Internal and External Metrics for Predicting Attack-prone Components

Michael Gegick and Laurie Williams
North Carolina State University

7 April 2008

Where Should Security Efforts Begin?

(Reliability context)

Fault-prone component → **Vulnerability-prone component**

Likely to contain faults → Likely to contain vulnerabilities

Failure-prone component → **Attack-prone component**

Likely to fail → Likely to be exploited

(Security context)

Fault- and vulnerability-prone

- Pre-execution context
- Some faults remain latent.
- Vulnerabilities can have a wide range of severity and likelihood of exploitation.

Failure- and attack-prone

- Execution context
- Execution of a fault is a failure.
  - Usage
- Exploitation of a vulnerability is an attack.
  - Ease of attack and value of asset (risk)

Research Outline

- **Goal** - identify ***where*** vulnerabilities most likely exist in a software system so fortification efforts can focus on those problem areas first.
- **Research objective** – create/validate statistical models that identify good and early predictors of security problems.
- **Candidate predictors**
  - Churn
  - Size (SLOC)
  - FlexeLint static analysis tool alerts (audited and un-audited)
    - All alerts
    - Null pointers
    - Memory leaks
    - Buffer overflows
  - Non-security failures (general reliability problems)
- **Methodology** - model values of the predictors and counts of security-based failure reports for a given component in the software system.
- ***Not*** identify exploits or qualify the vulnerabilities.

Case Study

- Commercial telecommunications software system.
- 38 components
  - 13 components left out → 25 components in analysis
  - Each component consists of multiple files
- 1.2 million lines of C/C++ source code (in the 25 components)
- Deployed to the field for two years
- 52 failure reports were classified as security-based problems
  - Vendor's security engineer verified our report

Attack-prone Components

- **Pre-release attack-prone components (10)**
  - Pre-release robustness testing at system level

- **Post-release attack-prone components (4)**
  - Customer-reported
    - “attacks” – vulnerabilities that could have been exploited
      - » No attacks reported

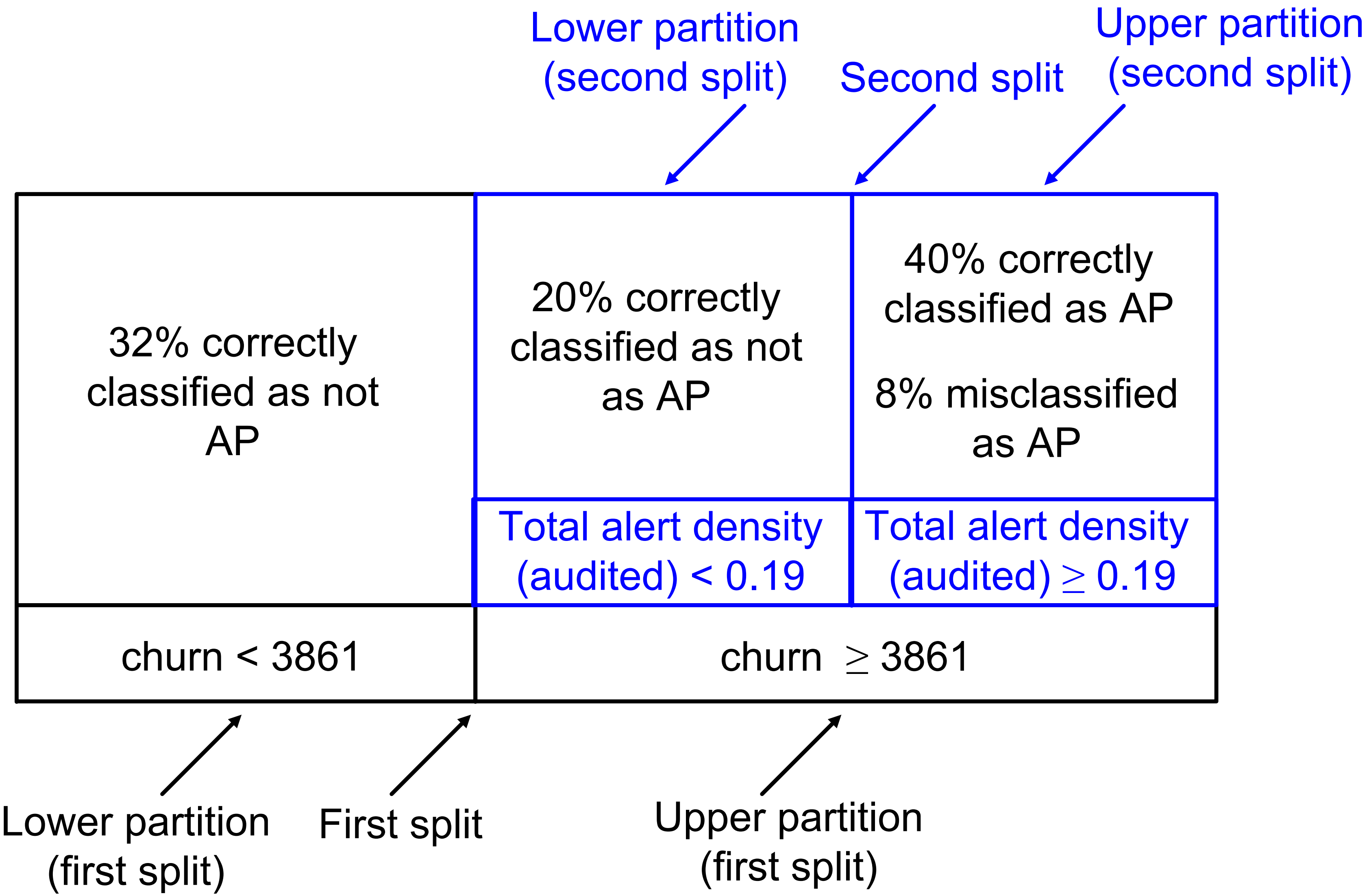
- Attack-prone (not vulnerability-prone)
  - Vulnerabilities were found during system execution

