Deliverable 1

eBPF-Compatible Mobile Device Environments

Name: Ramya Radjesh

Supervisor: Nicolas Dejon

Table of Contents

1 Mobile Operating System : *3*

1.1 Android 3

1.2 iOS 6

1.3 KaiOS 6

1.4 HarmonyOS 7

1.5 Tizen OS 8

1.6 Pure OS 9

1.7 Postmarket OS 9

1.8 Sailfish OS 10

1.9 Fuchsia 11

1.10 Ubuntu Touch 11

2 eBPF in Mobile Operating Systems *12*

3 eBPF in cloud environment *13*

4 eBPF functions- real time environment (not OS) *14*

5 eBPF in Windows *14*

5.1 Microsoft plan in eBPF 15

Bibliography *16*

1 Mobile Operating System :

- Mobile operating system (Mobile OS) is a software platform on top of which other programs called application programs, can run on mobile devices such as personal digital assistant (PDA), tablets, cellular phones, smartphones and so on.
- Software that allows smartphones, tablets, and other devices to run applications and programs.
- Mobile operating systems also provide application stores, which enable users to download and interface with mobile applications.
- Several mobile operating systems also have native GPS applications that allow users to search for locations, follow step-by-step directions and, in some cases, share locations with different devices.

Most widely used Operating Systems are:

1. Android
2. iOS
3. HarmonyOS
4. KaiOS

Methodology:

- To perform this research, I investigated various mobile operating systems and their architecture (both linux and non-linux architecture) to find out the possible usecase of using eBPF in Future.
- The details are gathered based on the resources that are available through research paper, articles, community posts and OpenAI.

1.1 Android

- Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework service.
- Advantage of android: device compatibility and opportunities for customization. Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.
- eBPF in Android: eBPF provides a framework for granular network traffic monitoring within mobile operating systems like Android (20). eBPF in Android was for measuring network traffic statistics.

Android devices running on kernel 4.9 or above and originally shipped with the P release MUST use eBPF-based network traffic monitoring (20)

accounting instead of xt_qtaguid (deprecated Linux kernel module previously used on Android for attributing network data usage to specific applications).

- Every Android phone uses eBPF to monitor traffic. Every single packet that goes in and out of a Meta datacenter is touched by eBPF (27).

Kernel Layer Functionalities

- Monolithic (Linux kernel)
- The Linux kernel (21) is the core of the Android operating system, providing essential services like hardware abstraction, memory management, process management, and device driver support.
- It allows Android to work across various hardware configurations, manages system memory efficiently, handles multitasking, and supports device drivers.
- The Hardware Abstraction Layer (HAL) bridges hardware-specific device drivers with the higher-level Android framework, ensuring hardware-agnostic operation on various hardware platforms.

Platform security architecture

Android seeks to be the most secure and usable operating system for mobile platforms by repurposing traditional operating system security controls to:

- Protect app and user data.
- Protect system resources (including the network)
- Provide app isolation from the system, other apps, and from the user.

To achieve these objectives, Android provides these key security features:

- Robust security at the OS level through the Linux kernel
- Mandatory app sandbox for all apps
- Secure interprocess communication
- App signing
- App-defined and user-granted permissions

Android Security Derivatives:

- eBPF in android security: Network usage monitoring, Power and memory profiling, Firewall, Intrusion detection, Kernel security monitoring.
- At the operating system level, the Android platform provides the security features of the Linux kernel and provides a secure inter-process communication (IPC) mechanism for secure communication between applications running in different processes.
- Android kernel security features:
  HWAddressSanitizer, KASAN, Top-Byte Ignore, KCFI, ShadowCallStack

- eBPF Traffic Monitoring (20): The eBPF network traffic tool in Android, which provides UID-based networking traffic blocking as a replacement for the xt_owner module inside the kernel. The feature can be configured using:

- traffic_powersave_uid_map - configures network traffic blocks for UIDs in power-saving mode, a battery optimization feature on Android devices that limits performance and background data usage.
- traffic_standby_uid_map - configure the UIDs for which network traffic should be blocked when the device is in standby mode. Standby mode is a state in which the device is idle and not actively used, with the screen turned off. o traffic_dozable_uid_map - configure the UIDs for which network traffic should be blocked when the device is in Doze mode. Doze mode is an Android feature that further optimizes battery life by restricting network access and background processes when the device is not being used for an extended period.

