Ahmad Humayun

540-824-8988 | ahmad35@vt.edu | linkedin.com/in/ahmayun | github.com/ahmayun | personal-website

Education

Virginia Tech Blacksburg, VA
PhD in Computer Science, Software Engineering *Aug 2021 - Current*
Relevant Courses: Advanced Topics in Software Engineering, Hot Topics in Machine Learning (ML) and Security

Lahore University of Management Sciences Lahore, Pakistan
Bachelor of Computer Science *Aug 2017 - May 2021*
Relevant Courses: Big Data Analytics, Distributed Systems, Data Mining, Hardware Acceleration for AI, Speech Processing, Topics in Computer and Network Security, Databases.

Publications

Paper | MLSys 2026 | W. Gill, A. Humayun, ... *2 more.* "ProToken: Token-Level Attribution for Federated Large Language Models".

Paper | ICPC 2026 | Y. Wu, X. Zhou, A. Humayun, ... *2 more.* "Generating and Understanding Tests via Path-Aware Symbolic Execution...".

Paper, Artifact | In Review | S. Haroon, A. Khan, A. Humayun, ... *5 more.* "How Accurately Do Large Language Models Understand Code?".

Paper, Code | FSE 2024 | Y. Wu, A. Humayun, M. Gulzar, M. Kim. "Natural Symbolic Execution-Based Testing for Big Data Analytics".

Paper, Code | IEEE ASE 2023 | A. Humayun, Y. Wu, M. Kim, M. Gulzar. "NaturalFuzz: Natural Input Generation for Big Data Analytics".

Paper, Code | ESEC/FSE 2023 | A. Humayun, M. Kim, M. Gulzar. "Co-dependence Aware Fuzzing for Dataflow-based Big Data Analytics".

Professional Experience

Amazon Web Services Santa Clara, CA
Applied Scientist (Intern) *May 2025 - August 2025*

- Developed a **RAG**-based, LLM-powered tool (using **LangGraph** with **AWS Bedrock**) that automatically models complex **distributed algorithms** in P, a resource-constrained language, deployed as both a standalone tool and **MCP server**. Applied **OWASP LLM security** controls: input **sanitization** against **prompt-injection**, **static analysis** validation of generated code, and containerized **sandboxing** to isolate code execution. Increased **pass@k** on internally developed benchmarks by ≈ 71% while reducing token usage by ≈ 12%.
- Engineered a Java library, with **Log4j logging**, to translate abstract network messages (generated by P models) into executable API calls for fault detection. Thoughtful **API-design** to handle both streaming and batch inputs.

Amazon Web Services Santa Clara, CA
Applied Scientist (Intern) *May 2024 - August 2024*

- Enhanced automated testing infrastructure for improving **security** of critical AWS APIs by building a **mutation-based fuzzer** for generating valid request chains. Delivered a PoC for **AWS S3** (based on Schemathesis) that increased "happy case" sequences by ≈ 40%. Uncovered a bug in the upstream library deemed critical after **triage**.

Virginia Tech, Department of Computer Science Blacksburg, VA
Research Assistant - Prof. Muhammad Ali Gulzar *August 2021 - Present*

- Improving security, **automated testing**, and **formal verification** for data-intensive distributed programs.
- Worked on developing novel methods to improve the state-of-the-art in security and reliability for **peta-byte scale**, data-intensive applications, including ML systems. Relevant projects: DepFuzz, NaturalFuzz, NaturalSym, LLMs
- My tools have automatically exposed several faults in production software, including Apache Spark and Apache Flink. Example. Through my work, I have developed a deep theoretical understanding of distributed consensus algorithms and fault tolerance, as well as familiarity with internals of complex practical distributed systems like Spark.

Technical Skills

Languages: Java, Scala, SQL, C/C++, Python, JavaScript, Haskell, Intel x86, JVM Bytecode, Go, TypeScript, R
Tools and Frameworks: LangChain, LangGraph, MCP Servers, Log4j, Spark, PySpark, Hadoop, SBT, Maven, Node.js, React, Flask, Flutter, FastAPI, HuggingFace, Keras, PyTorch, LLVM, ptrace, gcc, clang, Git, Docker, CVC5, Z3, Lean, SMT-lib, MLlib, AWS Bedrock

Selected Projects

Token-Level Attribution for Distributed LLM Training | *Backdoor Detection, Adversarial ML, Provenance, Security*

- Worked on **ProToken**, a security-oriented provenance system that attributes generated tokens in federated LLMs to specific clients, even under **adversarial** conditions.
- Implemented a attribution pipeline enabling tractable **forensic analysis** of compromised federated updates.
- Developed a rigorous backdoor evaluation harness: injected trigger–response attacks into selected clients and achieved **98.62%** accuracy in identifying poisoned contributors across 16 configurations.
- Validated robustness by scaling attribution to 55 clients, maintaining **92–95%** detection accuracy on **Gemma, Llama, SmolLM, and Qwen** families.
- **Under review at MLSys 2026**

Adversarial Stress-Testing of LLM Code Understanding | *Fault Injection, Semantic-Preserving Attacks*

- Conducted a large-scale **adversarial evaluation** of LLM code comprehension by systematically injecting faults into real-world programs and assessing models' ability to localize vulnerabilities. (Paper, Arifact)
- Developed a configurable framework for generating **semantic-preserving perturbations** (e.g., variable renaming, dead code, comment changes) to simulate subtle **evasion-style attacks** on model reasoning.
- Demonstrated that these semantic-preserving attacks cause up to **78% degradation** in debugging accuracy, revealing brittleness and shallow representational understanding.
- Designed a two-phase validation pipeline ensuring **specification integrity** and eliminating contaminated or unverifiable tasks prior to adversarial stress testing.
- Executed over **600K debugging tasks** across nine LLMs, quantifying model susceptibility to **non-functional adversarial code transformations** and identifying structural patterns correlated with failure.

