SecBox: a Lightweight Platform for Dynamic and Reproducible Malware Analysis

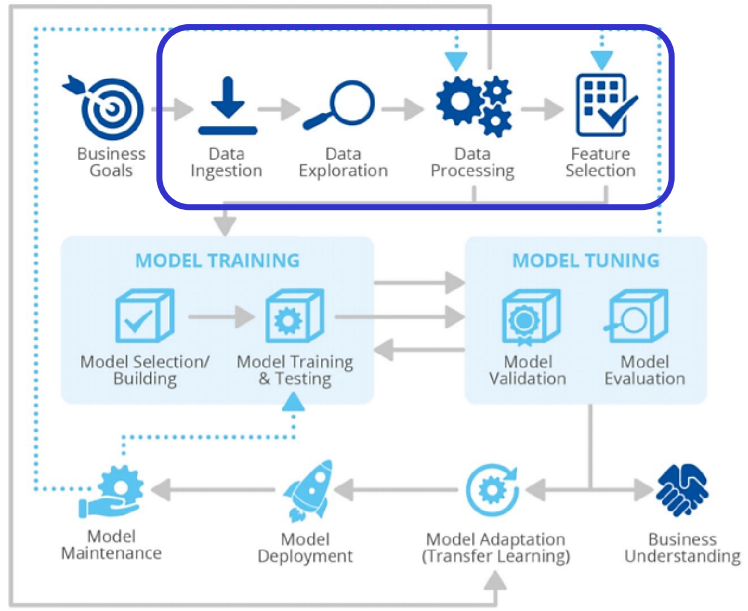
Chao Feng, Jan von der Assen, Alberto Huertas Celdrán, Raffael Mogicato, Adrian Zermin, Vichhay Ok, Gérôme Bovet, Burkhard Stiller

Communication Systems Group CSG, Department of Informatics Ifl
University of Zürich UZH, vonderassen@ifi.uzh.ch

Motivation, Challenges, Contributions
Platform Architecture
Implementation
Evaluations

Introduction and Motivation

- Malware: Still relevant in security
  - 20% of the malware persists for more than 400d[1]
- Accurate AI-based detection
  - Static features vs. behavior
  - Morphing threat vector
  - Requires dynamic execution
    - Frequency: 294 samples in 24h (April, 29)
- Challenges in Research and Practice
  - *ch1)* Secure execution of Linux-based samples
  - *ch2)* Lightweight isolation mechanisms not explored
  - *ch3)* Reproducible analysis of artefacts
  - *ch4)* Realtime, visual exploration of heterogeneous data



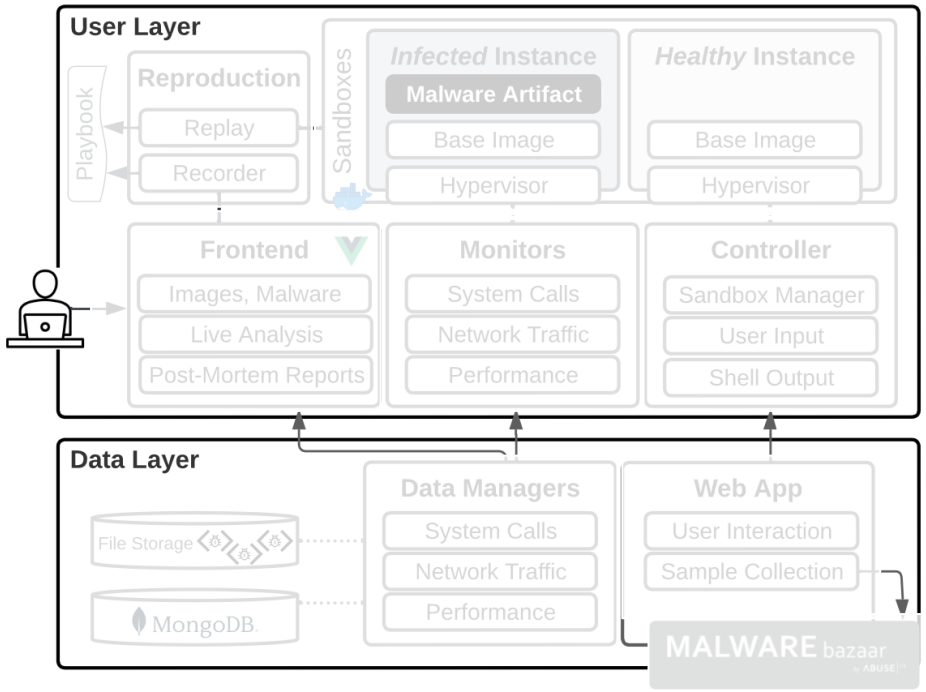
[1] Sean Cordey, ETH Zürich: "Software Supply Chain Attacks, An Illustrated Typological Review", 2023, Zürich, Available Online

