IEEE International Conference on Communications
Communications Technologies 4Good
8–12 JUNE 2025 // MONTREAL, CANADA

Malware Detection in Docker Containers: An Image is Worth a Thousand Logs

A. Nousias, E. Katsaros, E. Syrmos, P. Radoglou-Grammatikis, T. Lagkas, V. Argyriou, I. Moscholios, E. Markakis, S. Goudos and **P. Sarigiannidis***

*** University of Western Macedonia, psarigiannidis@uowm.gr**

Under DYNABIC & P2CODE

DYNABIC

P2CODE

K3Y Ltd

www.k3ylabs.bg

A. Nousias
E. Katsaros
E. Syrmos
P. Radoglou Grammatikis

International Hellenic University

www.cs.ihu.gr

T. Lagkas

Hellenic Mediterranean University

www.hmu/en/home/

E. Markakis

University of Western Macedonia

www.uowm.gr/en/

P. Sarigiannidis
P. Radoglou Grammatikis

Kingston University London

www.kingston.ac.uk/

V. Argyriou

University of Peloponnese

www.uop.gr/en

I. Moscholios

Aristotle University of Thessaloniki

www.auth.gr/en/homepage/

S. Goudos

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070455 (DYNABIC) and No 101093069 (P2CODE)

IEEE ICC®

INTRODUCTION & RELATED WORK

INTRODUCTION

Software Containers Broad Adoption and Benefits:

- Gained widespread adoption in recent years.
- Abstract system-specific dependencies and enable scalability.
- Represent standardized, self-contained units of software.
- Support diverse functionalities: OS, DB, ML models, etc.

Security Challenges with Container Adoption

- Significant security challenges arise despite benefits.
- Injection of malicious software into containers is a growing threat.
- Compromised containers can be entry points for further attacks.

RELATED WORK

2 MAIN LINES OF WORK

1. Classic Approaches in Malware Detection

Sathyanarayan et al. 2008

Signature generation and detection of malware families.

Aslan and Samet, 2020

A comprehensive review on malware detection approaches.

- **Signature-Based Detection**: Scans files for known malware patterns; fast for known threats but struggles with novel or polymorphic malware.
- **Heuristic-Based Detection**: Analyzes static features of files for suspicious characteristics (e.g., obfuscation, uncommon instructions); helps detect unknown malware but risks false positives.
- **Behavioral Monitoring**: Observes runtime behavior (e.g., unauthorized network communication, file modifications); effective but may miss dormant malware.

RELATED WORK

2 MAIN LINES OF WORK

2. ML-based Malware Detection

Cui et al., 2018

Detection of malicious code variants based on deep learning.

Gilbert et al., 2019

Using CNN for classification of malware represented as images.

CNN-Based Malware Recognition: Uses CNNs to classify whether a file is a malware or not.

Karn et al., 2020

Cryptomining detection in container clouds using system calls and explainable machine learning.

Machine Learning Advances in Malware Detection: Uses LSTMs system calls in Kubernetes pods, for cryptomining malware.

Landman and Nissin, 2021

Deep-hook: A trusted deep learning-based framework for unknown malware detection and classification in linux cloud environments.

Deep-Hook: Detects obfuscated malware by monitoring applications and analyzing memory dumps.

THIS PAPER'S RESEARCH SCOPE

FULL CONTAINERS

Previous research examines single-files, we focus on the attestation of the whole software container, i.e. a multi-GB file.

VIA THE FILE SYSTEM

We want to attest docker containers before deploying them, that is, without access to memory dumps or runtime behavior just the file system.

EFFICIENT

We want the method to scale to arbitrarily large containers and be able to run on small or no GPUs, with relatively low runtimes.

TOWARDS ZERO DAY

We want a method that will be able to learn patterns and perhaps recognize novel threats. Malware has basic recurring patterns such as changes in the registry or other system files.

CONTRIBUTIONS

4 MAIN CONTRIBUTIONS

C1. TASK FORMULATION

We formulate the Novel task of identifying malware-compromised dockerized software containers with ML-based methods and propose a streaming, patch-based CNN approach.

