An Investigation into Access Control in Various Types of Operating Systems

Mohamed A. Ismail
Military Technical Colleague,
Cairo, Egypt

H. Aboelseoud M.
Doctor, Dept. of computer,
Military Technical Colleague,
Cairo, Egypt

Mohamed B. Senousy
Professor of Computer Science
Sadaat Academy Computer Science and Information Technology, Cairo, Egypt

ABSTRACT
Access control is a security aspect whose importancy increases with technology advances as it forms the core of any security system. Access control can be applied at the operating system (OS) level, middle-ware level, or the application level. The objective of this investigation is to give a detailed overview of access control mechanisms implemented in various types of OSs like general purpose OSs, mobile OSs and distributed OSs. Finally, the paper outlines the main problems and challenges of access control, and proposes future directions in the access control field of research.

General Terms
Security, Access Control

Keywords
Access Control, Operating System Security, Usage Control

1. INTRODUCTION
Access control is a security aspect which is responsible for limiting or preventing unauthorized entities (e.g. users and processes) form accessing the computational resources and digital information while ensuring access for authorized entities. The goals of access control can be divided into three areas as follows: preventing unauthorized disclosure of information (confidentiality) and improper malicious modification (integrity), while ensuring accessibility of resources to authorized entities (availability) [1].

Starting with Lampson's access matrix in the late 1960's [2], dozens of access control mechanisms have been proposed. Only three, usually called traditional access controls, have achieved success in practice. They are:

1. DAC (Discretionary Access Control): in which objects or data are owned by a user (owner) and permission to act on them is given at the discretion of the owner [3]. DAC is widely implemented in many systems because of its flexibility and ease of implementation, enforcement, and policy configuration but the disadvantages of DAC are that it is highly vulnerable to Trojan horses and it is not suitable for an information flow control.

2. MAC (Mandatory Access Control): it is also known as lattice based access control (LBAC) or multilevel security (MLS). In MAC, access is based on labels assigned to subjects and objects and access decisions are made beyond the control of the individual owner of the object [4]. MAC policies are proposed to deal with the information flow control, meet the data protect requirements of secret information and to face the attacks of Trojan horses. The problem with MAC is that it is very rigid, hardly to manage by security administrators and suited, at best, for closed and controlled environments.

3. RBAC (Role-based Access Control): in RBAC, access is granted based on the roles individual users have in their organization based on their job functions. Permissions are assigned to roles based on the requirements of job functions and users are made members of roles, thus gaining permissions assigned to these roles [5]. RBAC is a policy independent mechanism that can be configured as MAC and/or DAC. The main advantage of RBAC that it simplifies the process of administration and management of privileges by making use of roles but the problem of it that it does not fit into open systems where entities are definitely unknown [1].

The above mentioned traditional access control mechanisms are usually considered user-oriented (i.e., access decisions are taken mainly based on the identity of users), the problem with the user-oriented access control comes from the fact that any process operating on behalf of a user usually takes his privileges. Thus, when a process is infected with malicious code like (viruses, worms, etc.) it can misuse the user's privileges to make any action that can compromise the OS's security or harm other applications installed on it so the need comes for what so called application-oriented access control (i.e., access decisions are taken mainly based on the concerned applications rather than on the identity of users) [6].

Application-oriented access control can be achieved by application restrictions and sandboxes techniques which are used to restrict an application's ability to access resources by devoting a set of resources to the application and preventing it from working outside of the sandbox [7].

Access control mechanisms are used in OSs to protect and control access to system resources (files, sockets, services, etc.). In general the security at the OS level is a critical issue. Since, if the OS is compromised then threats will definitely propagate to other layers leading to complete penetration of the entire system.

In the following sections we will describe the access control aspects (user-oriented and application-oriented) in various OSs types like general purpose OSs, mobile OSs and distributed OSs.

The rest of this paper is organized as follows. Section 2 shows access control aspects in general purpose OSs (UNIX, Linux, and Windows). Section 3 shows access control aspects in mobile OSs (Android and Apple's iOS). Section 4 shows access control aspects in distributed OSs (CORBA). Section 5 describes usage control (UCON) model as the successor of

access control. Section 6 presents the main problems and challenges of access control and UCON. Finally, we give our conclusions in Section 7.