Contributions

- SecBox, a container-based malware analysis platform
  - Modular platform → *ch1*
    - Secure execution decoupled from data collection, infrastructure
  - Container-based malware execution → *ch2*
    - Integration of Linux artefacts and images
  - Interactive, yet reproducible analysis → *ch3*
  - Post-mortem mining: Various dimensions → *ch4*
- Set of experiments using real malware → *ch1-4*
  - Performance
  - Isolation, Visibility
  - Monitorability
  - Automation, Reproducibility

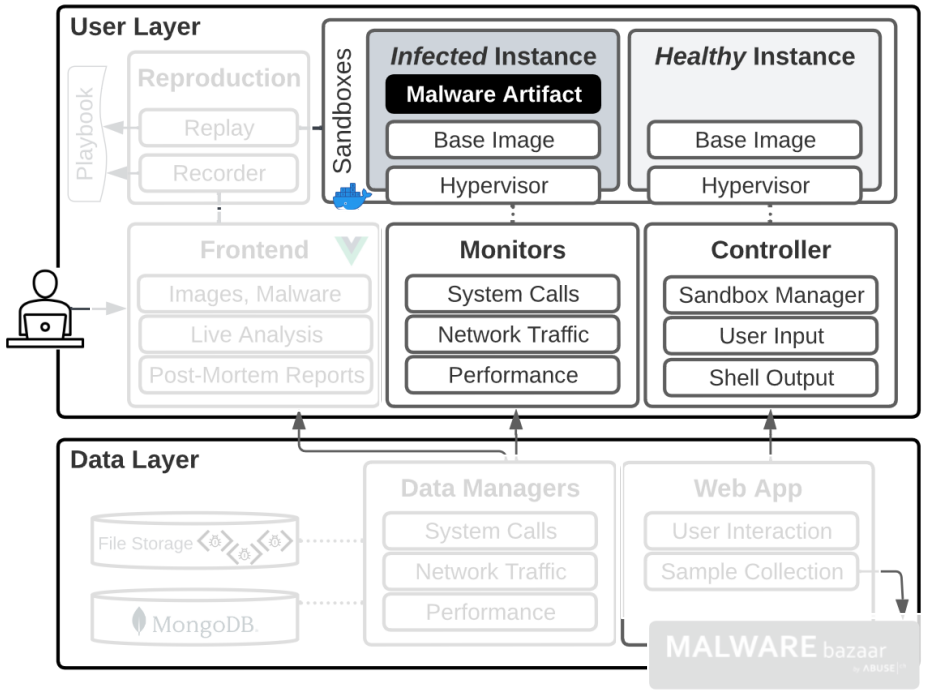
SecBox Platform Architecture (1/3)

- Sandboxes
  - Execute malware artifacts
    - 2 containers deployed
  - Docker with gVisor hypervisor
  - Base image
  - Malware artifact pre-loaded
- Controller
  - Manage life cycle of sandboxes
  - Relay Input/Output (shell cmds)
- Monitors
  - Capture data from Docker, gVisor
    - Network Traffic capture, System Calls, Performance Metrics



