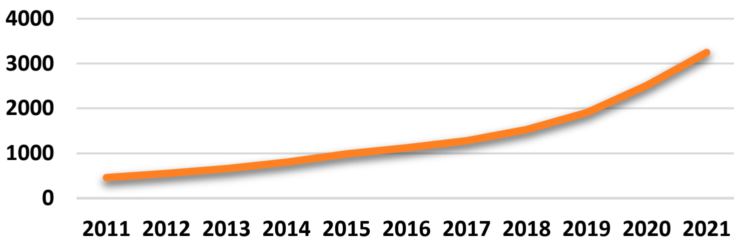
Expected Exploitability: Predicting the Development of Functional Vulnerability Exploits

Octavian Suciu[1], Connor Nelson[2], Zhuoer Lyu[2], Tiffany Bao[2], Tudor Dumitraș[1]

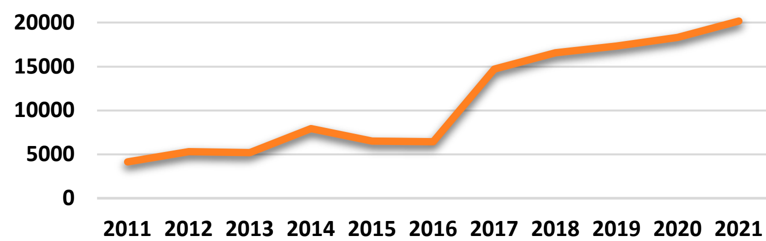
[1] University of Maryland, College Park
[2] Arizona State University

Progress in Vulnerability Detection

Number of papers on Google Scholar matching "vulnerability detection"



Number of vulnerabilities disclosed (cvedetails.com)



- Vulnerability detection techniques are improving

Impact of Exploits

- Exploits remain a principal tool for cybercrime

Are our techniques for identifying exploitable vulnerabilities also improving?

