**Operating System &  
Secure Operating System**

**(SE-Linux)**

A Case Study Report submitted in fulfilment of MTE project work.

**Bachelor of Technology  
(Software Engineering)**

Submitted by:  
**Shubham Sah(2K19/SE/121)**

**Suneet Prakash(2K19/SE/130)**

Guided by:

**Prof. Prashant Giridhar Shambharkar**

Assistant Professor



**Objective**

One of the fundamental concerns in the security of cyberspace and e-commerce is the security of operating systems that are the core piece of software running in all information systems, such as network devices (routers, firewalls, etc), Web servers, customer desktops, PDAs, and so on.

Many of known vulnerabilities discovered so far are rooted from the bugs or deficiency of underneath operating systems. We have discussed the security and lack of security of most commercial operating systems like Unix and Microsoft Windows, and its effect to the overall security of Web based applications and services. Based on DOD’s trusted computer system model, the current effort toward development of secure operating systems is presented, and as a case study, the publicly available security enhanced Linux, SE-Linux, is also analysed.

**Introduction**

Every modern computer system, from network servers, workstation desktops, to laptops and hand-held devices, has a core piece of software, called kernel or operating system, executed on the top of a bare machine of hardware that allocates the basic resources of the system (e.g., CPU, memory, device driver, communication port, etc), and supervises the execution of all applications within the system. Some popular commercial and Open Source operating systems are Microsoft Windows, Mac OS, and Linux.

Because of the crucial role of the operating system in the operation of any computer systems, the security (or lack of security) of an operation system will have fundamental impacts to the overall security of a computer system, including the security of all applications running within the system. A compromise of the underneath operating system will certainly expose danger to any application running in the system. Lack of proper control and containment of execution of individual applications in an operating system may lead to attack or break-in from one application to other applications.

**Proposed Work**

**Security of Operating Systems**

Most modern information computer systems provide concurrent execution of multiple applications in a single physical computing hardware (which may contain multiple processing units). Within such a multitasking, time-sharing environment, individual application jobs share the same resources of the system, e.g., CPU, memory, disk, and I/O devices, under the control of the operating system. In order to protect the execution of individual application jobs from possible interference and attack of other jobs, most contemporary operating systems implement some abstract property of containment, such as process (or task) and TCB (Task Control Block), virtual memory space, file, port, and IPC (Inter Process Communication), etc. An application is controlled that only given resources (e.g., file, process, I/O, IPC) it can access, and given operations (e.g., execution or read-only) it can perform.

However, the limited containment supported by most commercial operating systems bases access decisions only on user identity and ownership without considering additional security-relevant criteria such as the operation and trustworthiness of programs, the role of the user, and the sensitivity or integrity of the data. As long as users or applications have complete power, it will not be possible to control data flow. Because of such weakness of current operating systems, it is rather easy to breach the security of an entire system once an application has been compromised, e.g., by a buffer overflow attack. Some examples of potential exploits from a compromised application are:

* Use of unprotected system resources illegitimately. For example, a worm program launches attack via emails to all targets in the address book of a user after it gets control in a user account.
* Subversion of application enforced protection through the control of underneath system. For example, to deface a Web site by gaining the control of the Web server of the site, say changing a virtual directory in Microsoft IIS.
* Gain direct access to protected system resources by misusing privileges. For example, a compromised “send mail” program running as root on a standard Unix OS will result in super user privileges for the attacker and uncontrolled accesses to all system resources.

**Model of Security**

Generally, in an access control-based security model, there are set of objects, and set of subjects (a subject itself can also be an object). Every object has an associated security attribute, or security label; every subject also has a security label, or security clearance; and a defined set of control rule, or security policy that dictates which subject is authorized to access which object. *For example*, in military security model, a security label consists of two components: a security level with one of the four ratings: unclassified, confidential, secret, and top secret, where unclassified < confidential < secret < top secret, and “<” means “less sensitive than”. In general, security labels are partially ordered. That is, it is possible for two labels to be incomparable, in the sense that neither is more sensitive than the other.

The **Orange Book** defines fundamental security requirements for computer systems and specifies a series of criteria for various levels of security ratings of a computer system based on its system design and security feature.

**Ratings and their characteristics**

***D – Minimal Protection:*** no security is required; the system did not qualify for any of the higher ratings.

***C1 – Discretionary Security Protection:*** the system must identify different users running inside the system, and provide mechanisms for user authentication and authorization to prevent unprivileged user programs from interfere each other.

***C2 – Controlled Access Protection:*** clearing of newly allocated disk space and memory; and ability of auditing for security relevant events such as authentication.

***B1 – Labelled Security Protection:*** the system must implement the Mandatory Access Control in which every subject and object of the system must maintain a security label.

