RESEARCH AND AUTOMATION

In a Modern Security Company

Mariano Graziano

SILM 2019, Inria Rennes

$whoami

@emd3l

Italian, Hackademic, Malware, Memory forensics,
Cisco Talos, Eurecom (in random order)

TALOS

Cisco Security Research

Malware Research Team

- Malware analysis
  - Quick analysis (extraction of indicators, coverage)
  - In-depth reversing (manual)
- Automation
  - Signature generation (Bass)
  - Automated analysis tools (FIRST, Pyrebox, ROPMEMU)
  - Clustering

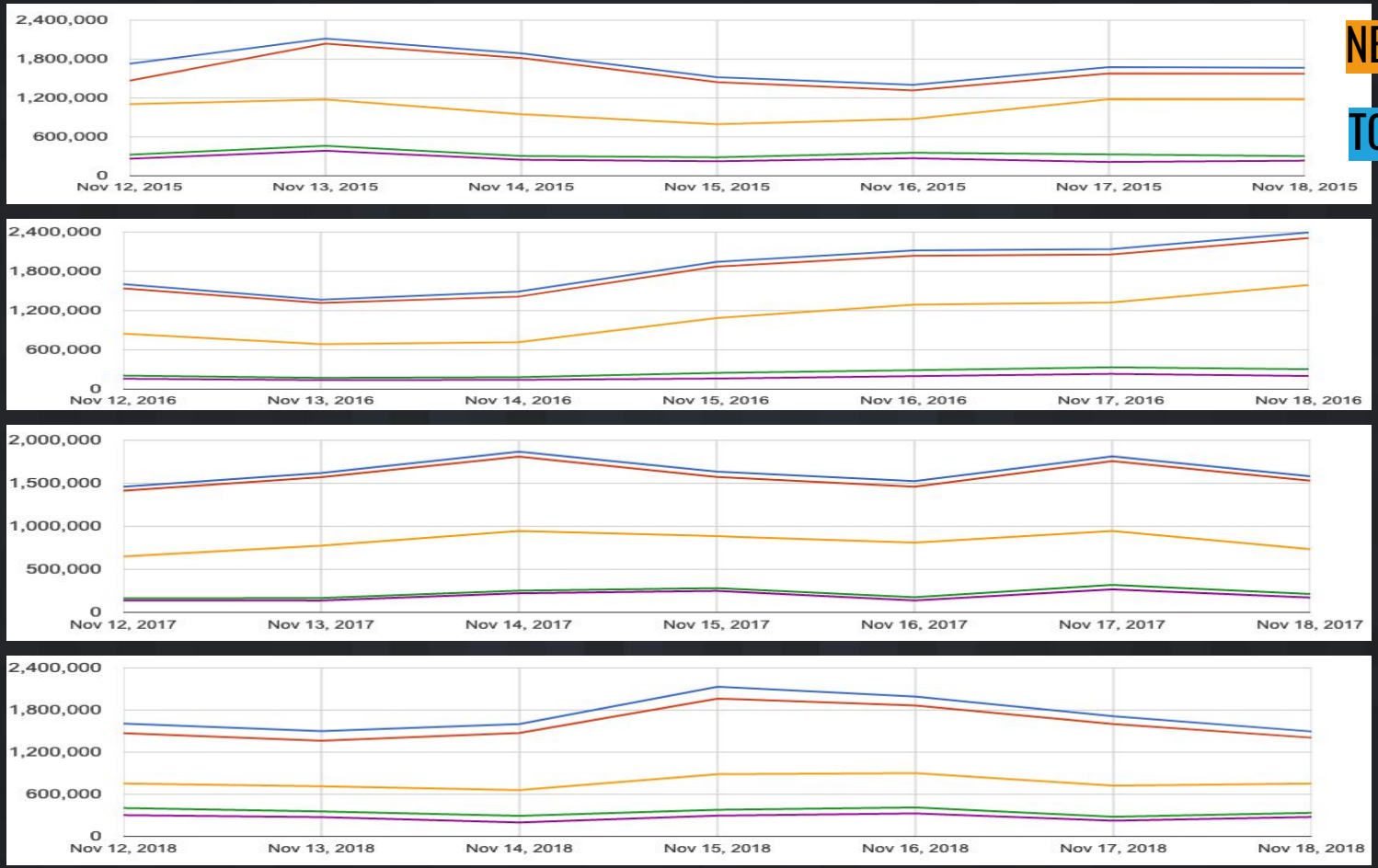
Malware **Research** Team

- Academic publications
  - 6 papers in 3.5 years
  - From future threats to open problems for the company
- Blogposts
  - Technical analyses of new malware families
  - Long-term investigations
- Industrial talks
  - Present new tools

