Ransomware Detection using Machine Learning with eBPF

Offensive Technologies Project Presentation

Max Willers Tomás Philippart

01/06/2023

Offensive Technologies
MSc Security and Network Engineering
University of Amsterdam

Agenda

1. Introduction
2. Background
   a. eBPF Recap
   b. ML Primer
3. Related work
4. Methodology
5. Results
6. Discussion & Conclusion
7. Further Research

Ransomware Detection using Machine Learning with eBPF

Offensive Technologies Project Report

Max Willers
Tomás Philippart

1 Introduction

Ransomware attacks continue to pose a significant and escalating threat to organizations worldwide, resulting in substantial disruptions and financial losses [17, 28, 25]. Recent reports indicate a worrisome surge in such attacks, with the percentage of organizations directly affected by ransomware rising from 37% in 2020 to a staggering 66% in 2021 [6]. While Windows systems have traditionally been the primary target of ransomware attacks, there has been a concerning uptick in attacks targeting Linux systems [13], which are often utilized for critical business applications and infrastructure.

Ransomware, a form of malware designed to extort victims, has gained notoriety due to its ability to encrypt files on compromised systems, holding them hostage until a ransom is paid [11, 17]. To combat this menace, existing ransomware detection methods primarily rely on two approaches: signature-based and behavioral-based techniques. Signature-based detection relies on predefined patterns or signatures to identify known ransomware samples, while behavioral-based detection focuses on analyzing application behavior, encompassing factors such as API calls and network behavior [12].

However, these conventional approaches have their limitations. Signature-based detection struggles to keep pace with the rapid evolution of ransomware variants, rendering it less effective against new and unknown threats. Behavioral-based detection, although valuable, may overlook subtle yet crucial indicators of ransomware activity, leading to potential false negatives. Moreover, attackers have become adept at obfuscating malware code, complicating static analysis and evading

How can **eBPF** be integrated with a **Machine Learning** pipeline to accurately **detect ransomware during runtime**?

1. How can **eBPF** be used to detect **ransomware?**
2. How can it be integrated into a **Machine Learning** pipeline?
3. How can this solution **accurately detect ransomware during runtime?**

Background - eBPF Recap

- Roots in BPF (*Berkeley Packet Filter*) technology
- **Run sandboxed programs** within the kernel
- **Hook** anywhere in the kernel to modify functionality
  - Can even attach directly to the NIC
  - JavaScript-like programmability to the kernel

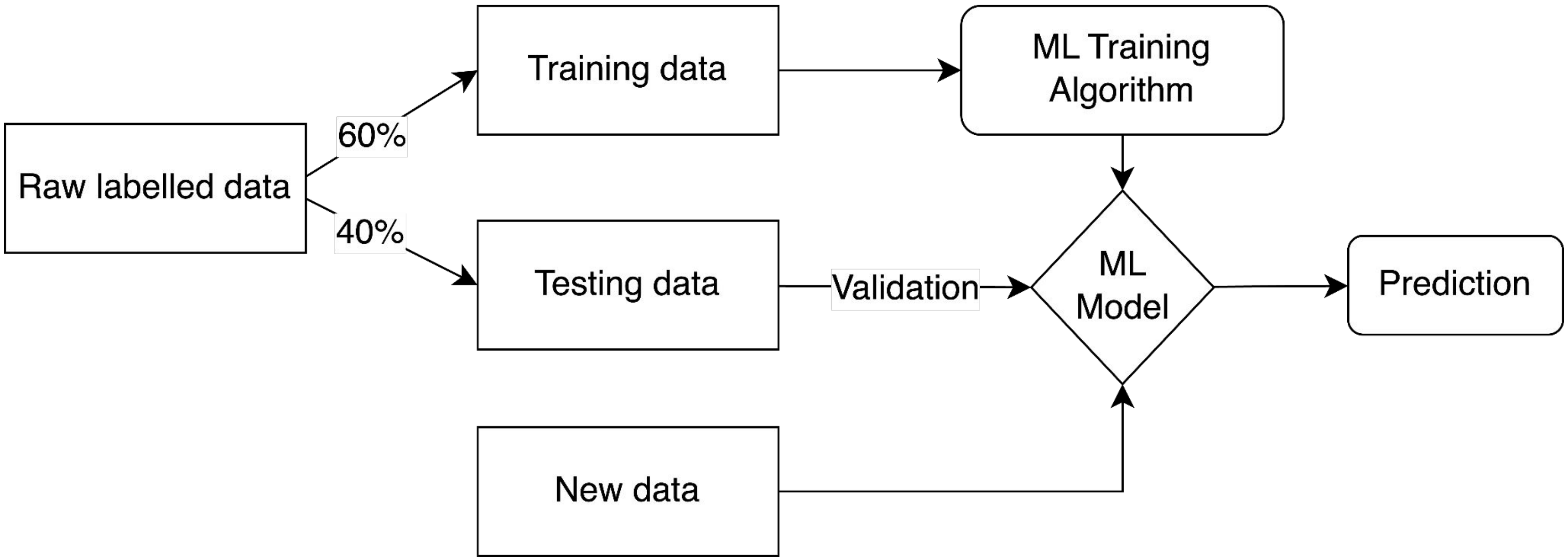
Use cases:

