Windows Security Reference Monitor

Daniel Cordeiro Marques

University of San Diego

CSOL-500-01-FA20 - Foundations of Cyber Security

Professor Ashton Mozano

October 5, 2020

Windows Security Reference Monitor

Schneier (2000) defines reference monitors as a mediator of access to objects by subjects. Tanenbaum & Bos (2015) expand on that definition, proposing that reference monitors allow the concentration of security decisions in one place. For example, reference monitors are responsible for handling access to memory and files by users and processes in operating systems. This paper briefly discusses the Windows Security Reference Monitor (SRM), Microsoft's security policy implementation, by mapping parts of the Windows security system components to a model reference monitor and its ramifications towards information protection.

Reference monitors

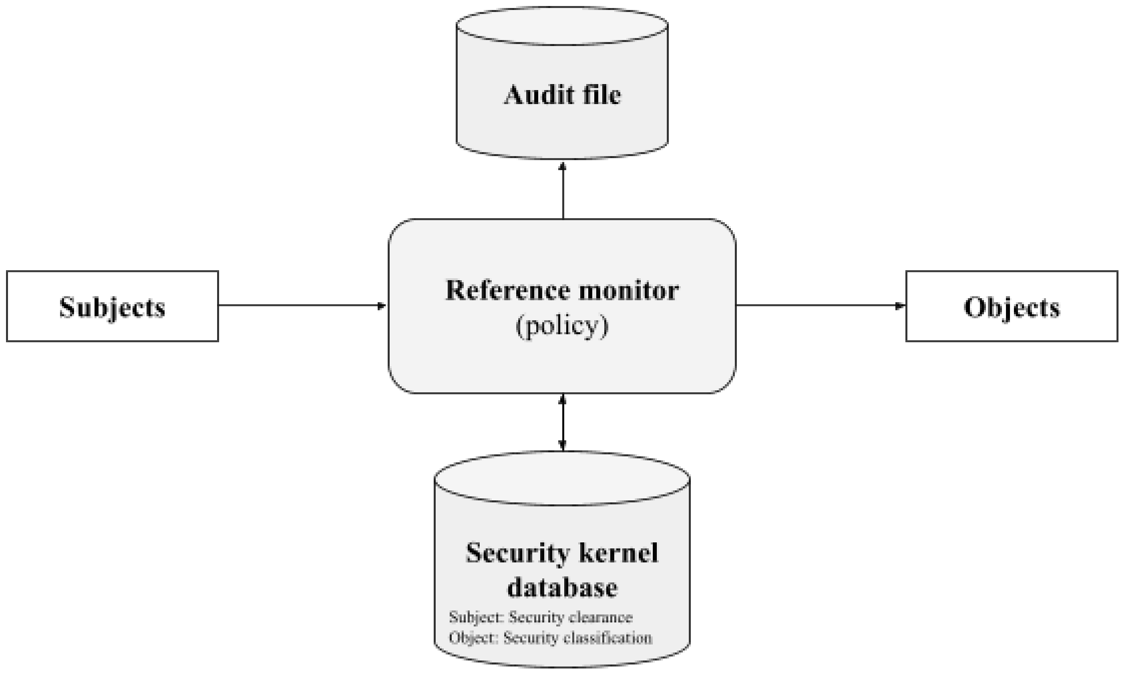
Reference monitor features

Stallings (2014) proposes a conceptual reference monitor illustrated by Figure 1 and highlights three properties essential for its successful implementation:

- Complete mediation. The reference monitor must verify and enforce security policies on every access, and therefore subjects should not bypass it.
- Isolation. Safeguards must protect the reference monitor and security policies from unauthorized modification.
- Verifiability. The reference monitor's ability to enforce policies should be provable. For example, the reference monitor code should be small enough that its correctness can be verified reliability.

Figure 1

Conceptual reference monitor (Stallings, 2014).



Following this model, users and processes will be treated as subjects, while operating system resources (such as memory and files) will be objects. The security kernel database stores the privileges required to access the objects and the classification of each object. For instance, if a user requests a file, the reference monitor first checks the database for the file's classification and the required privileges to access the file. If the user fulfills the necessary criteria, the reference monitor allows access to the file (Stallings, 2014). Finally, the reference monitor stores security-relevant events in the audit file, including policy changes and unauthorized access attempts.

