Context Sensitive File System

Britton Dennis

Tyler Travis

Clark Wood

Outline

- **Motivation**
- Related Works
- System Overview
- Context Monitor
- File System
- Demo
- Problems / Future Work
- Q and A

Motivation

- Context awareness is important:
  - Mobile devices, BYOD
  - Cheap Storage
  - Ubiquitous wireless connections
- Ways related works fall short:
  - Security
  - Transparency

Outline

- Motivation
- **Related Works**
- System Overview
- Context Monitor
- File System
- Demo
- Problems / Future Work
- Q and A

Related Works

- quFiles
- Ext3cow
- NCryptfs

Related Works
quFiles

- Way to implement context awareness at file level
- quFiles directory stores different versions of the same logical file; serves appropriate file to application on demand
- Transparent to applications
- Not transparent to users

Related Works
ext3cow

- Way of doing file versioning / auditing by using copy on write on the inode structure
- Appends snapshot epoch number to files
- Semi transparent to applications
- Not transparent to users

Related Works
NCryptfs

- Adds encryption to existing Linux file systems
- Balanced security, performance, convenience, portability
- Prevent clear text from being cached
- Context unaware

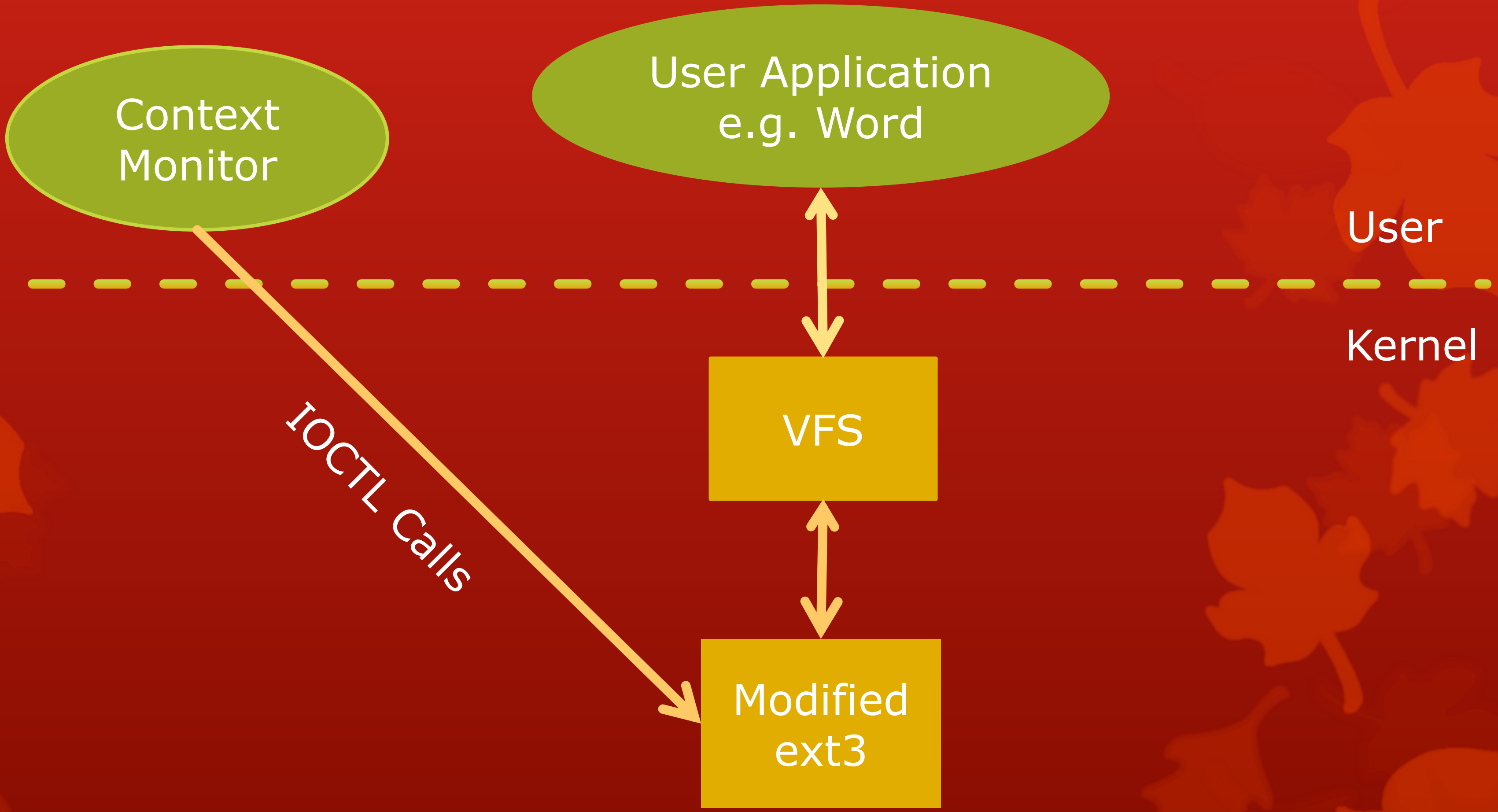
Outline

- Motivation
- Related Works
- **System Overview**
- Context Monitor
- File System
- Demo
- Problems / Future Work
- Q and A

System Overview

- Proposed by Michael Mitchell
- Our system is a proof of concept
- Two components
  - Context Monitor
  - File system
- Serves different files based on the security of the environment
- The metrics we wanted to optimize were portability and security

