eBPF Survey Check Point 01

Project participant: Zhaoyuan Su G01314332
Project participant: Renyuan Liu G01316367

EBPF Survey Objectives:

The objective of this eBPF survey is to give both an overview of the current general features and a detailed exploration for specific features of eBPF tools and applications, and to provide both qualitative and quantitative data support for both of these questions. The purpose of this survey is to give a quick insight to newcomers who are not familiar with eBPF tools and applications, and hopefully to give an inspiration to related researchers.

Survey Research Questions:

eBPF research questions in this survey into two dimensions:

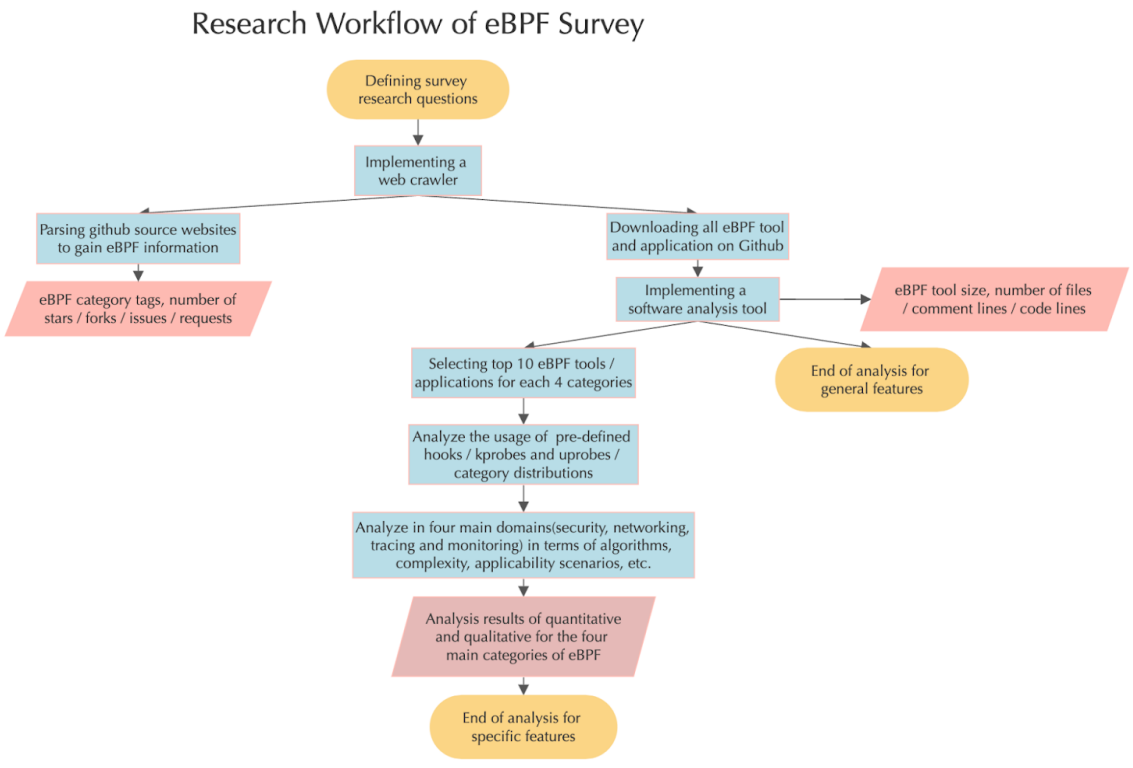
General Features	
Category	Research Question
Function classification	Which functional category does each eBPF tool or application belong to?
size	What is the software size of each eBPF tool or application?
	how many files and lines of code are there in total?
programming language	What programming language is used for each eBPF tool? Which programming language is the mainstream of eBPF tool development? what is the distribution of each programming language?
Open Source Community Reviews	How many stars / forks / issues / requests each tool has on github?
hooks and calls	Does each eBPF tool or application use hooks? Which hooks are used? What is the probability distribution like?
Running Platform	What is the distribution of the system platforms targeted by each eBPF tool or application, such as Windows, Linux, Mac, etc.?
bugs	What are the bugs of each eBPF tool or application? How are the types of bugs distributed? How many bugs has been fixed and how many bugs has still been open?

Table 1. General Features Measured by The eBPF Survey

Specific Features	
Category	Research Question
Security	What system security features are eBPF developers concerned about? What methods do current eBPF tools use to improve system security? What are the obvious advantages and disadvantages of these methods compared to the system default methods?
Tracing	What system runtime behaviors are of concern to eBPF developers in tracking the flow of users running applications?
	Instead of requiring the export of vast amounts of sampling data as typically done by default system, what data structures and algorithms are used by the advanced eBPF tools and applications to provide tracing system performance insight?
Networking	In which situations is eBPF best suited for programming networks? which properties of eBPF give it this advantage? Which compiler do most eBPF developers choose to use for networking and why?
Monitoring	Which system metrics are most concerned by eBPF developers?
	How does dBPF reduce the overall system overhead for monitoring?

Table 2. Specific Features Measured by The eBPF Survey

Research Workflow and Architecture:



Picture 1. The research workflow chart for eBPF Survey

Research Procedures and Development:

1. Design an analysis workflow and an architecture as the research line and technical support for the survey.(Done)

2. Implementing a web crawler and downloading all open source eBPF tools and applications from Github.(Done)
3. Parsing the structure of github's source website, and gaining open-source community reviews (such as category tags, number of stars / forks / issues / requests) of each eBPF tool and application using automatic web requests. (In-progress).
4. Implementing a software analysis tool that conducts multiple general feature statistics for each eBPF, including software size, number of files, number comment lines, number of code lines, etc.(In-progress)
5. Implementing an analysis tool that collects the information about distribution of programming languages used for eBPF.(In-progress)
6. For some top stars projects of every area, analyse what pre-defined hooks are used or what kernel probes and user probes are created. And analyse the function distribution of kprobes and uprobes.(In-progress)
7. Analyze the distribution of projects like Cilium, bcc, or bpftrace which provide an abstraction on top of eBPF.(In-progress)
8. Analyze the distribution of supported map types like Hash map, LRU, Ring buffer. (In-progress)
9. Find out the most popular way to ensure safety on the foundation of seeing and understanding all system calls. (In-progress)
10. Analyze the distribution of statistical data structures used for extract visibility data for tracing. (In-progress)
11. Analyze the distribution of different compilers used for networking solutions. (In-progress)
12. Analyze the distribution of customs metrics used for extending the depth of visibility for monitoring. (In-progress)

Deliverable Research Results

1. A survey paper on eBPF tools and applications.
2. Open-source analysis software.