- BPFroid (37): BPFroid is an Android dynamic analysis framework that uses eBPF technology to continuously monitor user application events in the real device.
  BPFroid traces all layer to keep in track of malware detection.
  Layers:
- API method invocation • Native librairies function calls
- Direct system call.
- Internal kernel function calls
- BPFroid leverages the fact that Android framework is generally compiled under ART runtime (to gain better runtime).
- Sifter: Its key approach is the use of fine-grained, highly selective filters to reduce the attack surface of these kernel modules and make their vulnerabilities unreachable for untrusted programs.
- Goal: Sifter is used to protect security-critical kernel modules in Android.
- Sifter is motivated by two simple observations. First, syscalls needed to interact with a kernel module follow a rich set of patterns, encoded in the module itself, often through a set of ioctl syscalls. Second, to exploit a vulnerability in a module, malware often issues syscalls with meticulously crafted arguments and sequences, which are not normally used by legitimate programs.

The main uses of eBPF on Android include (36):

- Data usage statistics and billing.
- Firewall/network restrictions for energy saving when entering power-saving mode.
- High-speed packet processing, such as shared network connections or 464xlat (a method of providing IPv4 connections over an IPv6 network since the Linux kernel itself does not support it).

Programmed in: C, C++, Java, Kotlin

OS family: Modified Linux kernel based.

Real Time device: Samsung, Google, Huawei, Xiaomi, OnePlus, Sony, LG, Motorola.

Evidence of eBPF: Found

1.2 iOS

- Unix-based mobile operating system. iOS is an intermediary between hardware and mobile applications. Instead of apps directly interacting with hardware, they connect via APIs, simplifying the creation of applications adaptable to different hardware specification.
- The heart of iOS lies the Core Framework, forming the base for other technologies and offering low-level functionality.
- eBPF in Mac: They have the compatibility issue with eBPF as Darwin was not supporting. But after setting up lima to create a linux virtual machine on the mac M1, M2 and configuring PyCharm helps running eBPF in Mac (3).

Kernel Layer Functionalities

- The iOS kernel is the XNU kernel of Darwin.
- Hybrid (XNU kernel combines monolithic and microkernel approaches)
- It facilitates efficient system resource management, including memory, processes, and file systems, while also providing a secure and stable environment for application execution.
- XNU's I/O Kit allows for a modular approach to developing device drivers, making it easier to support a wide array of hardware peripherals.

Programmed in: C, C++, Objective-C, Swift.

OS Family: Darwin

Real Time device: iPhone.

Potential eBPF usecase: none (Possible - through lima and Pycharm)

Future Possibility: Yes *(Link)*

Evidence of eBPF: Not Found

1.3 KaiOS

- KaiOS is a mobile operating system based on HTML5 and other web technologies like JavaScript.
- It supports 4G/LTE, GPS, Wi-Fi, NFC, and features the KaiStore, which is the first app store ever on feature phones.

- KaiOS uses Firefox v48 (in the latest stable release v2.2.0) to run web applications, basically using it as its JS runtime and rendering engine on top of a Linux kernel.
- The KaiOS developer ecosystem includes resources such as simulators and GitHub repositories to assist with app development and testing. KaiOS features

the KaiStore, which serves as an app store for feature phones, offering applications from various developers, including major companies like Facebook and Google
(1).

Linux Kernel Functionalities

- Monolithic (Linux kernel)
- 64-bit Kernel and binaries -Networking support -OpenSSH client and server support udhcpc support (Busybox DHCP Client) (23).
- Gonk's Linux Kernel is derived from the Android Open-Source Project (AOSP) with additional modifications for KaiOS.
- Manages system calls and hardware interrupts in a manner consistent with traditional UNIX-like operating systems.

Programmed in: HTML5, JavaScript

OS family: Firefox OS / Open Web (based on Linux kernel)

Real time device: Nokia 8110 4G, JioPhone, JioPhone 2, Alcatel Go Flip (and various regional variants), CAT B35, Doro 7050 and 7060.

Potential eBPF usecase: basic networking or security.

