Tool Support for Continuous Model-Based Verification of the Linux Kernel

Srinivas Dhanwada, Collin McIntyre, Matt Wall, Ben Weno

Overview

- L-SAP
- Problems with the Existing Verifier Pipeline
- Our Solution
- Patch Algorithm
- Differencing Algorithm
- Website Redesign
- Automation Design
- Testing & Evaluation Plan
- Project Schedule & Management
- Summary

L-SAP

L-SAP

- Scalable and Accurate Lock/Unlock Pairing for the Linux Kernel
- Looks for Lock/Unlock pairing inconsistencies throughout the kernel codebase
- Built on top of Atlas plugin to provide model-based verification
  - Other verification tools are unable to provide this for locking/unlocking problems
  - Example:
    - Lock a variable
    - Set it as a global variable
    - Unlock the global variable later in the code path.
- L-SAP Solves this problem by translating the code path into models and graphs
  - For each locking instance:
    - A graph is created tracking all the possible methods it can reach
    - A graph is created for each method tracking the status of the locking instance
  - L-SAP provides a conclusion, but also provides these graphs as evidence
  - Humans can look at this evidence and analyze inconclusive cases faster than looking at the codebase directly

Problems with the Existing Verifier Pipeline

Existing Verifier Pipeline Problems

- Currently L-SAP is unable to keep up with the pace of Linux development:
  - Have to manually create patch for each version
  - Have to manually upload results to website
- No existing way to map instances between different versions
  - How can we make sure a lock mismatch in a previous version was actually fixed?
- Current website is difficult to use
  - Consists of two lists for two different types of locking that each have thousands of links
  - No search tools available
  - Difficult to find mismatched locks

Requirements For Our Solution

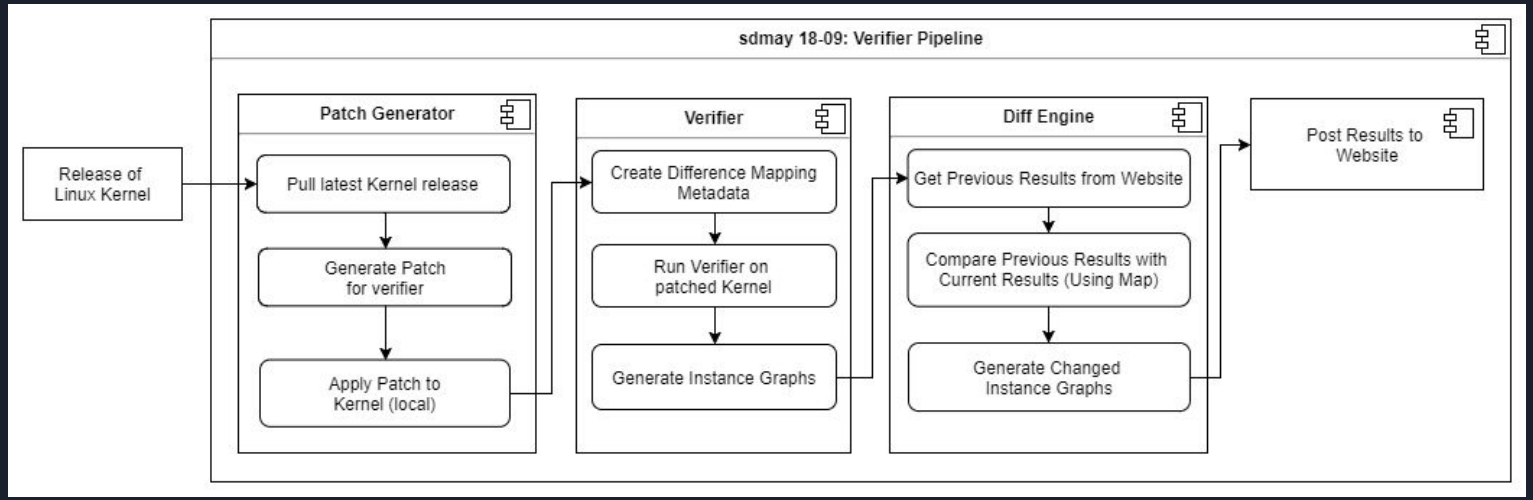
- Functional:
  - Automatically recognize kernel updates and start the tool
  - Create and apply a patch for the tool for each new version of the kernel
  - Run the tool with the new patch
  - Create a difference mapping for each locking instance
  - Post the results to the website
    - The website must properly display the new results in an organized manner
- Non Functional
  - The tool must run in a reasonable amount of time
  - The website must have a high level of usability, and scale to it’s demands