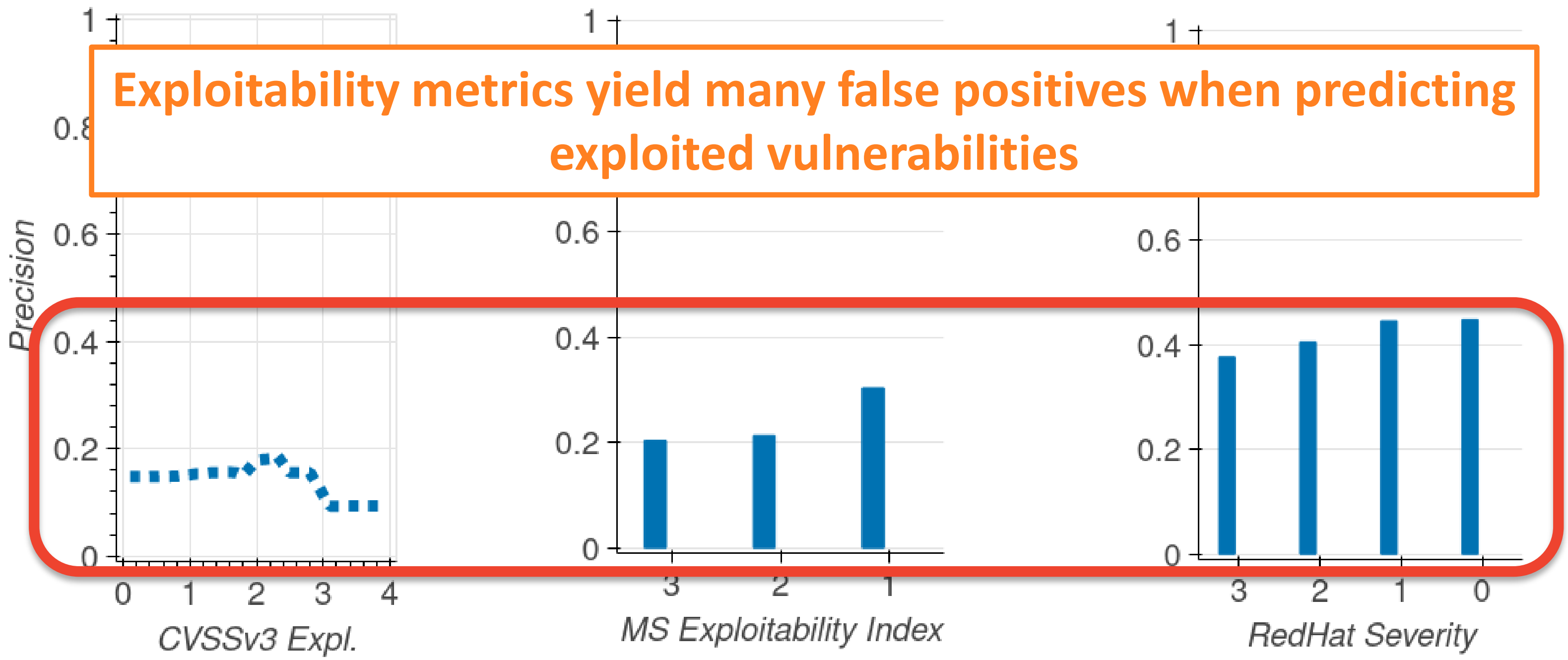
Identifying Exploitable Vulnerabilities

- Exploit evidence
  - Manual or Automatic Exploit Generation (AEG)[1]
  - Detect them in attacks in the wild[2]
  - Provide conclusive evidence about existence of exploits
- Exploit trackers
  - E.g.: CVSS Temporal, commercial platforms
  - Reactively capture existence of exploits
  - Do not provide evidence of non-exploitability
- **Exploitability assessment metrics**
  - Compute exploit likelihood, generally through heuristics
  - E.g.: CVSS Exploitability, Microsoft Exploitability Index, RedHat Severity

[1] Avgerinos+ "AEG: Automatic exploit generation.", 2011
[2] Sabottke+ "Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits.", 2015

Performance of Exploitability Assessment Metrics

- Predict “Exploited” if Score ≥Threshold
  - **Precision** = fraction of predicted vulnerabilities known to have functional exploits

