R210 Essay: Access Control

Chongyang Shi (*cs940*)

January 25, 2018

1 Summaries of research

Published a few years after Karger and Schroeder's evaluation [1] of the Multics system – which discovered a range of security vulnerabilities, a new report by Bell and LaPadula [2] gave a comprehensive overview of the then-improved system. Starting with a summary of previous studies on security models in general and on Multics, the authors formally defined the system's underlying access control model, as well as its definition of security. This was followed by a description of how the mathematical model was implemented in Multics through analogies. Finally, discussions were made on aspects of the security model that could still affect the security assurance of Multics, despite the improvements made. A potential criticism focuses on the outset mathematical analogy in Section III, which could have been more concise by invoking Multics' elements directly.

Later in research of access control, Badger et al. [3] published a prototype UNIX access control system with kernel modification known as Domain and Type Enforcement (DTE). Addressing the limitations of Domain Definition Table (DDT) systems in use at the time, the authors set out three design goals of DTE in an effort to simplify configuration and improve cross-host compatibility. The authors described and justified the designs of DTE's configuration language, attribute management, networking capabilities, and NFS interoperability. The implementation of the prototype was also described in detail, along with the state of work for a fully operational DTE-protected UNIX system. The authors concluded with discussions on issues yet to be solved in DTE's design and implementation. Under modern eyes, lack of encryption in DTE's network exchange of control information could lead to information leak and unauthorised modification.

In a 2013 magazine article, Watson [4] summarised preceding developments of extensible access control systems. Since the aforementioned efforts, access control systems had shifted towards extensibility due to demands from evolving computing devices. Through the MAC Framework for access control in FreeBSD, Watson contrasted drawbacks of previous systems with extensible design principles of MAC. This was followed with extensive discussions on how access control was implemented in line with the design principles in various contemporary operating systems, as well as the modifications each OS made and their effects. A potential addition to the article is a quantitative evaluation on performance overhead introduced by each access control implementation.

2 Key themes of research

2.1 Hierarchical arrangement of subject and object permissions

In order to organise subjects (e.g. processes) and objects (accessed items) for formalisable security, a hierarchical structure with recursive mapping [2, p. 36] had been extensively used in access control systems. This was the case for both Multics [2, p. 12] and DTE [3, pp. 52-55]. Each subject-object relation must follow a single path, and mediations were required to resolve hard-link and networking conflicts arose. To reduce complexities and improve performance, the more extensible MAC [4, p. 57] instead introduced the separation of access policies and object labelling.

2.2 Developing and integrating a security model into a system not designed for security is challenging

Before modern security threats to computer systems emerged, operating systems were often designed without rigid access control in mind, leading to complexities when access control models need to be integrated, which for Multics had to "disturb the designed operations as little as possible" [2, p. 25]. DTE was designed to address such inconveniences in DDT, but left out issues such as unencrypted transmission of control information [3, pp. 60-61]. In contrast, access control systems in modern operating systems such as FreeBSD's MAC were designed with minimal integration and upstream merging complexity [4, p. 58] to mitigate integration challenge.

2.3 It is tricky to work with trusted subjects

Trusted subjects have always been a difficult case in access control systems, which can be a system process on which all processes depend [2, p. 67] [3, p. 48], or a superuser with unrestricted access [4, p. 60], often forming the weakest link in systems' security. To address this, the MAC Framework replaced kernel privileges checks with fine-grained system privileges that can be specified by policies [4, p. 59], enabling greater control over trusted subjects in a secure system.

3 Ideas in current context

The attribute-based access control model of Multics influenced the designs of user and file access control systems in many modern systems. User groups and file access permissions in Linux distributions [5] and modern UNIX bear many similarities to that described by Bell and LaPadula in 1976. Similar entity mapping and access level models were also present in the policy syntax of DTE [3, Fig. 3]. Multics' recursive access control lists could be an inspiration for linked access principals in modern encrypted data frameworks such as Mylar [6, Fig. 3].

In order to exchange access control information between servers that may or may not have DTE, DTE attributes were transmitted in metadata of TCP and UDP packets in cleartext [3, pp. 60-62], exposing them to unauthorised interception and modification. Prior to the publication of DTE, ideas of transmitting identity information in access control securely were already suggested, such as by Gong [7] in 1989. A modern solution to the issue is to implement authentication with encryption and integrity check, such as through TLS [8].

Watson [4, p. 53] observed the trend of OS access control shifting from protecting multiple users from each other towards protecting single-user systems. This remains the case five years later. With further popularisation of smartphones and the emergence of Internet-of-Things (IoT) devices, contemporary focuses of access control research include protecting insufficiently secure IoT devices from botnet hijacking [9, p. 100]; and securing mobile operating systems (particularly Android devices with easily-obtained root access) against malicious applications [10].