Evidence of eBPF: Not Found

Future possibilities: Not Found

1.4 HarmonyOS

- It has a multikernel design with dual frameworks: the operating system selects suitable kernels from the abstraction layer in the case of devices that use diverse resources.
- HarmonyOS is designed with a layered architecture, which consists of four layers.
- HarmonyOS uses a microkernel that aims to provide enhanced security and low latency.
- HarmonyOS could leverage eBPF to virtualize network functions, allowing for the dynamic creation, scaling, and management of networking services.
- It can be tailored to fit the resource constraints and capabilities of different devices, from IoT devices to smartphones.

Linux Kernel Functionalities

- Kernel Layer Functionalities: HarmonyOS (22) uses a multi-kernel design so that appropriate OS kernels can be selected for devices with different resource limitations.
- Microkernel (HarmonyOS 2.0 and later), Monolithic (Linux kernel for HarmonyOS 1.x)

- The kernel abstraction layer (KAL) shields differences in kernel implementations and provides the upper layer with basic kernel capabilities, including process and thread management, memory management, file system, network management, and peripheral management.

Programmed in: C, C++, Java and ArkTS (developed by Huawei)

OS family: Unix-like (modified AOSP) and LiteOS on OpenHarmony until HarmonyOS NEXT

Real Time device: Huawei P30 series, Huawei P40 series, Huawei Mate 30 series, Huawei Mate 40 series, Huawei P50 series, Huawei Mate X2

Potential eBPF usecase: Network function virtualization (NFV)

Evidence of eBPF: Not Found

Future possibilities: Not Found

1.5 Tizen OS

- Tizen Mobile profile provides an effortless mobile experience with Tizen OS and a clutter-free, intuitive interface making it easy to understand and use.
- Tizen is a linux -based mobile operating system backed by the Linux Foundation, developed, and used primarily by Samsung and Intel. Tizen is faster during startup as it's lighter in weight.
- eBPF could be used to enhance Tizen's IoT capabilities by providing real-time data processing and decision-making at the edge.
- Tizen comes with a 64-bit processor that is not yet offered by Android TVs.
- In Tizen 4.0 Public M2 release D-Bus debugging and profiling tools (eBPF-based monitoring) has been added (14).

Linux Kernel Fucntionalities

- Monolithic (Linux kernel)
- Tizen OS utilizes the Linux kernel, which provides a stable and secure foundation for the operating system.
- Provides efficient resource management for processes, memory, and file systems, optimizing performance for both high-end and resourceconstrained devices.

OS family: Linux (based on a combination of Linux MeeGo and Samsung Bada)

Programmed in C++, Xamarin.Forms (.NET C#, F#, VB)

Real Time device: Samsung Z series.

Potential eBPF usecase: Network Security and Traffic Analysis

Evidence of eBPF: Not Found

Future possibilities: YES(Link)

1.6 Pure OS

- PureOS (4) is based on the Debian operating system, the most popular GNU+Linux OS in the world, powering everything from a Raspberry Pi to the world's biggest datacenters and supercomputers.
- PureOS strives for the strictest of security and privacy protection. It is maintained by Purism for use in the company's Librem laptop computers as well as the Librem 5 smartphone (eBPF - particularly relevant for the Librem 5 smartphone, where data privacy is a key selling point).
- Purism didn't develop most of the code that goes into PureOS. Most of it is the same code that goes into most versions of Linux, just packaged, and distributed in different ways.

Linux kernel Functionalities

- Monolithic (Linux kernel)
- PureOS (24) is one of a few strict GNU/Linux operating systems, meaning it does NOT include any non-free, proprietary software and/or drivers/firmware (aka binary blobs).
- PureOS (24) is based upon the most private and secure foundation in computing, the Linux kernel, that largely eliminates the need for antivirus and malware protection that plague Windows and OSX.

Programmed in C

OS family: Linux (based on Debian)

Real-Time device: Librem 5

Potential eBPF usecase: Privacy Enforcement

Evidence of eBPF: Not Found

Future possibilities: Not Found

1.7 Postmarket OS

- PostmarketOS (5) is an operating system primarily for smartphones, based on the Alpine Linux distribution. postmarketOS (abbreviated as pmOS) is an operating system primarily for smartphones, based on the Alpine Linux distribution.
- Alpine Linux was chosen as the base distribution due to its low storage requirements, making it more suitable for older devices. Excluding the kernel, a base installation takes up approximately 6 MB.