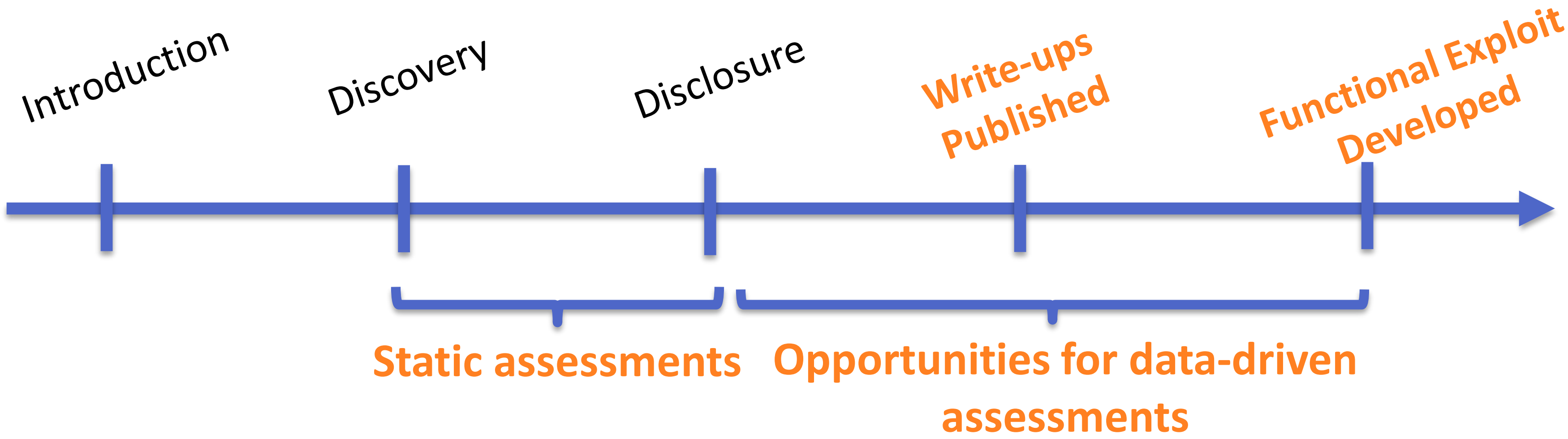


Research Agenda

- **Question:** Can we improve exploitability assessments through data-driven techniques?
- **Goal:** Develop a system to estimate exploitability from data
  - Track the **Expected** value of **Exploitability (EE)** over time
  - Build a platform to help practitioners

exploitability.app

Challenges Developing **EE**: CVE-2018-8174



- Exploitability 1.6/4.0 (9^{th} percentile) – **assessed difficult to exploit**
- Write-ups contain technical details – **exploit development made easier**

Challenge 1: Exploitability needs to capture information published after disclosure

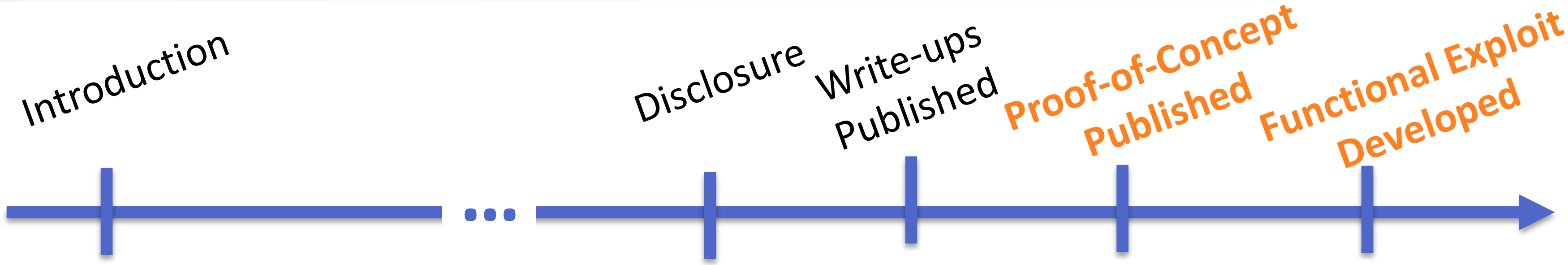
Assessments using existing work:

- Exploitability assessment metrics are static and based only on pre-disclosure information

Time-Varying Exploitability
[Contribution #1]

- **Time-varying view of exploitability influenced by artifacts published after disclosure**
  - Develop a machine learning system to compute **EE** and track exploitability over time

Challenges Developing EE: CVE-2019-1663



- Exploited 1 day after PoC– **overlooking useful PoC content features**

Challenge 2: Exploitability prediction requires extracting features from both code and descriptions in PoCs

Assessments using existing work:

- Features for predicting attacks in the wild : vulnerability characteristics, NLP on write-ups[1], social media[2], handcrafted[3]
- Existence of Proof-of-Concepts (PoCs) considered poor exploit predictors [4,5,6]

