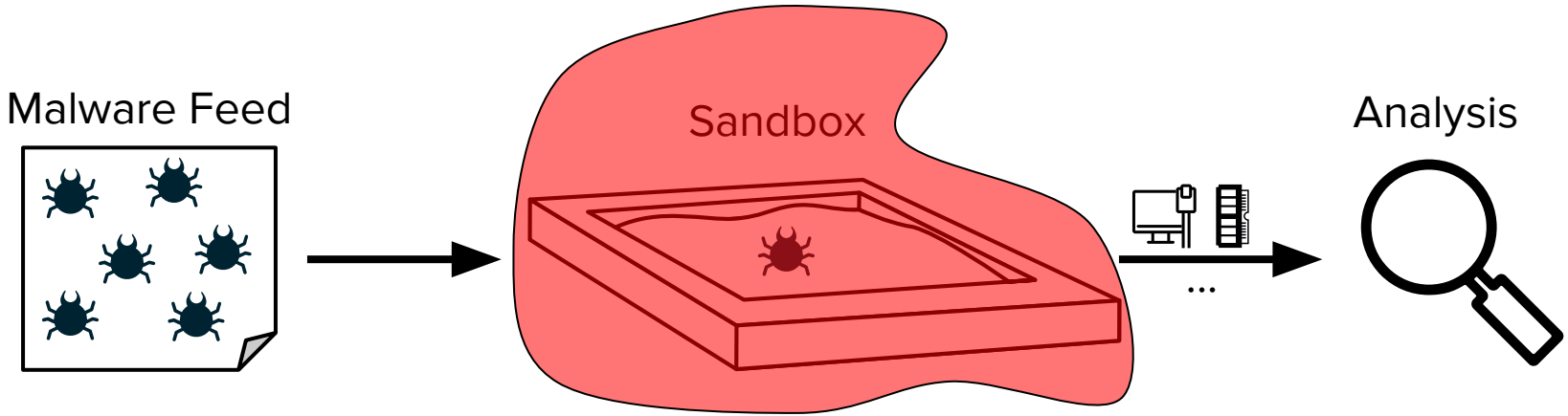
Evading Malware Sandboxes

Michael Brengel
mbrengel@mmci.uni-saarland.de
CISPA, Saarland University, Germany

Motivation



Two approaches:

1. Detect virtualization
2. Detect sandbox-inherent characteristics

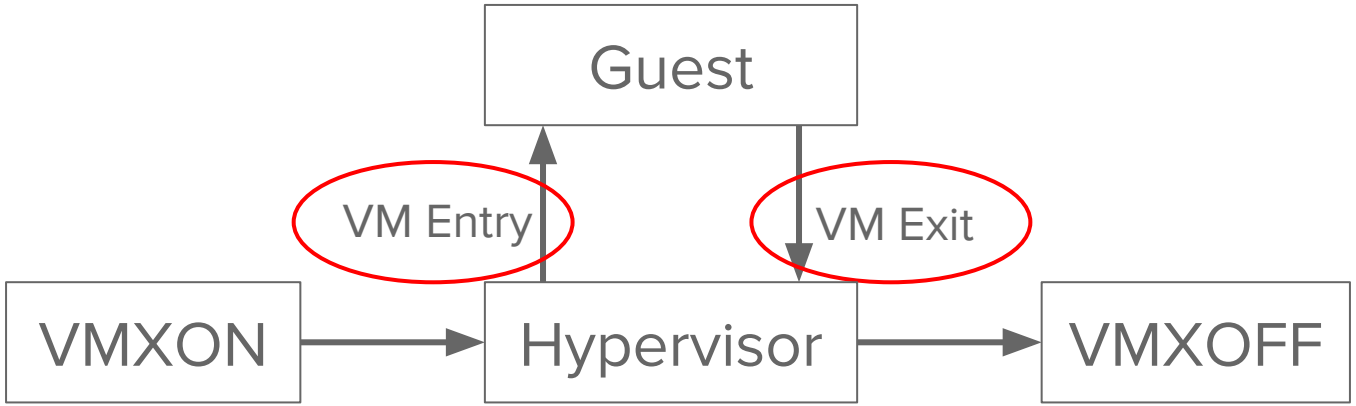
Virtualization Types

1. Software Emulators
   Emulate the machine completely in software
   ⇒ QEMU, BOCHS, ...

2. Reduced Privilege Guests
   Execute guest at a lower privilege level + handle interrupts
   ⇒ VMWare, VirtualBox, ... (if your CPU is really old)

3. Hardware-Assisted Virtualization
   Virtualization implemented on the CPU
   ⇒ Intel VT-x, AMD-V

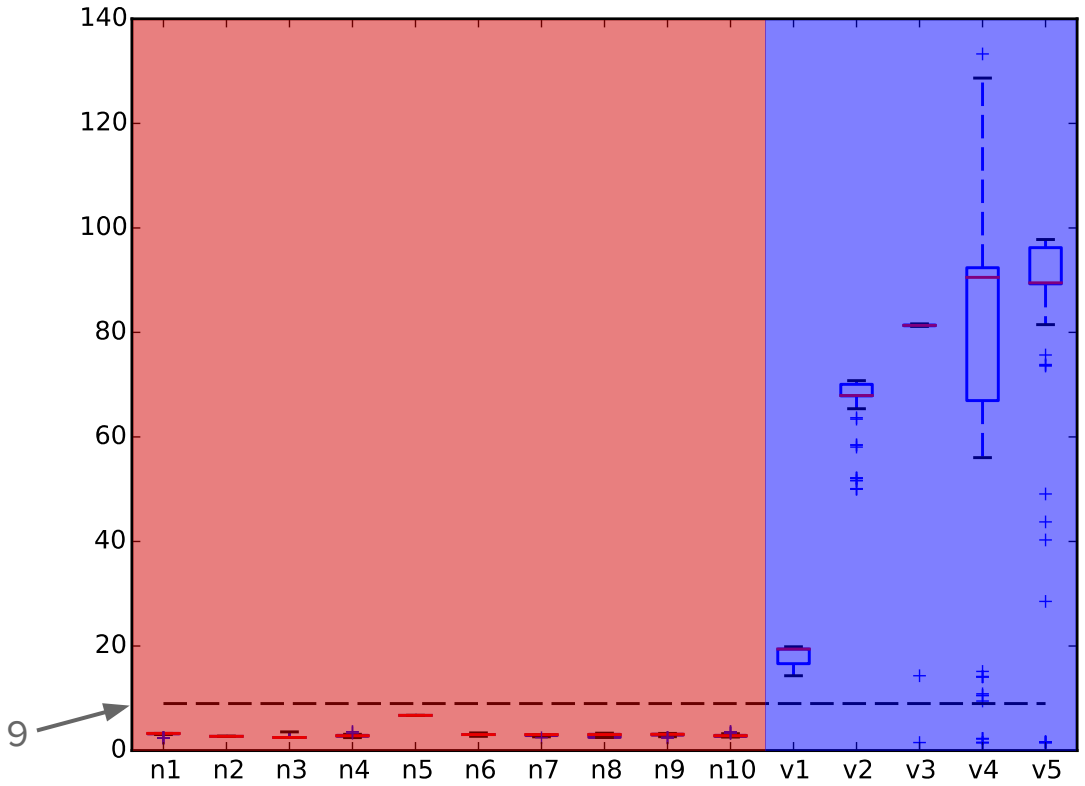
Intel VT-x



- A VM Exit is triggered whenever a sensitive instruction is executed
- We want to force a VM Exit: which instruction to use?
  ⇒ `cpuid`
  - Not privileged
  - Always triggers a VM Exit

