Compromised Software Containers (COSOCO) Dataset – README File

K3Y Ltd - https://k3ylabs.com/

Authors: Akis Nousias, Efklidis Katsaros, Evangelos Syrmos, Panagiotis Radoglou-Grammatikis, Thomas Lagkas, Vasileios Argyriou, Ioannis Moscholios, Evangelos Markakis, Sotirios Goudos and Panagiotis Sarigiannidis

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

1. Introduction

This report serves as documentation on the Compromised Software Containers (COSOCO) image dataset, that accompanies the publication '*A. Nousias, E. Katsaros, E. Syrmos, P.R. Grammatikis, T. Lagkas, V. Argyriou, I. Moscholios, E. Markakis, S. Goudos and P. Sarigiannidis, "Malware Detection in Docker Containers: An Image is Worth a Thousand Logs", ICC-W 2025 - IEEE International Conference on Communications Workshops, 2025*', where we address the vulnerability of software containers with Machine Learning (ML) methods and introduce a new dataset of benign and compromised dockerized software containers. By open sourcing the COSOCO dataset along with the description of the data generation pipeline, we aim to support ML research in malware detection on software containers and cybersecurity in general.

COSOCO is a synthetic dataset of 3364 images representing benign and malware-compromised software containers. Each image in the dataset represents a dockerized software container. The container-to-image conversion was performed with common byte-to-pixel tools widely used in malware analysis. Software container records are labelled (1) **benign** or (2) **compromised**: A **benign** software container will have installed commonly used harmless packages and tools, whereas 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.

COSOCO aims to support the identification of compromised software containers via the task of image classification task and the identification of compromised files and file system regions inside a container via the image segmentation task.

This document has the following outline:

Section **"1. Introduction"** is the current section.

Section **"2. Data Generation Pipeline"** is a description of the data generation pipeline along with the methodology and tools that were used to generate the COSOCO dataset. We describe the pipeline components, their functionality and outline all processing steps applied during dataset creation. In addition, we provide detailed information for all data artifacts including intermediate ones.

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

Section **“3. Dataset Structure”** provides an overview of the contents and organization of the COSOCO dataset. In addition, quantitative summaries of the dataset are included, such as the number of samples, distribution of classes or labels, data split proportions, and key metrics (e.g., average image size or mask coverage). Charts and tables are provided for ease of interpretation.

Section **“4. Citation”** includes instructions on how to properly cite the COSOCO dataset in research publications, including a BibTeX entry and acknowledgment requirements.

Finally, the **“APPENDIX”** section includes descriptions of each data process along with descriptions of the data assets that the process produces.

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

2. Dataset Generation Pipeline

2.1. Pipeline Architecture

The data generation pipeline is illustrated in Figure 01.

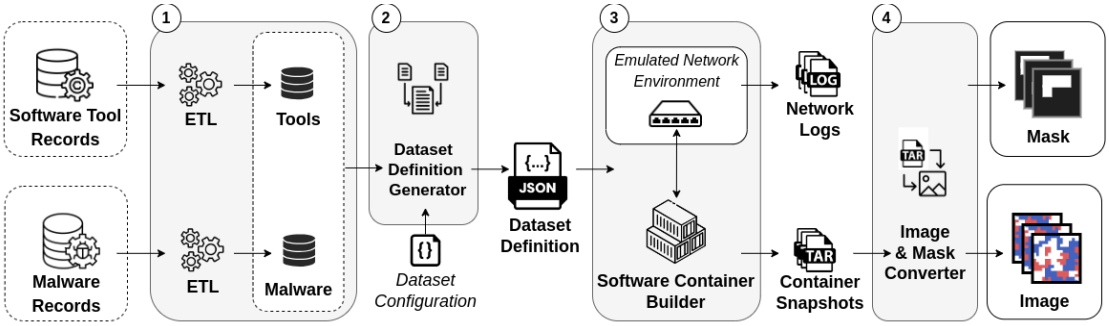

Figure 1: Data generation pipeline outline: (1) ingest tools from the APT package manager as well as malware records from MalwareBazaar, (2) sample tool and malware records to create a dataset definition, (3) for each record in the definition, build a container, convert it to a tarball file along with network traffic logs, and (4) generate filesystem images along with masks that highlight any malware changes.