Microsoft Windows reference monitor implementation

Security reference monitor (SRM)

Yosifovich et al. (2017) describe the SRM as being “responsible for defining the access token data structure to represent a security context, performing security access checks on objects, manipulating privileges (user rights), and generating any resulting security audit messages.” (p. 608). The SRM does not handle authentication. The Local Security Authority Subsystem Service (LSASS) manages these functions.

Like other core operating system components, the SRM runs in kernel mode. As noted by “*Windows Kernel-Mode Security Reference Monitor*” (2018), the operating system provides access to SRM functionality through a set of routines, commonly known as “Se routines.” For example, the function SeAccessCheck determines whether the operating system can grant the subject access rights to an object.

Local security authority subsystem service (LSASS) and LSASS database

The LSASS is a user-mode process that manages authentication, the local system security policies (such as privileges held by users and groups), and audit policies. The LSASS database stores these policies and cached logon information used by Windows domains.

Access tokens

The operating system creates an access token for a user and subsequently applies it to each process and thread started by the user (Howard & LeBlanc, 2003, p. 218). The SRM uses tokens to identify the security context of a process or thread (Yosifovich et al., 2017, p. 632). Security tokens store two essential pieces of information, among others:

Security identifiers (SIDs). SIDs are variable-length numeric value that identifies entities performing actions on the system. For instance, the Windows installation process assigns a SID to computers when completed; the operating system also creates a unique logon SID for each interactive logon session (Yosifovich et al., 2017, p. 626).

Privileges. Yosifovich et al. (2017) define privilege as "the right of an account to perform a particular system-related operation." (p. 668). The SRM is not always responsible for enforcing privileges; they can be defined and implemented by different components and modified (enabled or disabled) when their use is required. Example privileges include SeDebugPrivilege (checked by the process manager, allows the user to access any process or thread - except protected processes) and SeTcbPrivilege (allows the user to act as part of the operating system and is checked by the SRM).

Security descriptors and access control lists (ACLs)

While tokens relate primarily with subjects, security descriptors specify policies around objects. Security descriptors consist of attributes, including the object owner SID, the SID for the object's primary group, and access control lists. Access control lists (ACLs) are a collection of Access control entries (ACEs) used to identify the access rights given to a subject ("*Access Control Lists*", 2018). There are two types of ACLs:

Discretionary access control lists (DACLs). Yosifovich et al. (2017) summarized that DACLs "specify who has what access to the objects." ACEs in DACLs contain SIDs and an access mask that specifies the access rights granted to the SID holder (Yosifovich et al., 2017, p. 652).

System access control lists (SACLs). SACLs deal primarily with audit functions. Each ACE specifies the operations that should be audited and logged in the system audit log (Yosifovich et al., 2017, p. 653).

Audit records

The system's event log stores security audit records. At initialization time and on policy changes, LSASS informs the SRM of which events should generate logs. LSASS is also responsible for receiving audit events from the SRM and sending them to the event logger (Yosifovich et al., 2017, p. 678).

Security reference monitor mapping

Considering the model proposed by Stallings (2014), the SRM performs the reference monitor's role. It is responsible for verifying if subjects (users and processes) can access system objects using their tokens and requested access. As pointed out by Yosifovich et al. (2017, p. 623), "SRM's security model takes three inputs: the security identity of the thread, the access that the thread wants to an object, and the security settings of the object." The security descriptors represent policies and security settings through ACLs applicable to objects, and the ACEs that make up the ACLs are the access privilege and protection attributes. Finally, the LSASS database also stores additional policies applicable to the local system.

Discussion

This discussion evaluates the SRM against the three reference monitor properties proposed by Stallings (2014, p. 24-13) to reduce the scope of this work:

Complete mediation. As discussed in Yosifovich et al. (2017, p. 621), when a subject requests access to an object, the object managers leverage the SRM to check the object's security descriptor and verify if the subject is allowed to proceed. However, “*Windows Security Model For Driver Developers*” (2018) mentions that it is the object manager’s responsibility to verify the subject’s access rights and pass it to lower components if it cannot perform the check. Thus, SRM’s complete mediation cannot be verified, as object managers can perform the function themselves.

