Deep dive into eBPF-based system security tools in cloud-native environments

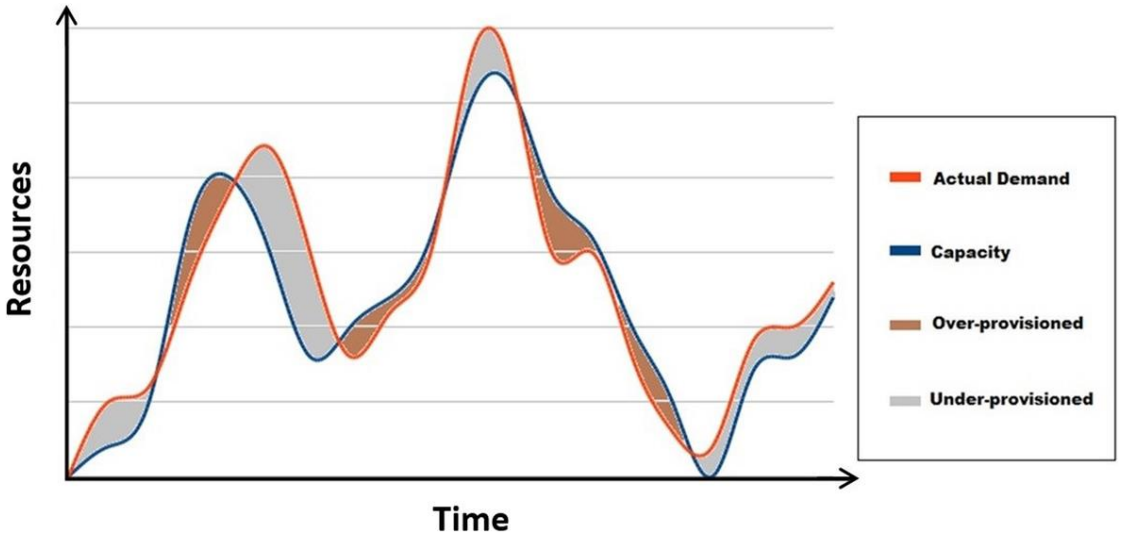
Jin Her

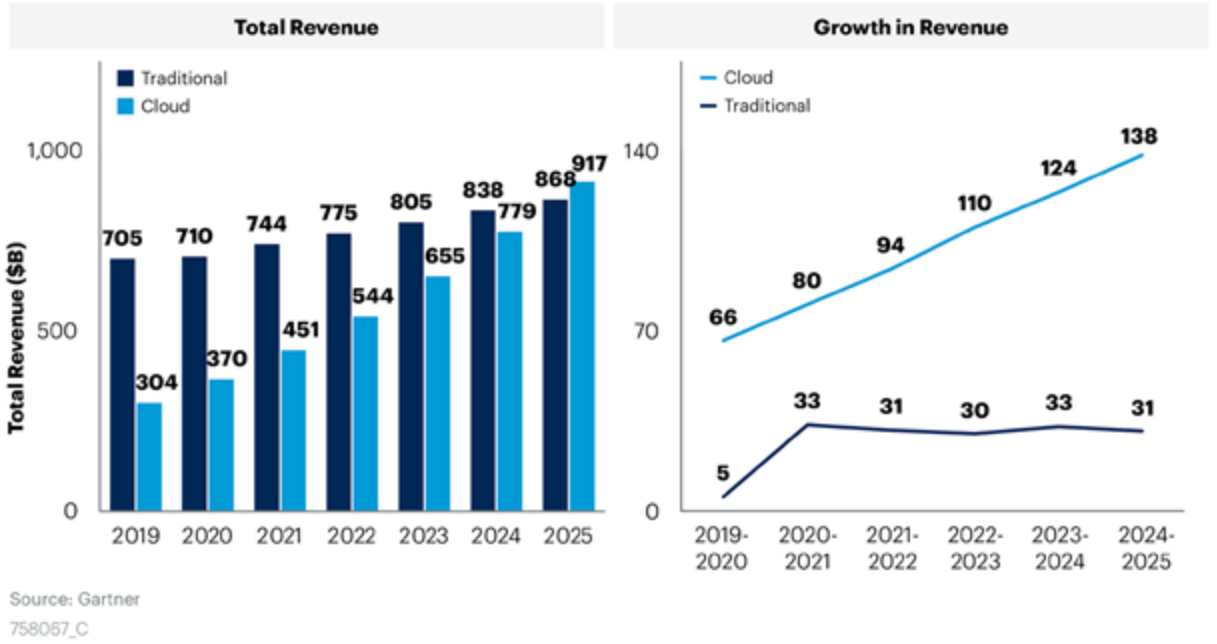
Department of Computer Science & Engineering

