<	...
Online team meeting	https://fau.zoom-x.de/j/9776061710?pwd=9BPbcHYQaVEf6L0IH3xbsSeNzajvJ0.1
Production system (if any)	...
Test system (if any)	...
GitHub repository	https://github.com/amosproj/amos2024ws03-android-zero-instrumentation
GitHub feature board	https://github.com/orgs/amosproj/projects/72/views/2
GitHub imp-squared backlog	https://github.com/orgs/amosproj/projects/76
Team T-shirt (white)	https://www.shirtinator.de/s/7TKe0UJeS-O8AjdcoWldFA
Team T-shirt (black)	https://www.shirtinator.de/s/4GNRKpvlQVWlYNAkMCx-4A
Additional materials	...
Team maling list	oss-amos-proj3@lists.fau.de

Last Name	First Name	GitHub User Name	Email Address
Krug	Maximilian	HaruspexSan	krugm03@gmail.com
Ayach	Mohammed Tamim	Tamemo99	Tamemayash@gmail.com / Ayachmoh@hu
Bretting	Luca	luca-dot-sh	luca.bretting@fau.de
Seidl	Robin	mr-kanister	robin.seidl@fau.de (main) / 68117355+Mr-k
Labroussis	Christos	clabrous	c.labroussis1@gmail.com
Hilgers	Felix	fhilgers	felix.hilgers@fau.de
Weisshuhn	Tom	der-whity	tom.weisshuhn@fau.de
Schlicht	Franz	ffranzgitHub	franz.schlicht@fau.de
Nawlo	Ali	alinawlo	ali.nawlo@campus.tu-berlin.de
Zinn	Benedikt	BenediktZinn	benedikt.wh.zinn@gmail.com

#	Meeting Day	Product Owners	Software Developer	Release Manager	Scrum Master	Comment
1	2022-10-16	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	
2	2022-10-23	Ali Nawlo	Everyone else	Maximlian Krug	Maximilian Krug	
3	2022-10-30	Mohammed Tamim Ayach	Everyone else	Benedikt Zinn	Maximilian Krug	
4	2022-11-06	Ali Nawlo	Everyone else	Tom Weißhuhn	Maximilian Krug	
5	2022-11-13	Mohammed Tamim Ayach	Everyone else	Robin Seidl	Maximilian Krug	
6	2022-11-20	Ali Nawlo	Everyone else		Maximilian Krug	
7	2022-11-27	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	Mid-term due
8	2022-12-04	Ali Nawlo	Everyone else		Maximilian Krug	
9	2022-12-11	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	
10	2023-01-11	Ali Nawlo	Everyone else		Maximilian Krug	
11	2023-01-18	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	
12	2023-01-25	Ali Nawlo	Everyone else		Maximilian Krug	
13	2023-02-01	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	
14	2023-02-08	Ali Nawlo	Everyone else		Maximilian Krug	Demo day!
15	2023-02-15	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	Retrospective
Product owners, software developers, and Scurm Master are set and ideally don't change over time; the critical part is the Release Manager role you need to define here						

Goals 1	
	Completing the objective and task given by our IP, becoming a well rounded team in the meantime
Meeting norms 2	Be punctual (with a 5min pardon time)
	Max. two times missing from IP meeting
	not having the camera off two consecutive times
Working norms 2	Don't push to main, keep main in working order
	Dependencies are a team effort
	all tests must pass
	criticism via pull/merge requests
Coordination norms 2	PR with one other member
	max keeps meetings on track
Communication norms 2	communication via discord - team meeting via zoom
	document major changes
Consideration norms 2	be repectfull
	small disagreement, discuss and vote
Cont. improvement norms 2	team meeting for tracking team's progress -> standup emails for gathering intel
	pushing non functional changes will trigger a workshop
Rewards 1	have cake together
Sanctions 1	Otheres choose a random virtual background
Signatures	
Scrum Master	Maximilian Krug
Product owner	Mohammed Tamim Ayach
Product owner	Ali Nawlo
Software developer	Luca Bretting
Software developer	Benedikt Zinn
Software developer	Christos Labroussis
Software developer	Robin Seidl
Software developer	Franz Schlicht
Software developer	Felix Hilgers
Software developer	Tom Weißhuhn
	https://oss.cs.fau.de/wp-content/uploads/2014/04/Team-Contract-Explanation-and-Examples.pdf

