Towards Adaptive Process Confinement Mechanisms

COMP5900I Literature Review

William Findlay

October 25, 2020

Abstract

[Come back hither when done.]

1 Introduction

Restricting unprivileged access to system resources has been a key focus of operating systems security research since the inception of the earliest timesharing computers in the late 1960s and early 1970s [cite]. In its earliest and simplest form, access control in operating systems meant preventing one user from interfering with or reading the data of another user. The natural choice for these early multi-user systems, such as Unix [23], was to build access control solutions centred around the user model—a design choice which has persisted in modern Unix-like operating systems such as Linux, OpenBSD, FreeBSD, and MacOS. Unfortunately, while user-centric permissions offer at least some protection from other users, they fail entirely to protect users from *themselves* or from their own *processes*. It was long ago recognized that finer granularity of protection is required to truly restrict a process to its desired functionality [20]. This is often referred to as *the process confinement problem* or *the sandboxing problem*.

1.1 The Process Confinement Threat Model

To understand why process confinement is a desirable goal in operating system security, we must first identify the credible threats to system stability and security that process confinement addresses. To that end, I first describe three attack vectors (items A1 to A3), followed by three attack goals (items G1 to G3) which highlight just a few of the credible threats posed by unconfined processes running on a given host.

A1. Compromised processes. Unconfined running processes have classically presented a valuable target for attacker exploitation. With the advent of the Internet, web-facing processes which handle untrusted user input are especially vulnerable, particularly as they often run with heightened privileges [cite]. The attacker may send specially crafted input to the target application, hoping to compromise its control flow integrity via a classic buffer overflow, return-oriented programming [cite], or some other means. The venerable Morris Worm, regarded as the first computer worm on the Internet, exploited precisely such a vulnerability in the `fingerd` service for Unix [cite].

A2. Semi-honest software. Here, I define semi-honest software as that which appears to perform its desired functionality, but which additionally may perform some set of unwanted actions without the user's knowledge. Without putting a proper, external confinement mechanism in place to restrict the behaviour of such an application, it may continue to perform the undesired actions ad infinitum, so long as it remains installed on the host. As a topical example, an `strace` of the popular Discord [cite] voice communication client on Linux reveals that it repeatedly scans the process tree and reports a list of *all applications* running on the system, even when the "display active game" feature is turned **off**. This represents a clear violation of the user's privacy expectations.

A3. Malicious software. **[Write this]**

G1. Installation of backdoors/rootkits. **[Write this]**

G2. Information leakage. **[Write this]**

G3. Tampering and denial of service. **[Write this]**

[Talk about how the internet has exacerbated this problem]

1.2 The Case for Adaptive Process Confinement

Despite decades of work since Lampson's first proposal of the process confinement problem in 1973 [20], the process confinement problem remains largely unsolved [5]. This begs the question as to whether our current techniques for process confinement are simply inadequate for dealing with an evolving technical and adversarial landscape. In this literature review, I present the status quo in process confinement, with an emphasis on Unix and modern Unix-like systems such as Linux. Further, I present a novel taxonomy, categorizing existing process confinement mechanisms into *maladaptive*, *semi-adaptive*, and *adaptive* solutions. Finally, I argue the case for the development and adoption of *adaptive process confinement mechanisms*.

Here, I define adaptive process confinement mechanisms as those which greatly help defenders confine their processes and are robust in the presence of attacker innovation. Roughly, this definition can be broken down into the following properties:

P1. Robustness to attacker innovation. **[Write this]**

P2. Low adoption effort. **[Write this]**

P3. High reconfigurability. **[Write this]**

P4. Transparency. **[Write this]**

P5. Usability. **[Write this]**

Ideally, an adaptive process confinement mechanism should have most—if not all—of the above properties.

1.3 Outline

The rest of this paper proceeds as follows. **[List sections and what is in them.]**

2 Traditional Process Confinement Approaches

3 Automating Policy Generation

4 Automating Policy Audit

5 Towards Truly Adaptive Process Confinement

5.1 Anomaly Detection Techniques

5.2 Extended BPF

6 Conclusion

References

[1] J. Anderson, "A Comparison of Unix Sandboxing Techniques," *FreeBSD Journal*, 2017. [Online]. Available: http://www.engr.mun.ca/~anderson/publications/2017/sandbox-comparison.pdf.

[2] AppArmor authors, *aa-easyprof*, Linux user's manual. [Online]. Available: https://manpages.ubuntu.com/manpages/precise/man8/aa-easyprof.8.html.

[3] A. Berman, V. Bourassa, and E. Selberg, "TRON: Process-Specific File Protection for the UNIX Operating System," in *Proceedings of the USENIX 1995 Technical Conference*, The USENIX Association, 1995, pp. 165–175. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.9149&rep=rep1&type=pdf.

[4] H. Chen, N. Li, and Z. Mao, "Analyzing and Comparing the Protection Quality of Security Enhanced Operating Systems," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2009. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2017/09/Chen.pdf.

[5] A. Crowell, B. H. Ng, E. Fernandes, and A. Prakash, "The Confinement Problem: 40 Years Later," *Journal of Information Processing Systems*, vol. 9, no. 2, pp. 189–204, 2013. DOI: 10.3745/JIPS.2013.9.2.189. [Online]. Available: http://jips-k.org/journals/jips/digital-library/manuscript/file/22579/JIPS-2013-9-2-189.pdf.

[6] L. Deshotels, R. Deaconescu, C. Carabas, I. Manda, W. Enck, M. Chiroiu, N. Li, and A.-R. Sadeghi, "iOracle: Automated Evaluation of Access Control Policies in iOS," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18, Incheon, Republic of Korea: Association for Computing Machinery, 2018, pp. 117–131, ISBN: 9781450355766. DOI: 10.1145/3196494.3196527.

