, intercrosses communication (IPC) objects, etc.

Let us consider the MPEG video player example from before. In order to improve MPEG video quality, we want to disable read security checks when a video is being played. However, some systems may have both publicly readable video and private videos that are not public. The system administrator may want to disable read security checks only on the public videos and not the private videos [18].

9. MPEG FRAME RATE

The MPEG frame rate benchmark is executed with a workload running in the background of the Berkeley MPEG player. The Berkeley MPEG player attempts to play a small MPEG video at 30 frames per second and 60 frames per seconds. The small video clip runs for 30 seconds long when run at 30 frames per second. The average frame size of this video clip is 2882 bytes.

When the benchmark attempted to play the MPEG video at 30 FPS, the average frame rate is approximately 0.8 less frames per second. When the Berkeley MPEG player tried to play at 60 FPS, the average frame rate decreased by over 2 frames per second. At workload sizes of 15, SPF is able to gain back approximately half the frames lost due to security in SELinux. At 30 FPS, SPF is able to regain approximately 0.4 frames per second and at 60 FPS SPF is able to regain a little over 1 frame per second[12].

These decreases in frame rate are quite small, so they may not seem like much. However, these small differences in frame rate may make a difference in a QoS aware operating system. In a QoS system, these differences in frame rate may make the difference in a MPEG video being schedulable or un-schedulable. The frame rate increase SPF brings will give the video a better chance of being schedulable.

<table>
<thead>
<tr>
<th>Workload</th>
<th>Redhat (frames/sec)</th>
<th>SELinux</th>
<th>SELinux System-SPF (frames/sec)</th>
<th>SELinux Process-SPF (frames/sec)</th>
<th>SELinux File-SPF (frames/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31.5714221</td>
<td>31.854801</td>
<td>31.9362</td>
<td>31.87568</td>
<td>31.92478</td>
</tr>
<tr>
<td>5</td>
<td>32.6630158</td>
<td>32.8379267</td>
<td>32.8307</td>
<td>32.81267</td>
<td>32.8407</td>
</tr>
<tr>
<td>10</td>
<td>32.8675641</td>
<td>32.80736</td>
<td>32.79403</td>
<td>32.77317</td>
<td>32.7903</td>
</tr>
<tr>
<td>20</td>
<td>4.7472983</td>
<td>4.9487652</td>
<td>4.60871</td>
<td>5.002774</td>
<td>4.806898</td>
</tr>
</tbody>
</table>

Table 1: MPEG Frame Rate, 30 Frames/Second, read () security disabled.

<table>
<thead>
<tr>
<th>Workload</th>
<th>SELinux No SPF Frame rate Decrease</th>
<th>SELinux System-SPF Improvement Over SELinux No SPF</th>
<th>SELinux Process-SPF Improvement Over SELinux No SPF</th>
<th>SELinux File-SPF Improvement Over SELinux No SPF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.2%</td>
<td>-0.05%</td>
<td>-0.25%</td>
<td>-0.09%</td>
</tr>
<tr>
<td>5</td>
<td>0.54%</td>
<td>-0.02%</td>
<td>-0.08%</td>
<td>+0.01%</td>
</tr>
<tr>
<td>10</td>
<td>-0.15%</td>
<td>-0.04%</td>
<td>-0.11%</td>
<td>-0.05%</td>
</tr>
<tr>
<td>15</td>
<td>-7.88%</td>
<td>+3.26%</td>
<td>+4.74%</td>
<td>+3.47%</td>
</tr>
<tr>
<td>20</td>
<td>4.24%</td>
<td>-7.32%</td>
<td>+1.14%</td>
<td>-3.00%</td>
</tr>
</tbody>
</table>

Table 2: MPEG Frame Rate Improvement, 30 Frames/Second, read () security disabled.
Surprisingly, for both 30 FPS and 60 FPS, SELinux has a better frame rate than Redhat when the workload is 20. The highest workload exhibited higher performance in SELinux without SPF than Redhat or any of the SPF implemented kernels.

Overall, it seems that SPF can help improve MPEG frame rate, but not too much. SPF does not help increase frame rate for a very wide range of workload conditions. It seems MPEG frame rate can only be improved under specific workload conditions. In most situations, the specific system load will not be known. Therefore, SPF may not be too useful for providing increased frame rate for MPEG video.

The results according to Table 1, 2, 3, 4 indicate that there are no performance advantages between System-SPF, File-SPF, and Process-SPF. The numbers indicate that the overhead of each Implementation is approximately equal. As was expected, the overhead of SPF is far less than the overhead incurred by SELinux security checks [14][19].

It is quite interesting that SPF may not be very useful for multimedia application. Multimedia is one of the initial workloads that SPF targeted for improvement. The numbers suggest that perhaps multimedia workloads are more dependent on scheduling algorithms and quality of service (QoS) for their performance, rather than the amount of time it takes to read a frame from disk [13].

9. CONCLUSION

- The novel idea that skipping non-essential security checks in a Trusted Operating System may increase system performance.
- The novel SPF framework for Trusted Operating Systems that gives system administrators the ability to balance security and performance needs of a system.
- The novel recognition that the SPF framework can be implemented at different levels within a Trusted Operating System.
- The novel development of system administrative commands that allow system administrators to configure a Trusted Operating System dynamically as the system executes.

10. FUTURE SCOPE

Future work can be done to analyze scheduling issues for highly secure operating systems and how scheduling algorithms might be modified. QoS is put into an operating system by creating a separate file system that stores QoS information for each application in a system. This separate file system is similar to how SELinux creates a separate table to store security label information. There are indications that the architectural ideas from QoS and Trusted Operating Systems can perhaps be integrated into one system together. This leads to the possibility that several of these features can be combined and implemented into a single kernel.

REFERENCES

[6]. Argus Systems Group, Inc. OS-level security for Solaris and AIX, 2001