Linux Kernel Functionalities :

- Monolithic (Linux kernel)
- The kernel driver core creates device nodes for all registered devices in that filesystem (25).
- PostmarketOS (pmOS) is built upon a lightweight Alpine Linux, which is known for its simplicity, security, and resource efficiency.
- Prioritizes privacy and security by offering a minimal base installation that reduces potential attack vectors and by supporting encrypted installations.

Programmed in Python install tool and shell script packages.

OS family: Unix-like

Real Time device: Pine64-pinephone, Purism-librem5, qemu-aarch64, qeumamd64, qemu-riscv64.

Potential eBPF usecase: Real-Time System Monitoring for Mobile Devices

Evidence of eBPF: Not Found

Future possibilities: Not Found

1.8 Sailfish OS

- Sailfish OS (7) is a Linux-based operating system based on free software, and opensource projects such as Mer as well as including a closed source UI.
- The Sailfish OS and the Sailfish software development kit (SDK) are based on the Linux kernel and Mer.
- Sailfish OS includes a multi-tasking graphical shell called "Lipstick" built with Qt by Jolla on top of the Wayland display server protocol.

Linux Kernel Functionalities

- Monolithic (Linux kernel)
- Support for ARM and Intel architectures, including the Intel Atom x3 processor, or any platform with kernel useable (settle-able) for MER core stack (also called middleware of Sailfish).

OS family: Linux (Unix-like)

Programmed in C++, QT framework.

Real Time device: Jolla Phone (original device by the creators of Sailfish OS), Jolla C, Jolla Tablet, Sony Xperia X, XA2, 10, and 10 II.

Potential eBPF usecase: Enhanced Mobile Network Security and Traffic Management

Evidence of eBPF: Not Found

Future possibilities: Not Found

1.9 Fuchsia

- Fuschia - Google's next Android, with a low latency rendering pipeline for the next generation of mobile devices (30)."
- Created by google's android and chrome OS. Based on kernel but not linux. Fuchsia is designed to power a diverse ecosystem of hardware and software. Fuchsia uses a new microkernel called Zircon.
- Fuchsia is designed to prioritize security, updatable components, and performance.
- There are two ways to work with Fuchsia: you can build products and software for Fuchsia using the SDK or contribute to the source tree to build the fuchsia platform.

Kernel Functionalities - Zircon (31):

- Microkernel (Zircon Kernel)
- The Zircon Kernel provides syscalls to manage processes, threads, virtual memory, inter-process communication, waiting on object state changes, and locking (via futexes).
- Zircon is an object-based kernel. User mode code almost exclusively interacts with OS resources via object handles. A handle can be thought of as an active session with a specific OS subsystem scoped to a particular resource.
- Zircon actively manages the following resources: processor time, memory, and address spaces, device-io memory, interrupts, signaling and waiting, inter-process communication.

Programmed in C++, Rust, and Dart.

OS Family: Not a part of traditional OS, based on Microkernel called ZIRCON. *Real time devices*: In January 2023, Google announced layoffs across the company with 16% of fuchsia employees being impacted. In May 2023, Google began rolling out a Fuchsia-based update to the second-generation Google Nest Hub (32). Not Officially declared by google (29).

Potential eBPF usecase: Capability-Based Security and Isolation and Sandboxing

Evidance of eBPF: Not Found

Future possibilities: Not Found

1.10 Ubuntu Touch

- Ubuntu Touch is a mobile version of the Ubuntu operating system, being developed by the UBports community.
- Ubuntu for Android was a variant of Ubuntu designed to run on Android phones.

- Ubuntu Mobile Internet Device Edition is a discontinued Ubuntu distribution planned to run on the Intel Mobile Internet Device platform, x86 mobile computers based on the Intel Atom processor.
- Ubuntu Touch can run on dozens of smartphones that originally shipped with Android (34).
- Ubuntu Touch uses Halium to communicate with the hardware using Android drivers.