2. Access Control in General Purpose OSs

In this section we will describe the access control aspects in three familiar OSs (UNIX, Redhat Enterprise Linux server version 6 (RHEL6) and windows server 2012) presenting which mechanisms are implemented in them.

2.1 UNIX (standard UNIX)

UNIX is a file-orientated OS [8]. On every UNIX system, the file system objects are stored in a hierarchical tree structure of directories starts from the root (denoted by a slash '/'). Each directory may contain a number of file system objects like (regular files, other directories, character and block device nodes or links (symbolic and hard links) to any of these objects, etc.) [9].

The access control in UNIX is based on the DAC mechanism with access control lists (ACL's). The access control model of the UNIX File system is implemented on a per object basis. Each file system object in UNIX has an ACL which has three sets of three access rights bits (read (r), write (w) and execute (x)) determine whether a specified right can be requested on the object or not by inspecting the identity of the requester. The three sets are corresponding for three categories of users which are respectively, the user who owns the object that can be identified by a user ID (UID), the object group that can be identified by a group ID (GID) and all other users [9].

Because files and directories are different entities, the meaning of these bits assigned to each differs slightly. In case of directory, the read permission allows the user to list the files in the directory, the execute permission allows the user to enter the directory, or access a file in the directory and the write permission allows the user to add, rename and remove a directory entry [9].

MAC is implemented in UNIX-based systems through Domain and Type Enforcement (DTE) access control mechanism which is an enhanced version of type enforcement (TE). In DTE the OS is divided as a collection of subjects (active entities) and objects (passive entities). Each subject (process) in system is assigned a security attribute called a domain and each object (file, directory, socket, etc.) is assigned a security attribute called a type. Each domain is defined as a collection of access rights where each right give subjects the ability to access objects of a specified type in one or more access modes (read, write, execute, create, send, receive, etc.)[10]. DTE has a Language for specifying access control policies called Domain Type Enforcement Language (DTEL) which is very expressive language capable of representing other common access control models [11].

Sandboxing is applied in UNIX-based systems by using chroot jail [12]. It is used to limit the access of the controlled process to a specific directory (virtual root directory) in the directory tree structure of the File system. In UNIX many daemons (e.g. Network Time Protocol (NTP) daemon) are executed in their own dedicated chroot jail [13].

2.2 LINUX (RHEL6)

Like many other Linux distributions, the traditional UNIX DAC mechanism is applied in RHEL6 at the file system level. There are also several MAC implementations (e.g. Security-Enhanced Linux (SELinux) and simple mandatory access control (SMAC)) based on the Linux Security Modules (LSM) project have been integrated into the kernel layer.

LSM is a lightweight, general purpose, access control framework merged into Linux kernel permitting many different access control mechanisms to be implemented as loadable kernel modules (LKMs) which are programs written with the intention to extend the kernel [14]. LSM allows modules to mediate access to kernel objects by placing hooks in the kernel code just ahead of the access [15], as shown in Figure (1).

