Feature Creation Towards the Detection of Non-Control-Flow Hijacking Attacks

Zander W. Blasingame[1], Chen Liu[1], Xin Yao[2]

ICANN 2021

[1] Department of Electrical and Computer Engineering
Clarkson University
10 Clarkson Ave, Potsdam NY, 13699, USA
[2] Research Institute of Trustworthy Autonomous Systems
Southern University of Science and Technology
Shenzhen, Guangdong, China

Problem

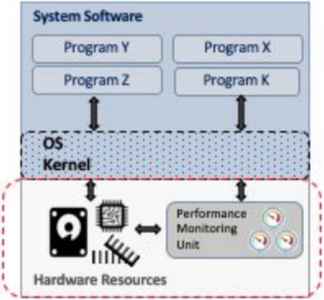
- *Non-control-flow* hijacking attacks are becoming increasingly prevalent
- They do not alter the execution of the program
- Which makes the difficult to detect

Solution

- Use Hardware Performance Counters (HPCs) to monitor program
- Use ML to compare current behavior to established baseline
- This can be used for detection

HPCs

Performance Monitoring Unit



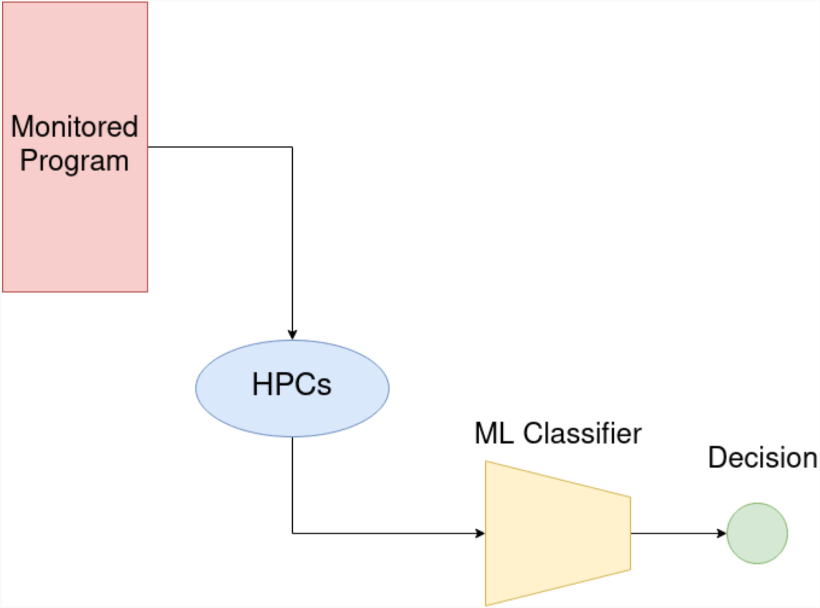
Hardware Performance Counters (HPCs)

- Special hardware registers
- Over 200 measurable HW events

Why use HPCs:

- Capture **raw execution** behavior
- Very **fast** to access
- **Difficult** for attackers to **manipulate**

Detection Architecture



Problem

- Very few or no samples from the attack
- Creates a very large imbalance in training data
- Leads to high False Negative Rates in classification

Use Genetic Programming (GP) to combat class imbalances

Solution

- Use GP to create new features from previous features
- Use the Hellinger distance as the fitness function
- Implement the Hellinger distance to work with discrete samples

Datasets

Vulnerability	Type	Program	Exploit	Type
bugtraq ID: 41956	FS*	ORZHTTPD	ORZHTTPD_ROOTDIR	Data leak
			ORZHTTPD_LEAKADDR	Mem leak
CVE-2013-2028	SBO†	NGINX	NGINX_ROOTDIR	Data leak
			NGINX_KEYLEAK	Data leak
CVE-2014-3566	ED‡	OPENSSL	POODLE	Data leak
CVE-2015-0204	ED‡	OPENSSL	FREAK	Data leak
CVE-2015-0400	ED‡	OPENSSL	LOGJAM	Data leak

(∗) Format String (†) Stack Buffer Overflow (‡) Encryption Downgrade

Objective

- $\mathcal{X}$ is the feature space
- Feature creation function $f : \mathcal{X} \to \mathbb{R}$
- Goal find function f that maximizes distance between the two classes