Timing Attack

$$\frac{\text{time}(\texttt{cpuid}_{\text{native}})}{\text{time}(\text{nop}_{\text{native}})} \leq \frac{\text{time}(\texttt{cpuid}_{\text{vt}})}{\text{time}(\text{nop}_{\text{vt}})}$$

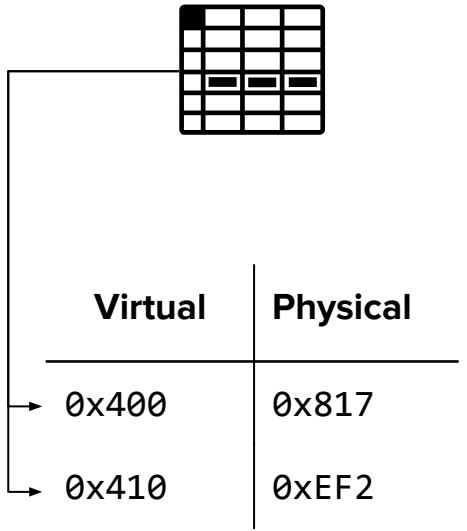


Note: We measure time with `rdtsc`

Translation Lookaside Buffer

TLB = A cache for page walks

```
mov eax, [0x400030]  <-- TLB Miss -> Page Walk  (0x817030)
mov ebx, [0x410044]  <-- TLB Miss -> Page Walk  (0xEF2044)
...
mov eax, [0x400F20]  <-- TLB Hit (0x817F20)
mov ebx, [0x410044]  <-- TLB Hit (0xEF2044)
```



TLB – Context Switches

0x40 → 0x87

P_1

```
mov eax, [0x40000]
cmp eax, ebx
je ...
```

0x40 → 0xDC

P_2

```
mov eax, [0x40000]
add ebx, eax
...
```

TLB Flush

Virtual	Physical
0x40	0xDC

⇒ Observation: VM Exit = Context Switch

TLB Attack

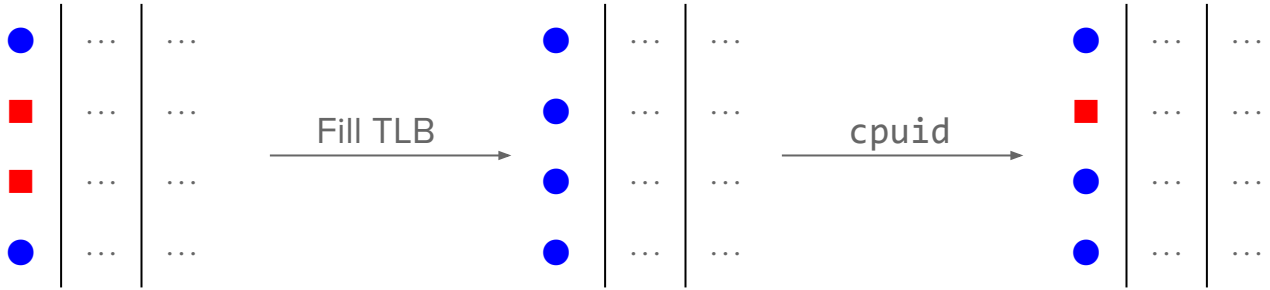
- General Idea:
  - Access a page
  - Execute cpuid
  - Access the page again and check how long it takes