It consists of the following functionalities organized into pipeline components:

1. **ETL:** This component extracts and transforms (ETL) raw data on APT packages records and Malware Bazaar [2] records. Additional information on the data schemas can be found in Appendix, sections A and B respectively.

2. **Dataset Definition Generator:** This component generates a Dataset Definition file from Malware and APT package records based on user-configured settings given as a Dataset Configuration file. The output Dataset Definition file is a `.json` file that holds software container configuration records along with metadata that will be used by the Software Container Builder component. Additional information on the schema of the Dataset Definition file can be found in Appendix section C.

3. **Software Container Builder:** Given a Dataset Definition file as input, this component builds and exports dockerized software containers as tarball files along with software container metadata. It consists of a series of sub-components that rely on a container runtime environment (e.g. gVisor) to manage the stages of the container-building process: (i) dockerfile generation, (ii) container image build and (iii) container tarball file export. Malware execution within each container occurs in a sandboxed environment with emulated network capabilities. For each malware-compromised container, a corresponding non-compromised container is generated, allowing for byte-level comparisons of malware impact. File changes between compromised and non-compromised containers are tracked using open-source container comparison tools.

4. **Image & Mask Converter:** This component converts tarball files into image - mask pairs. Converting byte values to pixels is straightforward as both take values in the

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

range [0-255]. The layout of the bytes on the image is achieved using Hilbert space-filling curves due to its locality-preserving properties.

The resulting image channels include the following information:

a. A byte-class channel, where each pixel gets a colour-code based on a crude classification of its value range (i.e., ASCII text, other bytes, and special colours for the 0x00 and 0xff byte values).
b. A byte-value channel, where each pixel holds the exact byte value of the input byte array.
c. A tarball file structure channel, where each pixel holds varying colours to indicate separate files in the original tarball file along with additional indicators between the file's header and content.

Given an image width w, the bytes transformed to pixel values are split into w^2-sized chunks which are then mapped to w × w square patches. This process continues until all bytes are exhausted. The image mask is created based on the container difference information stored in the container builder metadata file. Additional information can be found in Appendix section E.

2.2 Pipeline Data Assets

Table 1 describes the data assets that participate in the Data Generation Pipeline, their format and the Pipeline components that produce and consume them. Additional detailed information per data asset regarding their creation process, schema and data sample can be found in the Appendix.

Table 1: Overview of data assets that participate in the Data Generation Pipeline

Data Asset	Asset Format	Produced by Component	Consumed by Component	Description
APT packages records	tsv	ETL	Dataset Definition Generator	APT package records along with APT package metadata
Malware Bazaar records	tsv	ETL	Dataset Definition Generator	Malware records from Malware Bazaar along with malware metadata
Dataset Configuration	toml	User	Dataset Definition Generator	User Input how the Software Container dataset will be generated. Includes filters to be applied on the APT packages and Malware records along with configuration parameters.
Dataset Definition	json	Dataset Definition Generator	Software Container Builder	A dataset of software container configuration records, where each record has a unique identifier, packages and malware to be installed along with package and malware metadata.
Dockerfile	txt	Software Container Builder	Software Container Builder	Dockerfile with software container build instructions (one for each dataset record)
Software Container Tarball	tar	Software Container Builder	Image & Mask Converter	Tarball file representing an exported software container (one for each dataset record)
Network Log	txt	Software Container Builder	NA	Log file of malware recorded network activity from malware affected software containers (one for each compromised dataset record)

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

Container Builder Metadata	json	Software Container Builder	Image & Mask Converter	Metadata on the built software containers, both benign and compromised, such as container characteristics, installed packages, activated malware, and affected files (when applicable).
Image	png	Image & Mask Converter	NA (Final Artifact)	Image representing an exported software container tarball file (one for each dataset record)
Mask	png	Image & Mask Converter	NA (Final Artifact)	Mask representing affected container files (one for each compromised dataset record)

2.3. Dataset Generation

The COSOCO dataset was generated through an iterative process, where intermediate versions contributed to the progressive refinement of the overall pipeline. Intermediate datasets were used for filtering malware samples, adjusting the task formulation and refining key pipeline components and the input-output schemas. For the final dataset version:

- **ETL:** The APT package records were created by an APT package ETL process being executed on 12 Feb 2024 whereas the Malware Bazaar records were created via the Malware Bazaar ETL process being executed on 21 May 2024.