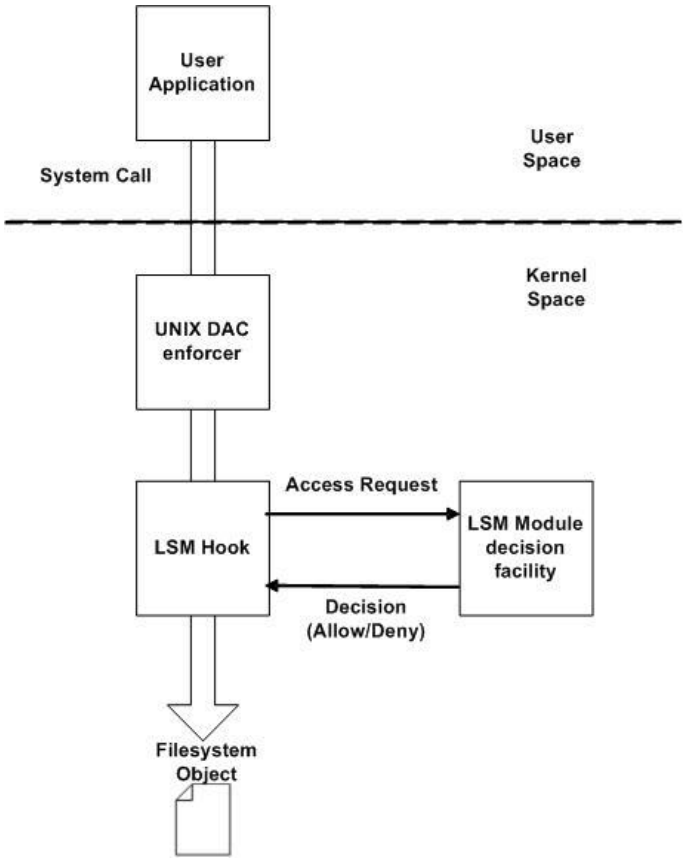

Fig 1: LSM Hook Architecture

One of the most widely used MAC mechanisms in Linux is SELinux which is implemented based on LSM and Flask architecture. SELinux implements a variation of the traditional TE due to the use of object classes along with types for objects and domain for subjects. SELinux also provides a form of Role-Based Access Control (RBAC) built upon TE in which roles are used to group domain types and relate these domains with users, but decisions are based on TE rules instead of RBAC permission assignments [16].

2.3 Windows Server 2012

In earlier releases of Windows OS the major access control mechanism used was discretionary access control list (DACL) which used to define permissions on objects and resources.

In windows server 2012 Microsoft introduces a new File system access control mechanism called Dynamic Access Control which make the administrators capable of specifying central file-access policies at the domain level that can be used in every file server in the domain [17], There are five key components in Dynamic Access Control that work together to achieve the mission [18]:

1. User and Device Claims: claims are Active Directory (AD) properties that can be used with Central Access Policies [17].

Administrators can set claims for both users and devices, for example a user claim can be (the department the user work in, role, clearance level, balance, etc.) and a device claim can be (location, managed, etc.). In fact, this model was originally named "claim-based access control," but Microsoft renamed it to Dynamic Access Control because it offers more than just claims [17].

2. File Classification Infrastructure (FCI): it allows the file server data to be identified and classified using NTFS file system tags so that the administrators can make policies based on this tags, This tagging can be done manually by the file server content owner or automatically by an application that search for certain formats or words in the file server content [17].

3. Expression-Based ACLS: now NTFS file system can use regular expressions in file system ACLS besides other security principals (users, groups, etc.). It is a new great feature for the administrators because it gives them the opportunity to make a flexible policies that manages a fewer security groups. In previous group-based policies there was no concept of an "and" operator for groups, you could only OR groups together, but now using Expression-Based ACLS an expression that means "user is in sales group and managers group" can be written [17, 18].

4. Central Access and Audit Policies: Central Access Policy (CAP) combines both of FCI and Expression-Based Access Control to define appropriate centralized policies that can be applied across multiple file servers in the organization. Central Access Policies are checked after the local DACL is checked but they dominates it which means that if a local DACL on a resource (R) allows access to user (U) but a Central Policy restricts access to this user, the user will not be able to access the resource, These policies are more flexible, powerful and precise than policies that were available in the previous Windows access control models [17, 18].

5. Access Denied Assistance: it helps the clients to know the reasons that prevent them from accessing a given resource. It can remind them to insert a physical passkey or explain to them the access rule they have violate [18].

MAC is implemented in Windows Server 2012 through a security feature called Mandatory Integrity Control (MIC) (it is also referred to as Windows Integrity Control (WIC)) which introduced in Windows Server 2008 and Windows Vista and implemented in subsequent releases of Windows. We can say that MIC is a kind of "User Access Control" oriented to processes by adding Integrity Levels (IL)-based isolation to running processes [19]. Multiple classes of applications can be isolated by using MIC, So scenarios like sandboxing potentially-vulnerable applications (such as Internet-facing applications) can now be achieved [20].

