Metrics-based Software Vulnerability Discovery Model to Support Cybersecurity Test & Evaluation

Julia Sorrentino, Priscila Silva, Baye Gaspard, Gokhan Kul, and Lance Fiondella

UMass | Dartmouth

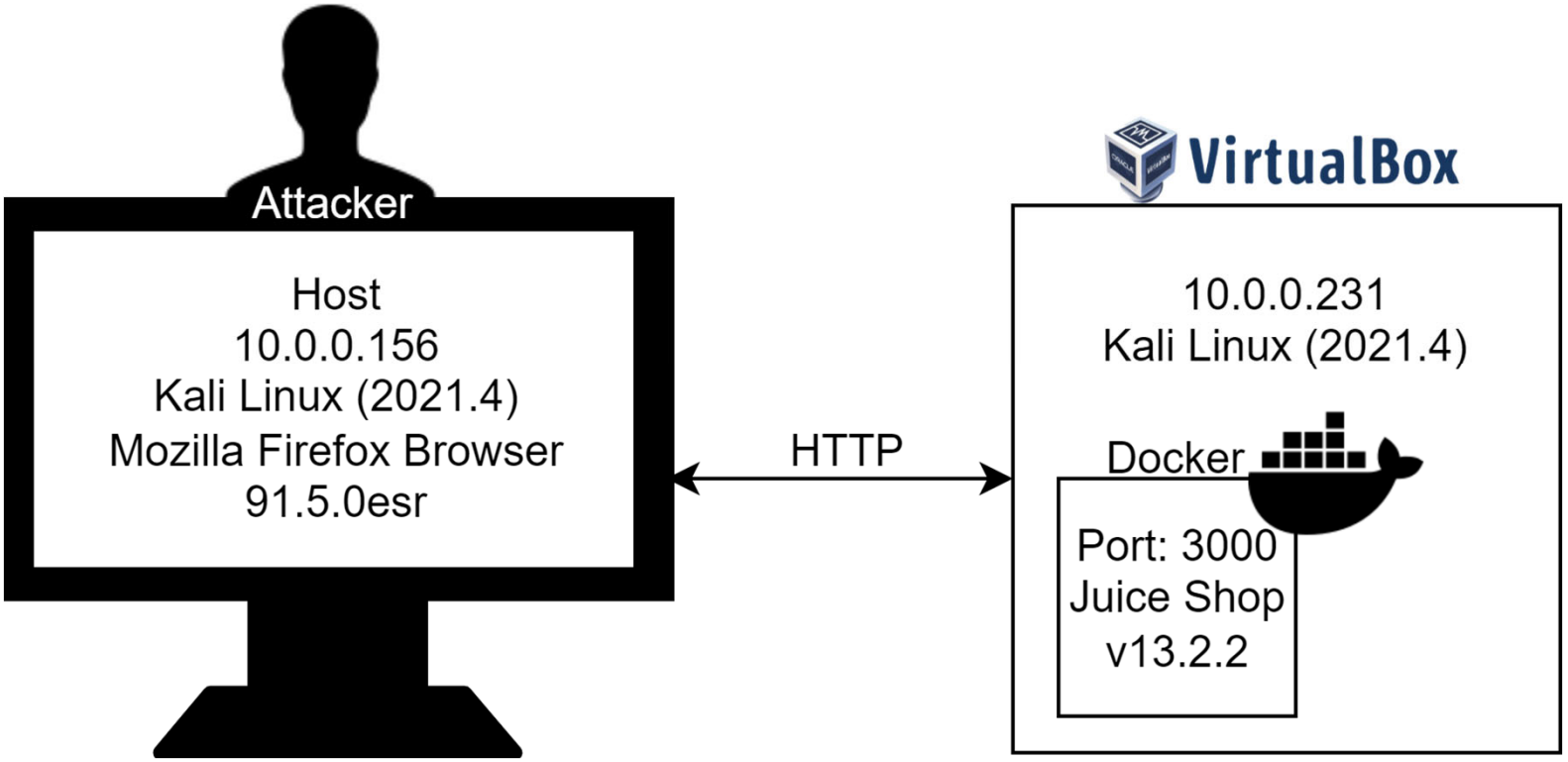
Background and Introduction

- Software highly versatile
  - Provides wide variety of functionality in applications and software-enabled systems
- Potentially negative impacts of software exploitation cast dark shadow over otherwise promising nature of software
  - Monitor and control infrastructure, homeland defense, and national security
- Effort dedicated to vulnerability taxonomies, techniques, and tools
- Some attention to vulnerability discovery models (VDM)
- Only consider amount of time spent testing
  - Prevents more detailed guidance on relative effectiveness of alternative vulnerability testing tools and techniques

