Work Plan
DSbD program / Catapult
KATLAS Technology (Cohort2, Tier1)

Marcos Mayorga, KATLAS CTO
20/09/22

Introduction.

This document highlights major waypoints between our discovery phase and final results, expanding on a timeline of six months.

Discovery.

During discovery we will spend our time studying the starting package and available documentation including:

- Receiving, unboxing and powering up the Morello board.
- CHERRI ISA spec.
- the debug port
- Installing available operating systems (debian gnu/linux, cherriBSD)
- Installing available development toolkits (gnu tookit, clang)

Development environment setup.

This activity include testing 2 operating systems (CherriBSD, debian gnu/linux) with two toolchains (GNU/clang) being able to compile on any of the 4 combinations the following code:

- A “Hello, world” minimum program.
- Our codebase dependencies, particularly the EC cryptography for the curve secp256k1.
- Our current codebase consisting on +100000 lines of code of C++, normally compiled using the GNU toolchain on debian GNU/Linux.

Any relevant complication found in the mature stack CherriBSD/clang or the less-mature stack GNU/Linux/gcc that could help in their strenghtening would be reported.
Once our compiled software runs its tests successfully we start looking with more detail at the features offered by CHERRI and how can our software benefit from them.

The expected amount of time for this milestoner is uncertain. The best case scenario could take only a

few hours, but as soon as any problem arise even if small could take days of investigation until the task can be resumed.

Our project and intended use-case.

KATLAS are contributing a secure network by design (SNbD), supported by system of intelligent wallets for web3.0 enabled federated machine learning on sensitive data with zero-trust on privacy enhancing technology (PET).
User Case - addressing the major gaps in the provision of existing national digital research infrastructure, especially those capable of dealing with personal identifiable and sensitive data. Overcoming the challenges of federated machine learning has enormous socio-economic potential to unblock the next generation of human knowledge. KATLAS has 4 years of R&D in a clinical setting to contribute, with a solution that is promises to level-up and democratise access to population data for specialist research, reducing monopoly power by a small number of global health organisations with undue power in times of pandemic and crisis. Success will bring the attention of data protection regulators in the public sector and be a global leader in privacy enhanced technology.

Development of a test scenario.

Our software is meant to be run as a privacy wallet with trading capabilities.
In other words the program is represented by a daemon process in the operating system.
The wallet daemon is meant to hold several simultaneous conversations with different remote peers using an independent e2e encrypted channel for each one of them.

Potentially, a remote adversary trying to obtain an edge would manipulate her side of the communication at harmful will. In lab we will give the possibility by alowing the attacker to run trojan malware in our target system.
In our software, the trading subsystem is based on plugins (aka R2R protocols), meaning the wallet-engine allows to link external software in-process.
With classic CPU architectures the malware could try to access any memory address at process scope (allowed to the process by the operating system with classic pointer architecture), including potential access to unathorized data structures related to other's privacy space.

The first task is building a 'malicious' wallet plugin (role-2-role protocol that could be named evil2honest) meant to be introduced in the victim's computer (target) as a trojan.
The pluging would define a protocol API, with one function named "spy_balance".
The attacker will successfully obtain victim's wallet balance in non-cherri architectures.

The expected amount of time for developing and testing such r2r protocol is only a few days of initial work, plus eventual correction iterations.

Code refactor.

Study ways the code should be refactored to leverage the new features of CHERRI using the Morello board: **fat pointers** and **compartimentalization**.

Implement the minimum set of changes in our source code needed to frustrate our attacker from accessing a remote balance.

The expected amount of time for this stage is the most critical for the proof we seek. A combination of research and development until the exact proof is achieved is difficult to estimate but it should feasible to achieve it in the allocated time.

Verify, review and ship.

Verifying that this particular type of attack can effectively be prevented by the cherri architecture, by comparing whether and how an attacker is un/successful against a Morello board/Classic architecture.
We'll write detailed reports about findings.