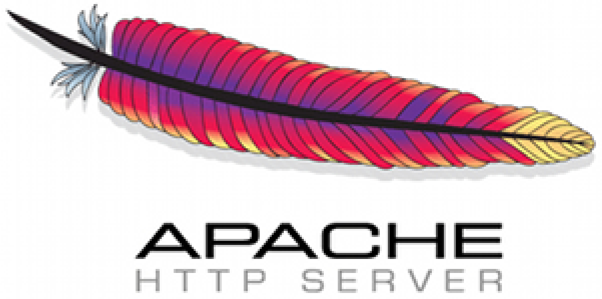
A Clairvoyant Approach to Evaluating Software (In)security

Bhushan Jain, Chia-Che Tsai*, Don Porter

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Stony Brook University *

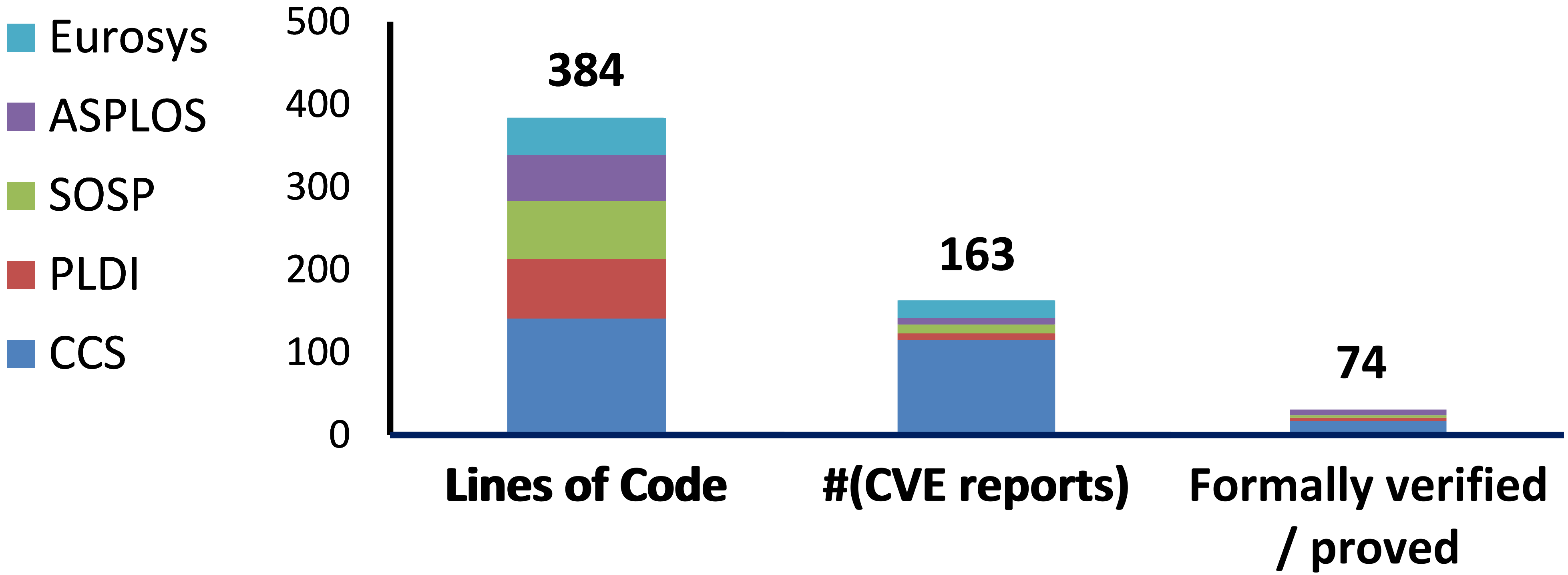
Which is More Secure?



Can we evaluate security empirically?

How do Researchers Evaluate Security Now?

# of papers using the approaches for evaluation or indication of security



Most just use LoC and #CVE reports as prediction of security

Is it a Good Idea to Use Lines of Code?

