# MQP A Term Wrapup

Zachary Porter

2022-10-12

## 1 Minimum A-Term Deliverables

- ☑ Code capable of tracing data through a program's execution
  - ☑ A plan for comparing the traces and identifying shared points between traces
  - ☑ A platform for my UI that I am comfortable with
  - ☑ Recording infastructure to support scrubbing along the timeline of a program
  - ☑ Tests for all code I have written (within reason)
  - ☐ Documentation for all code that reasonably requires documentation

    Failure: Librr is lacking documentation in many places that absolutely require documentation
  - ☐ An example use case to demonstrate value

    Failure: I am still working on this

## 2 Use Case: Profiling

Identify code that might work, but should not (use after free, memory leak, etc).

Problems:

- I am working with optmized assembly
- It is impossible (very difficult) for me to determine the correct length of an array and if it is overflown
- Other tools can likely do this better

## 3 Use Case: Fuzzing / Exploit Discovery

Identify inputs that do not crash but follow a similar code flow. This would be useful once you have identified a vulnerable path through the binary but it crashes.

Problems:

- Other technologies can do this with far less overhead.

# 4 Use Case: Version Ignorant Patches

Create a format for doing binary patches that is dependent on *program behavior* rather than binary layout. As such, in order to apply a patch, you first run through the program and then my program determines how to apply the patch to the binary.

This would allow for development of patches for version x to work on version x+1 with greater probability (not a certainty). This is useful if you want to apply a patch to something like your web browser which requires frequent but small updates.

Problems:

- Defining the specifications for this type of patch may be difficult
- Re-applying this patch may be quite fragile and it would require patch-tests

# 5 Use Case: Bypassing Simple Checks

Give the user tools to identify checks or instructions that the user does not wish to have run (Ex: ignore trial license expiration).

Problems:

- This is not a very noble persuit and won't impact much.

# 6 Use Case: Exposing GUI functionality as programmatic Functionality

Allow the user to define *functions* that set memory addresses, run blocks of code, and then return values in other addresses. This can be used to create programmatic interfaces to things that otherwise are locked deep inside of an App.

Example for a function that accepts a users zoom username and password and returns their personal meeting room id

How to use:
Start zoom via MQPProj
Signin using auth method (username, pw)
Navigate to personal page
Unhide your personal meeting room id.
Close Zoom.
Start MQPProj on recording that was just created.
Scrub start of timeline to before signin
Scrub end of timeline to seeing your personal room ID.
Indicate to MQPProj the function input (username, pw)
Indicate to MQPProj the output of the function (personal meeting room id)
Have it generate an executable for $f(username, pw) \rightarrow personalmeetingroomid$

Problems:

- This could be impossible

- Timing GUI events as well as internet events would break many assumptions of rr and require updates

# 7 Use Case: Identifying Performance Regressions

Given two traces of different binary versions running the same code, identify matching blocks of instructions and then compare size and performance to report to the user.

Problems:

- Exact performance timing information can be tricky in rr as it alters performance slightly (especially in programs that expect to be run on multiple cores).
- This would be more of a tool to compare compiler versions rather than comparing program versions

# 8 Use Case: Provide optimization hints back to the compiler

Many Java programs run as fast or faster than naive C or C++ or Rust programs due to optimizations allowed by runtime analysis. Create a framework that can analyze a trace in order to give hints to LLVM as to the best way to optimize the program.

Problems:

- I know very little about LLVM
- This might fail