- Kernel performance tracing
- Network security and observability
- Runtime security
- etc.

eBPF

Background - Machine Learning Primer

- In this project: **Supervised Learning Classifiers**
  - *Support Vector Machine* (SVM)



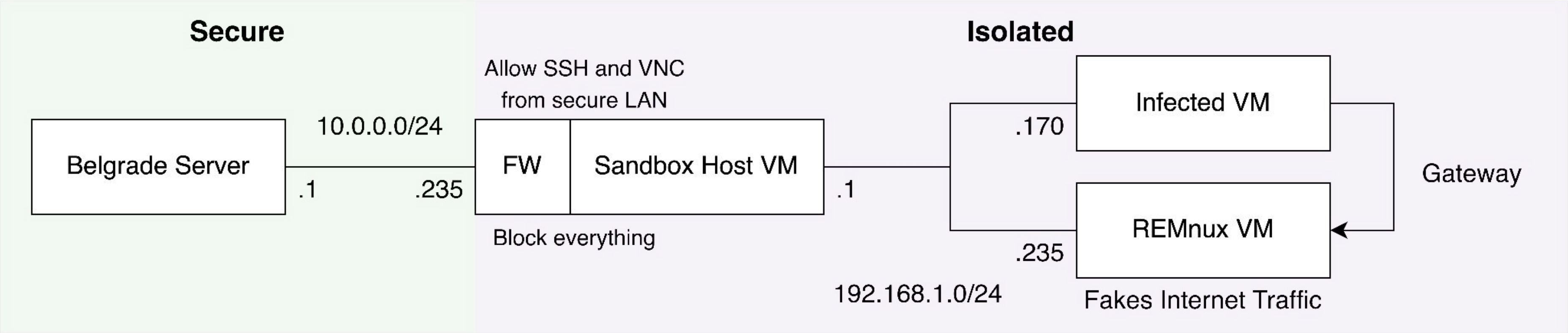
Related work

- Vehabovic et al. Ransomware Detection and Classification Strategies (2022)
  - Categorizes ransomware detection and classification systems into network-based and host-based
- Kharaz et al. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware (2016)
  - Dynamic analysis solution based on behavior
  - Able to identify and detect previously unreported ransomware
- Cozzi et al. Understanding Linux Malware (2018)
- Agman, Hendler. BPFroid: Robust Real Time Android Malware Detection Framework (2021)
  - eBPF-based malware detection for Android based on behavioral signature

Methodology - Experimental setup

Goal: **isolated environment** to experiment with ransomware

- Double-nested isolated virtualized environment
- Virtualized with KVM Hypervisor
- Use snapshots to run experiments under same conditions



Methodology - Ransomware detection techniques

- Static analysis (expensive)
- Fingerprint binary - compare hashes to known ransomware
- Analyzing behavioral traits and patterns
  - Host-based: filesystem and memory operations
  - Network-based: C&C communication
- Machine Learning methods:
  - Models trained on features (e.g., system calls, network traffic)
  - Effectively classify and identify in real-time
  - Focus of our paper!

