Challenges with Applying Vulnerability Prediction Models

Patrick Morrison[1], Kim Herzig[2], Brendan Murphy[2], Laurie Williams[1]

[1]Department of Computer Science, North Carolina State University

[2]Microsoft Research, Cambridge, UK

pjmorris@ncsu.edu kimh@microsoft.com
bmurphy@microsoft.com lawilli3@ncsu.edu

The Science of Security initiative is funded by the National Security Agency.

Problem Scale

- Windows 7-8
  - Over 70 million lines of code, ~ US Eastern Seaboard population
- Granularity
  - Binaries ~ cities
  - Files ~ neighborhoods
- Objects of interest
  - Defects ~ Doctors
  - Vulnerabilities ~ Vascular surgeons

Problem

In the absence of exhaustive software testing, software development teams have to choose what, and how much, to test.

Solution Idea

If there are machine-measurable differences between defect-prone code and more benign code, those differences can be used to automate identification of problematic code and focus verification efforts.

Software Metrics

- Measures of code size, complexity, change, dependency, and other characteristics of source code.

Defect Prediction Models

- Researchers and practitioners have applied statistical prediction modeling techniques to various software metrics to predict defect-prone sections of code.

The Science of Security initiative is funded by the National Security Agency.

Windows DPMs

Independent Variables	Precision	Recall
Organizational structure [17]	0.79	0.80
Code churn [5]	0.79	0.66
Code dependencies [9]	0.75	0.69
Pre-release defects [6]	0.74	0.63

- Used within Windows Development Teams for risk analysis, planning, resource allocation, dashboards

Windows DPMs

Independent Variables	Precision	Recall
Organizational structure [17]	0.79	0.80
Code churn [5]	0.79	0.66
Code dependencies [9]	0.75	0.69
Pre-release defects [6]	0.74	0.63

- Used within Windows Development Teams for risk analysis, planning, resource allocation, dashboards

Windows (Vista) VPM, Binary-level

Independent Variables	Precis ion	Recal l
Size, Churn, Organization, Dependencies [1]	0.40-0.67	0.20-0.40

The Science of Security initiative is funded by the National Security Agency.

Goal

The goal of this research is to measure whether ***vulnerability prediction models*** *built using* ***standard recommendations*** *perform well enough to provide* ***actionable results*** *for engineering resource allocation*

Vulnerability Prediction Models (VPMs)

- Dependent variable: Vulnerability-prone
- Independent variables: Software metrics
- Learner: Statistical models
- Train learner, predict presence of vulnerabilities

Standard Recommendations

- Size, Churn, Complexity, Dependency Metrics
- Multiple learners - “choice of learning method is far more important than which set of the available data is used for learning.” [12]
- Cross-validation

Actionable Results

- Actionable: Would an engineer use a VPM?
- Microsoft engineers use Defect Prediction Models (DPMs) to identify weak areas, and to plan resource allocation.
- If the VPM correctly identified vulnerability-prone sections of code small enough to be inspected by the engineer, yes.

Measuring Actionable Results

- Inspection effort required to perform security reviews on code areas suggested by the VPM.
  - 100-1000 lines per hour [37]
- Recall (true prediction rate)
- Precision (positive prediction rate)

Research Questions

- **RQ1** Can we replicate VPMs proposed by Zimmermann et al. [1] achieving comparable prediction accuracy on binary level for two newer version of Windows?
- **RQ2** How does granularity affect classification performance?
- **RQ3** How does the choice of statistical learner affect classification performance?
- **RQ4** Are VPMs predicting vulnerable Windows binaries actionable with respect to security inspection effort?

The Science of Security initiative is funded by the National Security Agency.

Experiments

- Built VPMs for Windows 7, 8
- Binary- and source file-level granularities
- Dependent variable: presence of post-release vulnerabilities in first six months
- 29 Metrics
- 6 Learners
- 100-fold cross-validation of 2/3 training, 1/3 testing SRS splits

The Science of Security initiative is funded by the National Security Agency.

Metrics Used

- Churn metrics [5].
  - Theory: that change is more likely to introduce error than its absence. Churn measures are relative to a time period; the period for all presented calculations is between the start and RTM date of the project.
- Complexity metrics [3]
  - Theory: that more complicated code is more likely to exhibit errors.
- Dependency metrics [9]
  - Theory: the degree to which a piece of code is depended upon, or depends upon other code, influences its impact on software vulnerabilities.
- Legacy metrics.
  - Theory: Code written before Microsoft's 'Security Reset' may be more likely to contain vulnerabilities.
- Size metrics.
  - Theory: Larger source files are more prone to defects and vulnerabilities.
- Pre-Release vulnerabilities
  - Theory: "usual suspects"

Learners Used

- Logistic Regression (LR)
  - Generalized linear model using a logistic function.
- Naïve Bayes (NB)
  - Simple probabilistic classifier assuming strong independence of the independent variables.
- Recursive Partitioning (RP)
  - Decision tree variant, model represented as a binomial tree
- Support Vector Machine (SVM)
  - Classifies data by determining a separator that distinguishes the predicted classes with the largest margin.
- Tree Bagging (TB)
  - Decision tree variant, uses bootstrapping to stabilize the decision trees.
- Random forest (RF)
  - Decision tree variant, builds ensemble of decision trees

RQ1: Replicate Windows VPM performance?