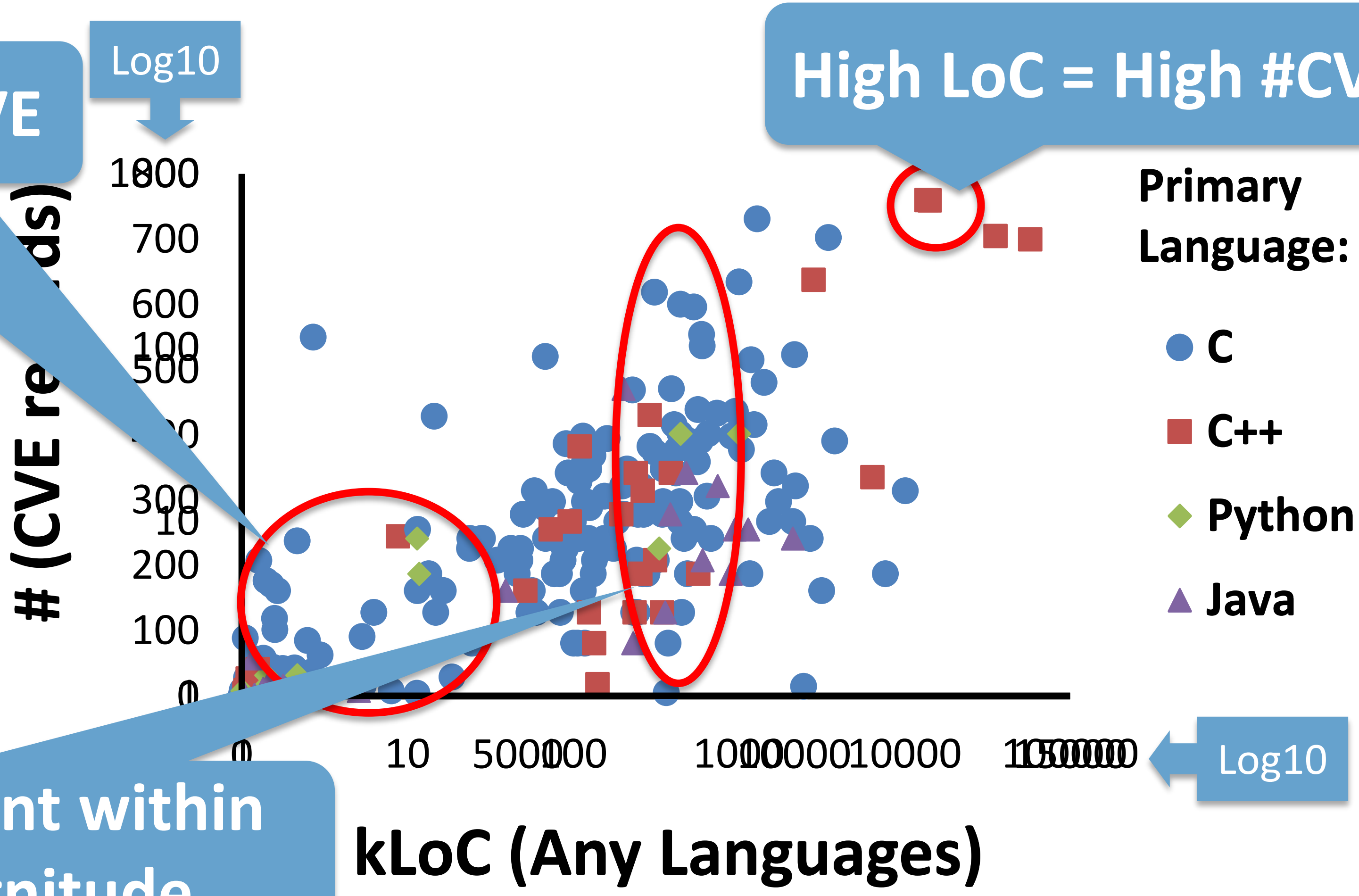
- Conventional wisdom:
  - # of LoC ➔ # of bugs
  - Easy to formally verify or code review small LoC
- "There are, on the average, about 21 bugs per KLoC discoverable" [Gaffney, TOSE '84]
- "Commercial software typically has 20 to 30 bugs for every 1,000 lines of code"

—CMU's CyLab quoted by WIRED magazine in 2004

LoC seems logical way to predict security problems

Is LoC Correlated to #(Vulnerabilities)?