Incheon National University

Introduction

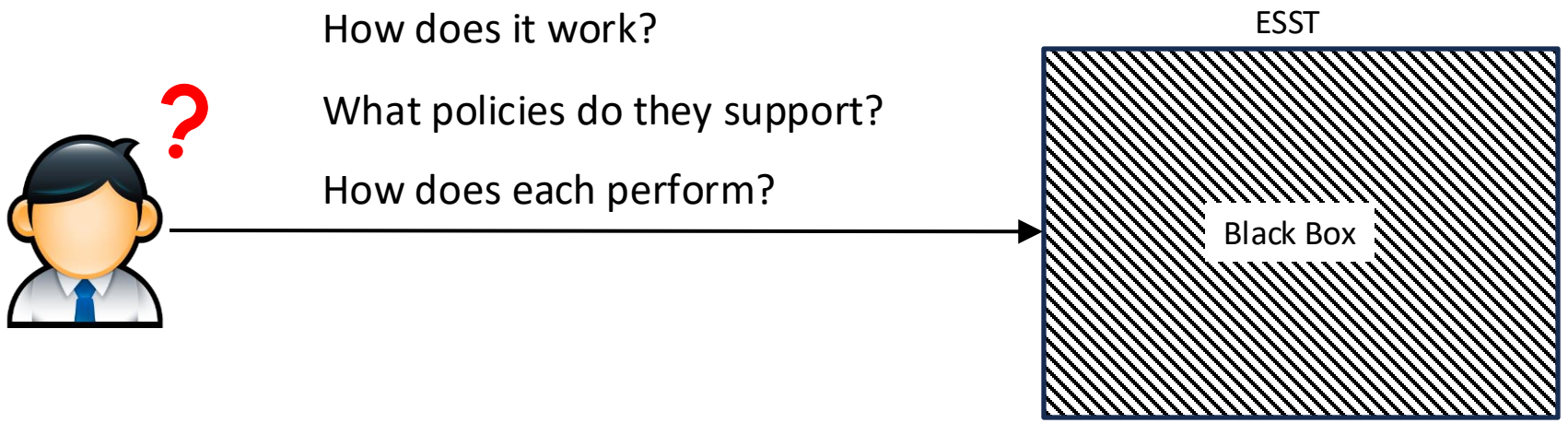
- Rapid adoption of microservice architectures(MSA)
- Misconfigured Kubernetes-based MSAs can lead to serious vulnerabilities
- Therefore, the US National Security Agency recommends the use of eBPF-based System Security Tools (ESST)