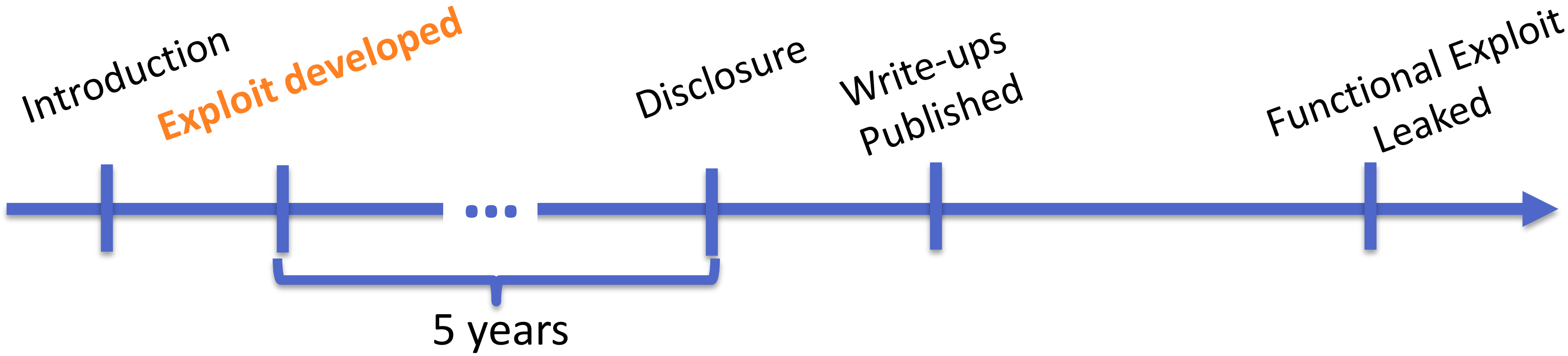
[1] Bozorgi+ "Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits.", 2010
[3] Jacobs+ "Improving vulnerability remediation through better exploit prediction", 2019
[2] Sabottke+ "Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits.", 2015

[4] Allodi & Massacci "A preliminary analysis of vulnerability scores for attacks in wild", 2012
[5] Tavabi+ "Darkembed: Exploit prediction with neural language models.", 2018
[6] Jacobs+ "Exploit prediction scoring system (epss).", 2021

Novel Features
[Contribution #2]

- Time-varying view of exploitability influenced by artifacts published after disclosure
- **PoC features semantically linked to the difficulty of creating functional exploits**
  - Features complement these from prior work and improve prediction performance

Challenges Developing EE: CVE-2017-0144



- Exploit initially unknown to the public – **incorrect label**

Challenge 3: Addressing label noise requires understanding its characteristics and impact

Assessments using existing work:

- Exploit datasets biases acknowledged since 2010[1] but not investigated before
- Ground truth biases cause label noise in machine learning[2]
- Label noise mitigation requires domain knowledge[3]

[1] Bozorgi+ "Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits.", 2010
[2] Frénay & Verleysen "Classification in the presence of label noise: a survey.", 2013
[3] DeLoach+ "Android malware detection with weak ground truth data.", 2016

Label Noise Robustness

[Contribution #3]

- Time-varying view of exploitability influenced by artifacts published after disclosure
- PoC features semantically linked to the difficulty of creating functional exploits
- **Characterize and quantify impact of label noise in exploit prediction**
  - Develop a noise mitigation strategy based on domain observations