- **Dataset Definition Generator:** The Dataset Definition, i.e. the configuration for each software container record, was created with the filters and conditions described in Table 2.

Table 2: Parameters and inputs for the final dataset configuration

Config Target	Config Type	Description
Malware Bazaar records	Filters	▪ 'file_type' == 'elf' ▪ 'cpu_arch' == 'x64' or 'cpu_arch' == 'x32' ▪ 'signature' in ('Mirai', 'Gafgyt', 'CoinMiner', 'XorDDos', 'Kaiji', 'Tsunami', 'GoBrut', 'BPFDoor', 'RotaJakiro', 'Unknown')
APT package records	Filters	▪ 'Installed-Size' < '50MB' ▪ 'Depends-Size' == '0 MB' ▪ 'Essential' != 'no' ▪ 'Section' != 'universe/doc'
Output records	Parameter	▪ Sample 1-3 APT package record ▪ Sample 1 Malware Bazaar record

- **Software Container Builder:** The benign and compromised Software Containers were generated in Virtual Machines hosted in Google Cloud Platform (GCP). For each software container, `ubuntu/jammy` was selected as base image. All images included the `wget` and `p7zip` packages which were required to facilitate the generation process. Malware activation was performed inside a sandbox container runtime environment based on `gVisor` with a network emulated by FakeNet-NG 3.2 [3]. For the activation process itself: (i) malware executables were downloaded from the Malware Bazaar API and copied inside the container, (ii) activated through execution as `elf` binaries and removed after activation, (iii) the resulting software container was compared with its benign counterpart using Google's `container-diff` tool [4] and changes on the filesystem, i.e. additions, modifications and deletions were recorded. (iv) Software containers in which no file

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

system alteration took place, or changes included only deletions were dropped from the dataset.

- **Image & Mask Converter**: The final images were generated with an in-house implementation of a byte-to-pixel transformations and visualization tool similar to `binvis` [1]. Besides the full image of the tarball file, down-sampled images of dimensions 1024 × 4096 and 2048 × 8192 are also provided. For the down-sampling, the input byte-array is sampled at regular intervals with an interval step defined in a way so that the input array fits in the target image size.

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

3. Dataset Structure

3.1. Dataset Files & Formats

The COSOCO dataset has the following folder structure:

- **dataset-root-folder:**
  - **1024_unrolled/**
  - **2028_unrolled/**
  - **4096_unrolled_n/**
  - **dataset_generation/**
    - **apt_packages**
    - **malware_bazaar**
    - **dataset_definition**
    - **dockerfiles**

The contents of each folder along with the file format are described in the Table 3.

Table 3: Dataset Folder Data Description

Dataset Folder	Description	Format
1024_unrolled	Image – Mask pairs of benign and compromised software containers, along with container metadata, downsampled in 1024x4096 separated per train, validation, test splits.	webdataset
2048_unrolled	Image – Mask pairs of benign and compromised software containers, along with container metadata, downsampled in 2048x8192 separated per train, validation, test splits.	webdataset
4096_unrolled_n	Image – Mask pairs of benign and compromised software containers, along with container metadata, in full resolution encoded in images of width 4096 and variable height separated per train, validation, test splits.	webdataset
dataset_generation/ apt_packages	APT package records along with APT package metadata	tsv
dataset_generation/ malware_bazaar	Malware records from Malware Bazaar along with malware metadata	tsv
dataset_generation/ dataset_definition	A dataset of software container configuration records, where each record has a unique identifier, packages and malware to be installed along with package and malware metadata.	json
dataset_generation/ dockerfiles	Dockerfile with software container build instructions (one for each dataset record)	txt

3.2. Dataset Statistics

The dataset consists of image-mask pairs in `.png` format representing 3364 software container records. The container records include benign tools sampled from 1297 unique Debian/Ubuntu packages coming from APT repositories as well malware samples focused on Linux OS (Ubuntu) and x86_64 architectures sampled from 495 unique malware samples, delivered as ELF binaries spanning 10 malware classes / signatures: Mirai, Gafgyt, CoinMiner, XorDDos, Kaiji, Tsunami, GoBrut, BPFDoor and RotaJakiro and an extra Unknown class for unclassified malware. More information on the represented malware classes can be found in the Appendix on section E. Each software container record has been converted to an image through byte-to-pixel visualization tools such as `binvis`. 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.