Product Vision	Project Mission
In systems with a high frequency of component changes, it is difficult to determine which component might be causing performance issues and affecting the entire system negatively. This is especially hard if the source code and/or build environment for the components is not present as they might be coming from external suppliers, which means they cannot easily be instrumented. This can result in a lot of communication and extra work. Using eBPF allows for tracking some of these issues at the kernel level, where for example blocking calls are made and can be tracked. It allows for hooking into Sys-Calls as well as calls to other userspace or kernel-level functions (uprobes and kprobes), all without needing to modify application code. This makes it possible to track down cross-cutting performance issues without needing additional support from the vendor of the component. The information about, for example the length of a blocking calls, can then be passed to various frontends, such as an Android application running on the target hardware or an external sink for displaying the data in visualization software like Grafana.	ZIOFA (Zero Instrumentation Observability for Android) aims to implement observability use cases relevant to performance specified by our industry partner using eBPF. Examples include tracing long-running blocking calls, leaking JNI indirect references or signals like SIGKILL sent to processes, all without instrumenting the observed application itself. The eBPF programs are loaded and unloaded using a backend daemon running as root that will collect metrics and send them to a client. For displaying these metrics to the user, we are implementing an on-device UI that can display visualizations for these use cases and allow for configuration of the enabled use cases, but using a decoupled Client SDK so that future work may easily make the data accessible the external processing.

Term	Definition

Sprint #	Sprint goal
1	None
2	None
3	None
4	Optional
5	Working, loading, and unloading of eBPF Programs from UI all the way to eBPF
6	Capturing Syscall Events
7	
8	
9	
10	
11	
12	
13	
14	
15	

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
Release						
Total			59	59		
Sprints						
1	**Get to know the Team**		8	59	6	59
2	**Get familiar with eBPF and other required technologies.**		8	51	6	53
3	**Start Developing, have a UI blueprint and a Backend beginning**		13	43	12	47
4	**Build a UI and work with Ebpf**		12	30	11	35
5	**Working, loading, and unloading of eBPF Programs from UI all the way to eBPF**		18		6	
Features						
1	**Get to know the Team**					
		Brain-storm Architecture	3		1	
		Preperation of Kotlin	3		3	
		Brain-storm ebpf use cases	2		2	
2	**Get familiar with eBPF and other required technologies.**					
		Docker Container	3		3	
		get information about android processes to list them	3		1	
		set aarch64 als target	1		1	
		use android 13 instead of 15	1		1	
3	**Start Developing, have a UI blueprint and a Backend beginning**					
		Preparation of CI	3		3	
		find timeseries visualization library	2		2	
		Sbom generation	2		1	
		Generation of sboms doesn't include kotlin	1		1	
		Communcation between Android side and Rust side	5		5	
4	**Build a UI and work with Ebpf**					
		unix domain socket traffic analysis (research)	5		3	
		Home Screen and Navigation Drawer	2		3	
		EBPF Program extension to load kProbes	3		3	
		Implement frontend load and list programs	2		2	
5	**Working, loading, and unloading of eBPF Programs from UI all the way to eBPF**					
		User eBPF programm Selection	5			
		kotlin interface for frontend loading and listing programs	1			
		test cli client: load and list programs	3			
		client library exported to kotlin	2			
		Running processes List	3		3	
		loading/unloading of ebpf functions in daemon	2		3	
		Display Installed Applications in UI	2			

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
Release						
Total			0	0		
Sprints						
1			0	0	0	0
2			0	0	0	0
3			0	0	0	0
...				0		0
Features						
1						
2						
3						

#	Feature Definition of Done	Sprint Release Definition of Done	Project Release Definition of Done
	1. Code for Components has been written. a. The code does comply to the naming conventions of the used programming language b. Code has been completed c. Unclear code parts are provided with a short comment, to explain what this part is supposed to do. 2. Developers submit a screenshots of the finished feature as a comment to the related issue 3. Feature has been reviewed by another team member 4. Feature has been merged and closed	1. Finished issues are marked as done 2. Code is tested and deployed 3. A short demo is available for each sprint (this is compliant with point 3 in DoD for Feature) so it can be the screenshots or a small video or even a short-live presentation 4. Bill of Material is kept in a current state	1. Team agrees on which features to be released 2. Features have been tested and reviewed by other team member 3. Documentations are kept updated 4. A short demo featuring major features is provided