GP Details

- Initial population size of 300
- Tournaments with three individuals
- Crossover and mutation with 80% and 10%
- Functions are created initially as trees with depth one or two
- Maximum bloat depth of 17
- Terminal node consists of
    1. The original 12 features
    2. Integer constant
    3. Floating point constant
    4. -1
    5. ADD, SUB, MUL, DIV, MAX, MIN, NEG, COS, SIN, LOG and ABS

Fitness Function $F : (\mathcal{X} \to \mathbb{R}) \to \mathbb{R}$

Hellinger Distance

- Let X, Y be r.v.s denoting the feature vector and class label
- Let ρ_i denote the density function associated with $P(f(X)|Y=i)$ with 1 for normal and 0 for anomalous
- The discrete Hellinger distance is then

$$F^2[f] = \frac{1}{2} \sum_{a \in f(\mathcal{X})} \left(\sqrt{\rho_1(a)} - \sqrt{\rho_0(a)} \right)^2$$

Implementation I

- Infeasible to evaluate distance over all of $f(\mathcal{X})$
- Two possible solutions
  1. Evaluate over samples in the dataset

$$F^2[f] = \frac{1}{2}\mathbb{E}_{a\sim P(f(\mathcal{X}))}\left[\left(\sqrt{\rho_1(a)} - \sqrt{\rho_0(a)}\right)^2\right]$$

     i.e., taking an expectation with respect to $P(f(X))$
  2. Or evaluate over a finite subset of $f(\mathcal{X})$, e.g., a uniformly discrete subset on $f(\mathcal{X})$

Implementation II

- Is it advantageous to normalize the inputs to f?

Implementation III

- Prior work has suggested this form for a fitness criterion loosely based on the Hellinger distance

$$F^2[f] = \mathbb{E}_{a\sim P(f(X))}\left[\left(\sqrt{\frac{\rho_1(a)P(Y=1)}{\rho(a)}} - \sqrt{\frac{\rho_0(a)P(Y=0)}{\rho(a)}}\right)^2\right]$$

where ρ is the density of $P(f(X))$

Implementation IV

- Estimates of ρ_i are needed to calculate the distance
- Could use histograms of the samples to create $\hat{\rho}_i$
- Or use Kernel density estimation to create $\hat{\rho}_i$

Classifiers

- Support Vector Machine (SVM) with linear kernel
- SVM with radial basis function
- Neural network with softmax output

Experimental Procedure

1. Run the GP algorithm with the different Hellinger distance implementations on the entire original dataset and select the top-4 performing individuals, i.e., the programs that construct the new features.
2. Create 5 pairs of new training and testing sets from the original data using stratified k-fold cross-validation.
3. The individuals created by the GP algorithm are used to construct new features. These new features are partitioned in accordance to the scheme from the previous step.

Hellinger Implementation Variants

- The baseline variant is

$$F^2[f] = \mathbb{E}_{a \sim P(f(\mathcal{X}))} \left[\left(\sqrt{\hat{\rho}_1(a)} - \sqrt{\hat{\rho}_0(a)} \right)^2 \right]$$

  where $\hat{\rho}_i$ is estimated via KDE
- The posterior variant uses the estimated posterior distributions
- The discrete variant uses histograms instead of KDE
- The full variant evaluates the distance over a resampled subset of $f(\mathcal{X})$
- The unnromalized variant does not normalize the input features