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

The dataset contains images in three different resolutions:

1. **4096_unrolled_n**: These images are a direct 1-to-1 map from the original tarball files. They have a fixed width of 4096 pixels and a variable height based on the input tarball size.
2. **2048_unrolled**: These images are down-sampled versions of the input tarball file mapped to a fixed target image size of 2048x8192 pixels. For the down-sampling, the input byte-array is sampled at regular intervals with an interval step defined in a way so that the input array fits in the target image size.
3. **1024_unrolled**: These images are down-sampled versions of the input tarball file mapped to a fixed target image size of 1024x4096 pixels. For the down-sampling, the input byte-array is sampled at regular intervals with an interval step defined in a way so that the input array fits in the target image size.

The dataset is split in train, validation and test set in a 70:10:20 split ratio using malware-class-based stratified sampling.

In addition, supplementary data files associated with the data generation process are provided for validation and replication purposes. Specifically, we provide:

- The APT package records and Malware Bazaar records, both raw data and parsed, that were used for the dataset definition generation as described Section 1.3.
- The Dataset Definition file with software container configurations records along with metadata that were used by the Software Container Builder component.
- The Dockerfiles of the final software containers, both benign and malevolent.

Table 4 includes information on the generated dataset overall and per split statistics.

Table 4: Generated Dataset Overall and per split 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
Fixed Image Size (1024_unrolled)	4.18 MP	4.18 MP	4.18 MP	4.18 MP
Fixed Image Size (2048_unrolled)	16.78 MP	16.78 MP	16.78 MP	16.78 MP
Avg. Image Size (4096n_unrolled-full)	158 MP	158 MP	157 MB	157 MP
Avg. Mask / Image Ratio	0.32%	0.35%	0.29%	0.24%

Table 5 includes statistics per malware type and their effect, after activation, on the underlying file system regarding total size of affected files. Additional information on Malware signatures can be found in Appendix section E.

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

Table 5: Generated Dataset Malware Statistics per Malware Signature

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

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

4. Citation

If you use the Compromised Software Containers (COSOCO) dataset, please cite the following paper:

'A. Nousias, E. Katsaros, E. Syrmos, P.R. Grammatikis, T. Lagkas, V. Argyriou, I. Moscholios, E. Markakis, S. Goudos and P. Sarigiannidis, "Malware Detection in Docker Containers: An Image is Worth a Thousand Logs", ICC-W 2025 - IEEE International Conference on Communications Workshops, 2025'

Or use the following bibtex entry:

```
@misc{
  nousias2025malwaredetectiondockercontainers,
  title={Malware Detection in Docker Containers: An Image is Worth a
Thousand Logs},
  author={Akis Nousias and Efklidis Katsaros and Evangelos Syrmos and
Panagiotis Radoglou-Grammatikis and Thomas Lagkas and Vasileios Argyriou
and Ioannis Moscholios and Evangelos Markakis and Sotirios Goudos and
Panagiotis Sarigiannidis},
  year={2025},
  eprint={2504.03238},
  archivePrefix={arXiv},
  primaryClass={cs.CR},
  url={https://arxiv.org/abs/2504.03238},
}
```

5. References

[1] Cortesi A., (2013), scurve (v0.2), GitHub repository, https://github.com/cortesi/scurve

[2] Abuse ch, "Malware Bazaar," https://https://bazaar.abuse.ch/.

[3] Kacherginsky P., (2016), Fakenet-NG (v3.2.alpha), GItHub repository, https://github.com/mandiant/flare-fakenet-ng

[4] GoogleContainerTools, (2021), container-diff (v0.19.0), GitHub repository, https://github.com/GoogleContainerTools/container-diff

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

APPENDIX

A. The APT Package Records data file

Description: This is a data file in `.tsv` format containing APT package records with useful APT package metadata. It is used for selecting and sampling benign packages to be installed in software containers. This data file is produced by the ETL component and consumed by the Dataset Definition Generator.