Our Solution

Our Solution

- Modularize and automate the pipeline
  - Patch Creation and Application
  - Instance Mapping
  - Running L-SAP
  - Differencing Results
- Redesign website to make tracking results easier
  - Provide search criteria
    - Text search
    - By driver search
  - Clearly show which instances are mismatched
  - Show the difference from previous versions to verify updates in the kernel quicker

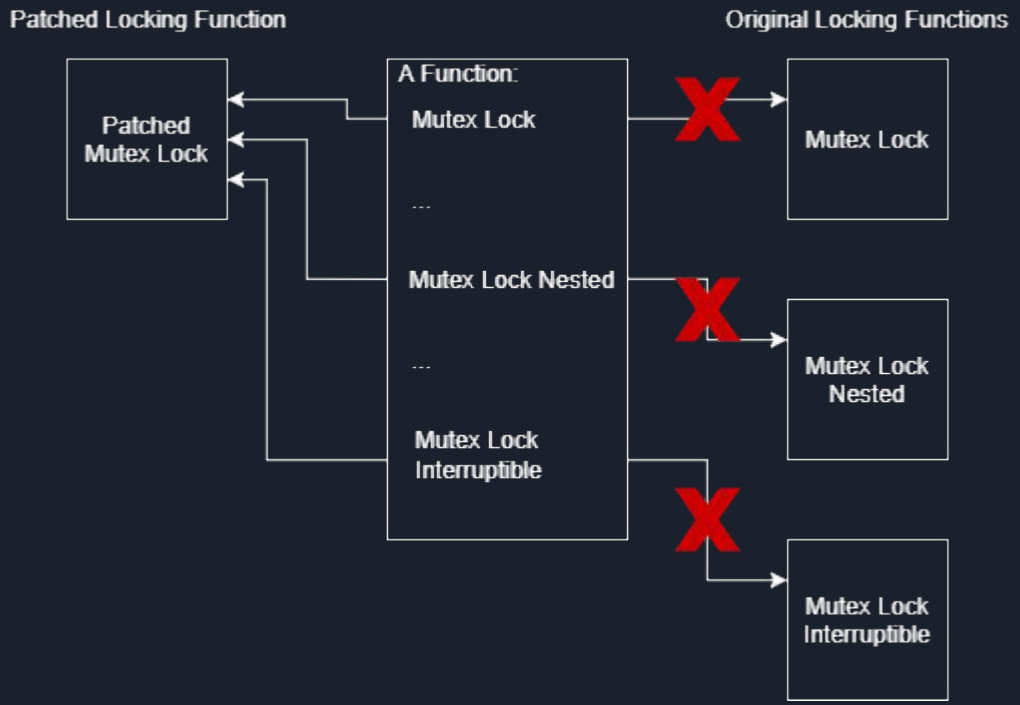
Our Solution



Patch Algorithm

Patching Algorithm

- What is the patch?
  - Redirection of all locking function calls to a single function per lock type
- Why is it needed?
  - L-SAP matches locks and unlocks based on function calls
  - Redirecting locking calls to a single function reduces computation time
  - Since kernel structure is all we care about, we can redirect to an empty function



Patching Algorithm

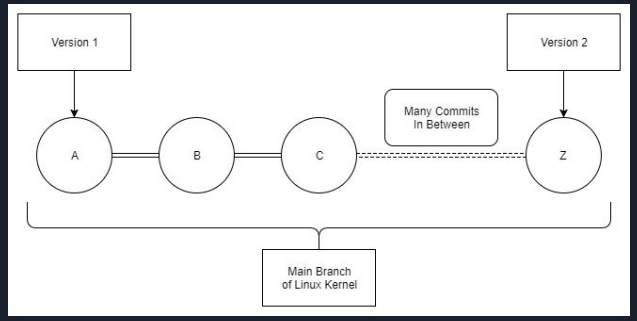
- Before the algorithm runs
  - Define parameters for mutex lock functions/macros
  - Define parameters for spinlock functions/macros
  - Determine kernel location
- Overview
  - Get function/macro information from existing headers
  - Generate patched header files
  - Using function/macro information…
    - Remove existing function/macro declarations
    - Remove existing function/macro implementations
- Process will be the same for mutex locks and spinlocks