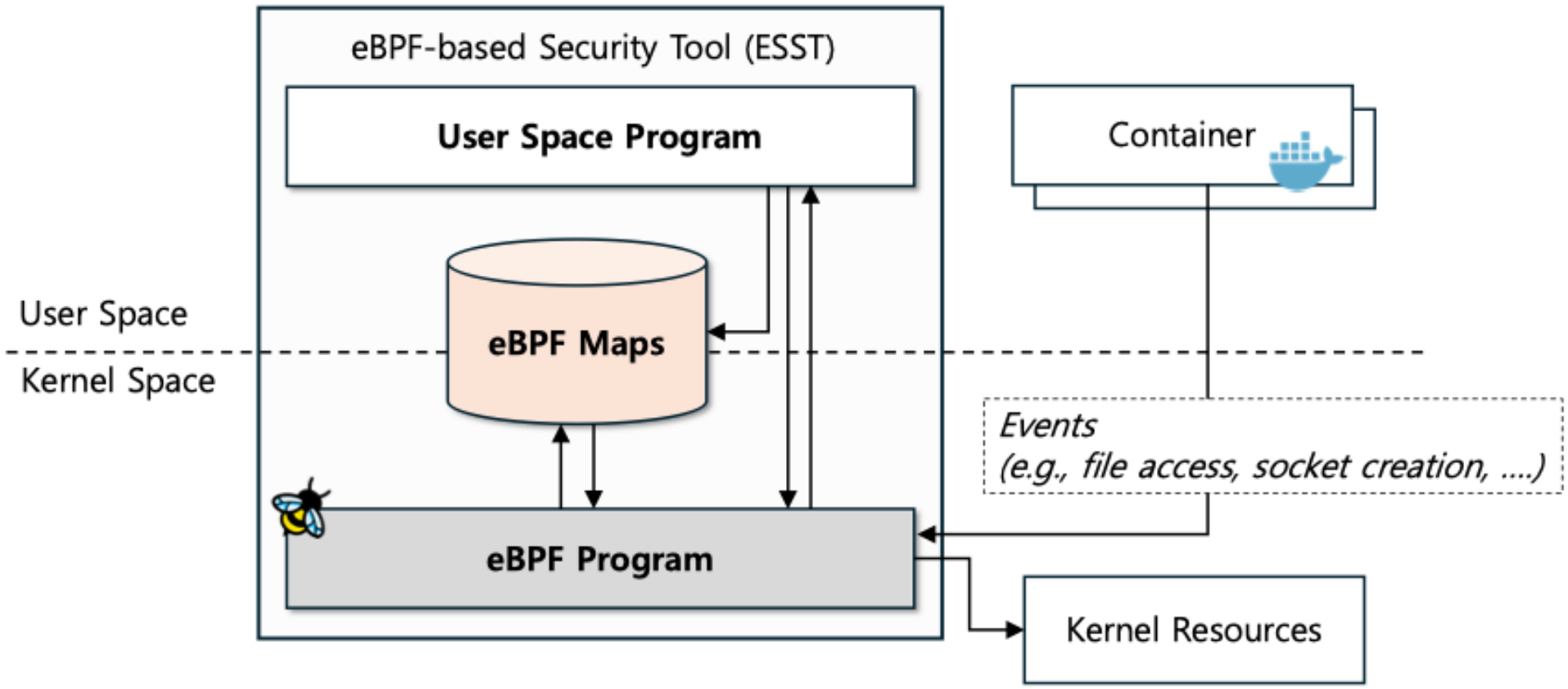


Problem Statement

- Limiting **visibility** into internal architecture
- independent **policy coverage**
- Lack of **performance insights**