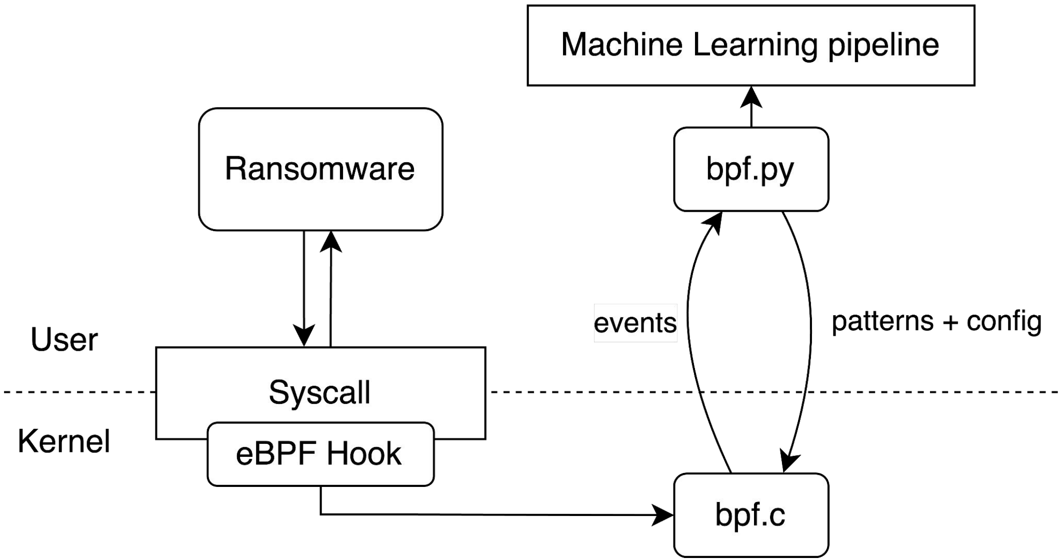
Methodology - Ransomware detection techniques **(with eBPF)**

Why use eBPF?

- **Event-driven nature** and direct execution within kernel
- **Unified mechanism** to intercept and handle events
- Optimizes **performance** by filtering irrelevant events in user space
- **Comprehensive kernel tracing**
- Actively maintained
- **Port to Windows** in progress
  - Can apply same techniques to Windows

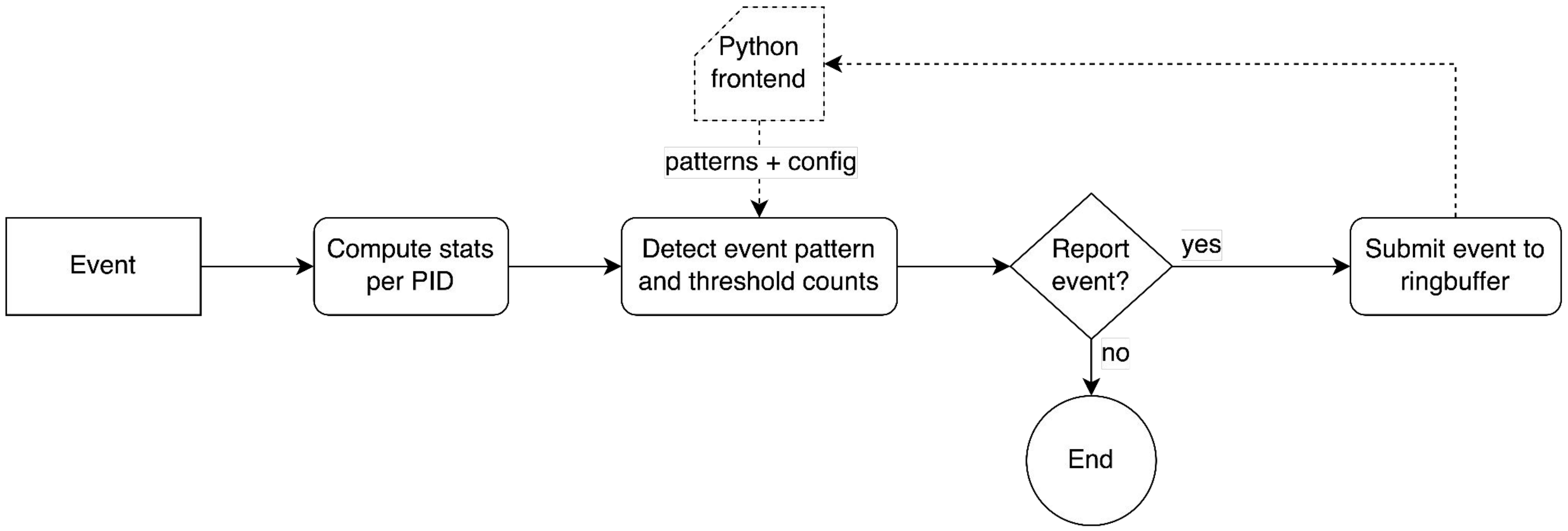
Methodology - Our detector (architecture overview)

- eBPF program attached to **critical system calls**
- Python frontend
  - Communicates back and forth with eBPF + ML pipeline