RBAC is implemented in Microsoft AD using groups (e.g., Administrators, Account Operators, Backup Operators, etc.) to control the access of users within those groups to the system resources based on their job functions, Microsoft has also provided a tool called AzMan to help security administrators accomplish RBAC using AD more simply [21].

A sandboxing mechanism called AppContainer is implemented in Windows Server 2012 providing a new isolation method applied to Metro applications. AppContainer protects the OS resources by restricting these applications from reading and writing to most of the file system, except the application's own AppData folder [22].

3. ACCESS CONTROL IN MOBILE OSS

The security of mobile devices is increasingly important as a result of their growing use. In this section we will describe the access control aspects in two familiar mobile OSs (Android and Apple's iOS).

3.1 Android

Android is the first free, open source, and fully customizable OS for mobile devices [23] which is developed and maintained by Google. Nowadays, we can see that it is one of the most popular mobile platforms. It offers a full software stack consisting of [24]:

1. Base OS: it is based on the Linux kernel which provides low-level services to the rest of the system such as file system support, device drivers, memory management, process management, and networking.

2. Middle-ware layer: it includes the Dalvik Virtual Machine (DVM), Java and native libraries, and provides system services, such as the application life cycle management.

3. Application layer: it consists of a collection of pre-installed and third party applications (available from the Google store) as well as some tools and APIs easing the development of third-party applications with the Java programming language.

Several access control mechanisms are applied in Android. We can classify them with respect to the software stack as shown in Figure (2).

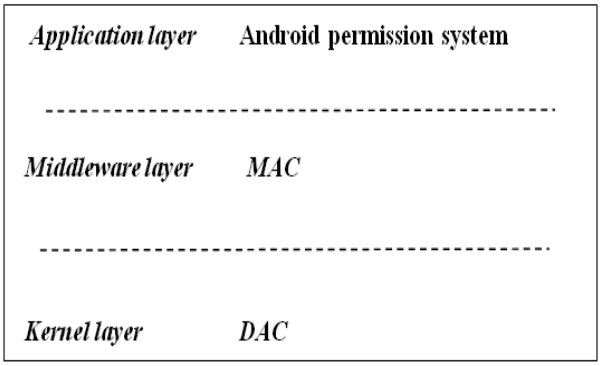

Fig 2: Android software stack with applied access control mechanisms

3.1.1 Kernel layer -specific mechanisms

The traditional UNIX DAC mechanism is applied in the underlying Linux kernel to control the access to the Android Files (both application and system files) and to enforce process isolation by making every application on the Android device runs as a separate user account with a unique (UID) and (GID). So Applications can only access their own files, or files that are explicitly defined as world-wide readable [25] and they will not be able to access the files of the other applications because they doesn't have the necessary permissions.

3.1.2 Middle-ware layer -specific mechanisms

Android's middle-ware layer provides MAC on inter-component communication (ICC) calls which enable android applications to communicate with each other. ICC calls are like Inter-Process Communication (IPC) calls but it is preferred to use the term ICC because these calls occur at the

granularity of application components [25]. ICC calls is controlled by making the Android's reference monitor checks permission assignments at run-time and refuses ICC calls if the caller does not have the necessary permissions [25].

3.1.3 Application layer -specific mechanisms

The core of the application level security in Android is the permission system [24] which controls the operations that an application can perform to limit the application abilities. Application developers must declare which permissions the application needs to be executed properly. At install time, The Package Manager is responsible for granting permissions to the application after the user approve for all the requested permissions and at run-time the application framework is responsible for enforcing system permissions [24].

A wide range of permissions in Android protects Security sensitive operations such as: dialing the phone (CALL_PHONE), taking photos (CAMERA), accessing the Internet (INTERNET), or writing an SMS (WRITE_SMS). Of course any Android application can define any new permissions it needs to protect access to sensitive application interfaces. In order to have permission the application must include it in its manifest file (the application's "contract" with Android) which is part of the application's installation package. Permissions have four protection levels [24]:

1. Normal: permissions that are not especially dangerous to have and automatically granted to the application without the user's approval before or during installation process.

2. Dangerous: permissions that are more dangerous than normal, or not normally needed by applications; such permissions may be granted to an application with the user's explicit confirmation at installation time of the application.