Contributions

- Study comparing software VDM incorporating covariates and Alhazmi-Malaiya Logistic (AML) model
  - Covariate VDM links specific test activities and tools to vulnerability discovery, organizations
  - AML flexible and highly cited VDM without covariates
- Target application identified and subjected to multiple vulnerability discovery tools and techniques
- Software VDM incorporating covariates
  - More accurately tracked and predicted number of vulnerabilities discovered in future intervals as function of penetration testing activities performed
  - Achieved significantly better goodness of fit, despite information theoretic measures penalized covariate models for additional parameters

Environment Architecture Setup



- Application emulates modern day web application for an online Juice Shop marketplace with complex functionality, containing a wide variety of vulnerabilities

- A01: Broken Access Control
- A02: Cryptographic Failures
- A03: Injection
- A04: Insecure Design
- A05: Security Misconfiguration
- A06: Vulnerable and Outdated Components
- A07: Identification and Authentication Failures
- A08: Software and Data Integrity Failures
- A09: Security Logging and Monitoring Failures
- A10: Server-Side Request Forgery

Data Collection

TABLE II
DATA COLLECTED DURING PENETRATION TESTING ACTIVITIES

T	K	OWASP Top 10										Tools		
		A01	*A02*	*A03*	*A04*	*A05*	*A06*	*A07*	*A08*	*A09*	*A10*	Z	B	M
1	4	2	0	0	1	1	0	0	0	0	0	147	42	0
2	12	2	1	3	1	1	1	2	0	1	0	375	402	104
3	8	1	1	2	0	0	0	3	0	1	0	0	521	195
4	6	4	1	0	1	0	0	0	0	0	0	11	123	34
5	3	0	1	1	1	0	0	1	0	0	0	0	45	6
6	4	2	0	0	1	0	0	0	1	0	0	0	71	97
7	3	1	0	1	0	0	0	1	0	0	0	0	214	0
8	3	1	0	1	0	0	0	0	0	0	1	0	86	107

- **T** – vector of discrete time interval (approximately 5 hours each)
- **K** – vector of number of vulnerabilities discovered in each time interval
- ***A##*** – Number of OWASP Top 10 category vulnerabilities discovered
- **Z**, **B**, **M** – vector of seconds spent executing tools (Zap, Burp, or manual inspection)

Discrete Cox Proportional Hazards NHPP SRGM

- MVF predicts number of vulnerabilities discovered up to and including n^{th} interval

$$m(x) = \omega \sum_{i=1}^{n} p_{i,x_i} \tag{1}$$

  - Given covariates x, where $\omega > 0$ denotes number of vulnerabilities that would be discovered with infinite testing
- Probability that vulnerability discovered in interval i, given it was not discovered in first $(i - 1)$ intervals

$$p_{i,x_i} = \left(1 - \left(1 - h(i)\right)^{g(x_i;\beta)}\right) \prod_{k=1}^{i-1} \left(1 - h(k)\right)^{g(x_k;\beta)} \tag{2}$$

  - $h(\cdot)$ - baseline hazard function
  - β - vector of covariate coefficient within Cox proportional hazards model

$$g(x_i;\beta) = \exp(\beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_m x_{im}) \tag{3}$$

Hazard Functions

- Can be incorporated into Equation (2)
  - Geometric (GM):

$$h(b) = b \tag{4}$$

    - where $b \in (0,1)$
  - Second order Negative binomial (NB2):

$$h(i;b) = \frac{ib^2}{1 + b(i-1)} \tag{5}$$

    - where $b \in (0,1)$ and 2 indicates order
  - Second order Discrete Weibull (DW2):

$$h(i;b) = 1 - b^{i^2-(i-1)^2} \tag{6}$$

    - where $b \in (0,1)$ and 2 indicates order

Log-Likelihood Function

- LL function of discrete Cox proportional hazard NHPP SRGM

$$LL(\boldsymbol{\gamma}, \boldsymbol{\beta}, \omega) = -\omega \sum_{i=1}^{n} p_{i,x_i} + \sum_{i=1}^{n} y_i \ln(\omega) + \sum_{i=1}^{n} y_i \ln\left(p_{i,x_i}\right) - \sum_{i=1}^{n} \ln(y_i!) \quad (7)$$