4 Literature review

The definition of security as well as the inductive process of state transitions leading to its proof, as described by Bell and LaPadula [2], became known as the *Basic Security Theorem*, describing a minimum standard for secure systems. McLean [11] however argued in 1985 that this interpretation is erroneous, as the model was insufficient to model every aspect of the operation of an implementation. Also addressing unsound applications of the model, an axiom-based model was built by Lin [12], significantly reducing the scope of trusted subjects. The inductive process of the model was additionally verified and formalised in Coq by Gureghian et al. [13] in 2013.

Adopting the concept of *sandboxing*, Goldberg et al. [14] designed an access control mechanism to protect the operating system from insecure applications, with DTE [3] as an inspiration. As an addition to DTE's fixed policy model, Provos [15] implemented a flexible policy model through system call interposition to support additional policy-related features. Originally designed for UNIX, many of DTE's features have also been adopted into the unified Linux Security Modules (LSM) framework [16] that can be loaded into the Linux kernel.

Watson's design principles for extensible access control [4, p. 53] have been implemented for Android by Heuser et al. [17] as a programmable interface. As a hardware-level support for extensible access control, a RISC instruction set providing application compartmentalisation was proposed by Watson et al. [18] in 2015. Another related idea was *Unikernals* developed by Madhavapeddy et al. [19], implementing type-safe access control to compartmentalised OCaml applications running on minimised kernels.

(1120 words according to texcount.)

References

[1] P. A. Karger and R. R. Schell, "Multics security evaluation volume ii: Vulnerability analysis," ELECTRONIC SYSTEMS DIV LG HANSCOM FIELD MASS, Tech. Rep., 1974.

[2] D. E. Bell and L. J. La Padula, “Secure computer system: Unified exposition and multics interpretation,” MITRE CORP, Hanscom AFB, Bedford, MA 01731, Tech. Rep. ESD-TR-75-306, 1976.

[3] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghighat, “A domain and type enforcement UNIX prototype,” *Computing Systems*, vol. 9, no. 1, pp. 47–83, 1996.

[4] R. N. Watson, “A decade of OS access-control extensibility,” *Communications of the ACM*, vol. 56, no. 2, pp. 52–63, 2013.

[5] “File permissions and attributes,” January 2018. [Online]. Available: https://wiki.archlinux.org/index.php/File_permissions_and_attributes

[6] R. A. Popa, E. Stark, J. Helfer, S. Valdez, N. Zeldovich, M. F. Kaashoek, and H. Balakrishnan, “Building web applications on top of encrypted data using Mylar,” in *NSDI*, 2014, pp. 157–172.

[7] L. Gong, “A secure identity-based capability system,” in *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on.* IEEE, 1989, pp. 56–63.

[8] T. Dierks and E. Rescorla, *The Transport Layer Security (TLS) Protocol, Version 1.2*, Network Working Group, August 2008.

[9] S. Khajuria, L. Sørensen, and K. E. Skouby, *Cybersecurity and Privacy-Bridging the Gap.* River Publishers, 2017.

[10] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, “The evolution of android malware and android analysis techniques,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, p. 76, 2017.

[11] J. McLean, “A comment on the ‘basic security theorem’of bell and lapadula,” *Information Processing Letters*, vol. 20, no. 2, pp. 67–70, 1985.

[12] T. Lin, “Bell and Lapadula axioms: a “new” paradigm for an “old” model,” in *Proceedings on the 1992-1993 workshop on New security paradigms.* ACM, 1993, pp. 82–93.

[13] E. Gureghian, T. Hardin, and M. Jaume, “A full formalisation of the Bell and Lapadula security model,” *Technical Report*, vol. 2003, no. 7, 2003.

[14] I. Goldberg, D. Wagner, R. Thomas, E. A. Brewer *et al.*, “A secure environment for untrusted helper applications: Confining the wily hacker,” in *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography*, vol. 6, 1996, pp. 1–1.

[15] N. Provos, “Improving host security with system call policies.” in *USENIX Security Symposium*, 2003, pp. 257–272.

[16] J. Morris, S. Smalley, and G. Kroah-Hartman, “Linux security modules: General security support for the linux kernel,” in *USENIX Security Symposium*, 2002.

[17] S. Heuser, A. Nadkarni, W. Enck, and A.-R. Sadeghi, “ASM: A programmable interface for extending android security,” in *USENIX Security Symposium*, 2014, pp. 1005–1019.

[18] R. N. Watson, J. Woodruff, P. G. Neumann, S. W. Moore, J. Anderson, D. Chisnall, N. Dave, B. Davis, K. Gudka, B. Laurie *et al.*, “Cheri: A hybrid capability-system architecture for scalable software compartmentalization,” in *Security and Privacy (SP), 2015 IEEE Symposium on.* IEEE, 2015, pp. 20–37.

[19] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, “Unikernels: Library operating systems for the cloud,” in *ACM SIGPLAN Notices*, vol. 48, no. 4. ACM, 2013, pp. 461–472.