- 164
  - Open-source
  - 5-20 years in CVE DB
  - Primary language:
    - C: 126
    - C++: 20
    - Python: 6
    - Java: 12



Low LoC = low #CVE

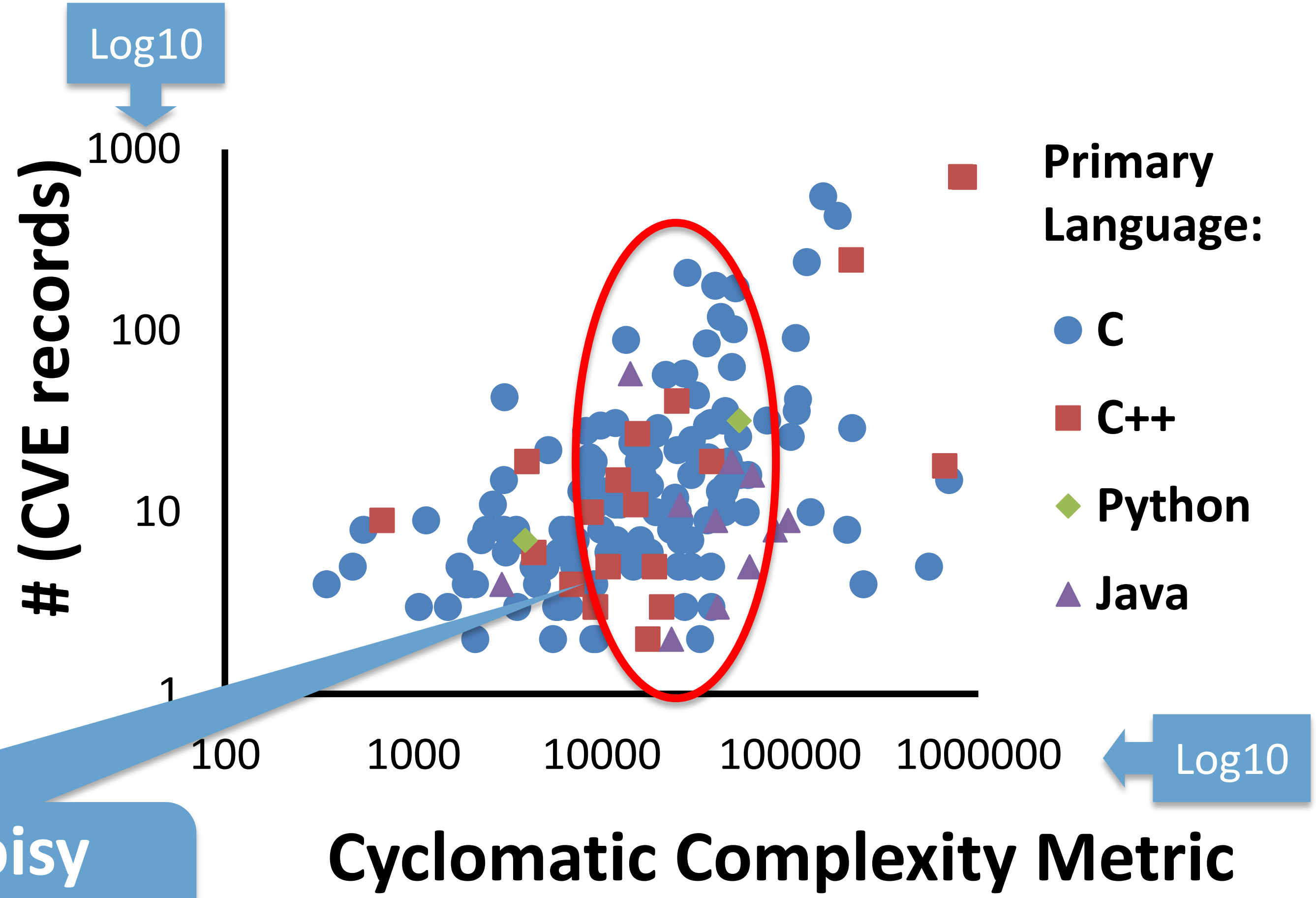
High LoC = High #CVE

LoC not significant within orders of magnitude

LoC not a reliable predictor of vulnerabilities

May be we can try program complexity?