Type	Link / reference

#	Context	Name	Version	License	Comment
1	Gradle Plugin	org.cyclonedx.bom	1.10.0	APACHE-2.0	https://github.com/CycloneDX/cyclonedx-gradle-plugin
2	Gradle Plugin	nl.littlerobots.version-catalog-update	0.8.5	APACHE-2.0	https://github.com/littlerobots/version-catalog-update-plugin
3	Gradle Plugin	com.github.ben-manes.versions	0.51.0	APACHE-2.0	https://github.com/ben-manes/gradle-versions-plugin
4	Gradle Plugin	com.android.application	8.6.0	APACHE-2.0	https://maven.google.com/web/index.html?q=com.android.applicat#com.android.application:com.android.application.gradle.plugin:8.6.0
5	Gradle Plugin	com.ncorti.ktfmt.gradle	0.20.1	MIT	https://github.com/cortinico/ktfmt-gradle
6	Gradle Plugin	org.jetbrains.kotlin.plugin.compose	2.0.21	APACHE-2.0	https://github.com/JetBrains/compose-multiplatform
7	Gradle Plugin	org.jetbrains.kotlin.android	2.0.21	APACHE-2.0	https://github.com/JetBrains/kotlin
8	Android UI	androidx.activity:activity-compose	1.9.3	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.activity#androidx.activity:activity-compose:1.9.3
9	Android UI	androidx.compose:compose-bom	2024.10.01	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.compose#androidx.compose:compose-bom:2024.10.01
10	Android UI	androidx.core:core-ktx	1.15.0	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.core#androidx.core:core-ktx:1.15.0
11	Android UI	androidx.lifecycle:lifecycle-runtime-ktx	2.8.7	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.life#androidx.lifecycle:lifecycle-runtime-ktx:2.8.7
12	Android DI	io.insert-koin:koin-android	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
13	Android DI	io.insert-koin:koin-androidx-compose	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
14	Android DI	io.insert-koin:koin-core	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
15	Android Test	io.insert-koin:koin-test-junit4	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
16	Android Test	androidx.test.espresso:espresso-core	3.6.1	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.test.es#androidx.test.espresso:espresso-core:3.6.1
17	Android Test	androidx.test.ext:junit	1.2.1	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.test.ext#androidx.test.ext:junit:1.2.1
18	Android Test	junit:junit	4.13.2	EPL-1.0	https://github.com/junit-team/junit4
19	Rust Ebpf	aya	0.13.0	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
20	Rust Ebpf	aya-ebpf	0.1.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
21	Rust Ebpf	aya-log	0.2.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
22	Rust Ebpf	aya-log-ebpf	0.1.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
23	Rust Ebpf	libc	0.2.159	MIT OR APACHE-2.0	https://github.com/rust-lang/libc
24	Rust Errors	anyhow	1.0.0	MIT OR APACHE-2.0	https://github.com/dtolnay/anyhow
25	Rust Build	cargo_metadata	0.18.0	MIT	https://github.com/oli-obk/cargo_metadata
26	Rust Build	clap	4.5.20	MIT OR APACHE-2.0	https://github.com/clap-rs/clap
27	Rust Build	which	6.0.0	MIT	https://github.com/harryfei/which-rs
28	Rust Logging	env-logger	0.11.5	MIT OR APACHE-2.0	https://github.com/rust-cli/env_logger
29	Rust Logging	log	0.4.22	MIT OR APACHE-2.0	https://github.com/rust-lang/log
30	Rust Async	tokio	1.40.0	MIT	https://github.com/tokio-rs/tokio
31	Rust Async	tokio-stream	0.1.16	MIT	https://github.com/tokio-rs/tokio
32	Rust API	prost	0.13.3	APACHE-2.0	https://github.com/tokio-rs/prost
33	Rust API	tonic	0.12.3	MIT	https://github.com/hyperium/tonic
34	Rust API	tonic-build	0.12.3	MIT	https://github.com/hyperium/tonic
35	Toolchain	python3	3.12.6	PSF-2.0	https://docs.python.org/3/license.html
36	Toolchain	rust	1.84.0-nightly	MIT OR APACHE-2.0	https://www.rust-lang.org/policies/licenses