Linux Kernel Functionalities (35):

- Ubuntu Touch consist of the Linux kernel for this device plus a minimal Android system that is used to enable all the hardware.
- There are three groups based on how their kernel and hardware abstraction is implemented:
    1. Android 5.1 based ports: These ports use a device-specific fork of the Linux kernel, heavily modified for the hardware, along with a minimal Android system for hardware enablement.
    2. Halium based ports: These ports also use a device-specific Linux kernel fork and a minimal Android system but are based on a more generic approach that allows for shared work across projects. Halium ports are based on newer Android versions (7.1 and above).
    3. Linux based ports: These refer to devices that use a pure Linux kernel without any Android components. This category includes devices like the Pinephone, Pinetab, and Raspberry Pi.

Programmed in QML, or the Qt Meta-Object Language

OS Family Ubuntu, Linux

Real time devices: Nexus 4, Nexus 7 2013 WiFi, BQ Aquaris E4.5, BQ Aquaris E5, BQ Aquaris M10, Meizu MX4, Meizu PRO 5.

Potential eBPF usecase: Enhanced Security

Evidance of eBPF: YES (Link)

Future possibilities: Linux scheduler implementations using sched-ext for eBPFloaded scheduler implementations.

2 eBPF in Mobile Operating Systems

- eBPF provides improved performance, simplified operations, and complete visibility for telecommunications.
- It provides additional functionality such as socket tagging, separating foreground/background traffic and per-UID firewall to block apps from network access depending on phone state (20).
- Usecases: eBPF helps mobile operating system in the following ways:

1. Performance Optimization (XDP and TC): eBPF, with technologies like XDP (eXpress Data Path) and TC (Traffic Control), provides improved performance, simplified operations, and complete visibility for telecommunications.

2. Security Enhancement (Socket Filters and LSM): eBPF is not directly used within mobile operating systems, but it can enhance security by implementing policies and monitoring system calls through socket filters and Linux Security Modules (LSM) to prevent unauthorized access or detect malicious activity.

3. Networking packet hits eBPF:

    - Data usage and and accounting
    - Firewalling: networking restriction for power saving
    - High speed packet processing (464xlat, tethering)
    - Network tracing via eBPF ringbuffers
    - TCP/UDP port reservations, DSCP remarking.

4. Performance Monitoring (BPF Maps and Perf Events): eBPF facilitates the collection of detailed performance metrics for system and application optimization using BPF maps for data storage and perf events for event-based monitoring.

5. Advanced Tracing and Debugging (Uprobes and Tracepoints): eBPF offers advanced tracing capabilities for debugging system and application issues with minimal overhead using uprobes (user-space probes) and tracepoints.

6. Kernel-Level Sandboxing (LSM and BPF Maps): eBPF enhances app sandboxing by enforcing additional security policies at the kernel level, utilizing LSM for security checks and BPF maps for policy enforcement.

3 eBPF in cloud environment

- Mobile OS often interact with cloud environment.
    - Android: Google cloud native services, Google Drive, Google Photos, Google docs. Android devices uses google cloud messaging and firebase cloud messaging.
    - iOS: iCloud for storage, backup, and synchronization of data.
- Goal: non-intrusive observability and security monitoring within cloud native environments.
- eBPF will become the new layer in the new cloud native infrastructure stack, impacting the observability, performance, reliability, networking, and security of all applications, supporters says (27).
- eBPF Cloud Tools: Falco, Hubble, Deepflow, Kindling, Apache SkyWalking, LoxiLB.

- New Concept is coming soon in Probe based operation: Linux Kernel Component Cloud Builder (LKCCB) - LKCCB is an automated system that determines structure offsets for every kernel version and distribution we want to run our BPF probes on.
- Kernel offset is loaded into probes at runtime.
- These Probes will be able to check the loaded offset and use them to navigate through kernel DS.

4 eBPF functions- real time environment (not OS)

Company	eBPF Functionalities
Meta (Facebook)	Layer 4 load balancing for facebook.com
Google	Networking in GKE, BPF LSM for security
Cloudflare	Load balancing, DDoS protection, security
Netflix	Network observability, performance diagnosis
Dropbox	Layer 4 load balancing with Katran
Microsoft	Ported eBPF and XDP to Windows
Apple	Kubernetes Pod security
Amazon	(Assumed usage in AWS infrastructure)