- Conventional wisdom:
  - Complex program ➔ high probability of vulnerabilities
- Cyclomatic Complexity [McCabe, TOSE '76]: # of linearly inde



Complexity too is noisy within orders of magnitude

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Stony Brook University

Complexity not necessarily correlated to the #CVE reports

Other Conventional Wisdom

- Large attack surface ➔ more opportunities for attacker
  - Relative Attack Surface Quotient (RASQ) [Howard et al., 2005]
  - Resources, communication channels, access rights for attackers
  - Specific to configuration
- Secure design guidelines ➔ less # of vulnerabilities
  - Design Security Standards
    - NIST 800-55, Common Criteria, ISO/IEC 27004
  - Qualitative, subjective, no precise evaluation model

These wisdom are mostly qualitative

Code Properties Reveal Security Aspects

Code Properties		Security Aspects
Choice of language	→	Safety of languages & runtimes
Lines of code	→	Difficulty of code-checking/verification
Cyclomatic complexity	→	Variant of execution paths
Attack surface	→	Number of paths to attack

Weighted aggregation covers more security aspects

Code properties in isolation doesn’t evaluate security. Aggregation may help.

Ideal Security Evaluation

- Predict risk of compromise
  - Attacker effort (qualitative)
  - Vulnerabilities (quantitative)
- Help improve code over versions
- Improved code = Improved metric score
- Compare similar software

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Stony Brook University

Predict # and severity of *all* vulnerabilities

Can We Just Predict Bugs Instead?

- “Many security holes in software are the result of software bugs...”

  — Seth Hallem, CEO of Coverity, 2004

- Vast research predict bugs based on code properties
  - A weighted correlation of code properties and bugs
  - Too many false positives
  - Need human intervention

Maybe #bugs is a good way of predicting vulnerabilities

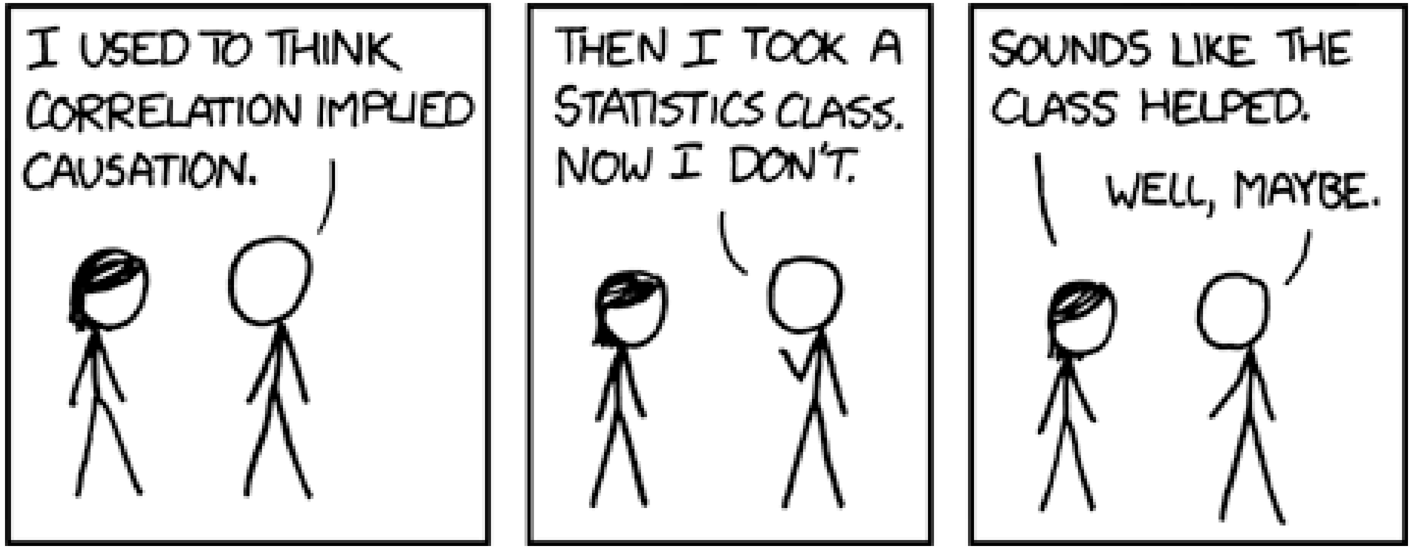
Bugs and Vulnerabilities: It’s Complicated!