System Diagram



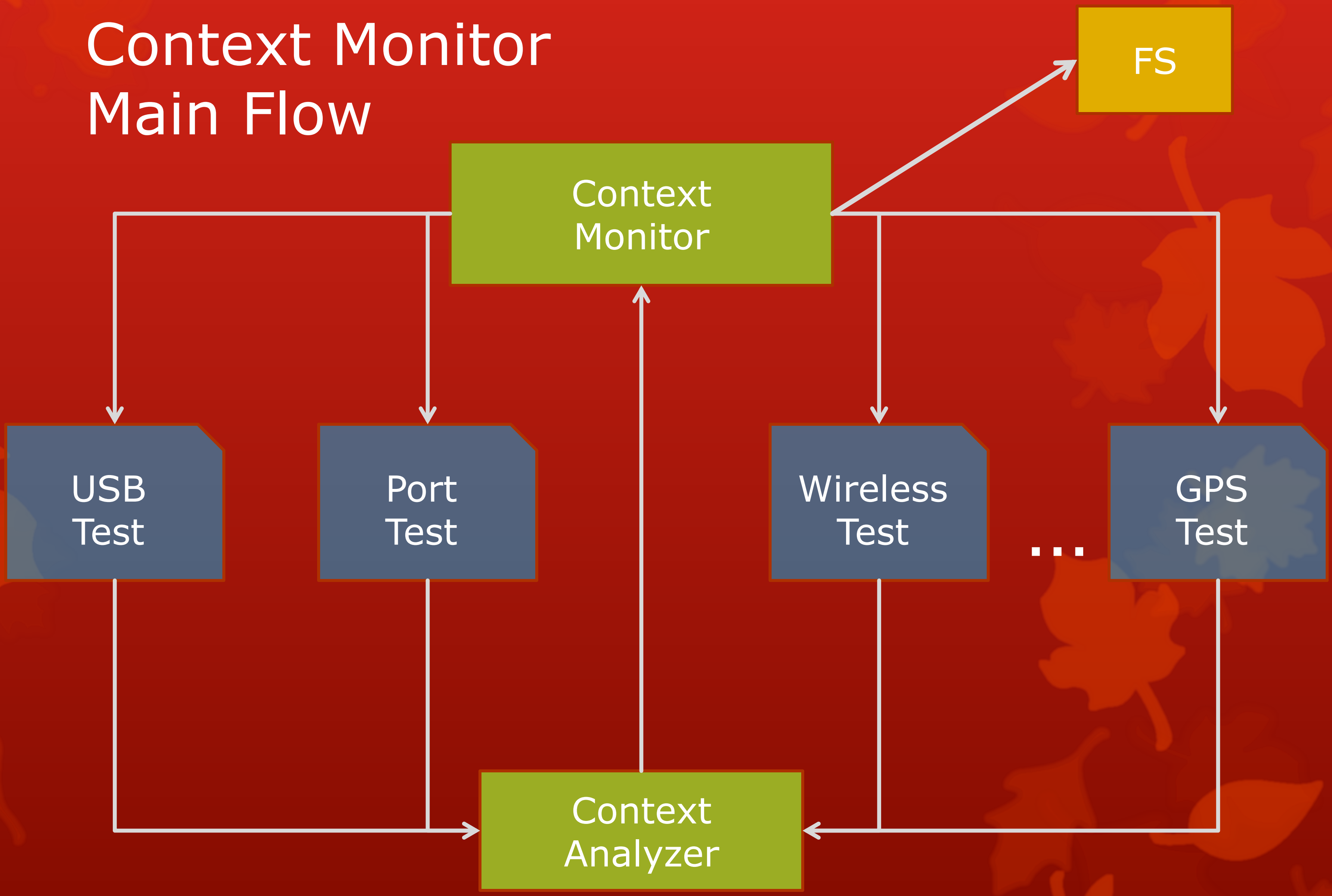
Outline

- Motivation
- Related Works
- System Overview
- **Context Monitor**
- File System
- Demo
- Problems / Future Work
- Q and A