5 eBPF in Windows

- Goal: Enhance Programmability, extensibility, and agility benefits of eBPF. Windows can use eBPF stack for eBPF based monitoring environment, for testing code running on Windows desktop or production on windows servers or in Azure (19). eBPF could bring performance enhancements to Windows, like those seen with SmartNICs and DASH on Linux.
- This project is a work-in-progress that allows existing eBPF toolchains and APIs familiar in the Linux ecosystem to be used on top of Windows (17).
- Windows uses drivers and Linux uses kernel modules and Api's rather than system call, so eBPF is implemented in slightly different way (18). (eBPF Bytecode -> PREVAIL Verifier (running in user mode) -> if Selected-> code compiled to native by the open source uBPF JIT Compiler).
- Verified eBPF code is either JIT compiled or interpreted in the kernel mode, running in the Windows driver ebpfcore.sys.
- eBPF for Windows enables complex verifier functions to run safely without impacting other applications and services.

5.1 Microsoft plan in eBPF

- Microsoft plans to enable anything that you can call as a public API and write a driver in Windows that's relevant to eBPF to be exposed to eBPF. (18 – Passage : Cross Platform Code).
- Microsoft's work with eBPF is "paving the way for other operating systems" such as BSD and Apple's OS X, to use eBPF (26).
- Microsoft's Jowett says standardization in the eBPF ecosystem around instruction sets is key to ensuring that eBPF programs function the same in all environments, which will make it easier to port applications back and forth (26).
- Higher-level tools like the L3AF lifecycle management and orchestration project for eBPF networking apps (including load-balancing, rate limiting, traffic mirroring, flow exporter, packet manipulation and performance tuning) that Walmart contributed to the LF Networking group are also currently only for Linux; again, Microsoft is working with the L3AF community to bring support for Windows (18).
- uBPF: uBPF is catching up with kernel developments, particularly with contributions from Microsoft for its eBPF Windows implementation (28).

Bibliography

1. Arif, H. (Apr 9, 2021). *KaiOS: Everything you need to know!* From Medium: https://medium.com/proximity-works/kaios-everything-you-need-to-know-383bd0c3d081
2. Touloupas, H. (Jun 22, 2023). *When Mac M1/M2 Met eBPF: A Tale of Compatibility*. From Medium: https://medium.com/@harry-touloupas/when-mac-m1-m2-met-ebpf-a-tale-of-compatibility-6b9a6bc53f3e
3. KING, B. (OCT 4, 2022). *Make us of*. From What Is PureOS? A Look at Purism's OS for Laptops and Phones: https://www.makeuseof.com/what-is-pureos-purism/
4. Purism. (n.d.). *PureOS – a pure Linux phone experience*. From Purism: https://puri.sm/products/librem-5/pureos-mobile/
5. PostmarketOS. (n.d.). *postmarketOS*. From Wikiwand: https://www.wikiwand.com/en/PostmarketOS
6. PostmarketOS. (n.d.). *PostmarketOS*. From PostmarketOS: https://postmarketos.org
7. Wikipedia. (n.d.). *Sailfish OS*. From Wikipedia: https://en.wikipedia.org/wiki/Sailfish_OS
8. Mulligan, B. (Aug 11th, 2023). *The New Stack*. From Performant and Programmable Telco Networking with eBPF: https://thenewstack.io/performant-and-programmable-telco-networking-with-ebpf/
9. DAVID SOLDANI 1, (. M. (13 June 2023). eBPF: A New Approach to Cloud-Native Observability, Networking and Security for Current (5G) and Future Mobile Networks (6G and Beyond). *IEEE Access*, 29.
10. Author PictureMarcos A. M. Vieira, A. P. (06 February 2020). Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications. *ACM* .
11. David Barrera, A. S. (November 9, 2020). bpfbox: Simple Precise Process Confinement with eBPF. 13.
12. Wikipedia. (October 31, 2023). *Wikipedia*. From Tizen: https://en.wikipedia.org/wiki/Tizen
13. Tizen. (Nov. 1, 2017). *Tizen*. From Tizen: https://developer.tizen.org/tizen/release-notes/tizen-4.0-public-m2
14. Mishra, Y. (October 3, 2022). *EULERNew OpenEuler 22.09 version support better features between Euler and HarmonyOS*. From EULER: https://www.huaweicentral.com/new-openeuler-22-09-version-support-better-features-between-euler-and-harmonyos/
15. yunwei37. (Oct 10, 2023). *Userspace eBPF Runtimes: Overview and Applications*. From Medium: https://medium.com/@yunwei356/userspace-ebpf-runtimes-overview-and-applications-d19e3c84c7a7#:~:text=The%20benefits%20of%20using%20eBPF,without%20modifying%20their%20core%20code.
16. Microsoft. (n.d.). *eBPF for Windows*. From Github: https://github.com/microsoft/ebpf-for-windows?tab=readme-ov-file
17. Branscombe, M. (Feb 8th, 2022). *Microsoft Brings eBPF to Windows*. From The News Stack: https://thenewstack.io/microsoft-brings-ebpf-to-windows/