ETL Process: Information on available APT packages were stored in data files in `.tsv` format via an ETL process via the following steps:

1. Update local APT package manager via the `apt-get update` command.
2. Retrieve a list of available APT packages via the `apt list` command.
3. For each package in the list:
    a. Retrieve additional metadata via the `apt show <pkg>` command.
    b. Parse retrieved metadata
    c. Store in data file.

Schema: The schema of the output `.tsv` data files of APT packages ETL process is described in Table A1.

Table A1: Schema of the generated APT packages data files

Field	Field Type	Field Description
Package	str	Name of the package
Version	str	Version number of the package
Priority	str	Priority of the package
Section	str	Section to which the package belongs (e.g. admin, libs etc.)
Source	str	Source package name (if different from the binary package
Origin	str	The origin of the package (usually the distribution name)
Maintainer	str	Name and email of the package maintainer
Bugs	str	Bug tracking URL or email address for reporting issues
Installed-Size	int	Estimated disk space required for the installed package
Pre-Depends	str	Packages that enhance the functionality of the main package
Depends	json	Dependencies required for the package to function properly
Breaks	json	Packages that the main package breaks
Replaces	json	Packages that the main package can replace
Homepage	str	URL of the package homepage
Download-Size	int	Size of the package in bytes
APT-Sources	str	List all sources that reference this package's repository
Description	str	A detailed description of the package including its purpose, features, and usage
Essential	str	Whether the package is characterized as essential
Depends-Size	int	Estimated disk space required for the dependencies of the main package

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

B. The Malware Bazaar Records data file

Description: This is a data file in `.tsv` format containing Malware Bazaar records with useful malware metadata. It is used for selecting and sampling malware bazaar records to be activated in software containers. This data file is produced by the ETL component and consumed by the Dataset Definition Generator.

ETL Process: Metadata of the Malware Bazaar available records are retrieved via the Malware Bazaar website from the bulk download section. These records are offered as a `.csv` dump file in the URL `https://bazaar.abuse.ch/export/csv/full/` [accessed 2025-Feb-05].

To facilitate the malware selection process, the dump file was augmented with additional tag information, available only through API and retrieved via the endpoint `https://mb-api.abuse.ch/api/v1/get_tag_info`.

Schema: The schema of the output `.tsv` data files of Malware Bazaar ETL process is described in Table B1.

Table B1: Schema of the generated APT packages data files

Field	Field Type	Field Description
first_seen_utc	date	TimeStamp when the file has been first seen by MalwareBazaar (UTC)
sha256_hash	str	SHA256 hash of the malware sample
md5_hash	str	MD5 hash of the malware sample
sha1_hash	str	SHA1 hash of the malware sample
reporter	str	Twitter handle of the report (or anonymous for anonymous submissions)
file_name	str	Malware sample's file name
file_type_guess	str	File Type
mime_type	str	MIME file type
signature	str	Malware family (if available)
clamav	str	List of ClamAV detections (empty column)
vtpercent	float	Percentage value related to VirusTotal (VT). It may indicate the level of confidence that the malware is indeed malicious based on the analysis by different security products
imphash	str	An imphash (only available for PE executables)
ssdeep	str	The hash from `ssdeep`, a program for computing context triggered piecewise hashes (CTPH). Also called fuzzy hashes, CTPH can match inputs that have homologies
tlsh	str	Trend MIcro Locality Sensitive Hash
tags	str	Pipe-separated ('|') string with tags assigned to a malware sample

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

C. The Dataset Definition data file

Description: This is a data file in `.json` format with configuration records for software containers. It is created based on user input and holds all information required by the Software Container builder to generate benign and compromised software containers, along with useful package and malware metadata. This data file is produced by the Dataset Definition Generator component and consumed by the Software Container Builder.

Schema: The schema of the Dataset Definition `.json` file is shown in Table C1.

