Atmosphere and the TCP/IP Stack for Maximizing Security

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Tirth Jain
ID No. 2019A7TS0120P

Under the supervision of:

Dr. Anton Burtsev
&
Dr. K Hari Babu

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS
November 2022

Declaration of Authorship

I, Tirth Jain, declare that this Undergraduate Thesis titled, ‘Atmosphere and the TCP/IP Stack for Maximizing Security’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

Certificate

This is to certify that the thesis entitled, "*Atmosphere and the TCP/IP Stack for Maximizing Security*" and submitted by Tirth Jain ID No. 2019A7TS0120P in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.

Supervisor	*Co-Supervisor*
Dr. Anton Burtsev	Dr. K Hari Babu
Associate Professor,	Asst. Professor,
University of Utah	BITS Pilani Pilani Campus
Date:	Date:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS

Abstract

Bachelor of Engineering (Hons.)

Atmosphere and the TCP/IP Stack for Maximizing Security

by Tirth Jain

Even after decades of work to make monolithic kernels more secure, serious vulnerabilities in them are still reported every year. Because the entire monolithic kernel is in one address space, an attacker is just one vulnerability away from owning the entire machine. We argue that it is time to decompose monolithic kernels like Linux into smaller parts that run in isolated compartments and communicate using secure interfaces. We think this is timely due to recent trends in hardware that make it easier and efficient to isolate kernel components.

In this Thesis, I discuss Atmosphere, our approach to microkernelization and fault isolation in Operating Systems. Specifically, I will focus on the network stack we built for Atmosphere. We argue that the network stack is a source for bugs and that isolation is the way forward to minimize the impact of these bugs.

Acknowledgements

Thanks to Anton for calling me to Utah and giving me this opportunity. Thanks to Prof. Hari Babu for teaching me NetProg. Thanks to Xiangdong for dragging me to the gym everyday and all of his compilers. Thanks to Zhaofeng for all the Nix shells and gatekeeping all our code. Without all of them, this Thesis wouldn't have been possible.

Contents

Dedicated to Kenkin, the emotional support capybara.

Chapter 1

Introduction

Operating system reliability is a topic that has been studied for decades but remains a major concern till date. Even thought its been 30 years since Linux kernel was developed, critical vulnerabilities are being found till date. On top of that, the monolithic design (ie everything in the kernel running in a single address space) makes it such that the attacker is only one exploit away from taking control of the entire operating system. This is concerning because device drivers, often developed by third parties are a major source of vulnerabilities. Redundancy in hardware and software protection mechanisms can protect against transient faults but persistent faults because of undetected logic flaws still remain a major concern. In recent times, with the advent of cloud services, this becomes even more significant because clouds often have to run untrusted code that needs to be isolated from other parts of the operating system.

Logic errors and faults can propogate from within a component and propogate to other parts of the system. Microkernelization aims to solve this problem by isolating various parts of an operating system and minimizing the effect of failures in a single subsystem. In a microkernel, only the bare minimum that is required to boot an operating system is included in the kernel. All other components such as the filesystem, the device drivers, and the network stack are run in separate isolated processes that interact with each other using some form of IPC. Inter component fault propogation can thus be reduced by introducing safe IPC mechanisms.

Atmosphere is a microkernel based operating system. In this thesis, I focus on the design of a single component of Atmosphere, the network stack and how this design can be used to model other services of the operating system. The network stack was built two main principles in mind: maximizing fault isolation and minimizing its cost on performance. For the first principle, we take microkernelization to its extremes and build a per connection (or per socket for UDP) network stack. This means, every new connection has its own TCP/IP stack that can process and dispatch packets. This ensures that if one connection is corrupted, the rest of the connections on the system can keep operating. Although this is not completely possible since there is some inherent shared state on any host machine on a network. In later sections we discuss how we minimize this shared state. For the second principle, we implement a zero copy model and explain how using Rust's guarantees help us implement this with confidence. On failure, a component can be safely restarted and meanwhile, all IPC calls to it can return an error.

The rest of this Thesis is organised as follows:

- In Background and Related Work we take a look at the contemporary approaches to microkernelization.
- System Design talks about the design choices we made in the development of Atmosphere.
- API Design discusses the API design and the internals of the network stack we built.
- Evaluation evaluates the performance of the network stack as compared to the current standards.

Chapter 2

Background and Related Work

2.1 Theseus

2.2 Singularity OS

2.3 sel4

Chapter 3

System Design

3.1 Safe IPC

Tirth *Describe RedLeaf RRef interface.*

Chapter 4

API Design

Chapter 5

Evaluation