SECURITY CONCEPTS OF SELECTED OS KERNELS
KICKOFF

Oliver Braunsdorf | Julian Horsch | Sascha Wessel

10.07.2020

Fraunhofer
AISEC

AGENDA

1. Organisatorisches
2. Zeitplan
3. Themen

ORGANISATORISCHES

- Das Seminar wird in Deutsch gehalten
- Report-Dokument
  - In Form eines wissenschaftl. Papers
  - Sprache: Englisch
  - Genau 8 Seiten (exkl. Referenzen)
  - Geschrieben in LATEX: acmart Template (https://ctan.org/pkg/acmart)
- Review
  - Erstellung von Reviews für 2 Paper von anderen Teilnehmern
  - ~1/2 Seite
  - Auf Basis der Reviews für eigenes Paper: Rebuttal + "Camera-Ready"-Version
- Präsentation
  - 20 Minuten Präsentation
  - ~10 Minuten Diskussion
  - Slide-Sprache: Englisch
  - Vortragssprache: Deutsch/Englisch
- Bewertung
  - "Camera-Ready" Paper
  - Review
  - Präsentation + Slides
  - Aktive Mitarbeit bei Diskussion

ZEITPLAN

- 10.07.2020 Vorbesprechung (heute)
- 15.07.2020 Themenwunsch oder ggf. eigene Themenvorschläge senden
- 16.07-27.07.2020 Registrierung für dieses Seminar in TUM-Matchingsystem
- 31.07.2020 Themenzuweisung (nachdem Ergebnisse aus TUM-Matching verfügbar)
- 15.10.2020 Einreichen der Gliederung + erste stichpunktartige Inhalte
- 29.11.2020 Einreichen des fertigen Papers
- 01.12.2020 Verteilung der Paper für Reviews
- 10.12.2020 Einreichen der Reviews
- 20.12.2020 Einreichen von Rebuttal + “Camera-Ready” Version
- 10.01.2020 Einreichen der Präsentations-Slides
- 13.+14.01.2021 (vorläufig) Abschlusspräsentationen und Diskussion

THEMEN

- Folgende Kernel/Betriebssysteme stehen zur Auswahl
  - Verve & Singularity
  - LittleKernel & Zircon
  - RedoxOS
  - seL4/Fiasco
  - BSD
  - XNU
  - ReactOS / Windows
- Vorschläge für andere Kernel, bitte bis 15.07. per Mail an <oliver.braunsdorf@aisec.fraunhofer.de>
  - Sollte kurzes Motivationsschreiben enthalten: Warum möchtest du genau diesen Kernel/OS untersuchen und welche spezifischen (Security-)Features sollen dabei betrachtet werden?

Inhaltliche Orientierungspunkte I

- Ziel des Seminars: Erarbeitung und Vermittlung der (Sicherheits-)Konzepte moderner Kernels/OS
  - Welche Abstraktionen und Mechanismen werden umgesetzt?
  - Welche Eigenschaften / Implikationen ergeben sich daraus?
  - Wie unterscheiden sie sich von anderen Kerneln, die in diesem Seminar vorgestellt werden?
- Linux als Basis. Betriebssystemkonzepte den meisten bekannt. => Unterschiede u. neue Ansätze herausstellen.

- Orientierungspunkte
  - Allgemeine Infos & "USPs"
  - **Architektur**
  - **Bewertung der Sicherheit:** bekannte Sicherheitslücken, Codesize, Validierungsgrad (formal verifiziert / auditiert / regelmäßiges Fuzzing / nicht getestet), Vorgehen bei Zero Days, CVEs
  - **Abstraktionen und Sicherheitsmechanismen**
    - Isolation (Prozesse/Tasks/.., Treiber, Privileged/Unprivileged Execution)
    - Access Control
    - Spezifische IPC-Umsetzung im Kernel
    - Access Control
    - Ressourcenverwaltung
    - Compilerbasierte Features (Shadow Stack, CFI)
    - Funktionale Sicherheitsfeatures (Crypto Subsystem, Integrity Protection)

Inhaltliche Orientierungspunkte II

- Weitere interessante Aspekte
  - Memory Management
  - Interrupt Handling
  - Übersicht System Calls
  - Ausgleich des Overheads für häufige Context-Switches in Microkernels
  - Standard Libraries & unterstützte Programmiersprachen
  - Ausblick: Was sind die aktuellen / kommenden Entwicklungen
- Möglicher praktischer Coding-Anteil:
  - Ggf. für neuere Kernels mit spärlicher Dokumentation
  - Zeigen wie man den Kernel erweitern kann
  - Stubs für neues Kernel-Komponente oder Treiber
  - Demonstration des IPC-Mechanismus in Microkernel
  - Erfahrungen im Report zusammenfassen