- ➢ Bugs don’t foreshadow vulnerabilities
  - – Study [Camilo et al., MSR '15] : # of bugs ⇏ # of vulnerabilities
  - – Buggiest files ≠ files with many vulnerabilities
- ➢ Code properties may have different relation to vulnerabilities
  - – Study [Shin et al., TOSE '11] : some code properties are indicative
  - – #functions, #declarations, #preprocessing lines, #branches, #input and output arguments to a function

Vulnerabilities may correlate with code properties differently

Let’s Learn the Correlation

- Hypothesis:
  - Machine learnable correlation between code properties & vulnerabilities


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

https://imgs.xkcd.com/comics/correlation.png

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Stony Brook University

May not be perfect, but we have to do SOMETHING

What Do We Need?

- Large ground truth data
  - More than 80,000 vulnerabilities in 400 applications and systems
- Representative data
  - #CVE Reports vary based on maturity and attention received
- May be missing security-indicative code properties
  - Any suggestions are most welcome!

Normalize for missing data

Calculating Other Code Properties

- Data flow analysis
  - # of expressions, functions, data structures
- Control flow analysis
  - # of calling and return targets
- Abstract interpretation
  - # of paths triggered by specific range of inputs

Static analysis can help collect more properties

Vulnerability Information

CVE-2016-8740 Detail

Impact

CVSS v3 Base Score:	7.5
Impact Score:	3.6
Exploitability Score:	3.9
Attack Vector (AV):	Network
Attack Complexity (AC):	Low
Privileges Required (PR):	None
User Interaction (UI):	None
Scope (S):	Unchanged
Confidentiality (C):	None
Integrity (I):	None
Availability (A):	High

Severity & attack properties

Vulnerability Type

Root causes

- Input Validation (CWE-20)
- Resource Management Errors (CWE-399)

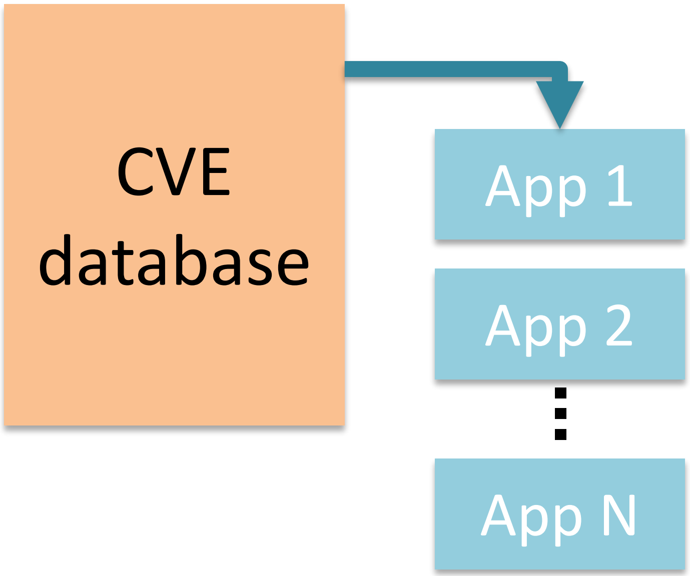
Configuration

- cpe:2.3:a:apache:http_server:2.4.17:*:*:*:*:*:*:*
- cpe:2.3:a:apache:http_server:2.4.18:*:*:*:*:*:*:*
- cpe:2.3:a:apache:http_server:2.4.19:*:*:*:*:*:*:*
- cpe:2.3:a:apache:http_server:2.4.20:*:*:*:*:*:*:*
- cpe:2.3:a:apache:http_server:2.4.21:*:*:*:*:*:*:*