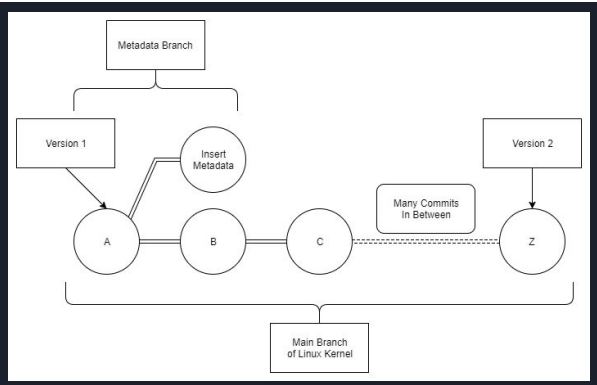
Differencing Algorithm

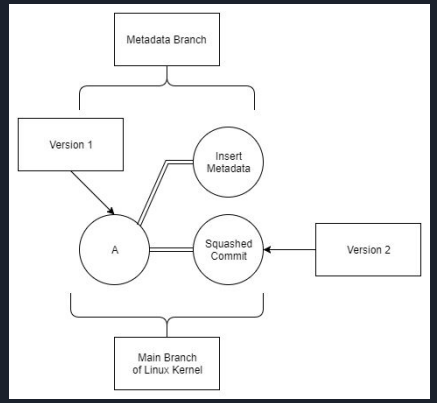
Differencing Algorithm

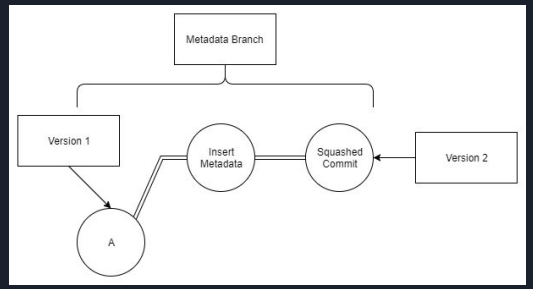
- Problem: Current Implementation of L-SAP assigns a random ID to each instance
  - No mapping between versions is possible, which means no comparison is possible
  - Looking at the same point in source code does not work either
    - Additions/Deletions could happen nearby causing a shift in the location of the instance
    - Changes to the locking variable name could also make it hard to find between two versions
  - Need some way to insert data within source code and have it move as the source code moves.
- Solution: Leverage the Linux Kernel's use of Git to help with tracking the differences between versions!
  - Insert a comment tag at the location of the instance -- commit these to a branch off of the first version
  - Use Git to rebase the differences between version onto the new branch
    - This inserts the comments first, then applies all the changes inserting the comments and having them move with the source code!

Differencing Algorithm









Differencing Algorithm

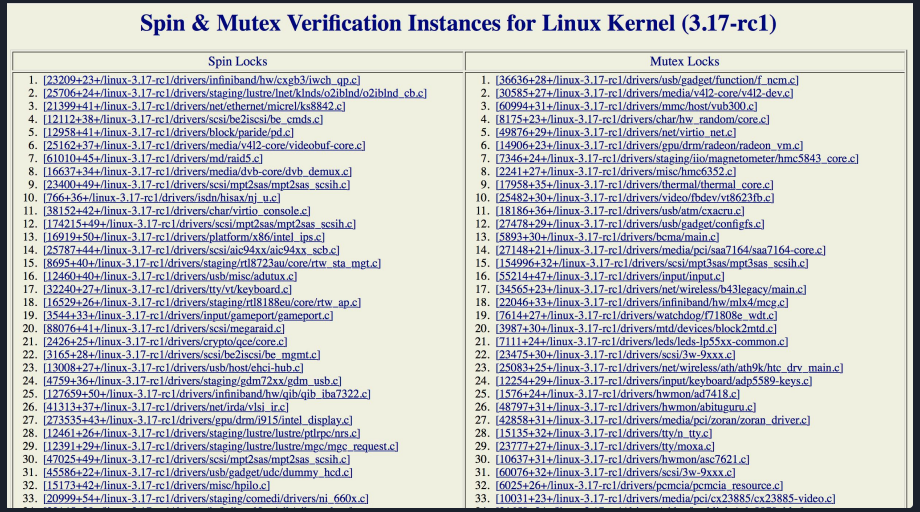
- Mapping between different versions has been created
- Need to fetch data and look for differences between results
  - This can be done by running the verifier on our modified version with metadata tags
  - Look at each instance for the metadata tag
  - Export these differences into a spreadsheet
- Due to the output of L-SAP, the only metric of differencing right now is the final status
  - An update to L-SAP might expose more data about the graph structures we can use
  - We will be able to compare the number of edges/nodes in a graph as well as the connections to see if the graph has changed even with the same status

