SABIN KC

443-540-1505 | sabin1_kc@yahoo.com | linkedin.com/in/sabin-k-c-118999170

EDUCATION

University of Maryland College Park, MD

Bachelor of Engineering: Computer Engineering Dec 2020

- **GPA**: 3.82 / 4.0
- **Honors and Awards**: COTS Scholarship, Yurie Scholarship, Destler Scholarship, Dean's List
- **Coursework**: Operating Systems, Microprocessors, Digital Computer Design, Computer Organization, Linear Algebra, Organization of Programming Languages, Object-Oriented Programming I/II, Algorithms

PROJECTS

Geek OS (C, x86 Assembly) Aug-Dec 2020

- Designed and Implemented system calls such as pipe, fork/exec, etc.
- Implemented OS functionality such as signal handling, pre-CPU segmentation, virtual memory, and gfs3 filesystem for x86 kernel of Geek OS.

Binary Scanner (C, x86_64 Assembly) Oct-Nov 2020

- Designed and implemented a binary scanner program that could extract sha1-hash of .text section, assembly instruction call counts, Levenshtein distance to previous binary, etc. from an elf64 binary.
- Implemented a custom binary format and obfuscation method to save the analyzed results in the disk's database.

Polynomial Evaluation Accelerator (System Verilog, C) Oct-Dec 2020

- Designed and implemented polynomial evaluation accelerator to speed up the polynomial calculation in hardware using concepts of Lightweight Dataflow Models.
- Synthesized and tested the design in Xilinx FPGA.

Tethi Software (C, ARM Assembly) Mar-May 2020

- Designed and implemented Tethi software for ARM Cortex-M microprocessor board(STM32L476), which could interact with components of the board via MoT-format messages sent from PC over the board's COM port.
- Implemented functionality to use DMA for communication with components like GPIO, ADC, SPI, etc.

Unix Shell (C) April-May 2019

- Designed and implemented a Unix shell program in C that could execute basic Unix commands such as cd, ls, cat, grep, wc, echo, owd, mkdir, clear, true, false, etc.
- Implemented shell features like Pipe, logical AND/OR, and Subshell by using concepts such as piping, duping, and forking.

Small C compiler (Ocaml) July-June 2019

- Designed and implemented the tokenizing module which would take the input c file, then use regex to generate the stream of the categorized tokens.
- Implemented parser module to parse the token and construct an abstract syntax tree based on C compilers context-free grammar. Then processed AST via evaluation module to produce final results.

SKILLS

Technologies: Git, Unix, VS Code, FPGA

Languages: C/C++, python, java, Assembly Language, System Verilog