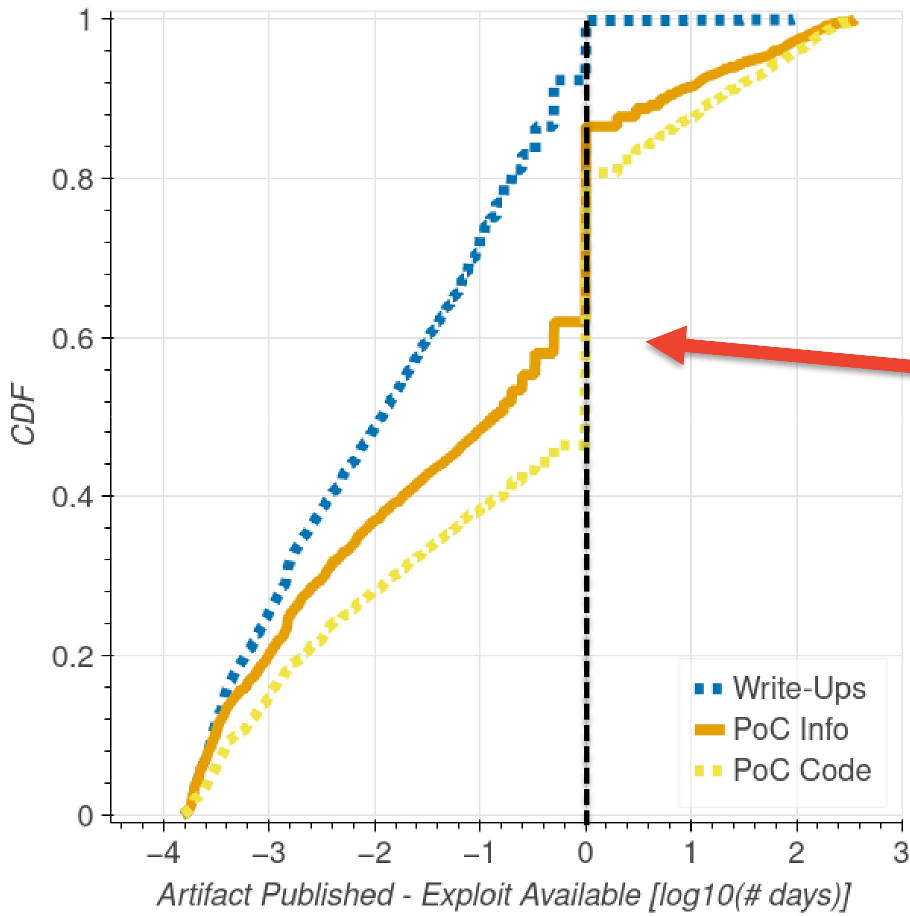
Outline

- **Time-varying exploitability**
- Novel features
- Label noise robustness

Dataset - Vulnerabilities

- 103,000 Vulnerabilities
  - NVD CVE-IDs disclosed between 1999 – 2020
- 327,000 Artifacts
  - 278,000 Write-ups from 76 sources
  - 49,000 Proof-of-Concepts (PoCs) for 22,000 vulnerabilities
- Ground truth: evidence of functional exploits
  - Aggregated from 12 public sources
  - **31%** vulnerabilities are **labeled as exploited** (32,093)

Factors Reflecting Exploit Development



- Measured how soon exploits become available after artifacts are published

Emergence of functional exploits is highly correlated with artifact publication

Exploitability changes over time and is reflected by the publication of artifacts