Website Redesign

Angular + Typescript

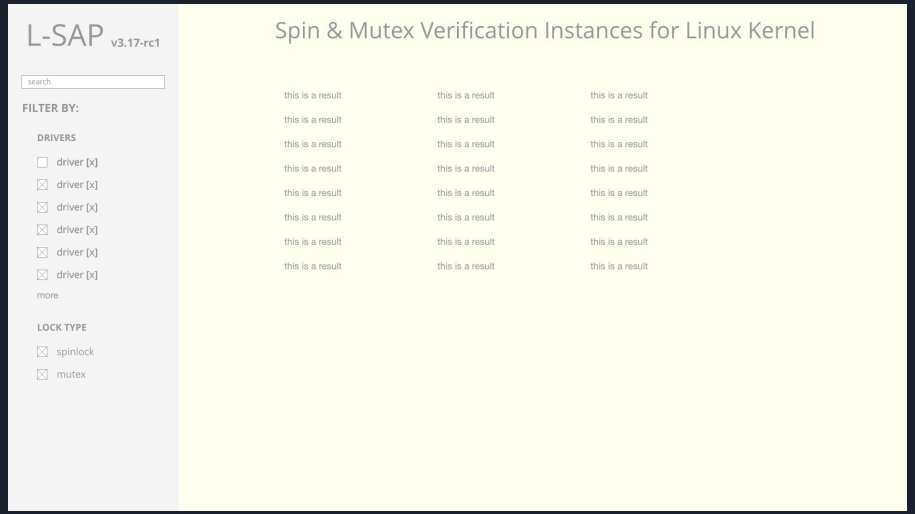
PROS	CONS
• Modularity • Data Binding • Cleaner Implementation • Optimization • Maintenance	• Harder to implement than static content • Some server side implementation required • Must have javascript enabled • One entry point to the site