Results I

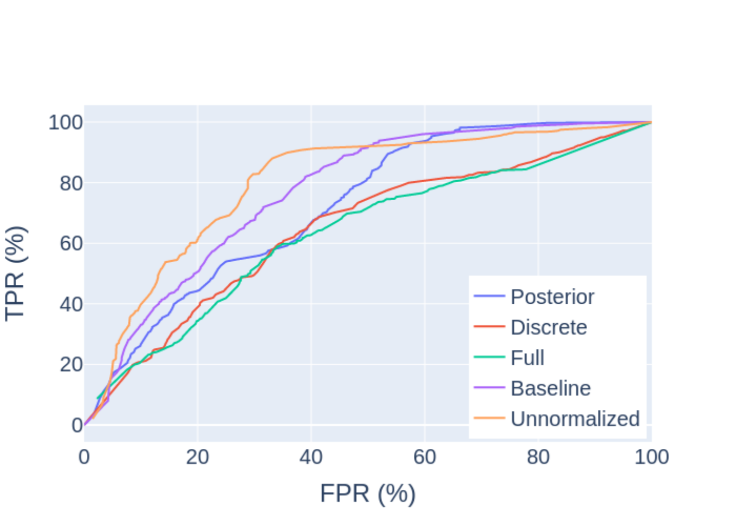


Figure 1: Average ROC curve of classifiers using different Hellinger estimates

Results II

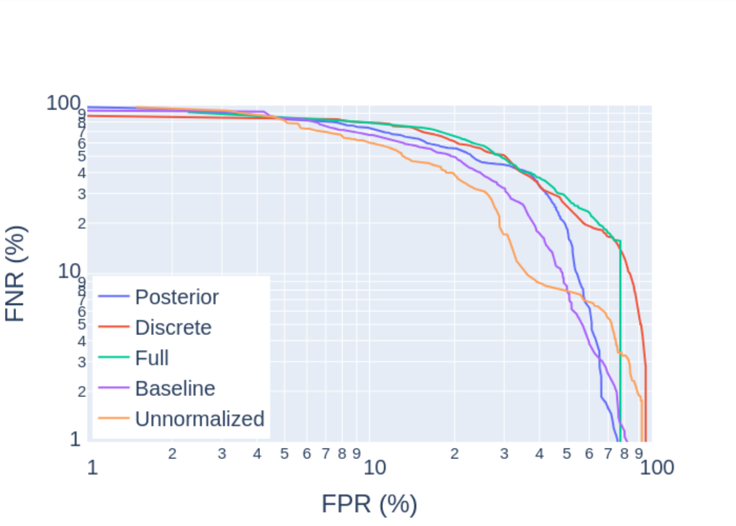


Figure 2: Average DET curve of classifiers using different Hellinger estimates

Results III

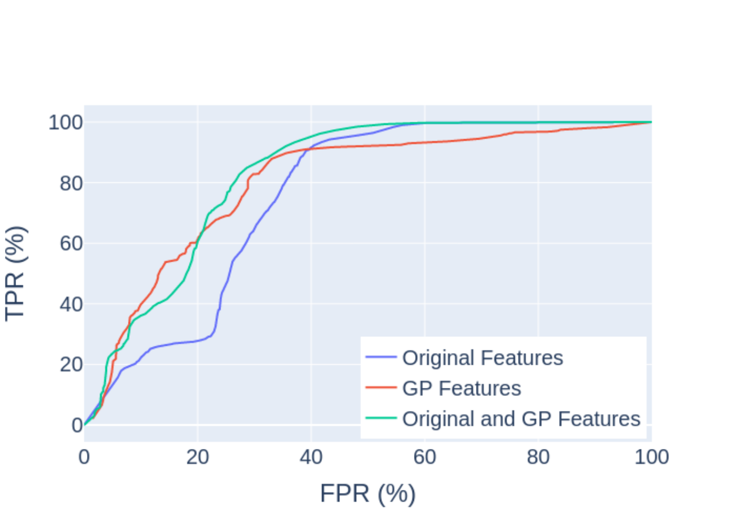


Figure 3: Average ROC curve of classifiers using different features

Results IV

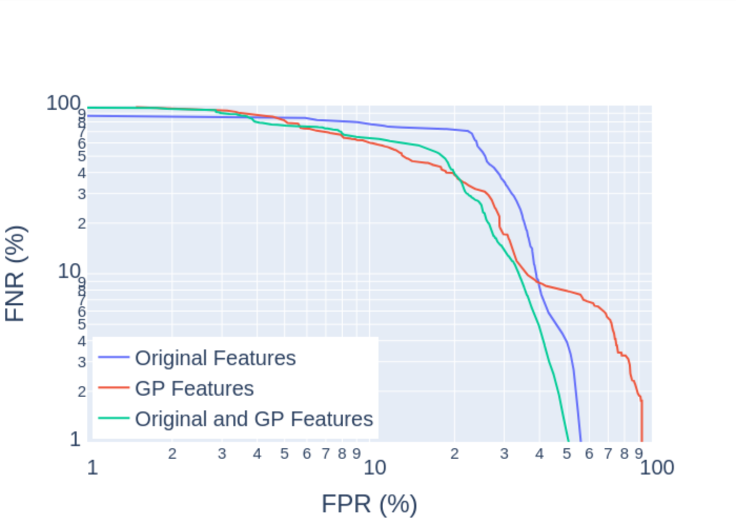


Figure 4: Average DET curve of classifiers using different features

Summary

- Used the Hellinger distance as a fitness function to create new features
- Studied different implementations of the Hellinger distance
- Using the unnormalized Hellinger GP to augment the original features vastly improves the performance