3. Signature: permissions that can only be granted to other packages that are signed with the same signature as the one declaring the permission.

4. SignatureOrSystem: a signature permission that is also granted to packages installed in the Android system image, these permissions are not available to 3rd party applications.

The application developer assigns the protection level during the development process due to his discretion.

3.2 Apple iOS

iOS (formally known as iPhone OS) is the OS that is running on Apples' iPhone, iPod Touch, and iPad devices. It is a proprietary OS developed and maintained by Apple.

The security model of iOS is not permission based as in Android [26]. When a developer submit his application to the Apple App Store, Apple inspects the application by making manual and automatic tests on it to ensure that the application do not have any malicious behavior. When the application goes through the inspection process Apple signs it digitally and make it available to be downloaded and installed on any apple device.

Once on the device, the application is free to access any resources on the device except few resources (e.g. user's location) that needs the user's approval for it at the first time the application use the resource. Later, whenever the user likes to revoke the application access to this resource he can do it by navigating to the iOS settings.

The traditional UNIX DAC mechanism is applied in iOS to manage the file system and achieve the basic privilege separation [27] while Controlling and separating the applications in iOS is done by an access control system current known as the Apple Sandbox which is implemented as a policy module in the TrustedBSD MAC framework [28]. A set of entitlements for the security permissions in iOS are declared for each application in its plist file (XML format file) to determine its sandbox policy [27].

4. ACCESS CONTROL IN DISTRIBUTED OSS

The integration of distributed computing systems and the object oriented model results in what so called distributed object computing systems, in which objects are distributed across multiple computers [29]. A good example of distributed object computing is the Common Object Request Broker Architecture (CORBA) which is defined and standardized by the Object Management Group (OMG) [30]. CORBA can interconnect multiple object systems providing interoperability between applications running on them in heterogeneous distributed environments [31].

The core element of CORBA is the object request broker (ORB) [29], which allows clients and servers to communicate with each other providing language transparency, location transparency and interoperability.

The distributed nature of CORBA (and middleware in general) makes it a perfect target for the attackers because there are many places where the attackers can exploit it to break into the system [32]. Thus, security requirements of CORBA systems must be taken into consideration.

Access control plays an important role in CORBA systems, When a client make a request and the target side receives it , the access control module should intercept it and decide if the caller is allowed to invoke the target method or not [32].

Access control in CORBA can be achieved at both sides (client-side and server-side). The ORB at each side is responsible for applying the client/server domain access policy which checks if the client is authorized to invoke the required operation or not [32].

The access control policy can be applied at the target objects level by inspecting the role or clearance of the principal and it can be also applied at the operations level by associating standard sensitivity levels to each operation and comparing the required level of access to the operation with the level granted to the client to see if the client's granted level is sufficient for access or not [32].

The specification provides a standard set of access rights includes g (get), s (set), u (use), and m (manage) and additional rights families may also be defined by developers to fit the requirements of their access control model. Thus, different access control mechanisms can be applied, such as DAC, MAC, and RBAC [32].

5. UCON MODEL

Researchers have studied various new solutions and enhancements for current classic access control models but these studies are usually dedicated to specific target problems (ad-hoc solutions) and not comprehensive enough to cover the broad traditional models, hence UCON model comes as a unified framework to extend traditional access control models

in a way that make it suitable for new challenges in the computer security.

$UCON_{ABC}$ model proposed by Sandhu et. al [33] formalizes the UCON model based on the concepts of authorization (A), obligations (B),and conditions (C) and also introduces new features like continuity (ongoing controls) and mutability of attributes, it encompasses and enhances traditional access control models, Trust Management (TM), and Digital Rights Management (DRM) and goes beyond them in its definition and scope.

A number of publications on UCON at the OS level are proposed such as [34, 35, 36, 37] to protect and control usage of OS sensitive resources by detecting and preventing kernel-level malicious attacks.

6. PROBLEMS AND CHALLENGES OF ACCESS CONTROL

This section will give an overview of the main problems and challenges of the access control systems that still need a substantial amount of research and identify some of the future research directions in UCON.