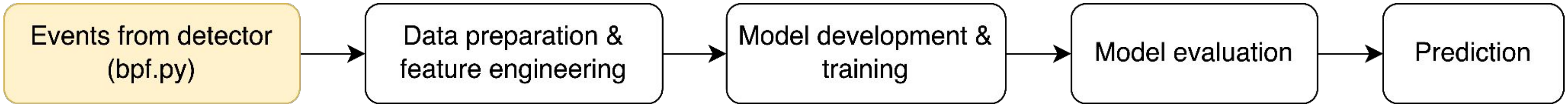


Methodology - Our detector (bpf.c)

Goal: trace all events and only submit relevant ones

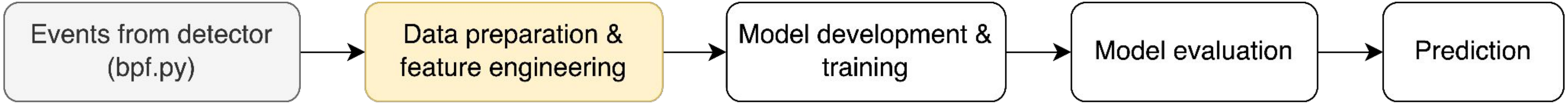


Methodology - Our detector (ML pipeline)




```
TS,PID,TYPE,FLAG,PATTERN,OPEN,CREATE,DELETE,ENCRYPT,FILENAME
2748377535267,3175,0,1,0,0,1,0,1,/sys/kernel/debug/tracing/events/syscalls/sys_enter_unlink/id
2748396149305,3175,0,1,0,0,1,0,1,/sys/kernel/debug/tracing/events/syscalls/sys_enter_unlinkat/id
2748396700388,3175,0,0,0,0,0,0,0,/usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
2748396841453,3175,0,0,0,0,0,0,0,/usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
2748396849806,3175,0,0,0,0,0,0,0,/usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
[…]
```

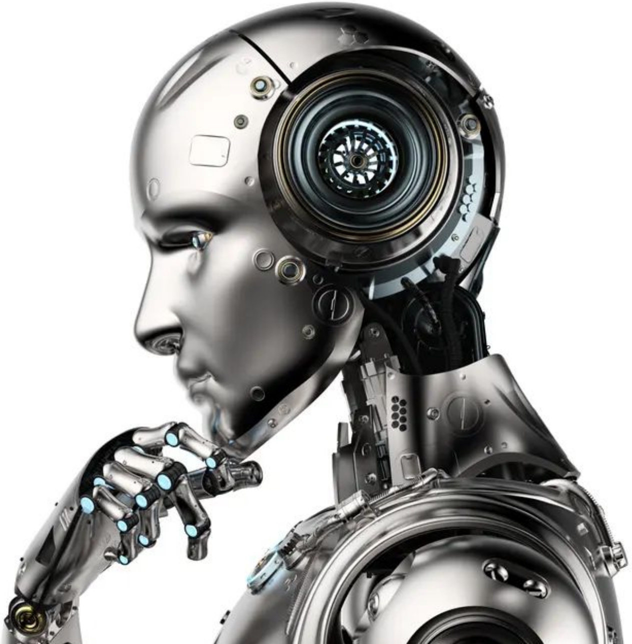
Methodology - **Our detector (ML pipeline)**



PID,C_max,C_sum,D_max,D_sum,O_max,O_sum,P_max,P_sum,CCC,CCO,CDD, [...], OOO
3101,1,1,2,2,7,7,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,5
3102,192,1251,0,0,315,3314,0,0,2,7,0,0,1079,162,0,0,0,0,0,0,0,7,0,1235,0,0,162,0,1909
3103,267,1557,0,0,635,4417,0,0,1,4,0,0,1314,237,0,0,0,0,0,0,0,4,0,1548,0,0,237,0,2627
3104,120,619,0,0,351,1985,0,0,0,2,0,0,585,31,0,0,0,0,0,0,0,2,0,615,0,0,31,0,1336
3105,177,1450,0,0,448,4074,0,0,4,7,0,0,1255,183,0,0,0,0,0,0,0,7,0,1432,0,0,183,0,2451
3106,139,1000,0,0,587,2921,0,0,3,4,0,0,820,172,0,0,0,0,0,0,0,4,0,989,0,0,172,0,1755
3107,267,1366,0,0,430,4100,0,0,3,5,0,0,1159,198,0,0,0,0,0,0,0,5,0,1353,0,0,199,0,2542
3108,275,1351,0,0,683,4611,0,0,0,7,0,0,1018,325,0,0,0,0,0,0,0,7,0,1337,0,0,326,0,2940
3109,267,1385,2,2,473,4630,1,1,3,7,1,0,1107,266,0,0,0,1,1,0,0,7,1,1367,0,0,267,0,2987
[…]