Home Page Design

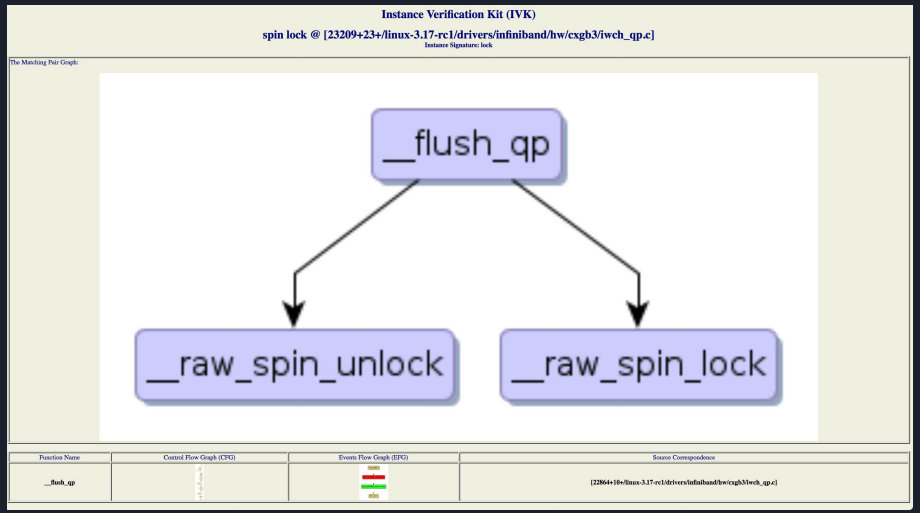


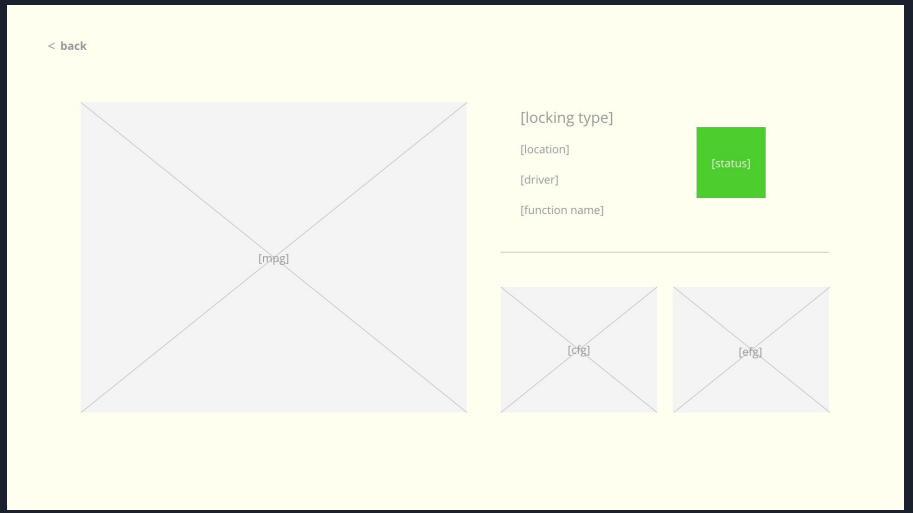
Spin & Mutex Verification Instances for Linux Kernel (3.17-rc1)