Conflict resolution. The problem of policy conflicts presents a challenge. Policy conflicts may happen as a result of the interaction of different access policies, leading to severe security problems. Research is required to identify the process of conflicts detection and resolving [38].

Usability. The usability of access controls should be taken into account by the access control designers and communities, they should resolve the tension between low level enforcement and a higher level controls for users [39].

Administration. The administration of access control systems is a tough challenge especially in systems like Grid computing, Cloud, social networks and other distributed systems where we can find several administrative domains. Splitting access control across different domains, making it hard to evaluate the effective permissions which a subject has [39], the main issue in multiple administrative domains environment is to how to map the local access policy to global access policy and vice versa.

Reliability in centralized administrative access control system is also a significant problem. Since, if the central administration server goes down, or communication problems occur between the server and clients the users will not be able to access their resources [40].

Lack of standardization. In our opinion, there is a lack of standardization generally in security and especially in access control. There is a real need to clarify and standardize many access control aspects like models, mechanisms, policy languages and even the concepts and definitions used in this field.

Scalability. In systems like Ultra-Large-Scale (ULS) systems [41] which have a huge number of users, resources, volumes of data, policies, objectives, and lines of source code the problem of access control is a challenge because it needs to scale beyond the normal systems which consists of few machines and centralized servers. The access control in ULS systems needs to handle issues such as scaling, performance of communications, fault tolerance and hence there is a need to develop novel access control mechanisms which is suitable for such environments [40].

UCON is largely open to research and there is still a lot of work to be done in this field because:

1. UCON is just a conceptual model and there is no concrete realization specification for it [42].

2. UCON is typically implemented at the application layer because at the OS level there is no support readily available for it [43]. So, further research is still required at this level.

3. There is a need to develop new policy specification languages that are capable of expressing complicated usage scenarios and policies that exist in modern systems.

4. Administration and delegation of rights issues in UCON are still active areas of research.

7. CONCLUSION

This paper investigated the existing access control mechanisms implemented in various types of OSs and presented a novel promising model called UCON as the successor of access control. Finally, it pointed to the main problems and challenges of the access/usage control research area. The results of this investigation show that the current access control solutions implemented at OS level are not sufficient enough and they need to be supported with appropriate UCON models that can provide substantial security benefits. The major advantage of UCON is that it is capable of expressing various access models such as DAC, MAC, RBAC, TM, DRM and going beyond them in its definition and scope.

Summary of the investigation is displayed in table 1. Some interested points about this summary include:

- General purpose OSs are typically used in multiuser environments. So, user-oriented access control comes at the first place in this type of OSs.

- Most, if not all, of the OSs reviewed in this paper use DAC mechanism as the base access control mechanism because of its flexibility and also, because system administrators are quite familiar with it.

As relying only on DAC mechanism for protecting OS resources is not sufficient to obtain a high level of security and often leads to make OS vulnerable to Trojan horse type attacks, OSs strengthens their security systems by using access control systems supporting MAC (e.g., SELinux, SMAC, MIC, etc.). The rigidity and complexity of MAC policies are problems that face security administrators and needs to be solved. So, it is recommended to develop tools that automate the labeling process to simplify the process of administration and management of this type of policies.

Table 1: Availability of Access/Usage Control Mechanisms in the Investigated OSs

OS Type	OS	DAC	MAC	RBAC	Sandbox	UCON
General Purpose OS	UNIX	✔	✔	✘	✔	✘
	Linux(RHEL6)	✔	✔	✔	✔	✘
	Windows Server 2012	✔	✔	✔	✔	✘
Mobile OS	iOS	✔	✔	✘	✔	✘
	Android	✔	✔	✘	✔	✘
Distributed OS	CORBA	Implementation Dependable				

RBAC is strongly supported in the investigated General purpose OSs as they are usually used in corporate environments where there is a need to mimics the organization's roles. Hence, RBAC is proposed as a policy independent model that can be configured as MAC and/or DAC with a major purpose of facilitating the security administration process in these organizations.

