```
diff --git a/arch/x86/entry/syscalls/syscall_64.tbl
b/arch/x86/entry/syscalls/syscall_64.tbl
index bbfc6d440870..abad2a394917 100644
--- a/arch/x86/entry/syscalls/syscall_64.tbl
+++ b/arch/x86/entry/syscalls/syscall_64.tbl
@@ -345,6 +345,9 @@
 334  common      rseq              __x64_sys_rseq
 424  common      pidfd_send_signal __x64_sys_pidfd_send_signal
 434  common      pidfd_open        __x64_sys_pidfd_open
+435  common      get_proc_log_level      __x64_sys_get_proc_log_level
+436     common  set_proc_log_level      __x64_sys_set_proc_log_level
+437     common  proc_log_message        __x64_sys_proc_log_message

 #
 # x32-specific system call numbers start at 512 to avoid cache impact
diff --git a/include/linux/syscalls.h b/include/linux/syscalls.h
index 8e5b2c6d5dea..d9ef69621a78 100644
--- a/include/linux/syscalls.h
+++ b/include/linux/syscalls.h
@@ -1114,6 +1114,9 @@ asmlinkage long sys_mmap_pgoff(unsigned long addr, unsigned
long len,
                  unsigned long fd, unsigned long pgoff);
 asmlinkage long sys_old_mmap(struct mmap_arg_struct __user *arg);

+asmlinkage int sys_get_proc_log_level(void);
+asmlinkage int sys_set_proc_log_level(int new_level);
+asmlinkage int sys_proc_log_message(int level, char *message);

 /*
  * Not a real system call, but a placeholder for syscalls which are
diff --git a/kernel/sys.c b/kernel/sys.c
index 0a1cdee858de..f70edf221fae 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -2807,3 +2807,77 @@ COMPAT_SYSCALL_DEFINE1(sysinfo, struct compat_sysinfo __user
*, info)
      return 0;
 }
 #endif /* CONFIG_COMPAT */
+
+int proc_log_level = 0;
+EXPORT_SYMBOL(proc_log_level);
+
+SYSCALL_DEFINE0(get_proc_log_level)
+{
+     return proc_log_level;
+}
+
+SYSCALL_DEFINE1(set_proc_log_level, int, new_level)
+{
+     if(new_level > 7 || new_level < 0)
+     {
+           pr_err("\nERROR: INVALID LEVEL\n");
+           return -1;
+     }
+
+     if(current_uid().val != 0)
+     {
+           pr_err("\nERROR: NOT A SUPERUSER\n");
+           return -1;
+     }
+
+     proc_log_level = new_level;
+     return proc_log_level;
+}
+
+SYSCALL_DEFINE2(proc_log_message, int, level, char, *message)
+{
+        if(level > proc_log_level)
+        {
+            return -1;
+        }
+
+        switch(level)
+        {
+                case 0:
+                    printk(KERN_EMERG "PROC_OVERIDE [ %s , %d ]: %s", current-
>comm, current->pid, message);
+                    return level;
+
+                case 1:
+                    printk(KERN_ALERT "PROC_ALERT [ %s , %d ]: %s", current->comm,
current->pid, message);
+                    return level;
+
+                case 2:
+                    printk(KERN_CRIT "PROC_CRITICAL [ %s , %d ]: %s", current-
>comm, current->pid, message);
+                    return level;
+
+                case 3:
+                    printk(KERN_ERR "PROC_ERROR [ %s , %d ]: %s", current->comm,
current->pid, message);
+                    return level;
+
+                case 4:
+                    printk(KERN_WARNING "PROC_WARNING [ %s , %d ]: %s", current-
>comm, current->pid, message);
+                    return level;
+
+                case 5:
+                    printk(KERN_NOTICE "PROC_NOTICE [ %s , %d ]: %s", current-
>comm, current->pid, message);
+                    return level;
+
+                case 6:
+                    printk(KERN_INFO "PROC_INFO [ %s , %d ]: %s", current->comm,
current->pid, message);
+                    return level;
+
+                case 7:
+                    printk(KERN_DEBUG "PROC_DEBUG [ %s , %d ]: %s", current->comm,
current->pid, message);
+                    return level;
+
+                default:
+                    return -1;
+        }
+         //shouldn't reach here but just in case
+         return -1;
+}
```