Fridalens: A Plugin for Instrumentation Across Multiple Binaries and Devices

Xavier Rosado, Leamsi Alicea Alamo, Gabriel Garcia Aviles

Advisor: Dr. Wilson Rivera Gallego

Department of Computer Science and Engineering

Computer Science and Engineering
University of Puerto Rico
Mayagüez Campus

1 Problem Statement

Current tools lack user-friendly interfaces for multi-binary instrumentation, impeding software analysis and testing efficiency.

By developing a tool with intuitive interfaces for multi-binary instrumentation and fully harnessing Frida's functionalities, we can enhance user productivity and streamline software analysis for developers, security researchers, and engineers.

2 Problem Background

Binary analysis workflows often consist of a combination of static and dynamic methods to properly understand program behavior. Static analysis tools like Binary Ninja provide decompilation and help understand the structure of the program, but lacks execution and performing dynamic modifications [1]. Frida provides dynamic of binaries, providing control over the process itself rather than altering the binary structure [2]. Existing tools such as frinja integrate both Binary Ninja and Frida capabilities, but struggle to meet more complex use cases such as multi-binary instrumentation [3] [4]. The project looks to address this gap and offer a user friendly interface with frida capabilities, all exposed to the user through the Binary Ninja UI.

3 Objectives

- Implement tracing of user selected functions across all frida-gadgets. Users are able to see function parameters and view return values when applicable.
- Implement seamless propagation of changes made to instrumentation scripts with hot reloading. Changes should occur within 2 seconds.

4 Technical Approach

Users can use the plugin through exposed commands in Binary Ninja which interface with fridalens functionality.
The frida-portal instance runs as a daemon thread under Binary Ninja. Communication to and from frida-gadgets are handled asynchronously in the portal, presented to the user through Binary Ninja's console.

```
[portal-server] Functions marked: [<func: x86_64@0x1036>]
[portal-server] Idling...
[portal-server] Functions marked: [<func: x86_64@0x1036>]
[portal-server] Idling...
[portal-server] Functions marked: [<func: x86_64@0x1036>]
[portal-server] Node connected: 1, Address: ('127.0.0.1', 41008)
[portal-server] Node joined: 1, Application: Application(identifier="/home/xv1r/Projects/capstone/trash/test", name="test", pid=37768, parameters={'system': {'arch': 'x64', 'os': {'id': 'manjaro', 'name': 'Manjaro Linux'}, 'platform': 'linux', 'name': 'QWERTY', 'access': 'full'}, 'config': {}})
[GADGETS] [PID 37768-info] int64_t sub_1036() not found in any loaded module
[portal-server] Node left: 1, Address: Application(identifier="/home/xv1r/Projects/capstone/trash/test", name="test", pid=37768, parameters={'system': {'arch': 'x64', 'os': {'id': 'manjaro', 'name': 'Manjaro Linux'}, 'platform': 'linux', 'name': 'QWERTY', 'access': 'full'}, 'config': {}})
[portal-server] Node disconnected: 1, Address: ('127.0.0.1', 41008)
[portal-server] Idling...
[portal-server] Functions marked: [<func: x86_64@0x1036>]
```

Figure 1: Console output of fridalens. Shows one marked function, as well as output of a gadget attempting to trace a function.

A testing environment using docker-compose setup was used to test system architecture as well as integration tests.

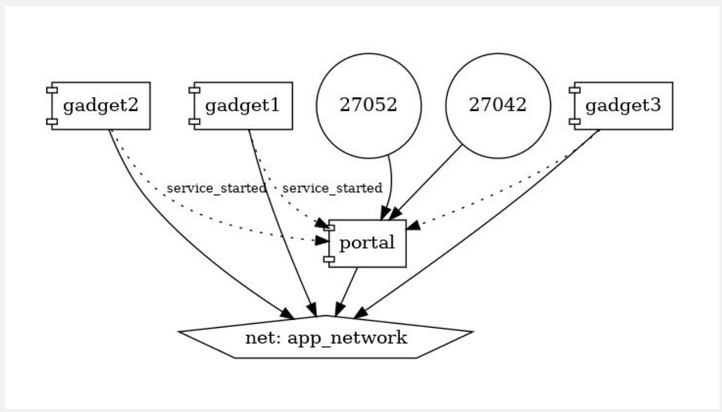

Figure 2: Block diagram of frida-portal testing environment generated using docker-compose-viz. Contains three gadgets, all connected to the portal instance under the same network.

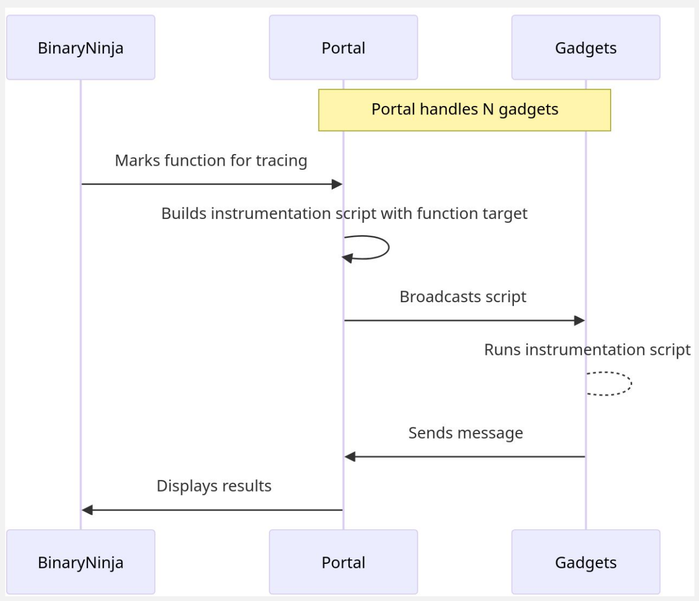

Figure 3: Sequence diagram for high level overview of function tracing

Fridalens implementation also allows for device tagging, allowing for devices to only take changes which correspond to a tag, enabling multi program instrumentation in any given device.

The design also preserves the frida-portal feature to be treated as a frida-server, allowing for future extension of features such as cross collaboration, as well as preserving frida-server compatible interfaces.

5 Results

- Functions can be marked and are instantly registered by the portal service to build instrumentation scripts with proper function name translation if possible. If function names are not usable, can be replaced with function addresses.
- Current implementation struggles with repeated function names and handling of architecture/OS specific quirks.

Table 1: Time measurements of hot reload new traced functions and for gadget response against amount of traced functions.

TABLE 1: Time measurements for function trace reloading and message passing

Traced amount	Hot reload	Message turnaround
1	127 ms	1.7 s
2	131 ms	2.1 s
3	143 ms	2.4 s
4	150 ms	2.7 s

6 Conclusion

Fridalens allows for quick and easy complex multi binary instrumentation environments to be used from only Binary Ninja. Integration of static analysis and instrumentation workflows allows for large scale setups to be created with little to no effort, allowing for more user productivity and focus on their uses.

R References

1. "Binary Ninja," Binary Ninja, https://binary.ninja/
2. O. A. V. Ravnås, "Frida • A world-class dynamic instrumentation toolkit | observe ...," Frida: Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers., https://Frida.re/.
3. D. Zervas, (2019) "Frinja" [Github Repository]. https://github.com/dzervas/Frinja/tree/main.
4. D. Andriesse, "9. Binary Instrumentation," in *Practical Binary Analysis: Build your own linux tools for binary instrumentation, analysis, and disassembly*, San Francisco, California: No Starch Press, 2019