Expected Exploitability (EE) System

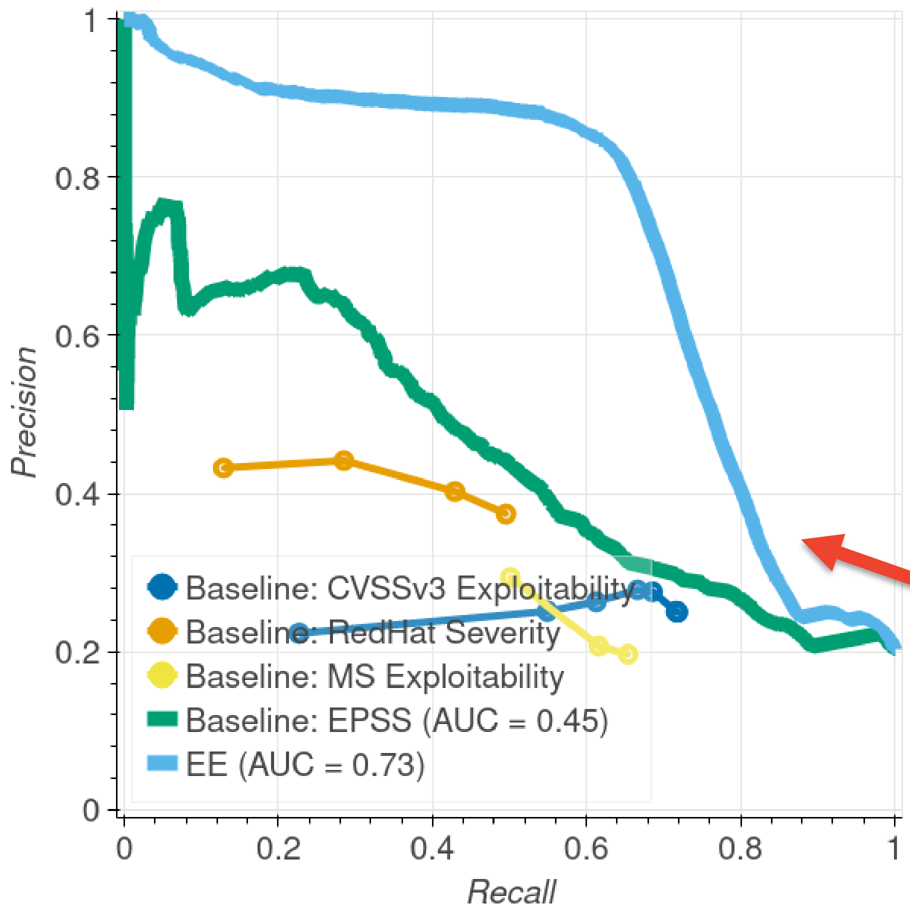
- Predict likelihood of functional exploits
- Use supervised machine learning
- Deployment: continuous exploitability estimates

Performance Evaluation

- Baselines:
  - **Static metrics:** CVSS Score, Microsoft Exploitability Index, RedHat Severity
  - **Data-driven (exploits in the wild)**: Exploit Prediction Scoring System (EPSS) [1]
    - Linear model based on **53 handcrafted features** from descriptors
- Performance Metrics:
  - **Precision**: fraction of predicted vulnerabilities known to have functional exploits
  - **Recall**: fraction of exploited vulnerabilities that are marked as such

[1] Jacobs+ "Improving Vulnerability Remediation Through Better Exploit Prediction", 2019

Performance Predicting Functional Exploits



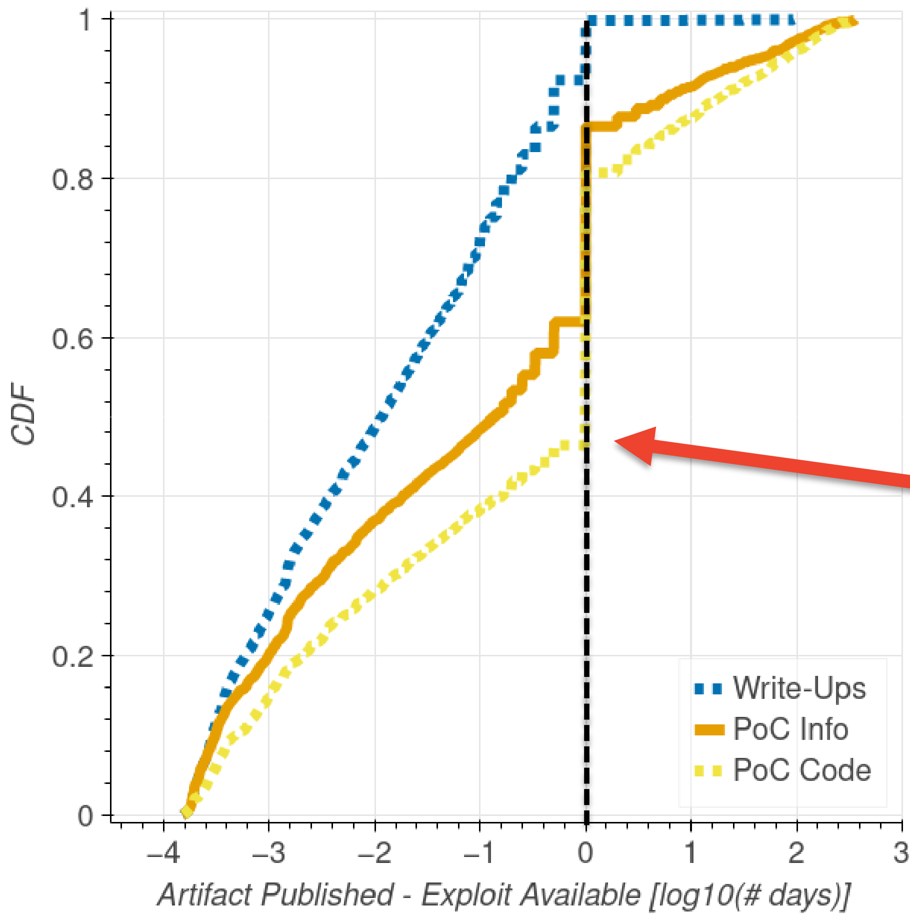
- Static exploitability metrics yield poor performance
- Features for exploits in the wild highlight potential for domain adaptation
- **EE** outperforms baselines by a large margin