    ⇒ Long access time ⇒ TLB Flush ⇒ Hypervisor present
- Problem: VPID

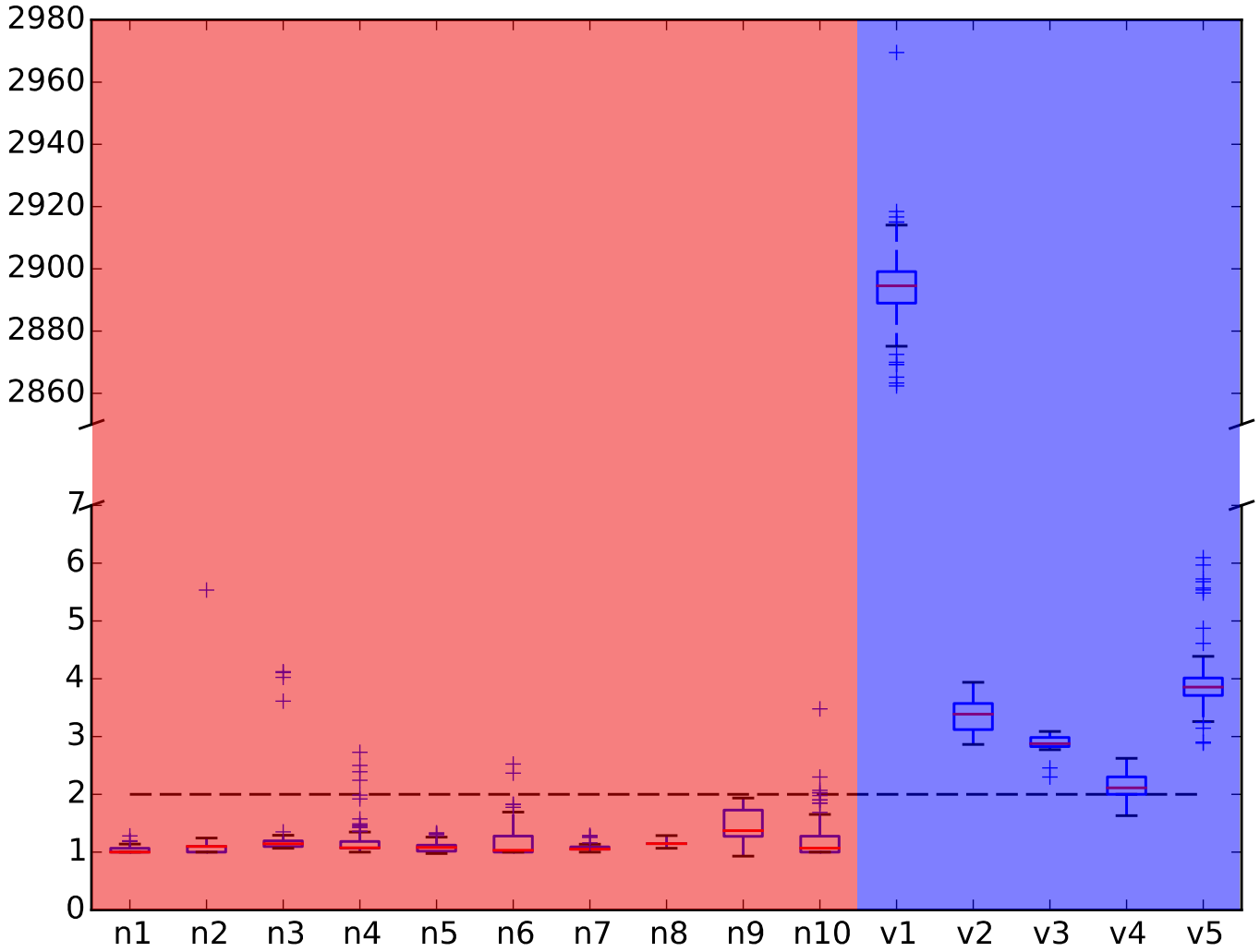
Tag	Virtual	Physical
🔵	0x40000	0x7F000
🟥	0x40000	0x86000
🟥	0x70000	0x6C000
🔵	0x70000	0x94000

⇒ No TLB Flush necessary anymore

TLB Attack (2)



- Rationale: Hypervisor evicts at least 1 TLB-Entry
- Caveat: Highly dependent on the TLB size
  ⇒ Execute multiple times with different TLB sizes
- We compute r = time(`pageaccess`$_{\text{after\_cpuid}}$) / time(`pageaccess`)