Affected versions

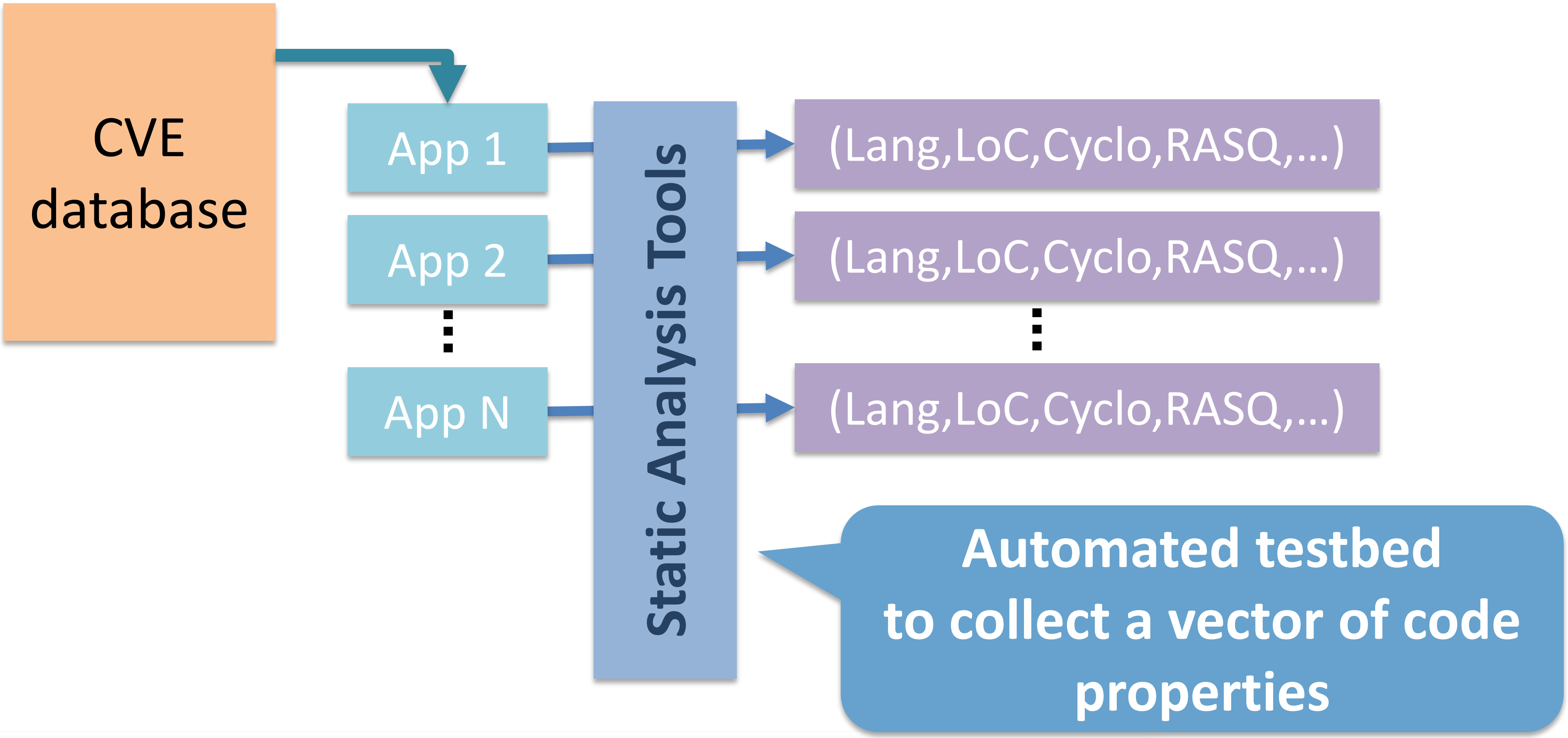
A vector of information available from CVE reports

