Kangjie Lu (UMN)

- ME: safe languages aren't enough
- "untrusted contributors" (well... typically have to submit PRs...)

examples
- log4J
Equifax vulnerability ?? "Apache Struts"

vulnerability known but not patched

11 min

3 pronged approach
4 directions

- foundational techniques
- analyses, automated patch fixing
"new classes"
- hardening systems, enforcing least privilege
- "secure compilers"
compiler - introduced vulns

understandability "assurability"??

ME: what about Rowhammer, Spectre etc.

This is so high level!

doesn't explain CFI

digging in on indirect calls & CFG analysis, wow

(can collect info at execution time easily, also ptr analysis, address taken, type analysis)

"MLTA"
type analysis of access to ptr for invocation?
every layer multi match
types of access for disambig.

kernel stuff

not clear how much this gains, no isolation yet

compare to Nooks, but no actual mechanism, so...

---

identify importance of vulns in Linux when in Android

"Diff CVSS"

seems pretty meh, & I don't understand why Android vendors don't have resources

---

Future Work

ME: LARGE body of work on type analysis in Java

Copilot observation not new
I think they are already looking at this at GH ~~bro~~ not bad

Kexin Pei talk

static program analysis - not always required, often problematic
Unusual take on static analysis requiring “rules & heuristics”
OK, use ML to automate program representation analysis, let’s see
ChatGPT, finds issues

Haha Scalene, nice touch (cite me! :))

Google experience?

ML4CODE.github.it

Limitations of code2vec (changing variable names from array to types, model changes)
(Did not actually understand semantics)

“Program semantics does not just manifest in static text”

[me: of course, we can extract more info, even dynamic info for Scalene]

- overfits to spurious textual and task-specific patterns
- distribution shift: program syntax and task requirement changes

Binary analysis (untyped not an issue with WASM, interesting)
- strips variables, arguments, etc. -> lowered to registers

“ML needs to understand semantics, not syntax”
Security requires rigorous understanding of program semantics

Search through binary (binary code similarity), compare to Yahav?

“Data-driven program analysis”

(I really like the program smoothing, which was an idea I tried to get some students interested in)

Oh no, not going into the white box thing for neural nets, too bad

“Programs only make sense when executed”

Hm, ML cannot understand instruction semantics (Carbin work?)

Dynamic analysis - large-scale vulnerability might not scale (maybe because execution is “slow”), potential false negatives

Key high level idea (22 min)
- Learn execution semantics of program
- Transfer knowledge w/o dynamic analysis

1. How do we collect and represent program behavior?
2. How do we train a model?
3. How do we avoid “expensive” dynamic analysis?

Me: guarantees?

Under-constrained micro-execution

ME: How does this not just lead to SEGVs all the time?

ME: Isn't this also time-consuming? (Running these tests, training the model…)

instructions + data flow states (values updated) + control-flow state (covered) + code addresses

I'd love to hear from Subrhansu how conventional or not this encoding is

transformer self-attention + masked language modeling

Embodies synthesis of instructions, or forward/backward interpretation

Mask more "dataflow states" (actual contents - concrete values of registers)

- Very rough initial response to Brian ("already discussed this" (!))
- Good follow-up though - difficult to train model to extrapolate ("best-effort")

Fine-tuning as static analysis using pretrained model

Binary similarity - maximize agreement between two codes compiled from same source code

Future work:
1. Multi-modal software (heterogeneous "modalities")
2. Security applications (automate existing security tasks and enable new security apps)
3. Robustness guarantees (develop & evaluate to ensure strong robustness guarantees)

1. ASTs, specs, physical interaction, natural language like comments, documentation, var names, etc.; configurations; execution traces, coarse-grained system logs
   A. Grounded data-driven program analysis
   B. Automate existing tasks

Correlate events from physical interactions - cyber-threat investigation or debugging system failures
Document-grounded invariant generation
Interactive debugging / interactive reverse engineering

Principled robustness measurement
- random testing
- "How can we systematically explore the program space?"
  - "Transformation-oriented testing" - focus on well-defined program transformations (?) - semantics-preserving syntax alteration, evaluate if model stays invariant wrt changes
- Develop data-driven program analysis with provable robustness by construction
  - Formalize transformations using "symmetry group"
  - "Open to neuro-symbolic approaches"

Yi Ding talk

A Holistic View on Machine Learning for Systems

Co-designs ML and software systems

UChicago, postdoc MIT
Meta

Heterogeneous hardware (CPU, GPU, FPGA, ASIC)
Software (MySQL, Cassandra, Httpd, Nguni, HDFS, Apache Flink, Transformers (hugging face))