DAG-based Fuzzing for Distributed Frameworks | *Program Synthesis, Property Testing, Scala, SQL, Optimizers*

- Built a program-synthesis framework to synthesize dataflow programs e.g., Tensorflow and Spark. The framework provides tools to generate and analyze DAGs, the user writes a module to lower the DAG to a language of choice.
- Designed a state-aware concretization engine to generate high-validity dataflows.
- Implemented a novel property test to find UDF non-determinism issues in the query optimizer. This property test found a bug in the Spark optimizer. Reported a reproducible, minimal bug example to Spark JIRA.
- Reported several faults in Apache projects: FLINK-38366, FLINK-38397, FLINK-38446, FLINK-38637, SPARK-51798, SPARK-54196

Fuzzing Distributed Programs using Data Provenance | *Scala, Spark, HDFS, Java, Maven, MapReduce, Git*

- Developed a novel technique for efficient and effective fuzzing of Big Data (DISC) applications.
- Engineered taint-analysis engine to track provenance of data, associating them with relevant program regions.
- Implemented automatic framework abstraction to isolate application code from the complex framework below.
- Modified Scala compiler plugin *scoverage* to capture fuzzing coverage efficiently.
- Reimplemented state-of-the-art fuzzing techniques to use as baselines. Improved fault detection rate by 3.4× vs Jazzer. The resulting paper was **Accepted** at **ESEC/FSE 2023**. Acceptance Rate: 12.9% (Unconditional).

Transformer LLMs for Code Comprehension | *HuggingFace, FastAPI, PyTorch, Jupyter, BERT, GPT*

- Trained various LLM encoder stacks (e.g. BERT) to create general-purpose embeddings for code understanding.
- Analyzed 12K+ webpage JavaScript files and performed unsupervised training and clustering (link) of embeddings using DistilBERT. Visualized after dimensionality reduction with t-SNE showing efficacy for code comprehension.

NaturalFuzz | *Scala, Taint-Analysis, Apache Spark, Apache Hadoop, HDFS, Java, GPT2, LLM, BERT, SQL*

- Developed a novel technique for efficiently generating natural-looking inputs for DISC applications.
- Designed algorithms to perform intelligent reduction of data using taint analysis.
- Engineered a modular system with complex interlinked components to fully automate the idea.
- Used state-of-the-art LLMs e.g. GPT2 and BERT to quantify the naturalness of synthetic data.
- Collaborated effectively with a team to bring the research idea into fruition.
- Wrote an end to end research paper that got **accepted at IEEE/ACM ASE 2023**.

RAFT-based Distributed Key-Value Store | *Go, Golang, Distributed Algorithms*

- Implemented a distributed key-value storage service using the distributed consensus algorithm RAFT in Golang, demonstrating proficiency in both distributed systems and the Go programming language.
- Designed and developed a fault-tolerant and highly available distributed system capable of electing leaders, replicating logs, and ensuring consensus among a cluster of nodes.

- Utilized Go's concurrency features, including goroutines and channels, to build a concurrent and efficient implementation that can handle concurrent requests and maintain system stability.
- Tested the RAFT implementation extensively, including scenarios involving network disruptions, node failures, and recovery, to validate its correctness and resilience.

LLVM Pass for Lazy Code Motion | *LLVM, Compilers, Optimization*

- Implemented Lazy Code Motion (LCM) optimization pass in LLVM using four dataflow analysis passes.
- Developed preprocessing pass to split critical edges and basic blocks for safe temporary insertion.
- Designed postponable expressions pass to minimize register pressure by delaying temporary placement.
- Measured performance of the implementation on microbenchmarks. Reduced dynamic instruction counts across seven microbenchmark categories with 100 variants each. Full technical report.

Omniscient Debugger for Python | *x86, ptrace, strace, ELF, Python, Cython, Linux, Kernel*

- Engineered a debugger for Python applications that has the capability of stepping backward temporally.
- Used *ptrace* to intercept system calls and record program state selected points to be replayed later.

Benchmarking Framework for Software Debloating | *LLVM, C, C++, Docker, Python, gcc, clang*

- Researched and planned the feasibility of developing a benchmarking framework for software debloating tools.
- Developed an orchestrator for Docker containers in Python for facilitating software debloating research.
- Led a group of researchers to integrate the software debloating tool *Chisel* into the benchmarking framework.

Blocking Non-Essential JavaScript on Web Pages | *JavaScript, HTML, CSS, MDN, Chrome*

- Developed a browser extension for Mozilla Firefox to classify and block non-essential JavaScript from running on resource-constrained devices. Developed on-board lightweight classifiers for JS classification.

React Website with ExpressJS Backend | *React, JavaScript, Bootstrap, Firebase, OAuth*

- Lead the development of an online food ordering system using the REACT JavaScript framework with Firebase integration and an ExpressJS backend. The system consisted of 3 modules (Admin, Customer, Server).

References

- **Dr. Muhammad Ali Gulzar (Ph.D. Advisor)** - *Assistant Professor, Virginia Tech*
  gulzar@vt.edu || https://people.cs.vt.edu/gulzar
- **Prof. Miryung Kim (Research Collaborator)** - *Professor and Vice Chair of Graduate Studies, UCLA*
  miryung@cs.ucla.edu || https://web.cs.ucla.edu/miryung/
- **Dr. Dan Williams (PhD Advisory Committee Member)** - *Assistant Professor, Virginia Tech*
  djwillia@cs.vt.edu || https://people.cs.vt.edu/djwillia/