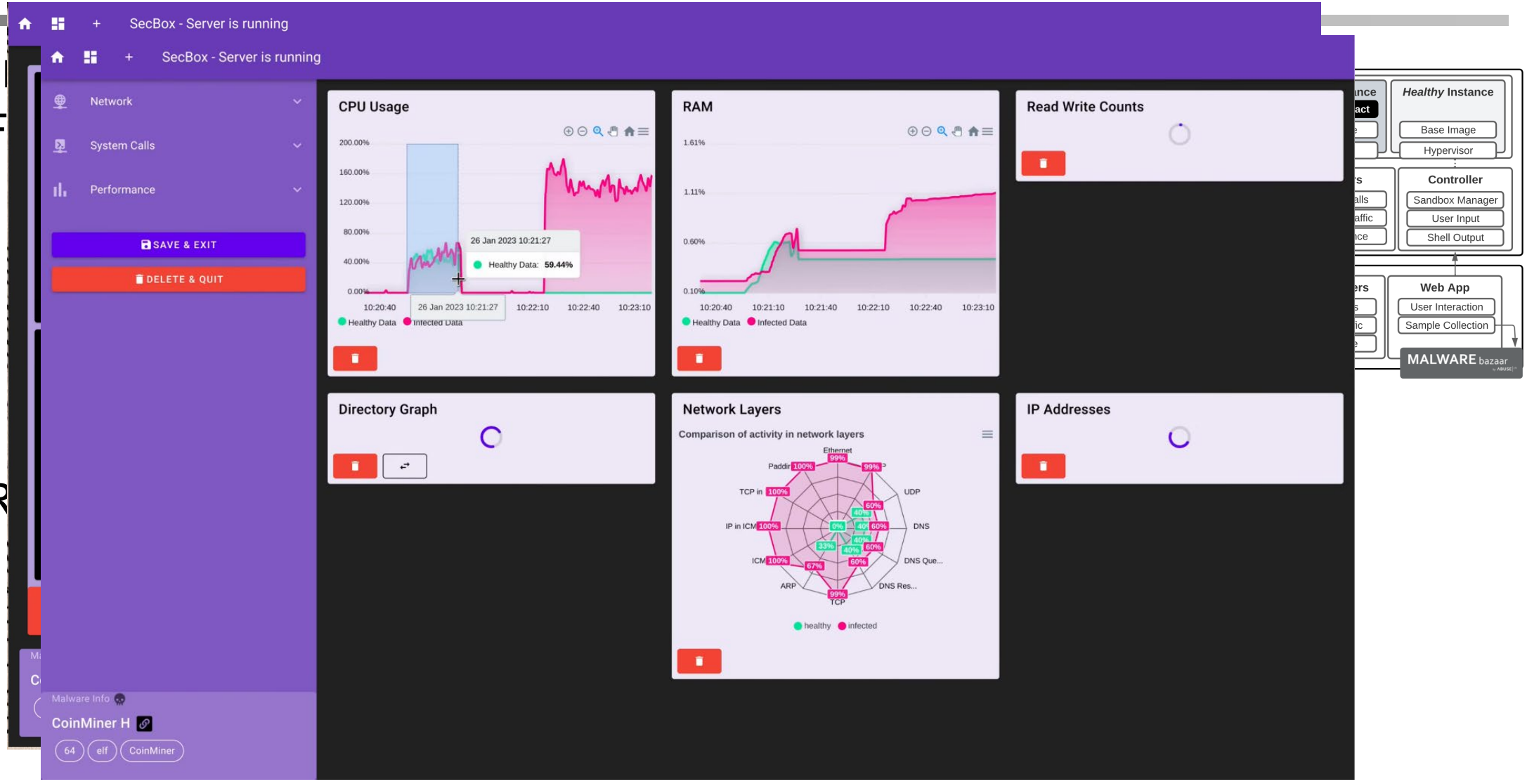
SecBox Platform Architecture (2/3)

- Data Layer
  - Data Managers
    - Persist collected data
    - Data Mining (post-mortem)
    - Real-time Processing (interactive)
      - File Storage, Document DB
  - Web API
    - Regularly fetches malware samples
      - Swiss non-profit malware exchange DB (research, threat intelligence)
    - Serving frontend (realtime data, interactions)