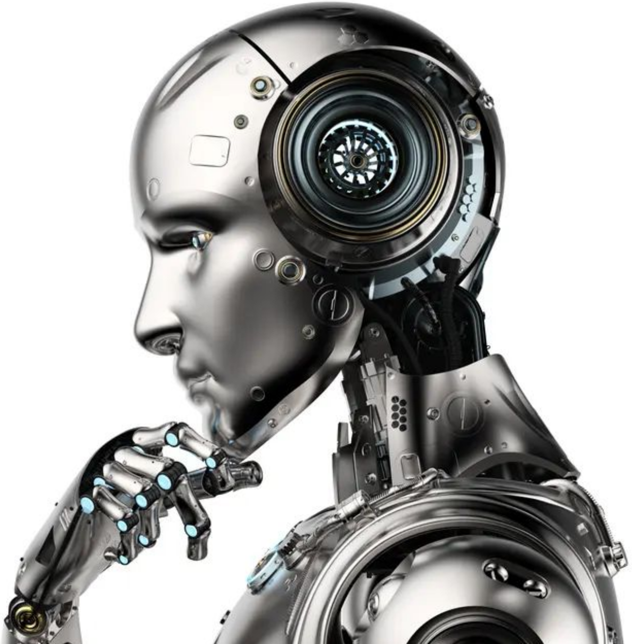
Methodology - Our detector (ML pipeline)

Events from detector (bpf.py) → Data preparation & feature engineering → Model development & training → Model evaluation → Prediction



Methodology - Our detector (ML pipeline)

Events from detector (bpf.py) → Data preparation & feature engineering → Model development & training → Model evaluation → Prediction



Methodology - Our detector (ML pipeline)


```
PID,C_max,C_sum,D_max,D_sum,O_max,O_sum,P_max,P_sum,CCC,CCO,CDD, [...], OOO     PREDICTION
3101,1,1,2,2,7,7,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,5                  BENIGN
3102,192,1251,0,0,315,3314,0,0,2,7,0,0,1079,162,0,0,0,0,0,0,0,7,0,1235,0,0,162,0,1909    RANSOMWARE
3103,267,1557,0,0,635,4417,0,0,1,4,0,0,1314,237,0,0,0,0,0,0,0,4,0,1548,0,0,237,0,2627    RANSOMWARE
3104,120,619,0,0,351,1985,0,0,0,2,0,0,585,31,0,0,0,0,0,0,0,2,0,615,0,0,31,0,1336         RANSOMWARE
3105,177,1450,0,0,448,4074,0,0,4,7,0,0,1255,183,0,0,0,0,0,0,0,7,0,1432,0,0,183,0,2451    RANSOMWARE
3106,139,1000,0,0,587,2921,0,0,3,4,0,0,820,172,0,0,0,0,0,0,0,4,0,989,0,0,172,0,1755      BENIGN
3107,267,1366,0,0,430,4100,0,0,3,5,0,0,1159,198,0,0,0,0,0,0,0,5,0,1353,0,0,199,0,2542    BENIGN
3108,275,1351,0,0,683,4611,0,0,0,7,0,0,1018,325,0,0,0,0,0,0,0,7,0,1337,0,0,326,0,2940    BENIGN
3109,267,1385,2,2,473,4630,1,1,3,7,1,0,1107,266,0,0,0,1,1,0,0,7,1,1367,0,0,267,0,2987    BENIGN
[…]
```

Results (so far)

- Number of events: ~1M
- Number of processes scanned: 507
- Ransomware families run: 9

Predicted

		Benign	Ransomware
Actual values	**Benign**	496	2
	Ransomware	0	9

Precision = 99.6%
F1 score = 99.80%

Discussion & Conclusion

- Good performance!

- Pipeline still requires heavy manual work
  - Labelling data can be automated
  - All the programs can be unified into a single mechanism
  - Goal is to do all the above

- Imbalanced dataset
  - Need more ransomware/benign runs!
  - How will it react against novel ransomware?

see code @ github (`TomasPhilippart/ebpfangel`)

Further Work

- Create a larger and more comprehensive dataset
  - More features
  - More runs (more samples)
  - More benign samples

- Implement more detection features and techniques
  - Network traffic to/from C&C
  - Data buffer entropy

- Optimize machine learning pipeline with other models
  - Neural networks
  - Decision Trees