Common Architecture



Compare Implementation

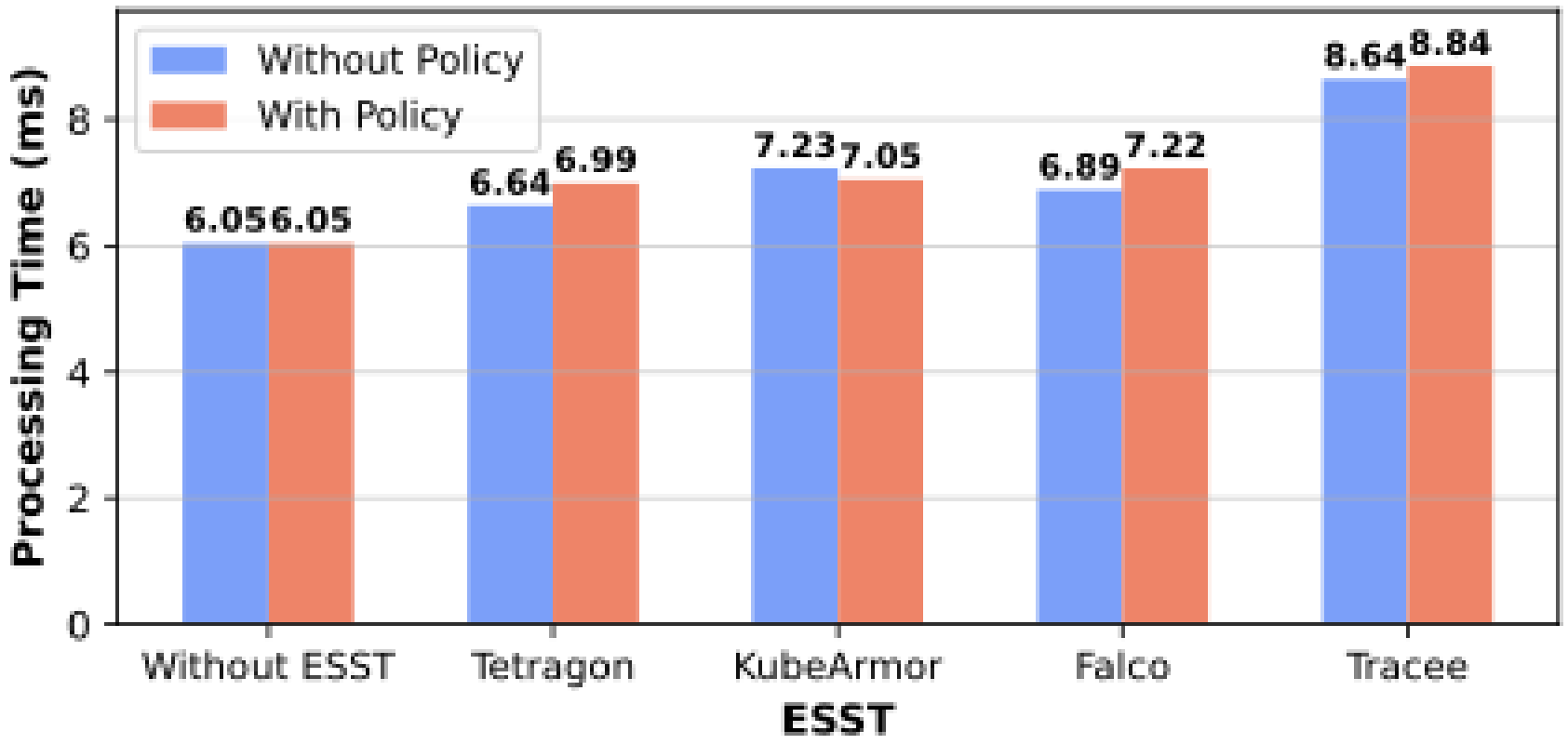
ESST	Policy Enforcement Method	Restart Required	eBPF Program
KubeArmor	Kubernetes API + Informer	X	Static
Tetragon	Kubernetes API + Informer	X	Dynamic
Tracee	Enforce in initialization phase	O	Static
Falco	Config File + *helm upgrade*	O	Static

Policy Coverage

ESST	Policy Scope
Tracee	Security, Network, Extra, Syscalls
KubeArmor	Process, File, Network, Capabilities, Syscalls
Falco	Syscalls, Tracepoints, Meta, Plugins
Tetragon	Kprobes, Tracepoints, Uprobes, LSM BPF