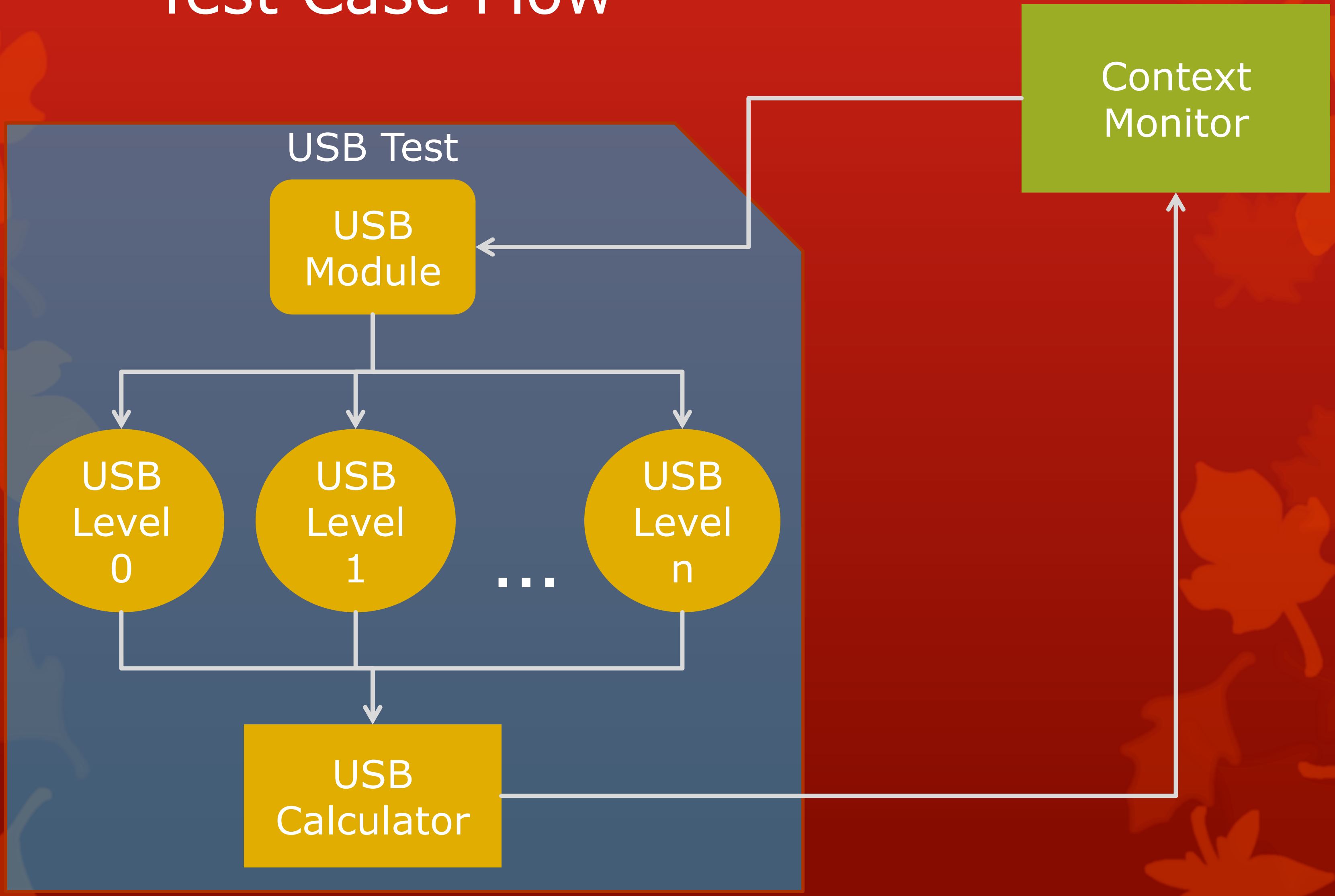
Context Monitor

- Modular user level deamon built mostly in python
- Each module polls a specific system internal (e.g. USB, Wifi, etc) for the current status
- The module then calculates how secure that component is at the moment and assigns a number
- Then all the numbers from all the modules are weighted according to the user's policy and a general 'security level' is generated
- This is then sent to the file system so that the file system knows whether to open 'up' or lock 'down' the file access
- The information is passed down through ioctl calls to give an extra level of security and flexibility as opposed to syscalls or procfiles as the caller needs to know the call number and the magic number