2980
2960
2940
2920
2900
2880
2860
7
6
5
4
3
2
1
0
n1
n2
n3
n4
n5
n6
n7
n8
n9
n10
v1
v2
v3
v4
v5

Sanity Checks

To account for obvious countermeasures:

1. time(`rdtsc`) $<$ 500 cycles
2. time(`cpuid`) $>$ 20 cycles
3. time(`nop`) $<$ 500 cycles

To improve the TLB attack:

4. time($\texttt{pageaccess}_{\text{after\_cpuid}}$) - time(`pageaccess`) $\leq$ 150 cycles

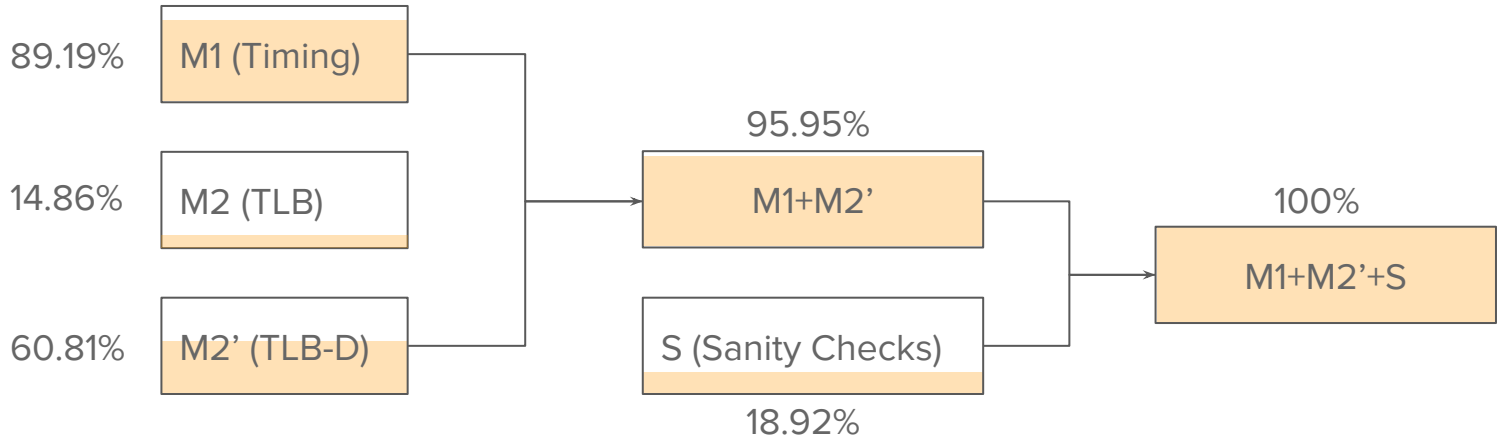
Evaluation: Planetlab

We ran experiments on 239 PlanetLab nodes.
233 native nodes, 6 virtualized nodes

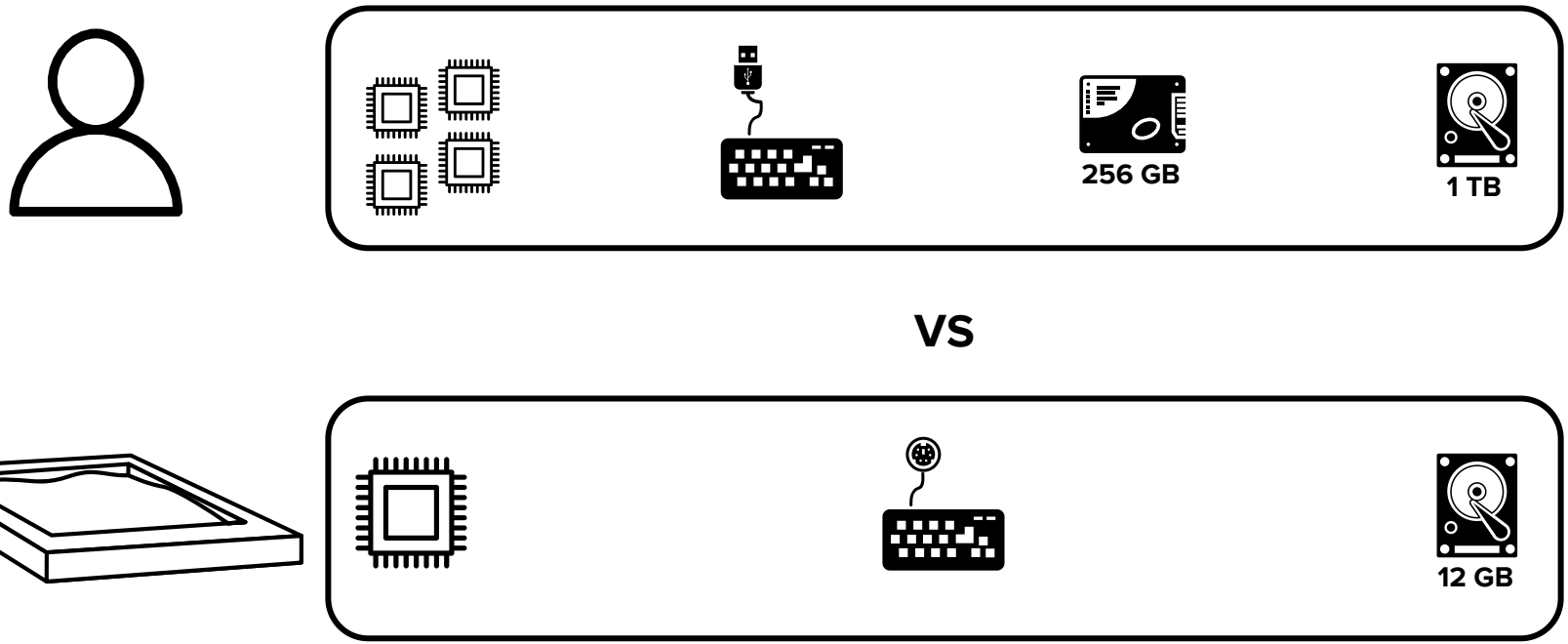
- Timing Attack (M1)
  - True Negative Rate: 99.99%
  - True Positive Rate: 100%
- TLB Attack (M2)
  - True Negative Rate: 99.96%
  - True Positive Rate: 100%

Evaluation: Sandboxes

- Sample submitted to malware services (VirusTotal, Threatexpert, ...)
- Results from 30 sandboxes



Fingerprinting Sandboxes – Motivation



Collecting Data

- SandPrint: a tool to extract characteristics from sandboxes
- Collects more than **40** features
- Submitted to **20** analysis services
- Reports sent to our server
- **2,666** reports
- Clustered into **76** sandboxes
- Also run on **50** user machines