  - $\boldsymbol{\gamma}$ is vector of hazard function parameters
  - $\boldsymbol{\beta}$ is vector of m covariate coefficients
  - ω is the total number of vulnerabilities to be discovered
  - $\boldsymbol{y}_i$ is the number of defects discovered in the i^{th} interval
  - $\boldsymbol{y}_n$ is vector of vulnerabilities discovered in each of the n intervals
  - $\boldsymbol{x}$ is the given covariate data

Log-Likelihood Function (2)

$$\hat{\omega} = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} p_{i,x_i}} \tag{8}$$

- Substitute for $\hat{\omega}$ in LL function to obtain reduced log-likelihood (RLL) function

$$\frac{\partial RLL}{\partial \boldsymbol{\beta}} = 0 \tag{9}$$

And

$$\frac{\partial RLL}{\partial \boldsymbol{\gamma}} = 0 \tag{10}$$

Alhazmi-Malaiya Logistic (AML) Model

- Mean Value Function

$$m(t) = \frac{B}{Bce^{-ABt} + 1} \qquad (11)$$

  - B - number of vulnerabilities that would be discovered with indefinite testing
  - A and c - constant of proportionality characterizing vulnerability discovery rate

- Vulnerability Discovery Intensity Function

$$\lambda(t) = \frac{B^3 cAe^{-ABt}}{(Bce^{-ABt}+1)^2} \qquad (12)$$

Alhazmi-Malaiya Logistic (AML) Model (2)

- Likelihood Function

$$LL(\mathbf{T},\mathbf{K};\theta) = \sum_{i=1}^{n} k_i \log(m(t_i) - m(t_{i-1}) - \sum_{i=1}^{n} \log(k_i!) - \sum_{i=1}^{n} (m(t_i) - m(t_{i-1})) \quad (13)$$

  - Where $<\mathbf{T},\mathbf{K}>=<(t_1, k_1), (t_2, k_2),\ldots,(t_n, k_n)>$
  - t_i is the time at which the i^{th} interval ended
  - k_i is the number of vulnerabilities discovered in interval i
  - $\theta = \{A, B, c\}$ is the vector of model parameters

$$\frac{\partial LL}{\partial \theta} = 0 \quad (14)$$

Model Assessment

- Sum of Square Error (SSE):

$$SSE = \sum_{i=1}^{n} \left(N(i) - \hat{m}(i) \right)^2 \qquad (15)$$

- Predictive Sum of Square Error (PSSE):

$$PSSE = \sum_{i=(n-l+1)}^{n} \left(N(i) - \hat{m}(i) \right)^2 \qquad (16)$$

  - $N(i)$ - vector of vulnerabilities discovered by time t_i
  - $\hat{m}(t_i)$- fitted model's estimate of number of vulnerabilities discovered

Model Assessment (2)

- Akaike Information Criterion (*AIC*):

$$AIC = 2\nu - 2LL(\hat{\gamma}) \quad (17)$$

  - ν - number of model parameters
  - Penalizes model by 2 points for each parameter

- Bayesian Information Criterion *(BIC)*:

$$BIC = \nu \log(n) - 2LL(\hat{\gamma}) \quad (18)$$

  - Penalty size includes the sample size (n)