System Proposal



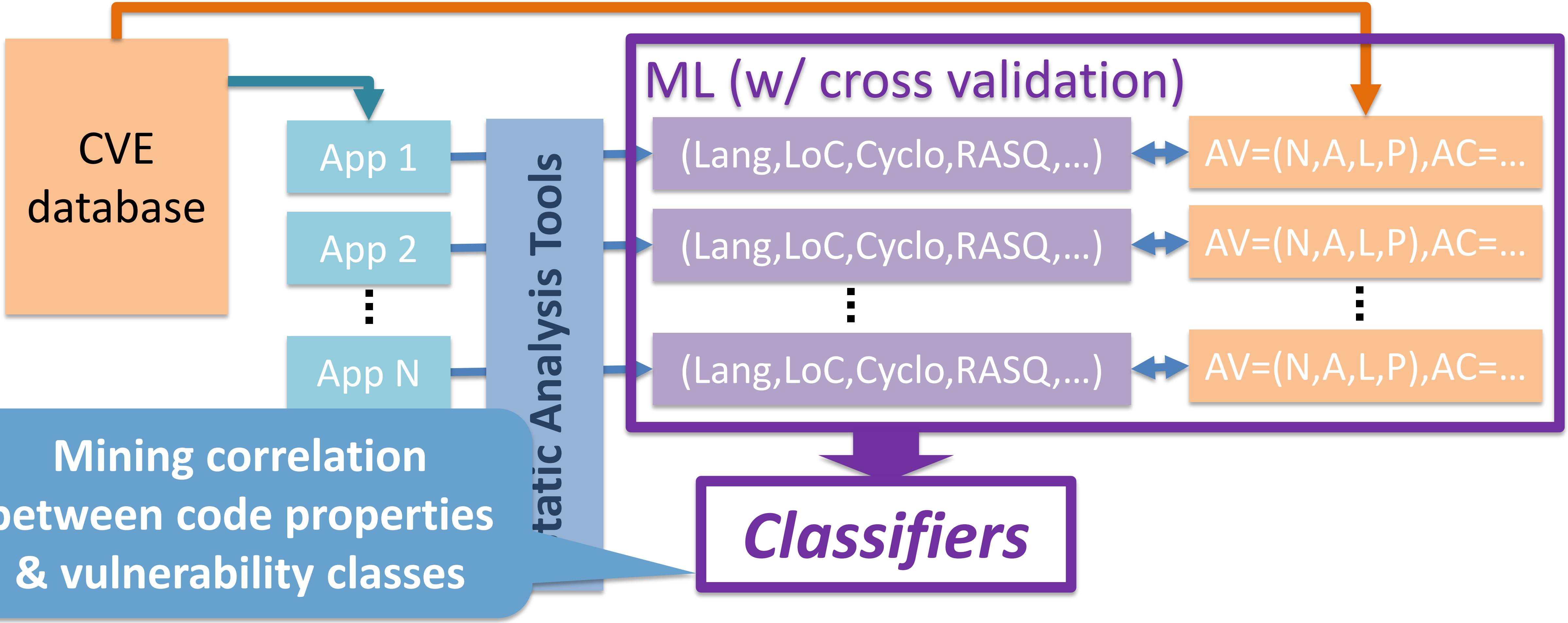
Selecting ground truth: Applications with converging CVE history

System Proposal



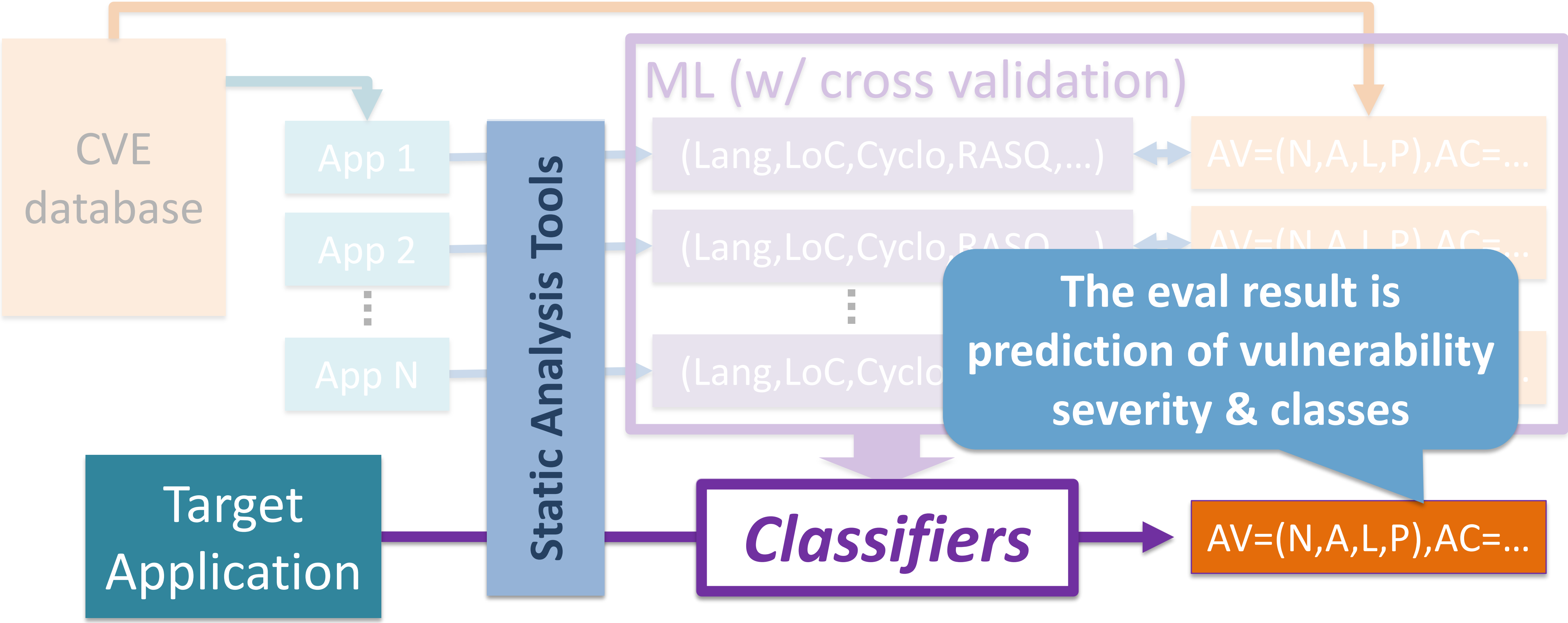
Automated testbed to collect a vector of code properties