C2. DATA GENERATION PIPELINE

We define a fully customizable and scalable data generation pipeline for creating images of benign and compromised software containers across various OS and CPU architectures.

C3. COMPROMISED SOFTWARE CONTAINER DATASET

We introduce a novel dataset containing 3,364 large-scale RGB image representations of benign and compromised dockerized software containers.

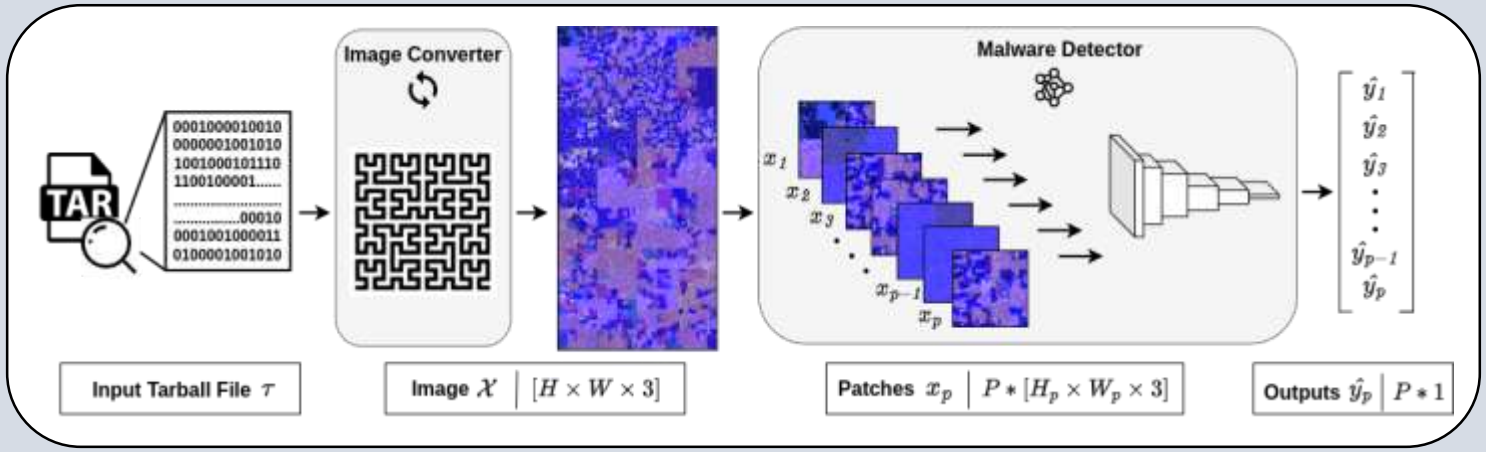
C4. EXPERIMENTAL ANALYSIS

We show experiments with various CNNs and demonstrate our approach outperforms commercial methods.

IEEE ICC®

PROPOSED METHOD

PROPOSED METHOD



Steps

Our method assumes as input the tarball file binary array representing the dockerized software container.