Microsoft's Research-Kernels: Verve & Singularity

- Singularity (2003) und Verve (2010) bei Microsoft Research entwickelt
- Betriebssysteme/Kernels für Memory Safety und Type Safety
- Interessante Aspekte: u.a. Programming Language-based / Compiler-based Security, Memory-Layout, SMT-Solving

- Yang, J., & Hawblitzel, C. (2010). Safe to the last instruction: Automated verification of a type-safe operating system. https://doi.org/10.1145/1806596.1806610
- https://channel9.msdn.com/Shows/Going+Deep/Verve-A-Type-Safe-Operating-System
- https://www.microsoft.com/en-us/research/project/singularity/?from=https%3A%2F%2Fresearch.microsoft.com%2Fos%2Fsingularity%2F#!publications
- Source Code für Verve u. Singularity: https://archive.codeplex.com/?p=singularity

LittleKernel & Zircon

- Zircon ist der Microkernel für Google's Fuchsia OS
- Basiert auf LittleKernel
- LittleKernel ist der Bootloader für Android und ist Teil der Trusty TEE
- Thema kann LittleKernel und Zircon betrachten
- Interessante Aspekte: Kernel-Objects, Handles + Rights, Garbage Collection

- https://fuchsia.dev/fuchsia-src/concepts/kernel
- https://github.com/littlekernel/lk/wiki
- https://fuchsia.googlesource.com/fuchsia/+/master/zircon
- https://fuchsia.dev/fuchsia-src/reference/

RedoxOS

- OS in Rust geschrieben, Kernel in Rust and x86-Assembly
- Architektur stark von Minix beeinflusst
- Interessante Fragestellungen:
  - Welche Security/Isolations-Mechanismen konnten mithilfe von Rust's Typsystem implementiert werden?
  - Wie ausgeprägt ist die Benutzung von "safe" Rust? An welchen Stellen muss "unsafe" Rust verwendet werden?

- https://www.redox-os.org/
- https://doc.redox-os.org/book/
- https://gitlab.redox-os.org/redox-os/kernel

seL4

- Formal Verifizierter Micro-Kernel basierend auf L4
- Interessante Aspekte: Verifikationsverfahren, Capability-System

- https://sel4.systems/About/seL4-whitepaper.pdf
- seL4: formal verification of an OS kernel https://dl.acm.org/doi/10.1145/1629575.1629596
- From L3 to seL4 What Have We Learnt in 20 Years of L4 Microkernels https://dl.acm.org/doi/pdf/10.1145/2517349.2522720

BSD

- Unix-basiertes OS & Kernel
- Lange Historie mit vielen Forks
- OpenBSD für Sicherheitsfokus bekannt
- Interessante Aspekte:
  - Adaption für CHERI-Architektur https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/cheribsd.html
  - Jails https://www.freebsd.org/doc/de_DE.ISO8859-1/books/handbook/jails.html
- http://wiki.netbsd.org/security/
- https://papers.freebsd.org/2019/bsdcan/stanek-improving_security_of_the_freebsd_boot_process/
- https://papers.freebsd.org/2020/linux.conf.au/paeps_improving_freebsd_security_process/
- https://papers.freebsd.org/2019/eurobsdcon/duleba-improving_security_of_freebsd_with_tpm_20_and_intel_sgx/
- [BUCH] Absolute **FreeBSD**: the complete guide to **FreeBSD**
- https://www.freebsd.org/doc/en/books/arch-handbook/

XNU

- Kernel der Apple-Plattformen: macOS, iOS, tvOS, iPadOS, etc.
- Hybridkernel: Mach + BSD
- Bildet zusammen mit BSD-Userland Darwin Betriebssystem ➔ Basis für macOS, iOS, etc.
- XNU Source Code für macOS: https://opensource.apple.com/source/xnu/
- Interessante Sicherheitsthemen: Sandbox/TrustedBSD MAC, User-space Drivers, System Integrity Protection (SIP), …
- Architekturbeschreibung: https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/Architecture/Architecture.html

ReactOS / Windows Kernel

- ReactOS: „Drop-in“-Replacement für Windows
- Bildet Windows-System nach und versucht Kompatibilität mit Windows-Anwendungen herzustellen
- Kernel bildet Windows NT Kernel nach
- Userland (Libraries etc.) mit Hilfe von Wine-Code realisiert
- Thema kann ReactOS-Kernel und Windows-Kernel betrachten

Q&A

Kontaktinformationen

Oliver Braunsdorf, Julian Horsch, Sascha Wessel

Abteilung Secure Operating Systems

Fraunhofer-Institut für Angewandte und Integrierte Sicherheit AISEC

Fraunhofer Institute AISEC
Lichtenbergstr. 11
85748 Garching b. München

https://www.aisec.fraunhofer.de/de/jobs/hiwi-stellen.html

E-Mail: oliver.braunsdorf@aisec.fraunhofer.de