SecBox Platform Architecture (3/3)

- Use
  - F
  - R



Evaluation Overview

```
ch1: secure execution
ch2: light-weight execution
ch3: reproducibility
ch4: effectiveness
```

- Show the functional and qualitative features of SecBox by investigating the following properties
  - **Isolation** capabilities of the sandboxing (ch1)
  - **Low visibility** against malware samples (ch1)
  - **Performance** requirements of the platform (ch2)
  - **Automation** of the process (ch3)
  - **Reproducibility** of the overall phase, leading to consistent datasets (ch3)
  - **Monitorability** of malicious behavior (ch4)

Experiment: Automation & Reproducibility

- Reproducibility: achieve consistent outcomes
  - Updating models based on a newly collected sample
    - Incident response team does initial analysis
    - Post-mortem analysis needed for model refinement
  - Recreating datasets across platforms
    - Recreate a dataset obtained on a RPI3 ona RPI4
  - Observe, persist and replay all parameters
    - Samples, execution parameters, human user input

- Cosine similarity of features
  - Samples are not deterministic
  - High similarity across 10 different runs

Malware	*Reference*	*Mirai*	*Monti*	*Coinminer*
Avg. cos_sim	0.998	0.999	0.985	0.996
Std. cos_sim	0.003	0.001	0.022	0.005

Experiment: Performance

- Host: Ubuntu 22.04, 4vCPU@2.4 GHz, 16 GB memory
- Three scenarios
  - Idle (no sandboxes)
  - Idle sandboxes (no workload)
  - CoinMiner executing in the sandbox
- Sandbox host, backend, frontend
  - Idle platform requires very few resources
  - Sandboxes are almost free
    - Container-based execution
  - Any user-generated load will define the performance requirements
  - Pipeline operates continuously during malware execution

	Sandbox Host		Backend		Frontend	
Scenario	*CPU*	*RAM*	*CPU*	*RAM*	*CPU*	*RAM*
Idle	0.5%	0.03 GB	0.6%	0.09 GB	0.21%	0.27 GB
Passive	0.6%	0.26 GB	3%	0.09 GB	0.3%	0.27 GB
Malware	26.0%	0.48 GB	1.0%	0.19 GB	1.5%	0.32 GB

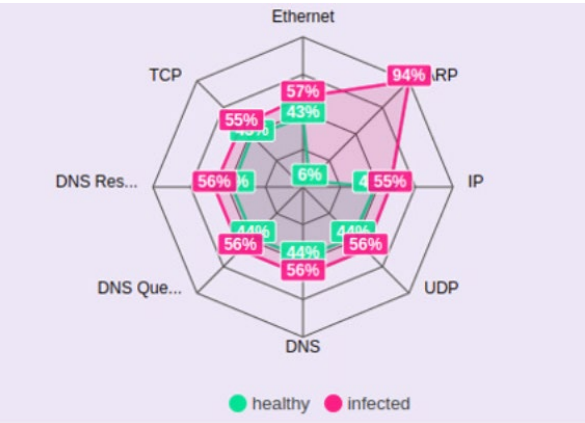
Experiment: Isolation & Visibility

- First container-based malware data extraction platform
  - Hard to formally *prove* security guarantees
- Maldet, signature-based malware detection on host
  - 10’822 signatures, updated daily
- Used during all experiments
  - 100 real samples: botnets, cryptojackers, rootkits, ransomware, backdoors
    - Established and less studied samples (*e.g., Conti* vs. *CoinMiner*)

- No breaches detected
- None of the samples seem to detect the sandboxing
  - Congruent with security research on gVisor

Experiment: Monitorability

- The most important (functional) aspect
  - Goal: Extract as much data as possible
- Performed several experiments on real-world malware, to assess the extraction of useful data
  1. Normal scenarios, e.g., Mirai running passively
  2. "Stress" scenarios: Many sandboxes in parallel
- Lots of visual feedback during and after execution
- Feasible to extract wide range of features
  - Resource consumption vs. system calls
  - Mirai: ARP-scanning
- Datasets are useful to train ADs and classifiers



Summary and Future Work

- Platform for reproducible data mining of dynamic malware execution
  - Secure, lightweight execution
  - Enables reproducible and automated execution
  - Rich data from different sources

- Future Work: What is *relevant* behavior?
  - Simulating other workloads (esp. benign)
    - Compare them to a realistic scenario (*e.g.,* desktop, server)
    - Deploying the host component in real device

Thanks for your attention!
Questions?