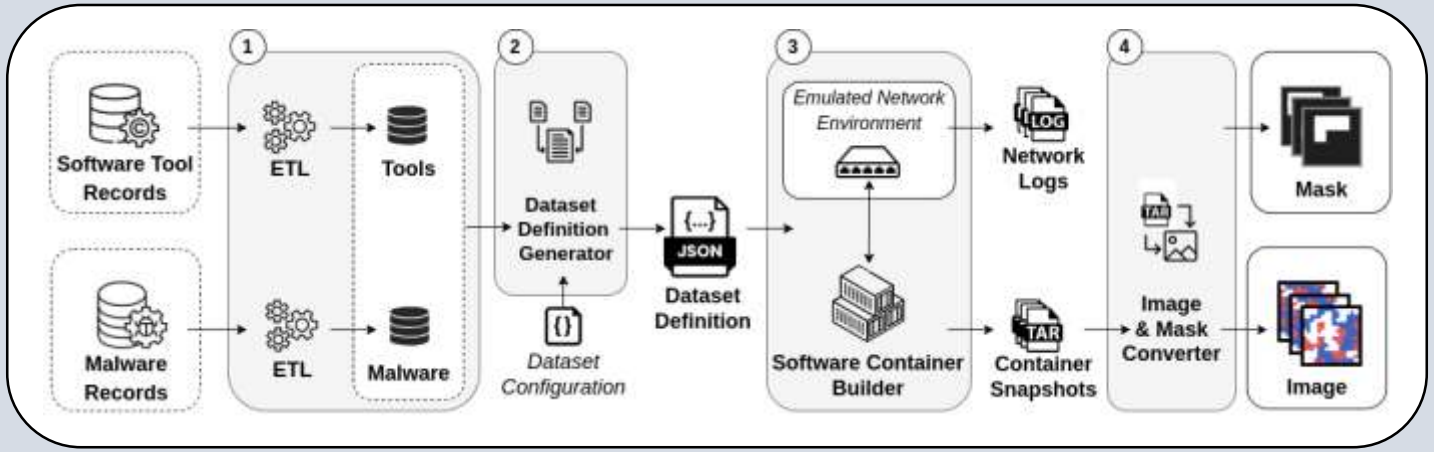
1. **Image Converter:** Casts the byte array onto an RGB image representation using an Image converter based on Hilbert space-filling curve.
2. **Malware Detector**: (i) Splits the image into patches of predefined shape and process them with a CNN model. (ii) Performs predictions on each patch whether it is malicious or not.

Advantages

- The patch-based approach allows to process larger containers (images) that would not fit into the GPU otherwise.
- Allows for early-exit inference, i.e. it can stop once it detects the first malicious patch, enabling faster runtimes.
- Explainable-by-design, as the malicious patch points to the malicious bytes when reversing the Hilbert curve from the image back to the bytes.

DATA GENERATION PIPELINE

DATA GENERATION PIPELINE



1. Extract raw tool & malware records

Ingest tools from the Linux APT package manager as well as malwares from MalwareBazaar.

2. Dataset Definition Generator

Generates a Dataset Definition by selecting malware, tools, OS and CPU architecture. Allows fine grained control over the data generation. The definition is a blueprint of the dataset to-be-produced.

3. Software Container Builder

Builds benign and compromised software containers from the dataset definition and stores them as tarfiles.

4. Image & Mask Converter

Converts tarfiles to images. Uses container difference tool to create segmentation masks.

COMPROMISED SOFTWARE CONTAINER DATASET

THE COMPROMISED SOFTWARE CONTAINER DATASET (I)