Research on Automation

- Large-scale studies:
  - A Close Look at a Daily Dataset of Malware Samples (TOPS)
  - Understanding Linux Malware (S&P)
- Manual analysis:
  - FIRST
  - Pyrebox
  - IDA/Ghidra server

VirusTotal



NEW FILES

TOTAL FILES

CATCH OF THE DAY

Clarification

- This presentation describes an academic paper developed in collaboration with Eurecom (France) [1]
- This research was started on the beginning of 2016
- Queries and sample processing were spread through several months by borrowing internal company resources

The dataset and our results should be representative and hold also after 3 years

[1] http://s3.eurecom.fr/docs/tops19_dailymalware.pdf

Catch of the day

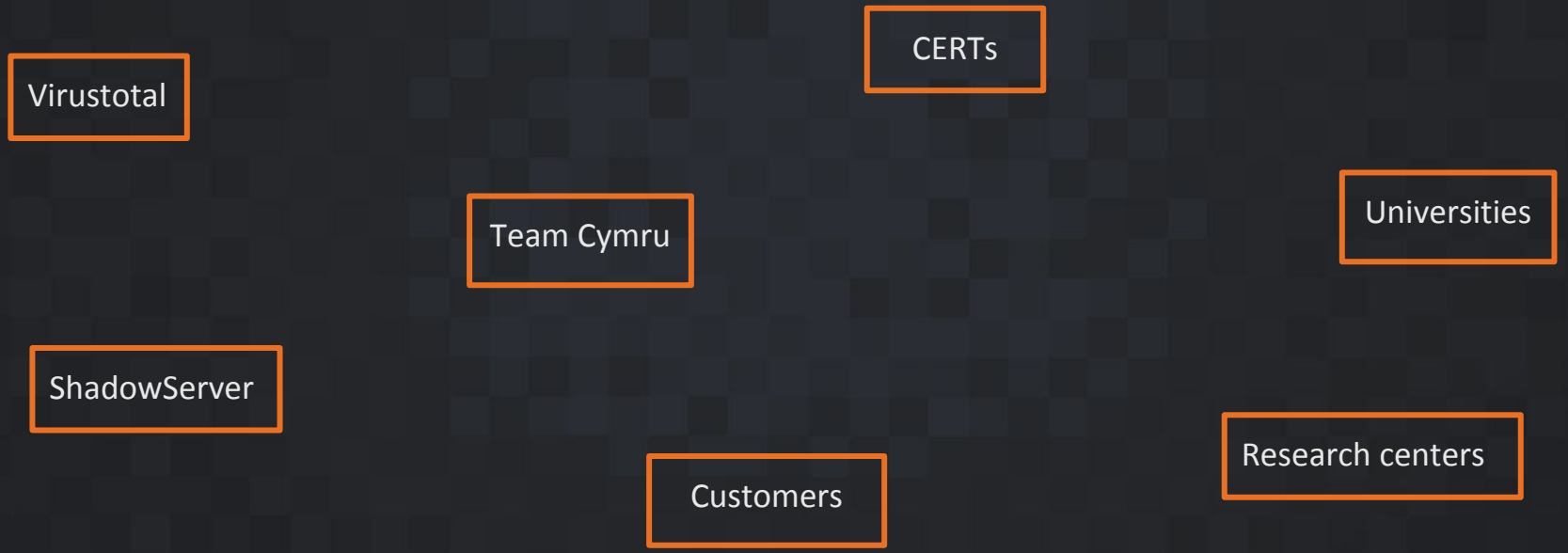
Everyday security companies collect millions of samples