- configuration parameters

Maximizing accuracy of ML is not the goal - instead, "holistic view"
- "understanding cost structure of systems problems" - trade offs

Trade model accuracy for reduced data collection (not all samples are equally useful)
- maximizing efficiency over all samples is not necessarily desirable
- Only configs that represent optimal power-performance are relevant
- <u>asymmetric costs - under prediction may be less or more costly than over prediction</u>

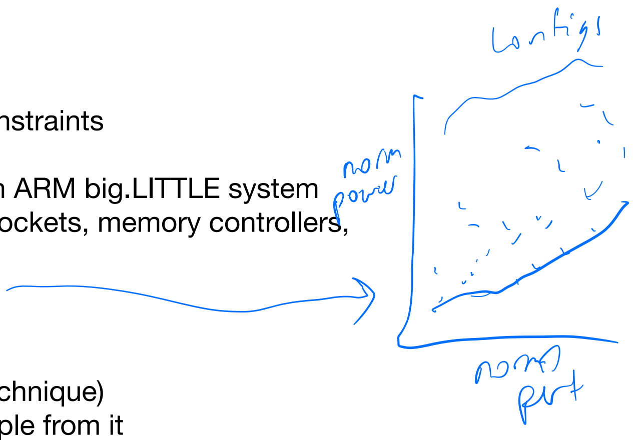
ML for systems
- Neural surrogates
- NURD
- Multi-phase sampling (ISCA)
- NURD (MLSys)

Fundamental ML thrust as well

Minimize energy under performance constraints

SRAD application - all configurations on ARM big.LITTLE system
(# of cores and clock speeds) - also # sockets, memory controllers,
# cores/socket, hyper threading

Optimal frontier - lower convex hull



Multi-phase sampling (new sampling technique)
- predict optimal frontier and then sample from it

Phases:
1. uniform sampling (from lowest to highest) - not random [me: could be quasirandom]
2. Predict
3. Selective sampling (predicted energy efficiency = pred perf / pred power)
   A. Pick from samples from phase 1 with highest efficiency
   B. Predict perf & power from other unsampled configs

(How do you predict the different ones?)

matrix completion methods & hierarchical Bayesian methods

[ME: is the perf not workload dependent? If not, how do you cope with it? Is average OK? How well does this generalize? **Compare to adaptive approaches?**] — used worst-case inputs
[**ME: also isn't perf very closely connected to power? If not, why not?]**

Cost structure: not all configs are equally useful (majority is not)

[ME: similar insight to what I said years ago about training for important customers]

Online-straggler prediction (**NURD**)
- Predict task latency based on task and node features, on-the-fly w/o prior knowledge

Trains model based on which tasks have completed by a time threshold

Use finished tasks, re weight predictions based on a function of dissimilarity between finished and running tasks

Gradient boosted tree for regression predictor
Weighting function:
- adjusted weight >= threshold —> straggler

Propensity score - conditional probability that task belongs to a class of finished tasks given its feature at time t
- use logistic regression since there are just two categories

Use this for reweighting

Evaluated on two public traces from Google and Alibaba Cloud

(For me, holistic here really is "end-to-end")

Meta - "Acela" [arxiv 22]

- over subscribing a server with too many maintenance yields server being withdrawn from production
    - Quantile regression
    - Currently deployed at Meta

AutoML - EuroSys paper - couldn't run experiments

Challenges in ML (or stats, or control theory) for systems?
- must capture causal relationship but ML captures stats
- Rethinking from causal perspectives
    - How to evaluate outcomes w/o bias when controlled experiments are not possible
        - Build causal models that permit observational studies
        - Answer counterfactual questions using offline log data
    - Uncover causality in systems when interference and unobserved confounders exist
        - Build causal models that incorporate domain expertise in computing
        - Develop new intervention strategy for causal discovery

-> resource scheduling and management

Sustainability & Climate in computing
- Reduce carbon footprints
    - Get high model accuracy w/reduced environmental costs

    - Reduce # of samples and environmental cost of each sample
- Explore trade offs between carbon, energy, perf & reliability

Ramesh: big roadblock - explainability of AI is a problem (performance)

GIL+
- interpretability of configurations that lead to low or high performance
- design for interpretability - characterize hierarchical structure (cache misses, cache miss rate, IPC) - new ML perf model