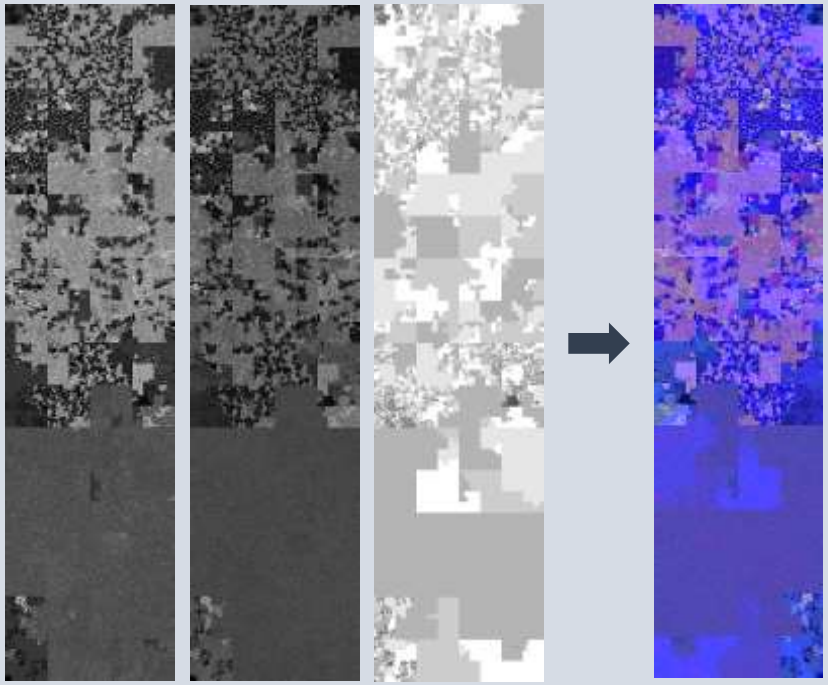


Figure: Channel decomposition of a 1024x4096 down-sampled software container image. From left to right: R-channel: byte-class, G-channel: byte-value, B-channel: tarball file structure, RGB image

Based on the Data Generation Pipeline, we introduce a synthetic dataset of 3364 images representing benign and malware-Compromised Software Containers (COSOCO).

Each image in the dataset represents a dockerized software container that has been converted to an image. Software container records are labelled **benign** or **compromised**:

- A benign software container will have installed commonly used harmless packages and tools.
- A compromised software container, will have, among harmless benign tools and packages, its underlying file system affected by some activated malware instance.

Each compromised instance is accompanied by a mask, i.e. a black and white image which marks the pixels that correspond to the files of the underlying system that have been altered by a malware.

THE COMPROMISED SOFTWARE CONTAINER DATASET (II)

DATASET STATISTICS

	Total	Train	Validation	Test
Nr. Images	3364	2360	328	676
Nr. Benign Images	2225	1564	214	447
Nr. Compromised Images	1139	796	114	229
Nr. Unique Packages	1297	1053	206	393
Nr. Unique Malware	495	347	49	99
Avg. Image Size	158 MP	158 MP	157 MB	157 MP
Avg. Mask / Image Ratio	0.32%	0.35%	0.29%	0.24%

MALWARE STATISTICS

Signature	Unique	Total	Avg. Bytes affected
Mirai	225	494	58 KB
Gafgyt	119	284	132 KB
CoinMiner	28	72	451 KB
XorDDos	27	50	18 KB
Kaiji	21	53	4.7 MB
Tsunami	16	43	1024 B
GoBrut	14	34	512 B
BPFDoor	7	16	20 KB
RotaJakiro	5	5	134 KB
Unknown	33	79	678 KB
Total	**495**	**1139**	**355 KB**

IEEE ICC®

EXPERIMENTAL ANALYSIS

QUANTITATIVE RESULTS (I) – CNN ARCHITECTURES COMPARISON

- We present results for multiple lightweight CNN architectures along with their parameter counts. All models were trained on a downscaled dataset version were software containers were represented by 1024 x 4096 sized images.
- We evaluate on multiple metrics including F1 score, Precision, Recall and Accuracy. We rank model performance based on **F1 Score**.

Key Metrics	F1 score	Precision	Recall	Accuracy	# Params (M)
VGG	0.552	0.928	0.393	0.784	132.9
ShuffleNet v2	0.586	0.857	0.445	0.787	2.3
MobileNet v2	0.622	0.855	0.489	0.799	3.5
AlexNet	0.680	0.821	0.581	0.815	61.1
EfficientNet	0.724	0.861	0.624	0.837	5.3
ResNet18	0.736	0.826	0.664	0.839	11.7

VGG has the worst performance despite highest parameter count.

EfficientNet is second in performance with half parameters than ResNet.

ResNet18 stands out in terms of performance with an F1 score of 0.736.

QUANTITATIVE RESULTS (II) - VIRUS TOTAL COMPARISON

- To establish a baseline we use **Virus Total**, an online platform with more than 70 commercial antivirus scanners to evaluate the test set of the Compromised Software Container dataset. We upload the test set and parse the results of their analysis.
- Our method outperforms all individual engines as well as all their ensembles while being a lot faster than the API call and not relying on external software.