Spin Locks	Mutex Locks
1. [23209+23+/linux-3.17-rc1/drivers/infiniband/hw/cxgb3/iwch_qp.c]	1. [36636+28+/linux-3.17-rc1/drivers/usb/gadget/function/f_ncm.c]
2. [25706+24+/linux-3.17-rc1/drivers/staging/lustre/lnet/klnds/o2iblnd/o2iblnd_cb.c]	2. [30585+27+/linux-3.17-rc1/drivers/media/v4l2-core/v4l2-dev.c]
3. [21399+41+/linux-3.17-rc1/drivers/net/ethernet/micrel/ks8842.c]	3. [60994+31+/linux-3.17-rc1/drivers/mmc/host/vub300.c]
4. [12112+38+/linux-3.17-rc1/drivers/scsi/be2iscsi/be_cmds.c]	4. [8175+23+/linux-3.17-rc1/drivers/char/hw_random/core.c]
5. [12958+41+/linux-3.17-rc1/drivers/block/paride/pd.c]	5. [49876+29+/linux-3.17-rc1/drivers/net/virtio_net.c]
6. [25162+37+/linux-3.17-rc1/drivers/media/v4l2-core/videobuf-core.c]	6. [14906+23+/linux-3.17-rc1/drivers/gpu/drm/radeon/radeon_vm.c]
7. [61010+45+/linux-3.17-rc1/drivers/md/raid5.c]	7. [7346+24+/linux-3.17-rc1/drivers/staging/iio/magnetometer/hmc5843_core.c]
8. [16637+34+/linux-3.17-rc1/drivers/media/dvb-core/dvb_demux.c]	8. [2241+27+/linux-3.17-rc1/drivers/misc/hmc6352.c]
9. [23400+49+/linux-3.17-rc1/drivers/scsi/mpt2sas/mpt2sas_scsih.c]	9. [17958+35+/linux-3.17-rc1/drivers/thermal/thermal_core.c]
10. [766+36+/linux-3.17-rc1/drivers/isdn/hisax/nj_u.c]	10. [25482+30+/linux-3.17-rc1/drivers/video/fbdev/vt8623fb.c]
11. [38152+42+/linux-3.17-rc1/drivers/char/virtio_console.c]	11. [18186+36+/linux-3.17-rc1/drivers/usb/atm/cxacru.c]
12. [174215+49+/linux-3.17-rc1/drivers/scsi/mpt2sas/mpt2sas_scsih.c]	12. [27478+29+/linux-3.17-rc1/drivers/usb/gadget/configfs.c]
13. [16919+50+/linux-3.17-rc1/drivers/platform/x86/intel_ips.c]	13. [5893+30+/linux-3.17-rc1/drivers/bcma/main.c]
14. [25787+44+/linux-3.17-rc1/drivers/scsi/aic94xx/aic94xx_scb.c]	14. [27148+21+/linux-3.17-rc1/drivers/media/pci/saa7164/saa7164-core.c]
15. [8695+40+/linux-3.17-rc1/drivers/staging/rtl8723au/core/rtw_sta_mgt.c]	15. [154996+32+/linux-3.17-rc1/drivers/scsi/mpt3sas/mpt3sas_scsih.c]
16. [12460+40+/linux-3.17-rc1/drivers/usb/misc/adutux.c]	16. [55214+47+/linux-3.17-rc1/drivers/input/input.c]
17. [32240+27+/linux-3.17-rc1/drivers/tty/vt/keyboard.c]	17. [34565+23+/linux-3.17-rc1/drivers/net/wireless/b43legacy/main.c]
18. [16529+26+/linux-3.17-rc1/drivers/staging/rtl8188eu/core/rtw_ap.c]	18. [22046+33+/linux-3.17-rc1/drivers/infiniband/hw/mlx4/mcg.c]
19. [3544+33+/linux-3.17-rc1/drivers/input/gameport/gameport.c]	19. [7614+27+/linux-3.17-rc1/drivers/watchdog/f71808e_wdt.c]
20. [88076+41+/linux-3.17-rc1/drivers/scsi/megaraid.c]	20. [3987+30+/linux-3.17-rc1/drivers/mtd/devices/block2mtd.c]
21. [2426+25+/linux-3.17-rc1/drivers/crypto/qce/core.c]	21. [7111+24+/linux-3.17-rc1/drivers/leds/leds-lp55xx-common.c]
22. [3165+28+/linux-3.17-rc1/drivers/scsi/be2iscsi/be_mgmt.c]	22. [23475+30+/linux-3.17-rc1/drivers/scsi/3w-9xxx.c]
23. [13008+27+/linux-3.17-rc1/drivers/usb/host/ehci-hub.c]	23. [25083+25+/linux-3.17-rc1/drivers/net/wireless/ath/ath9k/htc_drv_main.c]
24. [4759+36+/linux-3.17-rc1/drivers/staging/gdm72xx/gdm_usb.c]	24. [12254+29+/linux-3.17-rc1/drivers/input/keyboard/adp5589-keys.c]
25. [127659+50+/linux-3.17-rc1/drivers/infiniband/hw/qib/qib_iba7322.c]	25. [1576+24+/linux-3.17-rc1/drivers/hwmon/ad7418.c]
26. [41313+37+/linux-3.17-rc1/drivers/net/irda/vlsi_ir.c]	26. [48797+31+/linux-3.17-rc1/drivers/hwmon/abituguru.c]
27. [273535+43+/linux-3.17-rc1/drivers/gpu/drm/i915/intel_display.c]	27. [42858+31+/linux-3.17-rc1/drivers/media/pci/zoran/zoran_driver.c]
28. [12461+26+/linux-3.17-rc1/drivers/staging/lustre/lustre/ptlrpc/nrs.c]	28. [15135+32+/linux-3.17-rc1/drivers/tty/n_tty.c]
29. [12391+29+/linux-3.17-rc1/drivers/staging/lustre/lustre/mgc/mgc_request.c]	29. [23777+27+/linux-3.17-rc1/drivers/tty/moxa.c]
30. [47025+49+/linux-3.17-rc1/drivers/scsi/mpt2sas/mpt2sas_scsih.c]	30. [10637+31+/linux-3.17-rc1/drivers/hwmon/asc7621.c]
31. [45586+22+/linux-3.17-rc1/drivers/usb/gadget/udc/dummy_hcd.c]	31. [60076+32+/linux-3.17-rc1/drivers/scsi/3w-9xxx.c]
32. [15173+42+/linux-3.17-rc1/drivers/misc/hpilo.c]	32. [6025+26+/linux-3.17-rc1/drivers/pcmcia/pcmcia_resource.c]
33. [20999+54+/linux-3.17-rc1/drivers/staging/comedi/drivers/ni_660x.c]	33. [10031+23+/linux-3.17-rc1/drivers/media/pci/cx23885/cx23885-video.c]