- Yes

	[Zimmerman10]	Current paper
Granularity	Binary	Binary
N Entities	1000's	1000's
% Vulnerable	"very low"	9.5
Recall	0.20-0.40	0.04-0.42
Precision	0.40-0.67	0.11-0.76

RQ2: Impact of Granularity

- Recall and Precision are much worse at source file granularity

	Windows 7		Windows 8	
	Precision	**Recall**	**Precision**	**Recall**
		Binary level		
LR	0.5	0.12	0.32	0.09
NB	0.3	**0.42**	0.11	**0.4**
RF	**0.76**	0.27	**0.69**	0.07
RP	0.51	0.22	0.23	0.07
SVM	0.51	0.13	0.64	0.04
TB	0.69	0.13	0.45	0.1
		File level		
LR	0.01	0	0	0
NB	0.07	**0.14**	**0.01**	**0.01**
RF	**0.47**	0.02	0	0
RP	0.21	0.04	0	0
SVM	0.38	0.02	0	0
TB	0.36	0.03	0	0

RQ3: Impact of Learner

- Statistical learner choice does affect performance
- Naïve Bayes and Random Forests perform best on our highly imbalanced dataset

	Windows 7		Windows 8	
	Precision	**Recall**	**Precision**	**Recall**
	Binary level			
LR	0.5	0.12	0.32	0.09
NB	0.3	**0.42**	0.11	**0.4**
RF	**0.76**	0.27	**0.69**	0.07
RP	0.51	0.22	0.23	0.07
SVM	0.51	0.13	0.64	0.04
TB	0.69	0.13	0.45	0.1
	File level			
LR	0.01	0	0	0
NB	0.07	**0.14**	**0.01**	**0.01**
RF	**0.47**	0.02	0	0
RP	0.21	0.04	0	0
SVM	0.38	0.02	0	0
TB	0.36	0.03	0	0

RQ4: Is our VPM actionable?

- No
- Inspection effort at source file granularity is ~ 2-3 order of magnitudes smaller (< day, versus 100's of days) than binary granularity
- However, low Recall and Precision performance yield too few correct predictions and too many false positives

Other findings

- Complexity wasn't a consistent predictor of vulnerability. The correlations ranked differently between Windows 7 and 8.
- Churn was predictive
- Age was predictive (older is worse), implying that Microsoft's SDL efforts have been effective

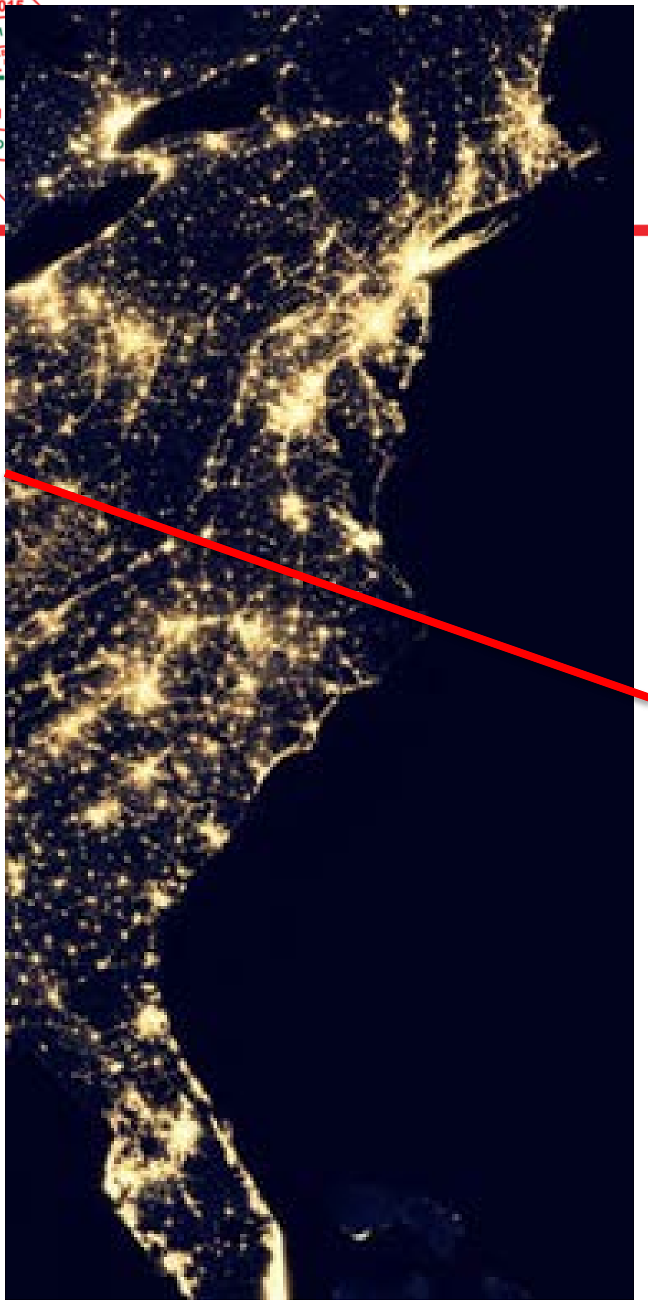
Why such a poor VPM?

- Vulnerabilities are rare - known vulnerable source files in Windows 8 comprise 1/3% of the codebase.
- Variability in learner performance suggests that we don't yet have a (good) model for how metrics indicate vulnerability-proneness. New metrics, and new approaches are needed.
- We conjecture that security domain knowledge must be added to VPMs before acceptable performance will be achieved.

The Science of Security initiative is funded by the National Security Agency.

Future Work

- New code, process metrics
- Focus the VPM on the Attack Surface



Attack Surface Approximation

- **What if we could cut that in half?**
- "Approximating Attack Surfaces with Stack Traces", C. Theisen, K. Herzig, P. Morrison, B. Murphy, L. Williams, ICSE 2015.