Tailored data-driven solutions can significantly improve exploitability assessments

Outline

- Time-varying exploitability
- **Novel features**
- Label noise robustness

Features of Exploitability



- Prior work features:
  - Vulnerability characteristics[1]
  - NLP on write-ups[2]
  - Handcrafted features[3]

PoCs are some of the most correlated artifacts with exploit development

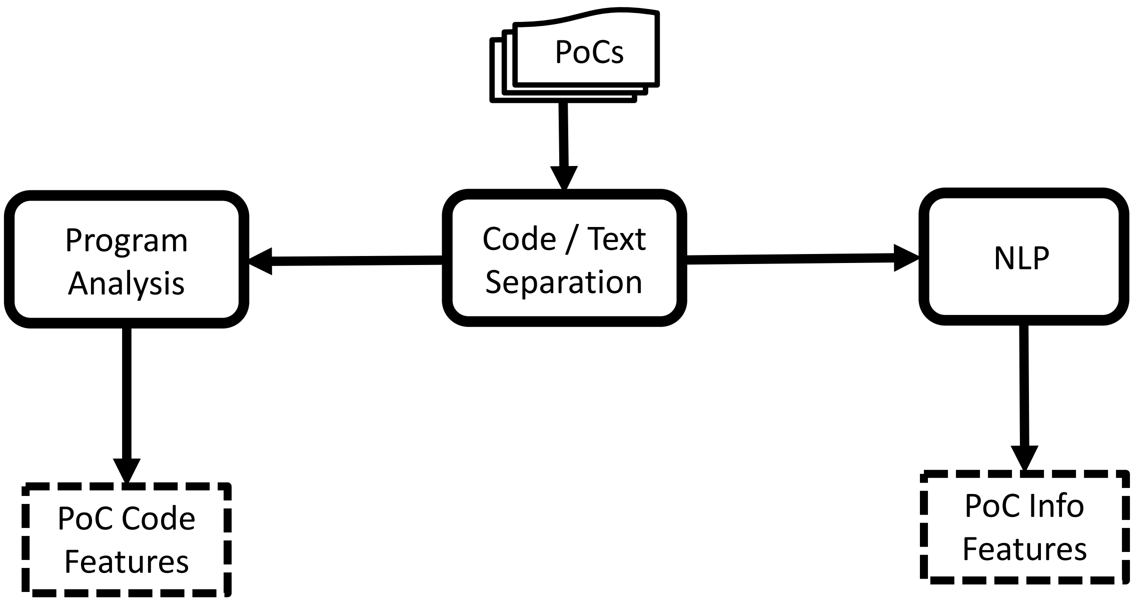
[1] Bozorgi+ "Beyond heuristics: learning to classify vulnerabilities and predict exploits", 2010
[2] Sabottke+ "Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits.", 2015
[3] Jacobs+ "Improving vulnerability remediation through better exploit prediction", 2019