- As can be seen in table 1, there are OSs (e.g. Linux and Windows) applying the three mechanisms (DAC, MAC and RBAC) together to achieve the flexibility, strength and ease of administration but the interaction of these different access policies may lead to the problem of policy conflicts. So, Security administrators should find suitable methods to identify and resolve these conflicts.

- Since mobile devices are typically personal devices, the access control systems in mobile OSs are not primarily focus on the users(user-oriented access control), but it mainly focuses on the applications (application–oriented access control) by limiting the access for them using application sandboxing like in Apple's iOS or by applying a restricting permission system like in Android.

- RBAC is not supported in the investigated mobile OSs as current mobile devices are considered to be personal devices and usually not used in corporate environments but soon or later, mobile devices will be adopted for use in such environments. So, RBAC and other access control features will be needed to protect business data that may be stored on these mobile devices.

- OMG has designed the CORBA Security model in a sufficient generic way that allows for applying various access control mechanisms like (DAC, MAC, RBAC, etc.) which in turn gives great flexibility in security policy specification. It makes use of proper abstractions that allow fine-grained access control at the operations level besides the target objects level.

- None of the OSs that were investigated in this paper apply UCON model. So, it is recommended to bring UCON into these OSs because there are a lot of usage scenarios that can be applied at OS level to secure and control usage of OS resource besides preventing malicious attacks that target OS kernel. Researchers and developers should also design new UCON models, policy specification languages and enforcement mechanisms suitable for the evolution of OSs.

8. REFERENCES

[1] Lazouski, Aliaksandr, Fabio Martinelli, and Paolo Mori. "Usage control in computer security: A survey." Computer Science Review 4.2 (2010): 81-99.

[2] LAMPSON, B.W. 1971. Protection. 5th Princeton Symposium on Information Science and Systems. Reprinted in ACM Operating Systems Review 8, 1, 18–24, 1974

[3] Russell D, Gangemi GT. Computer security basics. Sebastopol, CA:O'Reilly and Associates; 1991.

[4] Ramachandran R, Pearce DJ, Welch I. AspectJ for multilevel security. In: The 5th AOSD workshop on aspects, components, and patterns for infrastructure software (ACP4IS). Bonn, Germany; 2006. p. 1–5.

[5] SANDHU, R., COYNE, E., FEINSTEIN, H., AND YOUMAN,C. 1996. Role based access control models. IEEE Computer 29, 2.

[6] Schreuders, Z. Cliffe, Tanya McGill, and Christian Payne. "The state of the art of application restrictions and sandboxes: A survey of application-oriented access controls and their shortfalls." Computers & Security 32 (2013): 219-241.

[7] Andress, Jason. The basics of information security: understanding the fundamentals of InfoSec in theory and practice. Access Online via Elsevier, 2011.

[8] Dalton, Chris I., Tse Huong Choo, and Andrew P. Norman. "Design of secure UNIX." Information Security Technical Report 7.1 (2002): 37-56.

[9] Mellander, Jim. "Unix Filesystem Security." Information Security Technical Report 7.1 (2002): 11-25.

[10] Sterne, Daniel F., et al. "Scalable access control for distributed object systems." Proceedings of the 8th USENIX Security Symposium. 1999.

[11] Carr, Steve, and Jean Mayo. "Teaching access control with domain type enforcement." Journal of Computing Sciences in Colleges 27.1 (2011): 74-80.

[12] R. E. Smith, Mandatory protection for internet server software," in Proceedings of the 12th Annual Computer Security Applications Conference, ser. ACSAC '96. Washington, DC, USA: IEEE Computer Society,1996,pp.178{.[Online].Available:http://dl.acm.org/citation.cfm?id=784588.784626

[13] Matthews, Christopher James. Isolating Legacy Applications with Lind. Diss. University of Victoria, 2013.

[14] Rául Siles Peláez. Linux kernel rootkits: protecting the systems ``ring-zero". GIAC Unix Security Administrator (GCUX), May 2004.

[15] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman. Linux Security Modules: General Security Support for the Linux Kernel. In Proceedings of the 11th Annual USENIX Security Symposium, pages 17–31, San Francisco,California, August 2002.

[16] Mayer, F., MacMillan, K., & Caplan, D. (2007). SELinux by example: using security enhanced Linux. Upper Saddle River, NJ: Prentice Hall.