18. YANAKIEVA, N. (16 JULY, 2021). *DevStyleR*. From PROGRAMMING THE WINDOWS KERNEL WITH EBPF: https://devstyler.io/blog/2021/07/16/programming-the-windows-kernel-with-ebpf/
19. Source, G. (n.d.). *eBPF traffic monitoring*. From Git Source: https://source.android.com/docs/core/data/ebpf-traffic-monitor
20. Subramanian, A. (Sep 18, 2023). *Exploring the Layers: A deep dive into Android OS Architecture*. From Medium: https://medium.com/@ayyappansubramanian77/exploring-the-layers-a-deep-dive-into-android-os-architecture-31a2cd7a4036#:~:text=Key%20functions%20of%20the%20Linux,allocation%20and%20deallocation%20of%20resources.
21. HarmonyOS. (2022-12-09). *About HarmonyOS.* From HarmonyOS: https://developer.harmonyos.com/en/docs/documentation/doc-guides/harmonyos-overview-0000000000011903#section17198101363120
22. KaiOS. (n.d.). *kaios64 v1.1.* From GitHub: https://github.com/kaios/kaios
23. PureOS. (n.d.). *PureOS Frequently Asked Questions FAQ*. From PureOS Tracker: https://tracker.pureos.net/w/faq/
24. PostmarketOS. (n.d.). *Kernel configuration*. From PostmarketOS: https://wiki.postmarketos.org/wiki/Kernel_configuration
25. Wikiwand. (n.d.). *Sailfish OS*. From Wikiwand: https://www.wikiwand.com/en/Sailfish_OS
26. Foundation, e. (January 2024). *The State of eBPF.* The Linux Foundation.
27. yunwei37. (Oct 10, 2023). *Userspace eBPF Runtimes: Overview and Applications*. From Medium: https://medium.com/@yunwei356/userspace-ebpf-runtimes-overview-and-applications-d19e3c84c7a7#:~:text=The%20benefits%20of%20using%20eBPF,without%20modifying%20their%20core%20code.
28. Which devices will support Fuchsia OS? (n.d.). *Quora*.
29. Byrne, B. (Feb 2, 2020). *Google Fuchsia To Run Magenta Kernel, Not Linux*. From ValueWalk: https://www.valuewalk.com/google-fuchsia-magenta-kernel/
30. *Zircon Kernel objects*. (2024-03-15). From Fuchsia: https://fuchsia.dev/fuchsia-src/reference/kernel_objects/objects
31. Bradshaw, A. f. (May 2 2023). *Nest Hub 2nd Gen updates to Google's Fuchsia operating system*. From 9TO5Google: https://9to5google.com/2023/05/02/nest-hub-2nd-gen-fuchsia-update/
32. Larabel, M. (26 February 2024). *Ubuntu Blog Talks Up Rust Schedulers, Potential For Micro-Kernel Design Future*. From Phoronix: https://www.phoronix.com/news/Ubuntu-Rust-Scheduler-Micro
33. LINDER, B. (04/15/2021). *Ubuntu Touch for the PinePhone is moving to a new kernel for better hardware support*. From liliputing: https://liliputing.com/ubuntu-touch-for-the-pinephone-is-moving-to-a-new-kernel-for-better-hardware-support/
34. *Kernel and hardware abstraction*. (2019-2023). From UB Ports Docs: https://docs.ubports.com/en/latest/systemdev/kernel-hal.html

35. *Head First eBPF*. (n.d.). From eBPF and Network Trends Forecast for 2024: https://www.ebpf.top/en/post/network_and_bpf_2024/
36. yanivagman. (2021). *BPFroid*. From Github: https://github.com/yanivagman/BPFroid/blob/main/README.md