Isolation. Robbins & Schroeder (2017) demonstrated that attackers with the necessary privileges could target DACLs and modify ACEs, interfering in the ACE evaluation process and abusing the SRM to create backdoors. Similarly, Alexander & Breen (2017) and Pierini (2019) illustrate how an attacker may abuse misconfigured “Se” privileges to obtain access to additional system resources. For instance, Microsoft defines SeTcbPrivilege as a privilege that gives the user the right to “act as part of the operating system.” (“*Privilege Constants (Authorization)*”, 2020). Attackers can abuse this feature to impersonate other users and gain unauthorized access to resources. Therefore, the SRM cannot be considered tamperproof.

Verifiability. The SRM is part of a larger and more complex system. An essential characteristic of the Windows security model is the flexibility of the permissions assigned to a user, allowing multiple combinations to be defined. Similarly, the windows security policy is extensible, and security rules can be arbitrarily defined. As such, verification of the SRM correctness is a complex problem.

Finally, the existence of administrative accounts can add another layer of exposure. Users with the same privileges as administrator accounts have broad access to operating system functionality and can, ultimately, bypass the controls provided by the SRM. Malicious device drivers can also become a threat; code running in the kernel mode is not verified and is considered part of the trusting computing base (TCB). An attacker with the ability to run crafted device drivers might target the privileged parts of the system and circumvent the security mechanisms provided by the SRM.

Conclusion

The Windows security architecture is complex and involves multiple parts of the operating system. The flexibility and extensibility of the model allow administrators to implement policies aligned with the overall company strategy. However, the Windows security reference monitor does not strictly enforce security goals. As described during this paper, the SRM does not mediate all sensitive operations; other system components also perform mediation functions, such as the object managers. The SMR is not tamperproof, as high privilege users can perform actions that circumvent the SMR verification and abuse the access check process. Finally, the windows security model provides discretionary access control, which can be modified by processes that are not part of the trusting computing base (TCB). Future research not part of this paper includes expanding on abusing system privileges to gain additional access to the SMR and techniques to load untrusted device drivers and target the SMR process in the kernel-mode.

References

Access Control Lists. (2018, May 31). Microsoft Docs. Retrieved October 3, 2020, from https://docs.microsoft.com/en-us/windows/win32/secauthz/access-control-lists

Alexander, B., & Breen, S. (2017, August 25). *Abusing token privileges for windows local privilege escalation*. Foxglove security. Retrieved October 2, 2020, from https://foxglovesecurity.com/2017/08/25/abusing-token-privileges-for-windows-local-privilege-escalation/

Howard, M., & LeBlanc, D. (2003). *Writing secure code: Practical strategies and techniques for secure application coding in a networked world* (Second ed.). Microsoft Press.

Pierini, A. (2019, June 19). *Show me your privileges and I will lead you to SYSTEM*. Hack in Paris Conference 2019. https://hackinparis.com/data/slides/2019/talks/HIP2019-Andrea_Pierini-Whoami_Priv_Show_Me_Your_Privileges_And_I_Will_Lead_You_To_System.pdf

Privilege Constants (Authorization). (2020, July 27). Microsoft Docs. Retrieved October 4, 2020, from https://docs.microsoft.com/en-us/windows/win32/secauthz/privilege-constants

Robbins, A., & Schroeder, W. (2017). *An ACE up the sleeve: Designing Active Directory DACL backdoors*. Black Hat Conference Briefings 2017. https://www.blackhat.com/docs/us-17/wednesday/us-17-Robbins-An-ACE-Up-The-Sleeve-Designing-Active-Directory-DACL-Backdoors-wp.pdf

Schneier, B. (2000). *Secrets & lies: Digital security in a networked world*. John Wiley & Sons. https://doi.org/10.1002/9781119183631

Stallings, W. (2014). Operating system security. In *Computer security handbook* (6th ed., 24.1-24.21). John Wiley & Sons, Inc.

Tanenbaum, A. S., & Bos, H. (2015). *Modern operating systems* (Fourth ed.). Pearson Education.

Windows kernel-mode security reference monitor. (2018, October 17). Microsoft Docs. Retrieved October 3, 2020, from https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-security-reference-monitor

Windows security model for driver developers. (2018, February 1). Microsoft Docs. Retrieved October 4, 2020, from https://docs.microsoft.com/en-us/windows-hardware/drivers/driversecurity/windows-security-model

Yosifovich, P., Ionescu, A., Russinovich, M., & Solomon, D. (2017). *Windows Internals, Part 1: System architecture, processes, threads, memory management, and more* (Seventh ed.). Microsoft Press.