[17] http://windowsitpro.com/windows-server-012/exploring-windows-server-2012-dynamic-access-ontrol.

[18] http://www.infoq.com/news/2012/l0/Dynamic-Access-Control.

[19]http://www.informit.com/guides/content.aspx?g=windowsserver&seqNum=306

[20] http://en.wikipedia.org/wiki/Mandatory_Integrity_Control

[21] http://www.sans.org/reading-room/analysts-program/access-control-foxt

[22] http://blog.avecto.com/2012/05/application-sandboxing-in-windows 8/

[23] Ni, Xudong, et al. "DiffUser: Differentiated user access control on smartphones." Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on. IEEE, 2009.

[24] Shabtai, Asaf et al. "Google Android: A state-of-the-art review of security mechanisms." arXiv preprint arXiv: 09l2.5l01 (2009).

[25] Bugiel, Sven, et al. "Towards taming privilege-escalation attacks on Android." Proceedings of the 19th Annual Symposium on Network and Distributed System Security. 2012.

[26] Mylonas, Alexios, et al. "On the feasibility of malware attacks in smartphone platforms." E-Business and Telecommunications. Springer Berlin Heidelberg, 2012. 217-232.

[27] Wang, Tielei, et al. "Jekyll on iOS: when benign apps become evil." Presented as part of the 22nd USENIX Security Symposium}. USENIX}, 2013.

[28] Blazakis, Dionysus. "The Apple Sandbox." Arlington, VA, January (2011).

[29] Narasimban, P., Louise E. Moser, and P. Michael Melliar-Smith. "Using interceptors to enhance CORBA." Computer 32.7 (1999): 62-68.

[30] Hartman, Bret, Donald J. Flinn, and Konstantin Beznosov. Enterprise Security with EJB and CORBA. Vol. 16. John Wiley & Sons, 2002.

[31] Deng, Robert H., et al. "Integrating security in CORBA based object architectures." Security and Privacy, 1995. Proceedings., 1995 IEEE Symposium on. IEEE, 1995.

[32] Lang, Ulrich, and Rudolf Schreiner. Developing secure distributed systems with CORBA. Artech house, 2002.

[33] Park, Jaehong, and Sandhu Ravi (2004). The UCONabc usage control model. ACM Trans. Inf. Syst. Secur., 7:128–174.

[34] Teigão, Rafael, Carlos Maziero, and Altair Santin. "Applying a usage control model in an operating system kernel." Journal of Network and Computer Applications 34.4 (2011): 1342-1352.

[35] M. Xu, X. Jiang, R. Sandhu, X. Zhang, Towards a VMM-based usage control framework for OS kernel integrity protection,in: SACMAT' 07: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, ACM, NewYork, NY,USA, 2007, pp. 71–80.

[36] D. Kyle, J.C. Brustoloni, Uclinux: A linux security module for trusted-computing-based usage controls enforcement,in: STC'07: Proceedings of ACMWorkshop on Scalable Trusted Computing, ACM, New York, NY, USA, 2007, pp. 63–70.

[37] M. Alam, J.-P. Seifert, Q. Li, X. Zhang, Usage control platformization via trustworthy SELinux, in: ASIACCS'08: Proceedings of ACM Symposium on Information, Computer and Communications Security, ACM, New York, NY, USA, 2008, pp. 245– 248.

[38] Ray, Indrakshi, and Indrajit Ray. "Access Control Challenges for Cyber-Physical Systems."

[39] Usability Meets Access Control Challenges and Research Opportunities 2009.

[40] Garnes, Håvard Husevåg. "Access Control in Multi-Thousand-Machine Datacenters." (2008).

[41] http://en.m.wikipedia.org/wiki/Ultra-large-scale_systems

[42] Danwei, Chen, Huang Xiuli, and Ren Xunyi. "Access control of cloud service based on ucon." Cloud Computing. Springer Berlin Heidelberg, 2009. 559-564.

[43] Suhendra, Vivy. "A survey on access control deployment." Security Technology. Springer Berlin Heidelberg, 2011. 11-20.