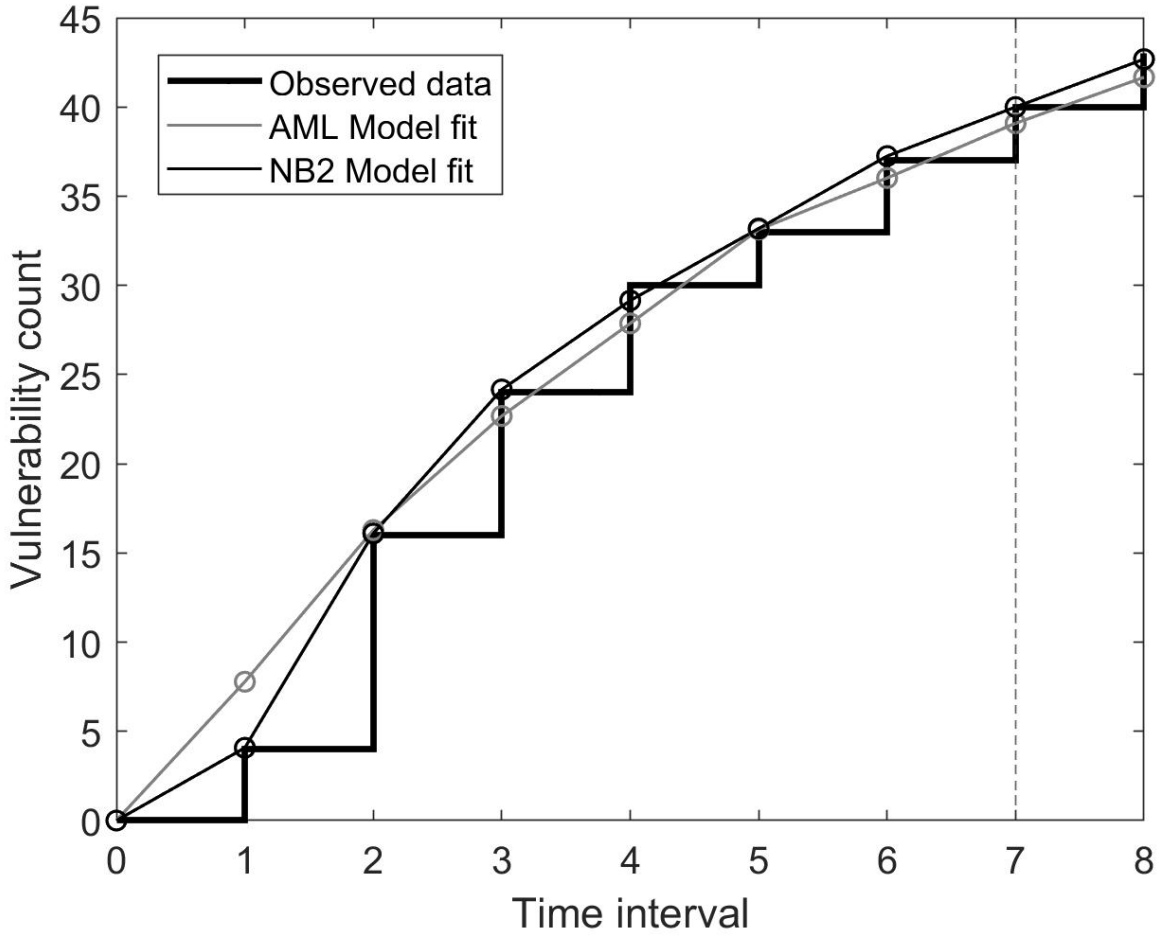
Goodness of Fit Model Assessment

TABLE III
GOODNESS OF FIT VULNERABILITY DISCOVERY MODELS WITH AND WITHOUT COVARIATES

Model	SSE	$PSSE$	AIC	BIC
AML	34.2907	5.6025	47.0145	47.2528
AML ($Z_i + B_i + M_i$)	21.5592	1.7770	41.8812	42.1195
GM	6.6994	2.7028	35.5228	35.9200
NB2	0.9573	0.1007	**34.9565**	**35.3537**
DW2	13.8840	2.8550	40.8420	41.2390
DW3	1.2286	0.3291	36.9901	37.4667
S	**0.8891**	**0.0857**	36.9387	37.4153
TL	6.4349	2.5106	37.5281	38.0047
IFRSB	95.1872	8.9720	65.7081	66.1053
IFRGSB	6.4503	2.5154	37.5306	38.0072