***B2 – Structured Protection:*** the focus is on the structure (design) of the system to maintain greater levels of assurance so that the system behaves predictably and correctly.  
***B3 – Security Domains:*** more requirements to maintain greater assurance that the system will be small enough to be subjected to analysis and tests, and not to have bugs that might allow something to circumvent mandatory access controls.

***A1 – Verified Design:*** no additional features in an A1 system over a B3 system; rather there are formal procedures for the analysis of the design of the system and more rigorous controls on its implementation.

*Most existing commercial operating systems are with the ratings of C2 or below.*

**Requirements of Secure Operating Systems**

Most current operating systems provide discretionary access control, that is, someone who owns a resource can make a decision as to who is allowed to access the resource. Due to the lack of built-in mechanisms for the enforcement of security policies in such systems, the access control is normally a one-shot approach: either all or none privileges are granted.

The basic philosophy of discretionary controls assumes that the users and the programs they run are the good, and it is up to the operating system to trust

them and protect each user from outsiders and other users. Such perception could be extremely difficult to hold true and no longer be considered as secure

enough for computer systems of “information era” with broad connectivity through

the Internet and heavily commercialization of e-commerce services.

Requirements of a secure operating system are discussed below:

* Mandatory security – a built-in mechanism within the operating system (system security administrator) that implements and tightly controls the definition and assignment of security attributes and their actions (security policies) for every operation or function provided by the system.
* Trusted path – a mechanism by which a trustworthiness relationship is established among users and application software.
* Assurance – a process or methodology to verify the design and implementation of the system that should actually behave as it claims to be and meet the security requirements.
* Support of diverse security policies –A secure architecture requires flexibility for support of a wide variety of security policies.

**Result and Discussion**

**SE-Linux:** Security Enhanced Linux Operating System  
SE-Linux is discussed as a case study of recent efforts in the development of secure operating systems. A brief summary of architecture and how SE-Linux meets the general requirements of secure operating systems.

**Background**

One notable feature of SE-Linux release is that it follows the same Open Source Initiative as that of the Linux. All documentation and source code of SE-Linux are publicly available under the same terms and conditions of Linux.

**Architecture**

The SE-Linux is an adoption of the Flask security architecture in Linux operating system. The integration of the security architecture with Linux is accomplished in a way that a new kernel module, called the Security Server (SS) that implements the security policy decision logic, is added into a non-security enhanced Linux (ordinary Linux) that is patched with LSM (Linux Security Module) for maintaining security attributes in kernel data structures and for the mechanism of security control enforcement.

The SS of SE-Linux implements the MAC in the form of identity and role-based access control with type enforcement. A new data type called security context is maintained by the SS, which contains security attributes of identity, role, and type. The combinations of these attributes are used for access decisions with given security policies.

**Meeting Requirements of SE-Linux**

* Mandatory security – with the hooks in LSM and the decision-making policy logics of SS, all system operations in SE-Linux (even initiated by a super-user process) are subjected for mandatory access controls (permission checks) based on security labels of source and target objects.
* Trusted path – permission checks based on the security context of user identity, role definition, and type enforcement of the source and target objects involved in an operation ensures the security policies for transitions or transactions between user authentications, change of roles, and process domains/types, etc.
* Support of diverse security policies – all policy-independent security data types in the kernel and decision-making logics in SS make it a clear separation of mechanism from policy in the implementation of SE-Linux. A variety of diverse security policies can be defined to meet specific security objects. The example of a general-purpose security policy included in the release of SE-Linux could be used as default configuration for a normally secured system.
* Assurance – No formal assurance is provided for the security of SE-Linux, although the well-defined SS module and LSM framework could be a good basis for system assurance of SE-Linux. In addition, the issues of covert channels and security audit trails are not addressed in SE-Linux.

**Conclusion**

With ever growing security alerts and CERT Advisory for systems like Microsoft Windows and the ordinary Linux. The approach covered in this case study is executing applications from a strongly guarded, secure operating system – certainly opens an alternative frontier in battling with many of existing cyber-space threats of the real world

Although, the security of individual applications may still suffer from the vulnerabilities of their own, with the strong containment of a secure operation system, the damages caused from a compromise within one application would be much localized, and the impacts among various applications could be much well controlled.

The release of SE-Linux to general public assures that the usage of secure operating systems is not necessarily an expensive endeavour limited only to academic and defence related institutions, and encourages further efforts in research and development of secure operating systems.

**References**

1. DOD 5200.28-STD, “DOD Trusted Computer System Evaluation Criteria” (Orange Book), 26 December 1985
2. SELinux and grsecurity: A Case Study Comparing Linux Security Kernel Enhancements, Michael Fox, John Giordano, Lori Stotler, Arun Thomas University of Virginia, Department of Computer Science.
3. Linux Security Modules: General Security Hooks for Linux
4. Security Enhanced Linux documentation, selinuxproject.org