Intuition For PoC Features

- PoCs are designed to trigger the vulnerability
  - Cannot always be easily weaponized
  - Their existence was considered poor exploit predictor by prior work[1,2,3]
- Observation: Any functional exploit needs to trigger the vulnerability
  - A common goal with PoCs
- PoCs and functional exploits share characteristics
  - PoC code complexity reflects exploit complexity
  - PoC comments and descriptions highlight exploit steps

[1] Allodi & Massacci "A preliminary analysis of vulnerability scores for attacks in wild", 2012
[2] Tavabi+ "Darkembed: Exploit prediction with neural language models.", 2018
[3] Jacobs+ "Exploit prediction scoring system (epss).", 2021

PoC Feature Extraction System



In the paper: PoC features complement these from prior work

Outline

- Time-varying exploitability
- Novel features
- **Label noise robustness**

Biases in Ground Truth

- Inherent class biases
  - Exploits might be private or published later
  - P[true_label=1|ground_truth=0] > 0
  - Acknowledged since 2010 [1] but not investigated before
  - Results in **class-dependent label noise** during training
- Observation: Individual exploit evidence sources also have biases
  - E.g., Symantec did not have a product for Linux
    - Linux exploits are absent from a Symantec ground truth
  - This suggests presence of **feature-dependent label noise**
  - Studying this category is an open problem in ML[2]

[1] Bozorgi+ "Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits.", 2010
[2] Frénay & Verleysen "Classification in the presence of label noise: a survey.", 2013

Evidence of Feature-Dependent Noise

- Statistical tests for independence between sources of ground truth sources and vulnerability features
- Independence can be ruled out for each label source with at least 4 features
  - All label sources might depend on some vulnerability features

Exploit prediction is subject to class- and feature-dependent label noise

	Functional Exploits						Exploits in the Wild					
	Tenable	X-Force	Metasploit	Canvas	Bugtraq	D2	Symantec	Contagio	Alienvault	Bugtraq	Skybox	Tenable
CWE-79	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ (0.006)
CWE-94	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	X (1.000)
CWE-89	✓	✓	✓	✓	✓	X (1.000)	✓	✓	✓	✓	✓	X (0.284)
CWE-119	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ (0.001)
CWE-20	✓	✓	✓	✓	✓	✓	✓	X (1.000)	✓	✓ (0.002)	✓	X (1.000)
CWE-22	✓	X (0.211)	✓	X (1.000)	X (1.000)	✓	X (1.000)	X (1.000)	X (0.852)	X (1.000)	X (1.000)	X (1.000)
Windows	✓	✓	✓	✓	✓	X (0.012)	✓	✓	✓	✓	✓	✓
Linux	✓	✓	✓	✓	✓	X (1.000)	✓	✓	✓	✓	✓	✓

In the paper: Measuring the Impact of Noise

- Simulate label noise and measure impact on performance

Label noise can significantly degrade the performance of exploit predictors

In the paper: Mitigating Impact of Noise

- Embedding noise mitigation during training phase
  - Off-the-shelf solutions do not fully address the problem
- Feature Forward Correction (FFC) loss
  - Use priors about noise distribution and probabilities
  - Key idea: reduce penalty for mistakes on instances that might have noisy labels

FFC corrects the performance degradation caused by class- and feature-dependent label noise

Conclusions

- Exploitability evolves over time function of artifacts
  - Expected Exploitability captures evolution and predicts exploitation likelihood
- Prior work features are not sufficient for the task
  - We propose new features from PoCs that can complement existing ones
- Exploit predictors are affected by ground truth biases
  - We characterize these biases and provide a mitigation strategy against label noise

Thank you!

- Expected Exploitability platform:
  - Updating daily with new vulnerabilities, artifacts and predictions
  - Provides Web tools and API for practitioners to analyze and prioritize vulnerabilities

exploitability.app