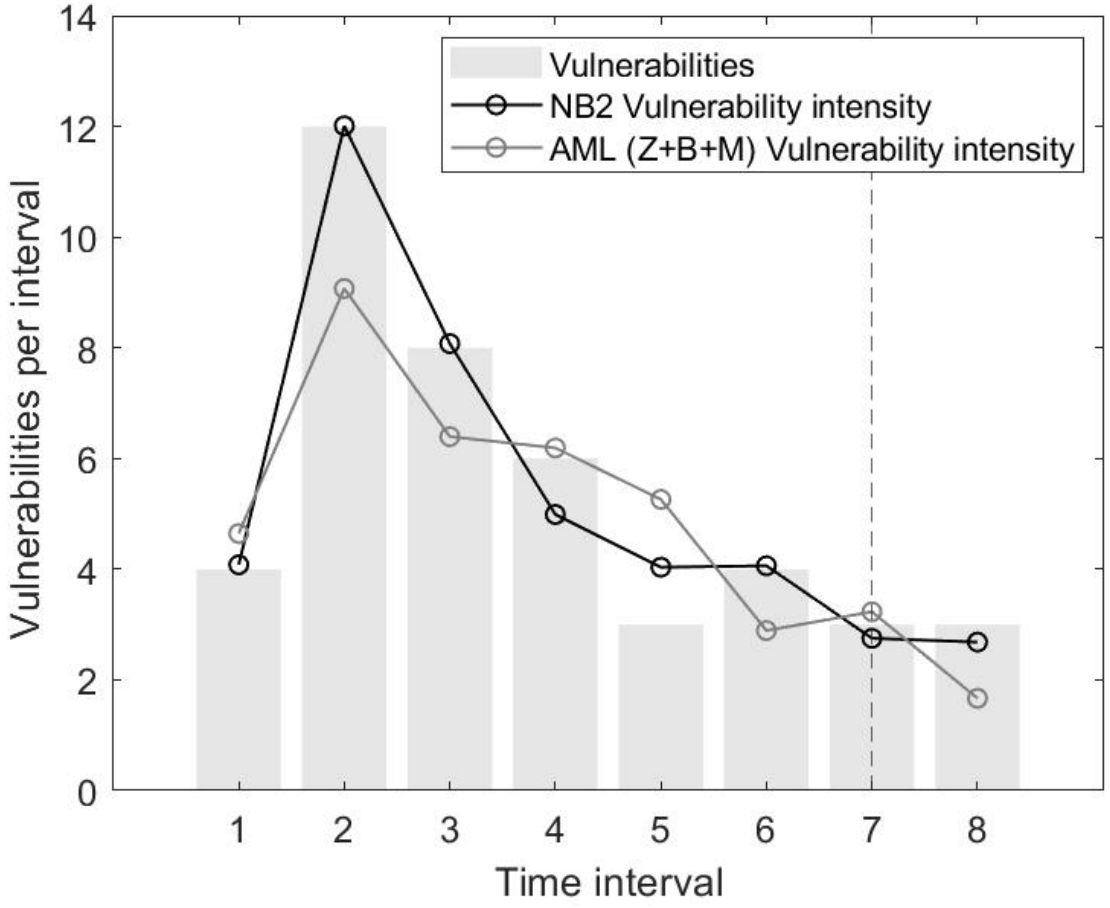
- VDM with covariates performed substantially better, despite AIC and BIC measures penalized inclusion of additional parameters for covariates associated with vulnerability discovery activities

Model Fit



- Covariate VDM with NB2 hazard function tracked and predicted better than AML model

Vulnerability Discovery Intensity



- Covariate VDM with NB2 hazard function tracked and predicted better than AML model

Summary & Conclusion

- Presented comparative study of software VDM incorporating covariates with Alhazmi-Malaiya Logistic (AML) model
- Software VDM incorporating covariates
  - More accurately tracked and predicted number of vulnerabilities discovered in future intervals as function of penetration testing activities performed
  - Achieved significantly better goodness of fit, despite information theoretic measures penalized covariate models for additional parameters
- Open source tool available from
  - https://lfiondella.sites.umassd.edu/research/software-reliability/

Next Steps and Future Work

- Combine techniques presented here with other reliability engineering techniques to provide more comprehensive, accurate, and usable methods that support systematic test and evaluation of software
- Demonstrating applicability of models to test & evaluation of machine learning with domain specific techniques and tools such as adversarial machine learning, generative adversarial training, and incremental learning

Acknowledgements

This research was supported by the National Science Foundation under Grant Number 1749635 and the Homeland Security Community of Best Practices (HS CoBP) through the U.S. Department of the Air Force through under award number SCR1158132. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, U.S. Department of Homeland Security or U.S. Department of the Air Force.