



# 3 Benchmarks

Bei dem Einsatz von Mandatory-Access-Control-Systemen kommt immer wieder die Frage auf, wie viel *Overhead* diese Systeme erzeugen. Hierzu können Benchmarks zur Messung herangezogen werden. Auf <http://lvs.sourceforge.net/> werden verschiedene Benchmarks für Linux aufgeführt. Die meisten Benchmarks sind jedoch synthetischer Natur und können daher nicht direkt auf Applikationen übertragen werden, die in der Realität eingesetzt werden.

Ich verwende hier die Benchmarks LMBench<sup>1</sup> und UNIXbench<sup>2</sup>. Im Folgenden finden Sie die Vergleichswerte mit und ohne Mandatory-Access-Control-System.

Die Prüfungen fanden jeweils im Single-User-Modus statt, um möglichst alle Verfälschungen durch weitere laufende Applikationen zu vermeiden (Mailserver, Cron-Daemon etc.). Als Hardware wurde ein Dell Optiplex 170L eingesetzt. Dieses System hat einen Pentium-4-Prozessor mit 3 GHz und aktiviertem Hyperthreading und 512 Mbyte RAM.

Besonders der *Micro-Benchmark* LMBench zeigt bei SELinux durchaus eine Verringerung der Geschwindigkeit. Diese ist jedoch bei *UNIXbench* mit Ausnahme der gleichzeitigen Shellscript-Ausführung nicht nachzuvollziehen. Als Fazit kann man nur sagen, dass eine Applikation durch SELinux wie auch durch AppArmor gebremst wird. Ob dies aber signifikant ist, kann nur von Applikation zu Applikation durch einen Test geprüft werden. Ob ein Vergleich derartig unterschiedlicher MAC-Systeme überhaupt sinnvoll ist, sollten auch Sie selbst entscheiden. Hier stelle ich lediglich meine Ergebnisse vor. Diese mögen in Abhängigkeit der Architektur auch von System zu System unterschiedlich ausfallen.

## 3.1 Novell AppArmor

Die Benchmark-Prüfung erfolgte auf einer OpenSuse 10.2-Distribution mit allen Patches. Die Zeiten wurden mit und ohne Novell-AppArmor-Modul ermittelt.

Um tatsächlich auch AppArmor zu testen, wurden für die Applikationen jeweils Profile erzeugt, sodass die Benchmark-Applikationen unter der Kontrolle des AppArmor-Systems liefen. Die hierbei verwendeten Profile für die Applikationen sind im Anhang abgedruckt (siehe Abschnitt B.1).

---

<sup>1</sup> <http://www.bitmover.com/lmbench/>

<sup>2</sup> <http://www.tux.org/pub/tux/benchmarks/System/unixbench/>

Der UNIXbench-Benchmark ergab die folgenden Ergebnisse:

Test	ohne AppArmor	mit AppArmor	Verlust
Dhrystone 2 using register variables	343,9	345,5	0%
Double-Precision Whetstone	191,9	192,0	0%
Execl Throughput	694,2	477,5	32%
File Copy 1024 bufsize 2000 maxblocks	732,7	697,1	5%
File Copy 256 bufsize 500 maxblocks	506,4	476,3	6%
File Copy 4096 bufsize 8000 maxblocks	1118,9	1050,5	7%
Pipe Throughput	445,3	440,7	1%
Process Creation	792,8	785,6	1%
Shell Scripts (8 concurrent)	1048,3	934,3	11%
System Call Overhead	390,1	390,3	0%

Die Ergebnisse des LMBench-Benchmarks hier aufzuführen, möchte ich Ihnen ersparen. Sie befinden sich daher im Anhang (siehe Abschnitt C.1). Die wesentlichen Punkte möchte ich jedoch zusammenfassen. Der LMBench-Benchmark weist kaum Unterschiede mit und ohne AppArmor auf. Lediglich bei den folgenden Operationen erkennt man einen deutlichen Effekt durch AppArmor:

- AppArmor bremst die Ausführung eines Shellsriptes um den Faktor 1,2.
- Die Ausführung eines neuen Prozesses ist 35% langsamer.
- Das Öffnen und Schließen einer Datei dauert mit AppArmor doppelt so lange.

## 3.2 SELinux

Die Benchmark-Prüfung erfolgte auf einer Fedora Core 6-Distribution mit sämtlichen verfügbaren Patches. Die Benchmarks wurden mit und ohne SELinux-Unterstützung<sup>3</sup> durchgeführt. Die verwendeten Profile sind im Anhang abgedruckt (siehe Abschnitt B.2). So konnte der Geschwindigkeitsverlust durch die zusätzlichen Kontrollen durch SELinux gemessen werden. Der UNIXbench-Benchmark ergab die in der Tabelle auf der folgenden Seite gezeigten Ergebnisse.

Die Ergebnisse des LMBench sind ebenfalls im Anhang (siehe Abschnitt C.2) zu finden. Hier wieder nur in Auszügen:

- Das Öffnen und Schließen einer Datei benötigt mit SELinux 12% mehr Zeit.
- Der Start eines neuen Prozesses benötigt 5% mehr Zeit.
- Der Start eines Prozesses mit eigener Shell dauert 8% länger.
- Der Start einer Datei benötigt 23% mehr Zeit.

<sup>3</sup> Beim Booten wurde `selinux=0` angegeben.

- Auch das Erzeugen von Dateien dauert bis zu doppelt so lange. Das Löschen einer Datei dauert bis zu 58% länger.

Test	ohne SELinux	mit SELinux	Verlust
Dhrystone 2 using register variables	392,5	375,9	5%
Double-Precision Whetstone	198,9	192,4	4%
Execl Throughput	727,5	679,5	7%
File Copy 1024 bufsize 2000 maxblocks	612,7	585,7	5%
File Copy 256 bufsize 500 maxblocks	418,4	391,9	7%
File Copy 4096 bufsize 8000 maxblocks	1157,9	1124,7	3%
Pipe Throughput	366,4	327,5	11%
Process Creation	745,4	740,4	1%
Shell Scripts (8 concurrent)	1054,5	118,5	89%
System Call Overhead	286,2	285,4	1%