- All post-release attack-prone components are also pre-release attack-prone

Correlations

Metric	Security failure count	Spearman rank correlation (p-value)
FlexeLint alerts	Sum pre- and post-release	0.39 (.06)
Churn	Pre, post- or both	No correlation
SLOC	Post-release	0.43 (0.03)
Sum pre- and post-release non-security failure count	Sum pre- and post-release	0.82 (< .0001)

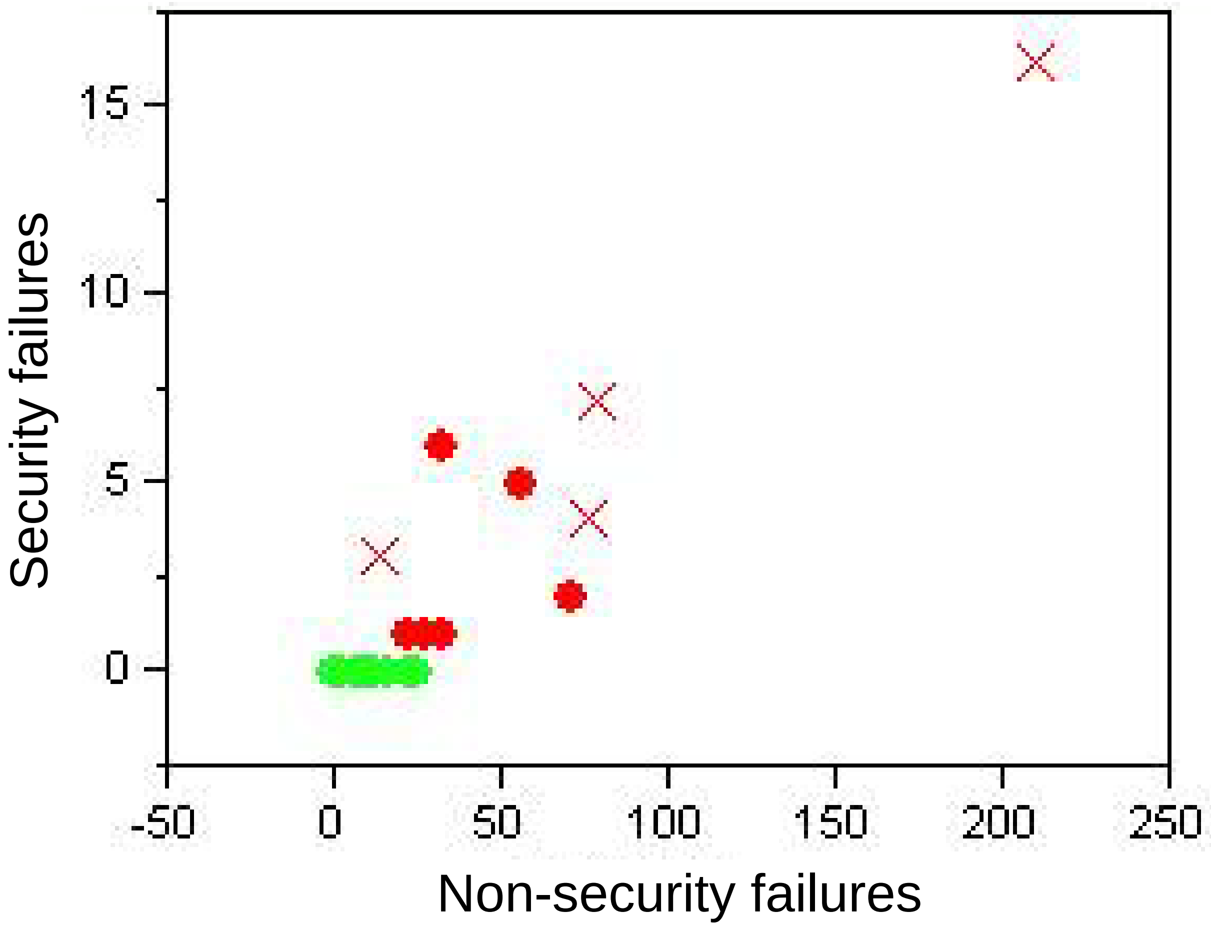
Classification and Regression Tree Analysis (CART)



Attack-prone Prediction Results from CART

Metric	Type I	Type II	R^2	Cross-validated R^2	ROC
alerts	7 (28%)	0%	31.5%	19.4% X	76.7%
churn	7 (28%)	0%	32%	30% X	77%
SLOC	--	--	--	--	--
alerts, churn, SLOC	2 (8%)	0% ✓	68%	61% ✓	93% ✓
total pre-release failure count	2 (8%)	0% ✓	68%	64% ✓	93% ✓

Non-security and Security Failure Counts



X post-release attack-prone ● pre-release attack-prone ● non attack-prone

All post-release attack-prone components are also pre-release attack-prone components.

Predicting Attack Counts

Pre-release non-security failures are good predictors of pre- and post-release security failures (in our setting).

- Negative binomial distribution
  - Standard error = 0.56
  - $p<.0001$
  - Value/DF = 0.92

Limitations

- Small sample size – 25 components
- Moderate R^2 values
- Only one data set
- Only one static analysis tool
  - Not representative of all static analysis tools.
- Testing effort not necessarily equivalent on all components

The Coupling Effect

- Coupling effect – “simple” problems found by FlexeLint are ***coupled*** to more complex problems in design and operation.
  - E.g. - buffer overflow (simple) in same file as an access control issue.
    - Developer does not understand buffer overflows (a potential security problem) which could indicate that they do not understand the encryption requirements for an authentication mechanism.
    - Customer requirements are unclear → design is ambiguous[1] → developers make guesses about the ambiguous designs.
  - Failure reports
    - 60% - coding bugs **(hopefully found by static analysis tools)**
    - 40% - design flaws and operational vulnerabilities
    - The “simple” 60% can predict the “complex” 40%

McGraw, G. Software Security: Building Security In, Boston, Addison-Wesley, 2006.

Summary

- Components with high code churn and FlexeLint alerts are attack-prone.
- Components with many non-security failures are attack-prone.
- Reliability testers can find security vulnerabilities.

IAD

Looking for industrial partners!

Thank you!

mcgegick@ncsu.edu
williams@csc.ncsu.edu