Classifier – Feature Selection

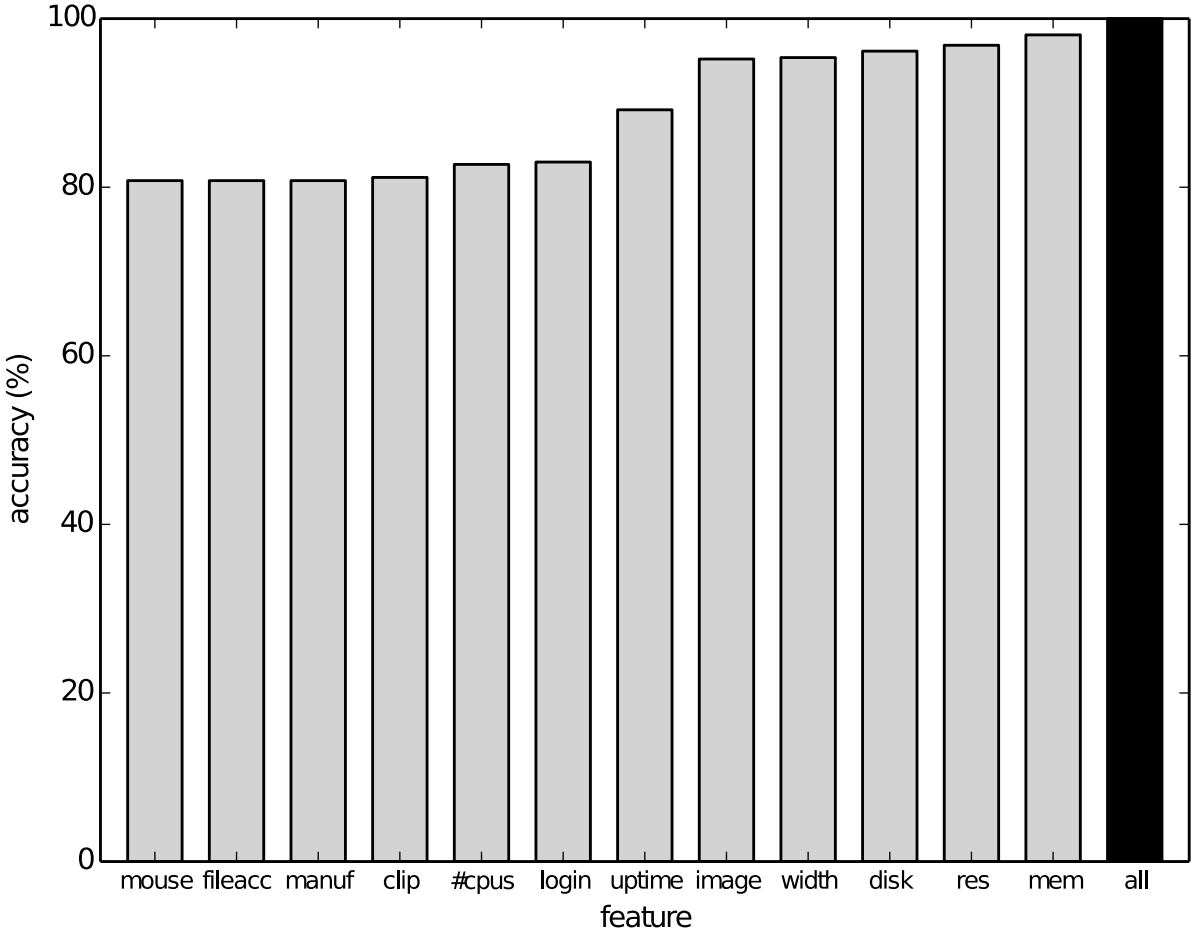
	Display Resolution
	Display Width
	RAM size
	PS/2 mouse
	#CPU cores
	Disk size