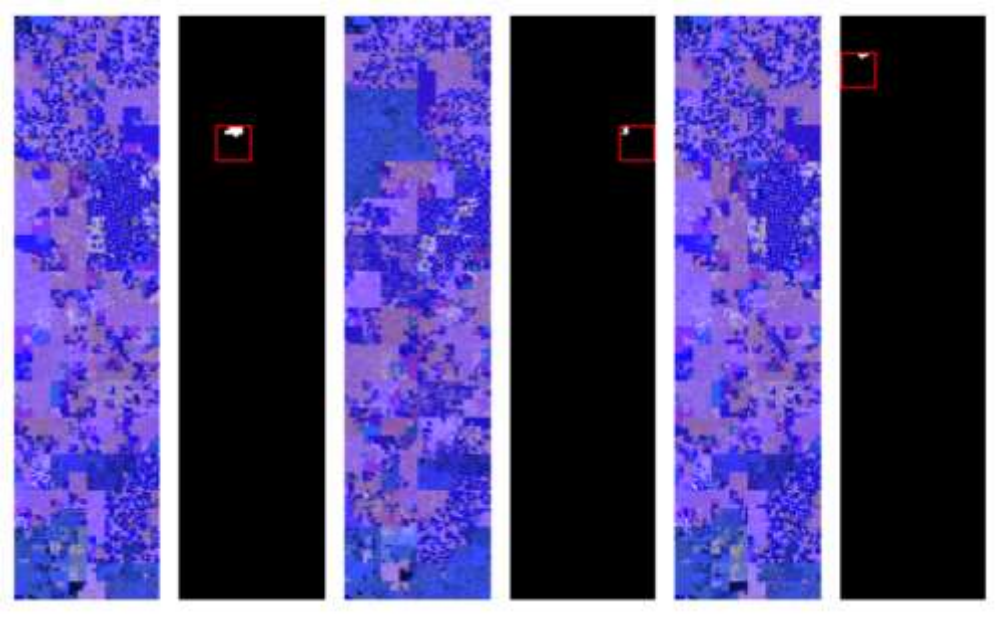
Key Metrics	F1 score	Precision	Recall	Accuracy
Pandas (#32)	0.085	1.000	0.044	0.679
Karspersky (#18)	0.598	1.000	0.427	0.807
Ikarus (#1)	0.703	1.000	0.542	0.846
VirusTotal (≥ 20)	0.601	0.990	0.431	0.807
VirusTotal (≥ 1)	0.709	0.992	0.551	0.848
ResNet18	**0.736**	0.826	0.664	0.839

Evaluation results of popular engines and their relative ranking (#) among other engines

Evaluation results for "ensembles of engines". With '≥ a' we denote the scores where at least a engines classify a record as malevolent.

Our approach **outperforms** all commerial solutions.

QUALITATIVE RESULTS



Each pair illustrates an image and its respective mask. White pixels refer to the bytes compromised by malware activity, whereas the red bounding boxes indicate the patch predicted as malevolent using ResNet18.

Our streaming, patch-based approach is:

a) **Scalable:** Reduces GPU memory required, as it processes the image in patches.

b) **Efficient:** Implicitly allows early-exit inference to improve the runtime, as it finishes inference after the first malicious patch is detected.

c) **Explainable-by-design:** By identifying the patch with malicious content, provides explanations by design. One can easily reverse engineer the image generation process using the Hilbert space-filling permutation and derive the exact byte locations of the malicious component within the container.

CONCLUSIONS & FUTURE WORK

CONCLUSIONS

- We introduced the task of identifying compromised software containers given their file system with machine learning and proposed a streaming, patch-based CNN approach.
- To encourage further research, we introduced a novel dataset of Compromised Software Containers (COSOCO) along with a Dataset Generation Pipeline.
- We used the COSOCO dataset to train CNN classification models and produce baseline results

FUTURE WORK

- Investigate Transformer architectures for direct training on byte-level representations.
- Examine the performance vs efficiency trade-offs when training at higher image resolutions.
- Investigate techniques under the framework of Multiple Instance Learning (MIL) that may be more suitable.

IEEE ICC®

APPENDIX B: TOOL DEMONSTRATION

TOOL DEMO

TOOL DEMONSTRATION