Chao Feng, Jan von der Assen, Alberto Huertas Celdrán, Raffael Mogicato, Adrian Zermin, Vichhay Ok, Gérôme Bovet, Burkhard Stiller

Communication Systems Group CSG, Department of Informatics Ifl
University of Zürich UZH, vonderassen@ifi.uzh.ch

Universität Zürich[UZH]

CSG Communication Systems Group

ifi

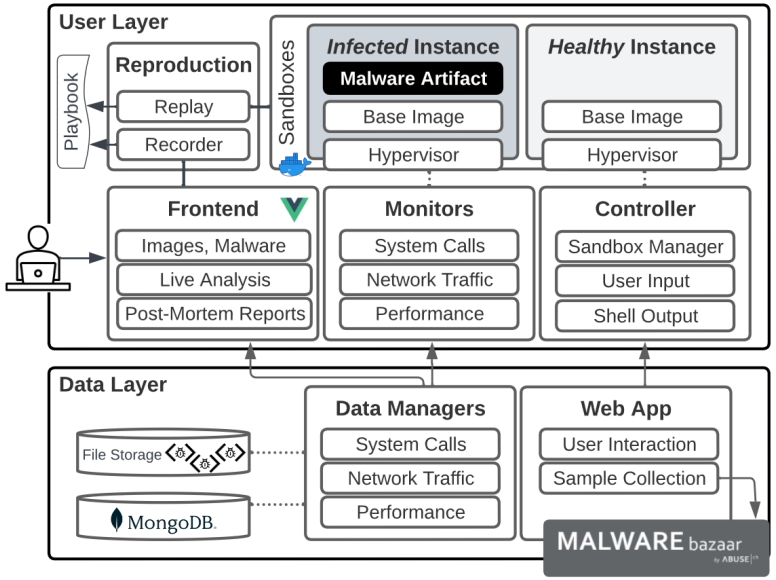
Related Work

- Literature Review
  - Academic Sources
  - Open-source Solutions

Solution	*Year*	*Platform*	*Isolation*	*Maintained*	*Real-time*	*Data Reproducibility*
TTAnalyze [5]	2006	Windows	Emulation	Unknown	No	No
CWSandbox [6]	2007	Windows	VM, API hook	No	No	No
Ether [7]	2008	Windows	VM	No	No	No
DRAKVUF [4]	2021	Windows, Linux	VM	Yes	Yes	Only Collection
Cuckoo [8]	2019	Windows, Linux, macOS	VM	No	No	Python-based
ANY.RUN [9]	2023	Windows	VM	Yes	Yes	Only Basic Presets
LiSa [10]	2019	Linux	QEMU	No	No	via RESTful API
Tamer [11]	2022	Linux/ARM	QEMU	Yes	No	Automated Interaction
V-Sandbox [12]	2020	Linux	QEMU	No	No	ELF configurator, C&C simulator
SecBox	2024	Linux	gVisor Container	Yes	Yes	Interaction, Artifacts, Exec. Environment

Implementation Aspects

- 3 independent modules
  - Sandboxes + backend + frontend
  - Run virtually or on real devices



- Technical Details
  - Modules interconnected by WebSockets
  - Python-based web API and client
  - Docker runtime + gVisor host
  - Vue.js frontend
  - Block storage + MongoDB

Future Work

- Directions for future research
  - Analyze isolation capabilities from an offensive perspective
    - Difficulty of proving security guarantees
    - Exploratory approach
  - Simulating other workloads (esp. benign)
    - Compare them to a realistic scenario (*e.g.,* desktop, server)
    - Deploying the host component in real device
  - Demonstrating usefulness of collected data
    - Training Anomaly Detection
    - Training Classifiers