	Image Name
	Clipboard
	System Manufacturer

	System Uptime
	Last Login
	Last File access

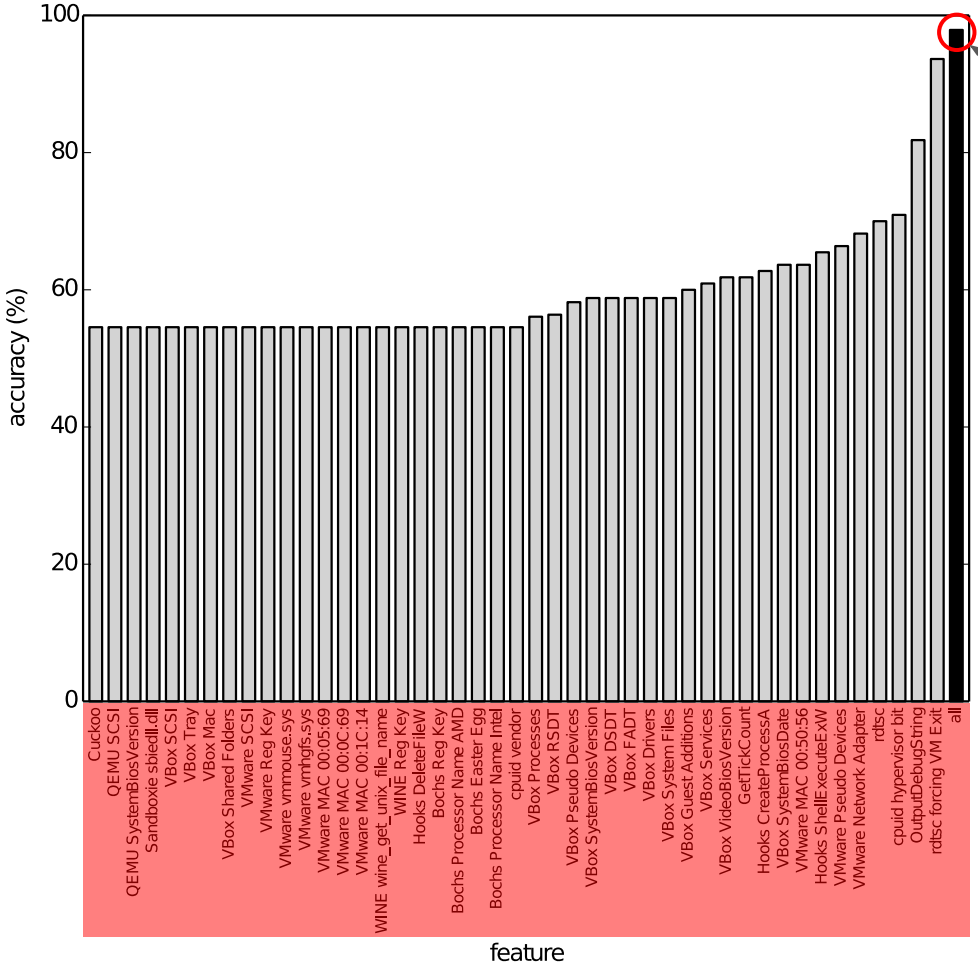
Classifier – Support Vector Machine (SVM)

- 202 reports
  - 50 user reports
  - 152 sandbox reports (sampled up to 3 per sandbox)
- Hyperparameter Optimization with 10-fold cross validation
- Classifier for each individual feature
- Classifier for all features combined


100
80
60
40
20
0
accuracy (%)
mouse
fileacc
manuf
clip
#cpus
login
uptime
image
width
disk
res
mem
all
feature

Comparison With Pafish

- “Pafish is a demonstration tool that employs several techniques to detect sandboxes and analysis environments in the same way as malware families do.” (https://github.com/a0rtega/pafish)
- Mainly virtualization detection
- Encoded **45** Pafish features in Sandprint



Virtualization

Stealthiness

- Collecting the features might raise suspicion
- Executed SandPrint on 3 popular appliances
  ⇒ 40 reports
- A lot of features (i.a. Pafish) raise a security alert
- Restriction to **11 non-alerting** features
- 100% accuracy

Summary

- In Practice, sandboxes can still be detected
- Virtualization Detection
  - Timing Attack
  - Cache (TLB) Attack
- Sandbox-inherent features 
  - Hard/Unpleasant to mitigate
  - Stealthy (security appliances)

Our Papers

1. **Michael Brengel**, Michael Backes and Christian Rossow, “Detecting Hardware-Assisted Virtualization”, in *Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2016

2. Akira Yokoyama, Kou Ishii, Rui Tanabe, Yinmin Papa, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Daisuke Inoue, **Michael Brengel**, Michael Backes and Christian Rossow, ”SANDPRINT: Fingerprinting Malware Sandboxes to Provide Intelligence for Sandbox Evasion”, in *Proceedings of the International Conference on Recent Advances in Intrusion Detection (RAID)*, 2016