Table C1: The JSON Schema of the Dataset Definition data file

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "title": "Compromised Software Container Dataset Definition",
    "description": "A dataset of software container configuration records, where each record has a unique
identifier, packages and malware to be installed along with package and malware metadata.",
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "uid": {
                "type": "string",
                "description": "A unique identifier assigned to the software container configuration."
            },
            "packages": {
                "type": "array",
                "items": {
                    "type": "object",
                    "properties": {
                        "Package": {
                            "type": "string",
                            "description": "Name of the package."
                        },
                        "Version": {
                            "type": "string",
                            "description": "Version of the package."
                        }
                    },
                    "required": [
                        "Package", "Version"
                    ]
                }
            },
            "malware": {
                "type": "array",
                "items": {
                    "type": "object",
                    "properties": {
                        "sha256_hash": {
                            "type": "string",
                            "description": "SHA256 hash of the malware sample."
                        },
                        "file_type_guess": {
                            "type": "string",
                            "description": "The malware's file type."
                        },
                        "signature": {
                            "type": "string",
                            "description": "Malware family (if available)."
                        },
                        "tags": {
                            "type": "string",
                            "description": "Pipe-separated ('|') string with tags assigned to a malware sample"
                        }
                    },
                    "required": [
                        "sha256_hash", "file_type_guess", "signature", "tags"
                    ]
                }
```

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