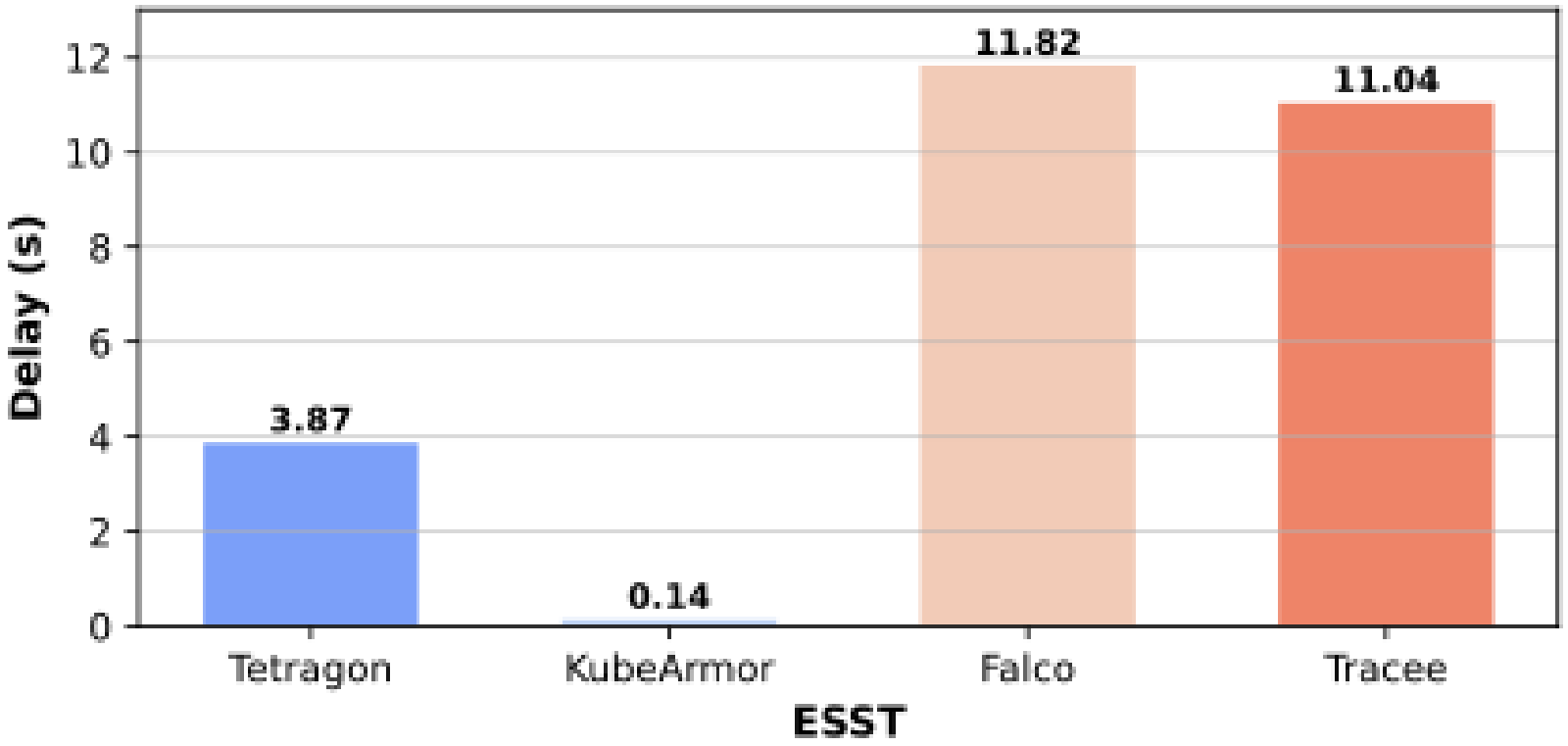
Evaluation [1] Micro Benchmark

- Measure the execution time of ***ls*** commands before and after the policy is applied



Evaluation [2] Enforcement Benchmark

- Run ***ls*** Commands Repeatedly After Deploying a Policy
- Measure the time until a policy is applied and logs occur



Conclusion

- This paper presents a comprehensive analysis of the operating mechanisms of ESST operating within the Kubernetes environment
- This is expected to provide a practical basis for ESST-related research

- Future work
  - Extend to analysis of eBPF-based security tools in a variety of environments