[7] T. Eisenberg, D. Gries, J. Hartmanis, D. Holcomb, M. S. Lynn, and T. Santoro, "The Cornell Commission: On Morris and the Worm," *Communications of the ACM*, vol. 32, no. 6, pp. 706–709, 1989. DOI: 10.1145/63526.63530.

[8] W. Findlay, "Host-Based Anomaly Detection with Extended BPF," Honours Thesis, Carleton University, Apr. 2020. [Online]. Available: https://williamfindlay.com/written/thesis.pdf.

[9] W. Findlay, A. B. Somayaji, and D. Barrera, "bpfbox: Simple Precise Process Confinement with eBPF," in *Proceedings of the 2020 ACM Cloud Computing Security Workshop (CCSW'2020)*, To appear, Nov. 2020. DOI: 10.1145/3411495.3421358.

[10] M. Fleming, "A thorough introduction to eBPF," *LWN.net*, Dec. 2017. [Online]. Available: https://lwn.net/Articles/740157 (visited on 09/26/2020).

[11] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for Unix Processes," in *Proceedings 1996 IEEE Symposium on Security and Privacy*, May 1996, pp. 120–128. DOI: 10.1109/SECPRI.1996.502675.

[12] T. Gamble, "Implementing Execution Controls in Unix," in *Proceedings of the Seventh Large Installation System Administration Conference (LISA)*, The USENIX Association, 1993. [Online]. Available: https://www.usenix.org/legacy/publications/library/proceedings/lisa93/full_papers/gamble.pdf.

[13] G. Gheorghe and B. Crispo, "A Survey of Runtime Policy Enforcement Techniques and Implementations," University of Trento, Tech. Rep., 2011. [Online]. Available: http://eprints.biblio.unitn.it/2268/1/techRep477.pdf.

[14] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer, "A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker)," in *Proceedings of the Sixth USENIX UNIX Security Symposium*, The USENIX Association, 1996. [Online]. Available: https://www.usenix.org/legacy/publications/library/proceedings/sec96/full_papers/goldberg/goldberg.pdf.

[15] R. M. Graham, "Protection in an information processing utility," *Communications of the ACM*, vol. 11, no. 5, pp. 365–369, 1968, ISSN: 0001-0782. DOI: 10.1145/363095.363146.

[16] B. Gregg, *BPF Performance Tools*. Addison-Wesley Professional, 2019, ISBN: 0-13-655482-2.

[17] H. Inoue, "Anomaly detection in dynamic execution environments," Ph.D. dissertation, University of New Mexico, 2005. [Online]. Available: https://www.cs.unm.edu/~forrest/dissertations/inoue-dissertation.pdf.

[18] H. Inoue and S. Forrest, "Inferring Java Security Policies through Dynamic Sandboxing," in *International Conference on Programming Languages and Compilers (PLC'05)*, 2005. [Online]. Available: https://www.cs.unm.edu/~forrest/publications/inoue-plc-05.pdf.

[19] K. Jain and R. Sekar, "User-level infrastructure for system call interposition: A platform for intrusion detection and confinement," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2005.

[20] B. W. Lampson, "A Note on the Confinement Problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973, ISSN: 0001-0782. DOI: 10.1145/362375.362389.

[21] K. MacMillan, "Madison: A new approach to policy generation," in *SELinux Symposium*, vol. 7, 2007. [Online]. Available: http://selinuxsymposium.org/2007/papers/08-polgen.pdf.

[22] N. Provos, "Improving Host Security with System Call Policies," in *Proceedings of the 13th USENIX UNIX Security Symposium*, The USENIX Association, 2003. [Online]. Available: https://www.usenix.org/legacy/events/sec03/tech/full_papers/provos/provos_html.

[23] D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System," in *Proceedings of the Fourth ACM Symposium on Operating System Principles*, ser. SOSP '73, New York, NY, USA: Association for Computing Machinery, 1973, p. 27, ISBN: 9781450373746. DOI: 10.1145/800009.808045.

[24] Z. C. Schreuders, T. J. McGill, and C. Payne, "Towards Usable Application-Oriented Access Controls," in *International Journal of Information Security and Privacy*, vol. 6, 2012, pp. 57–76. DOI: 10.4018/jisp.2012010104.

[25] J. R. Smith, Y. Nakamura, and D. Walsh, *audit2allow*, Linux user's manual. [Online]. Available: http://linux.die.net/man/1/audit2allow.

[26] B. T. Sniffen, D. R. Harris, and J. D. Ramsdell, "Guided policy generation for application authors," in *SELinux Symposium*, 2006. [Online]. Available: http://gelit.ch/td/SELinux/Publications/Mitre_Tools.pdf.

[27] A. B. Somayaji, "Operating System Stability and Security through Process Homeostasis," Ph.D. dissertation, University of New Mexico, 2002. [Online]. Available: https://people.scs.carleton.ca/~soma/pubs/soma-diss.pdf.

[28] A. B. Somayaji and H. Inoue, "Lookahead Pairs and Full Sequences: A Tale of Two Anomaly Detection Methods," in *Proceedings of the 2nd Annual Symposium on Information Assurance Academic track of the 10th Annual 2007 NYS Cyber Security Conference*, NYS Cyber Security Conference, 2007, pp. 9–19. [Online]. Available: http://people.scs.carleton.ca/~soma/pubs/inoue-albany2007.pdf.

[29] D. A. Wagner, "Janus: An Approach for Confinement of Untrusted Applications," M.S. thesis, University of California, Berkeley, 1999. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/1999/CSD-99-1056.pdf.