```
            },
            "label": {
                "type": "string",
                "description": "The label of the software container record.",
                "enum": [
                    "benign", "malevolent"
                ]
            }
        },
        "required": [
            "uid", "packages", "malware", "label"
        ]
    }
}
```

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

D. Images & Masks

Description: The image and mask representations of benign and compromised software containers were created with an in-house implementation of a byte-to-pixel visualization tool similar to `binvis` [1]. Figure D1 shows a decomposition of an RGB image sample to its 3 channels.

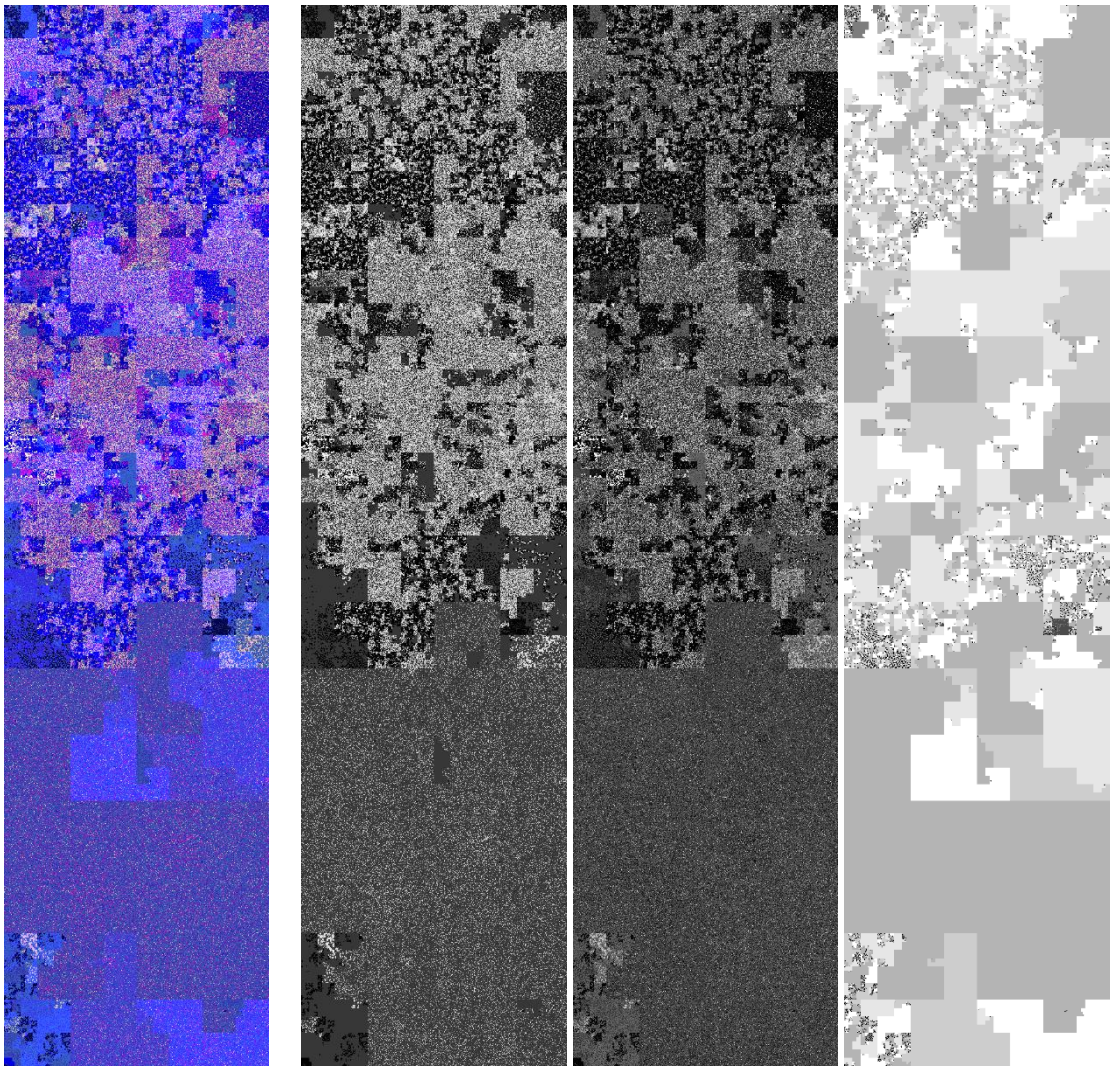

Figure D1: Channel decomposition of a 1024x4096 down-sampled software container image. From left to right: RGB image, R-channel: byte-class, G-channel: byte-value, B-channel: tarball file structure.

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

E. Malware Signatures Descriptions

Table E1: Description of Malware signatures included in the COSOCO dataset.

Malware Signature	Malware Type	Description
Mirai	botnet	**Mirai** is malware that turns networked devices running Linux into remotely controlled bots that can be used as part of a botnet in large-scale network attacks. It primarily targets online consumer devices such as IP cameras and home routers but can also target web servers and data centres.
Gafgyt	botnet	**Gafgyt**, also known as BASHLITE, is malware written in C, which infects Linux systems to launch distributed denial-of-service attacks (DDoS).
CoinMiner	coin-miner	**CoinMiner** is an OS agnostic malware which uses the victim's computational power (CPU and RAM mostly) to mine for coins (for example Monero or Zcash). The malware achieves persistence by adding one of the opensource miners on startup without the victim's consensus. Most sophisticated coin miners use timer settings or cap the CPU usage to remain stealthy.
XorDDos	ddos	**XoRDDoS** is a Linux-based malware family known for its DDoS capabilities. It mostly targets Docker servers and its hosted containers but can also infect Linux servers and IoT devices. The affected devices are turned into bots that can be used to launch DDoS attacks against targeted websites or services.
Kaiji	ddos	**Kaiji** is a Linux-based malware, written in Go, that spreads through brute force attacks on SSH ports on Linux servers and IoT devices. It tries to get root access and after activation, spreads to other devices in the network. The affected devices are then manipulated to carry out DDoS attacks on targets.
Tsunami	ddos	**Tsunami** is a Linux-based malware family known for its DDoS capabilities. It infects Linux servers and IoT devices, turning them into bots that can be used to launch large-scale DDoS attacks against targeted networks or services.
GoBrut	botnet, worm	**GoBrut** (also known as StealthWorker) is a 2019 observed worm and botnet written in the Go programming language. Its variants target both Microsoft Windows Server and UNIX systems. Once installed, it will at first connect to a Command and Control (C2) server and then attempt to gain access to other systems and devices on the network.
BPFDoor	backdoor	**BPFDoor** is a custom backdoor targeting Linux-associated systems which utilizes multiple protocols including TCP, UDP and ICMP, that allows it after activation to communicate with a Command and Control (C2) server. It was identified in 2021 and is associated with a China-based threat actor named Red Menschen. [PwC Cyber Threats 2021 report, link]
RotaJakiro	backdoor	**RotaJakiro** is a stealthy Linux backdoor which uses rotating encryption to encrypt the resource information within the sample, and Command and Control (C2) communication, using a combination of AES, XOR, ROTATE encryption and ZLIB compression.

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