Catch of the day

Everyday security companies collect millions of samples

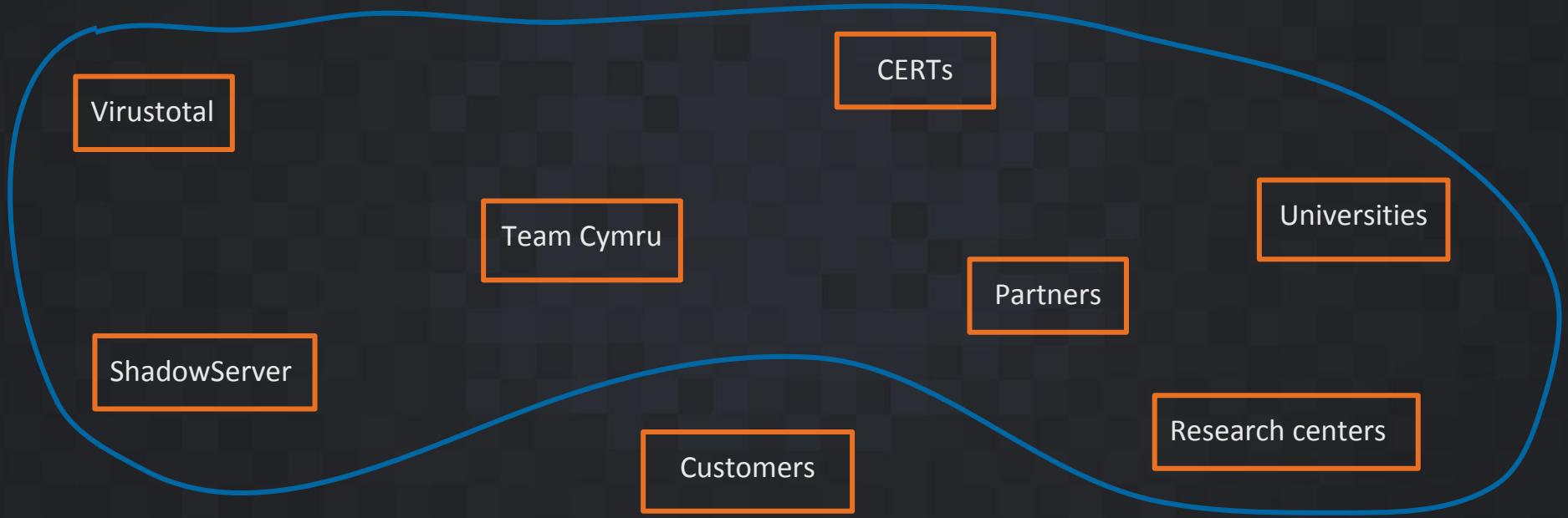
Catch of the day

Everyday security companies collect millions of samples



Catch of the day

Everyday security companies collect millions of samples



Catch of the day

Everyday security companies collect millions of samples

17 different feeds

Open questions

Open questions

- What the dataset contains?
- How many samples belong to known families?
- How much effort to analyze the remaining samples?
- How effective are the state-of-the-art techniques?

but most importantly:

- How much effort would it take?
- How many people? How many VMs? Cores?
- How many resources are wasted?
- What are the challenges?