37	Toolchain	cargo-ndk	3.5.7	MIT OR APACHE-2.0	https://github.com/bbqsrc/cargo-ndk
38	Toolchain	protoc	28.2	BSD-3-Clause	https://github.com/protocolbuffers/protobuf
39	Toolchain	bpf-linker	0.9.13	MIT OR APACHE-2.0	https://github.com/aya-rs/bpf-linker
40	Toolchain	nix	2.18.7	LGPL-2.1	https://github.com/NixOS/nix
41	Toolchain	cyclonedx-cli	0.25.1	APACHE-2.0	https://github.com/CycloneDX/cyclonedx-cli
42	Toolchain	gradle	8.10.2	APACHE-2.0	https://github.com/gradle/gradle
43	Toolchain	openjdk	21.0.3	GPL-2.0-with-classpath-exception	https://openjdk.org/legal/gplv2+ce.html
44	Toolchain	android-cmdline-tools	16	android-sdk-license	https://developer.android.com/studio/terms
45	Toolchain	android-emulator	35.3.6.0	android-sdk-license	https://developer.android.com/studio/terms
46	Toolchain	android-ndk	28.0.12433566	android-sdk-license	https://developer.android.com/studio/terms
47	Toolchain	android-tools	35.0.0	android-sdk-license	https://developer.android.com/studio/terms
48	Toolchain	platform-tools	35.0.2	android-sdk-license	https://developer.android.com/studio/terms
49	Toolchain	platforms-android	35	android-sdk-license	https://developer.android.com/studio/terms
50	Rust API	uniffi	0.28.2	MPL-2.0	https://github.com/mozilla/uniffi-rs
51	Rust API	thiserror	1.0.68	MIT OR APACHE-2.0	https://github.com/dtolnay/thiserror
52	Gradle Plugin	com.android.library	8.6.0	APACHE-2.0	https://maven.google.com/web/index.html?q=com.android.libr#com.android.library:com.android.library.gradle.plugin:8.6.0
53	Gradle Plugin	org.mozilla.rust-android-gradle.rust-android	0.9.4	APACHE-2.0	https://github.com/mozilla/rust-android-gradle
54	Android Rust	net.java.dev.jna	5.15.0	Apache-2.0 OR LGPL-2.1	https://github.com/java-native-access/jna
55	Android Navigation	androidx.navigation:navigation-compose	2.8.0	Apache-2.0	https://maven.google.com/web/index.html?q=androidx.navigation#androidx.navigation:navigation-compose:2.8.0
56	Android Logging	com.jakewharton.timber:timber	2.8.0	Apache-2.0	https://github.com/JakeWharton/timber
57	Android Visualization	com.partykandpatrick.vico:compose	2.0.0-beta.2	Apache-2.0	https://github.com/patrykandpatrick/vico
58	Android Visualization	com.partykandpatrick.vico:compose-m2	2.0.0-beta.2	Apache-2.0	https://github.com/patrykandpatrick/vico
59	Android Visualization	com.partykandpatrick.vico:compose-m3	2.0.0-beta.2	Apache-2.0	https://github.com/patrykandpatrick/vico
60	Android Visualization	com.partykandpatrick.vico:core	2.0.0-beta.2	Apache-2.0	https://github.com/patrykandpatrick/vico
61	Rust Serialization	serde	1.0.214	MIT OR APACHE-2.0	https://github.com/serde-rs/serde
62	Rust Serialization	serde-json	1.0.0	MIT OR APACHE-2.0	https://github.com/serde-rs/json
63	Rust Tracing	tracing	0.1.40	MIT	https://github.com/tokio-rs/tracing
64	Rust Tracing	tracing-subscriber	0.3.18	MIT	https://github.com/tokio-rs/tracing
65	Rust System	procfs	0.17.0	MIT OR APACHE-2.0	https://github.com/eminence/procfs

Last Name	First Name	Value					
Krug	**Maximilian**			**#DIV/0!**	**#DIV/0!**		
Ayach	**Mohammed Tamim**						
Bretting	**Luca**						
Seidl	**Robin**						
Hilgers	**Felix**			**0**	**No size**		
Weisshuhn	**Tom**			**1**	**Trivial size**		
Schlicht	**Franz**			**2**	**Small size**		
Nawlo	**Ali**			**3**	**Medium size**		
Zinn	**Benedikt**			**5**	**Large size**		
				8	**Very large size**		
				13	**Too large (size)**		
Team members left							
Labroussis	**Christos**						
How to play planning poker							
1. Everyone type their number into their value field, don't hit return yet							
2. Someone, perhaps a product owner, count down 3.. 2.. 1..							
3. Then, everyone hit return to submit their value							