CSE 231—Advanced Operating Systems
"Nooks"

Andrew Quinn

Summary. Nooks introduces a technique for improving the reliability of software systems by isolating and sandboxing the kernel extensions (i.e., modules) that are executed on the system [3]. Nooks targets the interesting but often unexplored set of errors that arise from misuse rather than malice. As a result, it provides a best-effort approach that limits the effect of misbehavior.

Discussion. This paper is the first paper that we will read that introduce some guardrails to prevent the effects of a software bug. It introduces a very compelling vision that has been applied in other domains [1]. Their results are extremely effective, with nearly all crashes averted using Nooks.

The two major issues that I have with the paper are around performance and manual effort. First, their system is extremely slow when used for many extensions. Perhaps they could have limited the extensions that they supported to only device drivers, since their performance was much better in these cases. The second issue was massive manual effort—almost 15k lines of code for wrappers! It seemed to me that much of wrapper generation could be automated (see, for example, modern RPC libraries that automate serialization), and I would love to see the limits of applying those approaches here.

After Nooks, the authors produced follow up work that introduced a shadow driver to make device driver failures transparent to applications—essentially, improving the reliability of not only commodity operating systems, but also the applications that run on commodity operating systems [2].

As mentioned above, sandboxing is an extremely important technique for applications today. It has been applied most effectively in web browsers

[1]although, many of the works that build similar sandboxes explicitly target malice in addition to misuse

as a means for isolating the data and logic of one web-page from another. Relevant systems include NaCl [4] and Embassies [1]. Sandboxing is a great was of limiting explicit flow from one privilege domain to another, but we're now increasingly asking "what about side-channels...".

References

[1] Jon Howell, Bryan Parno, and John R. Douceur. Embassies: Radically refactoring the web. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 529–545, Lombard, IL, April 2013. USENIX Association.

[2] Michael M. Swift, Muthukaruppan Annamalai, Brian N. Bershad, and Henry M. Levy. Recovering device drivers. In *6th Symposium on Operating Systems Design & Implementation (OSDI 04)*, San Francisco, CA, December 2004. USENIX Association.

[3] Michael M. Swift, Brian N. Bershad, and Henry M. Levy. Improving the reliability of commodity operating systems. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, page 207–222, New York, NY, USA, 2003. Association for Computing Machinery.

[4] Bennet Yee, David Sehr, Gregory Dardyk, J. Bradley Chen, Robert Muth, Tavis Ormandy, Shiki Okasaka, Neha Narula, and Nicholas Fullagar. Native client: A sandbox for portable, untrusted x86 native code. In *2009 30th IEEE Symposium on Security and Privacy*, pages 79–93, 2009.