Single Instance Design





Automation Design

Automation Design

- Major Automated Steps:
  - Periodically poll Linux release RSS feed to kick off the process when a new kernel is released
  - Download new kernel
  - Generate and apply patch
  - Run L-SAP tool
  - Run mapping and differencing algorithm on results
  - Upload to website

Testing & Evaluation Plan

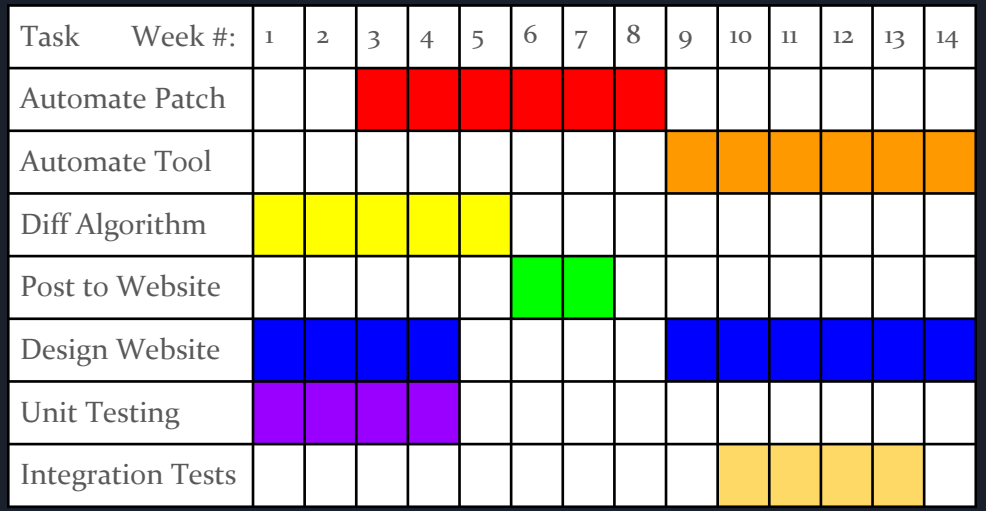
Testing & Evaluation Plan

- Unit Testing
  - Patching Algorithm, Differencing Algorithm, and The Automation Design are all written in Java
    - Unit Testing will be done through JUnit testing framework
  - Website will be build using TypeScript and the Angular 2 Framework
    - The Karma framework will be used to provide testing to the components on the website
  - Unit Testing will be done every time we intend to merge code with our master branch
- Integration Testing
  - Due to the stages of the pipeline being developed simultaneously, integration testing cannot be done immediately
  - Once the basic components of every component have been created, an Integration Testing project can be built to run through the pipeline (with a mock version of the verifier)

Project Schedule & Management

Project Schedule

- The next semester will be spent implementing existing designs
  - The differencing tool will be completed based on the mapping tool and research from this semester
  - The automation of the patch will be created based on the manual patch created this semester
  - The website will be implemented based on designs from this semester
  - Finally, all the pieces will be tied together for total automation of the tool
  - We will also implement unit testing and integration testing into our development cycle



Project Management

- We are using GitLab issues to document work that needs to be done, and to assign it in order to stay on track with our project plan
- Work will be lead in the following ways:
  - Srinivas Dhanwada - Differencing algorithm lead
  - Collin McIntyre - Patch automation lead
  - Matt Wall - Web design lead
  - Ben Weno - Tool automation lead
- Any resources for the project are provided by our client, such as a computer capable of running L-SAP and our tool

Potential Risks:

- A sweeping change in the Linux kernel code could render L-SAP unable to produce proper results
- L-SAP takes a large amount of memory to run, and our tool must run on the same machine, so we will need to keep resource usage low

Summary

Summary

- L-SAP Provides Accurate Results and Human Readable Evidence to aid in verification
- Problems that exist with the current process of running L-SAP
- Our Solution solves those problems and aims to automate the process
  - Patch Creation and Application
  - Difference Mapping and Summarization
  - User-Friendly Website to aid in Understanding of Results
  - Automation of Patching, Verification and Differencing
- Testing Framework will be integrated into development cycle
- Week-by-week schedule and Project Management

Questions?