System Proposal



Mining correlation between code properties & vulnerability classes

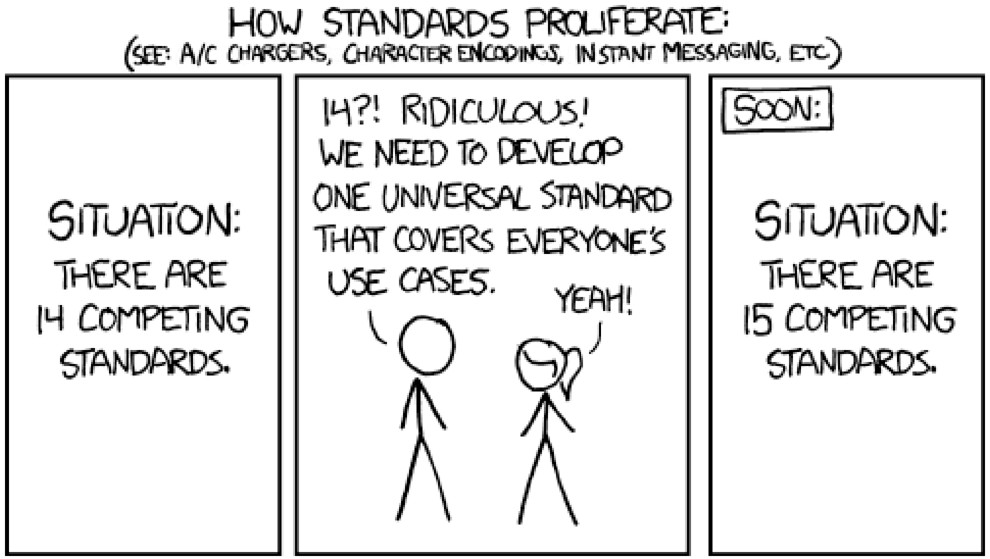
System Proposal



The eval result is prediction of vulnerability severity & classes

Classifiers predict #, severity, and classes of vulnerabilities

Oh No! Not Another Security Metric!


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

https://imgs.xkcd.com/comics/standards.png

- Our metric is:
  - Easily extendable
  - Can only improve with time (more CVE data)
  - Doesn't rely on only one code property
  - Gives useful feedback to developers

Supposed to be the one metric to rule them all!

Using the Metric

we propose to build a series of classifiers for SW vulnerability:

EX: $\mathbf{E\ [AV\ _{(Attack\ Vector)} = N\ _{(Network)}\ ] =}$
$\mathbf{Lang \times W_0 + Log10(LoC) \times W_1 + Cyclo \times W_2 + RASQ \times W_3\ + \ldots}$

- Confirm or update conventional wisdom
- Balance multiple properties
- Hint possible security enhancement:
  - Defenses against potential attacks
  - Improve code property

More than just another "security score"!

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Stony Brook University

Metric can be integrated with regression testing

Conclusion

- LoC, complexity, other metrics are noisy
- We propose to approximate risk of having a vulnerability
- Learn weighted relation of code properties to vulnerabilities
- Challenge:
  - Extract meaning from incomplete ground truth