Context Monitor Main Flow



Context Monitor Test Case Flow



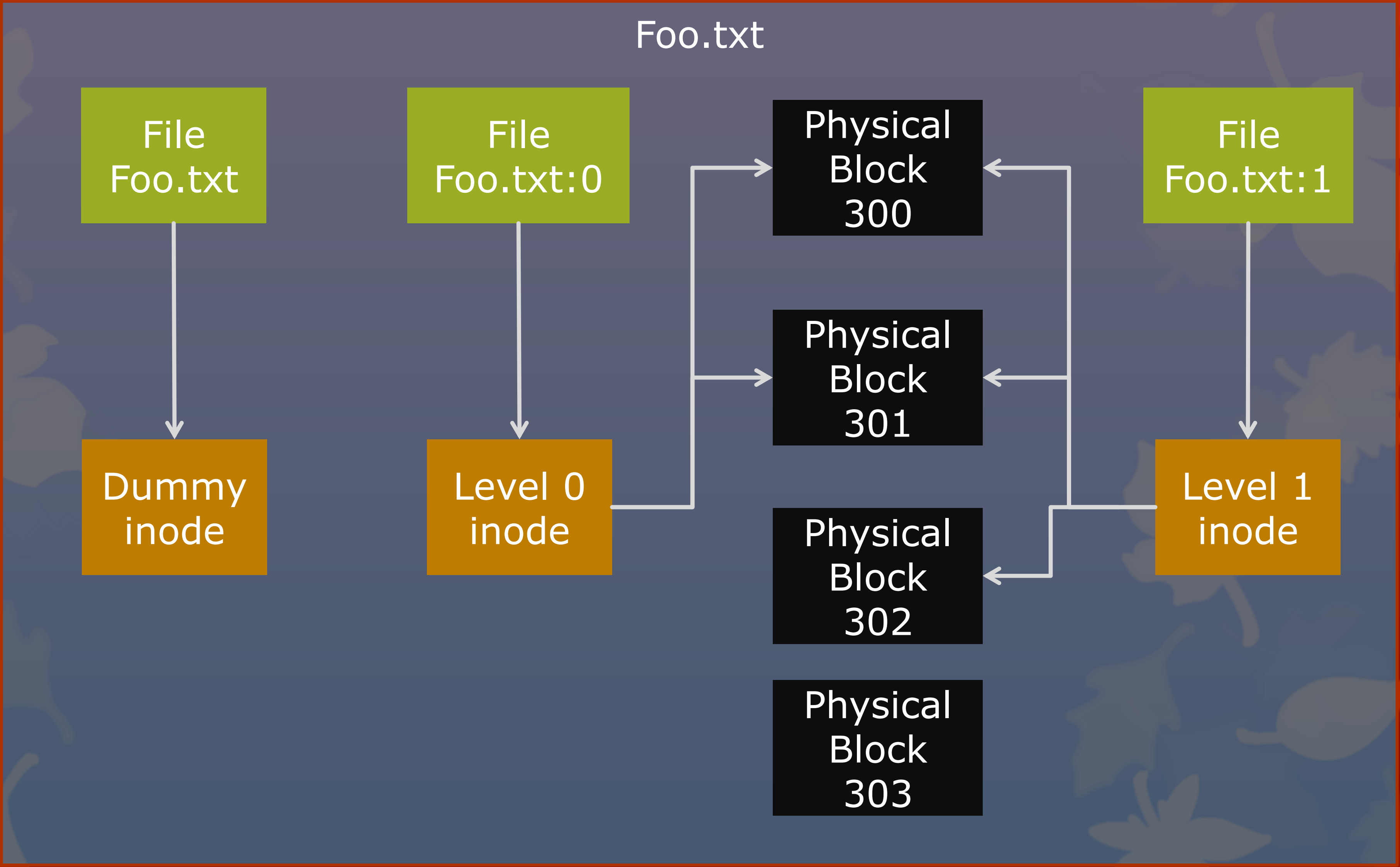
Outline

- Motivation
- Related Works
- System Overview
- Context Monitor
- **File System**
- Demo
- Problems / Future Work
- Q and A

File System

- Built from ext3 from kernel v3.8.5
- For each file the user sees there are really *n+1* files where *n* is the number of security levels
- The extra *n* files have the same base name put append a separator string as well as the security level number
- To keep secure data from being leaked, during a security level change, the file system will close all processes (as well as their children) that opened a file owned by the file system
- To prevent name space collisions and secure information leakage, the read directory file operation will hide all files ending with our suffix (except the ones the user created which can be determined through a recursive scan)
- To offer more data at higher security levels, the inode from the file at level *i* has blocks that point to all the blocks that inode at *i-1* points to as well as some blocks only it can see.

File System Organization



Outline

- Motivation
- Related Works
- System Overview
- Context Monitor
- File System
- **Demo**
- Problems / Future Work
- Q and A

Demo

Outline

- Motivation
- Related Works
- System Overview
- Context Monitor
- File System
- Demo
- **Problems / Future Work**
- Q and A

Problems / Future Work

- File system
  - Get write to work
  - Try to eliminate the need to open/close files in kernel context
  - Fairly high chance that code contains locking issues and memory leaks
  - Run some benchmarks and get some performance numbers to see if this is even within in the realm of usability
  - Look into possible caching issues
  - Resolve naming collisions that occur from non regular files
  - Support other file operations
  - Add similar mechanism for other types of files, e.g.. Directories
  - Add support for an arbitrarily large number of different security levels
- Context Monitor
  - Add other tests
  - Close file system if security level goes below acceptable limits
  - Add config options
- Provide a second user level program
  - Allow users to view data blocks belonging only at a certain levels
  - Allow users to switch to a level lower than what is deemed safe

Outline

- Motivation
- Related Works
- System Overview
- Context Monitor
- File System
- Demo
- Problems / Future Work
- **Q and A**

Q and A