Find a good day

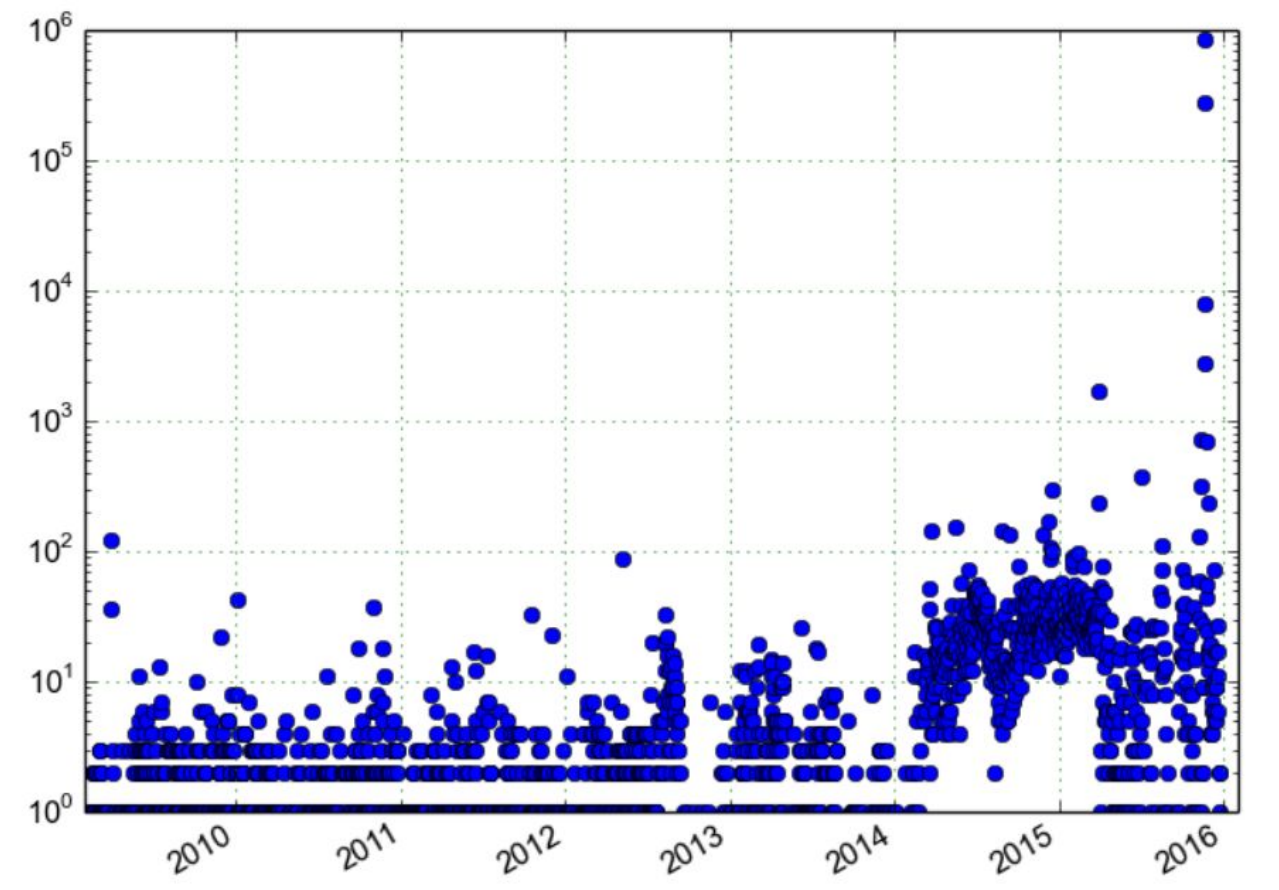
Find a good day

Day: Wednesday, November 18 2015
Number of samples: 1,261,882

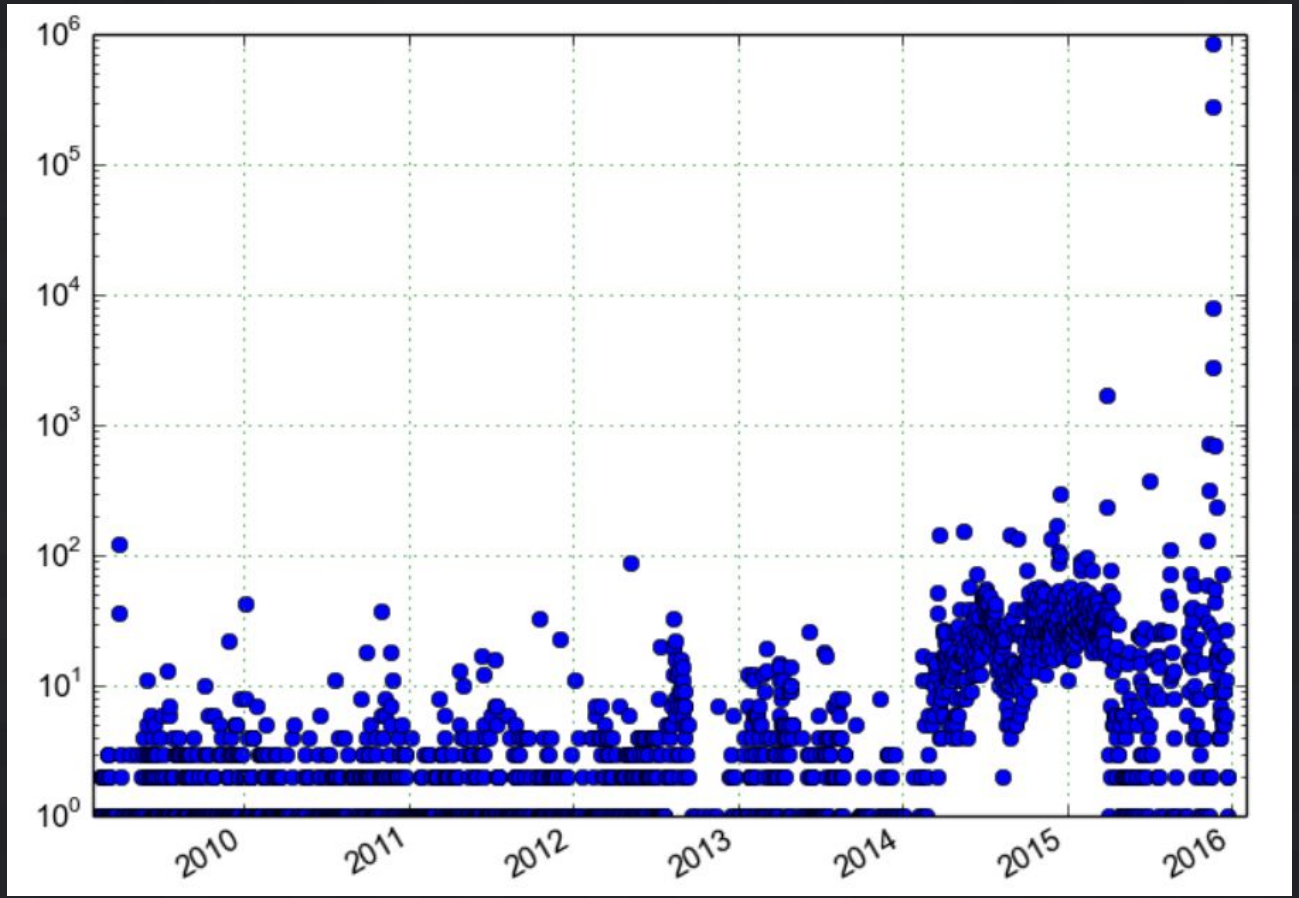
Worst case:
the highest possible number of samples

First look in VT

First look in VT

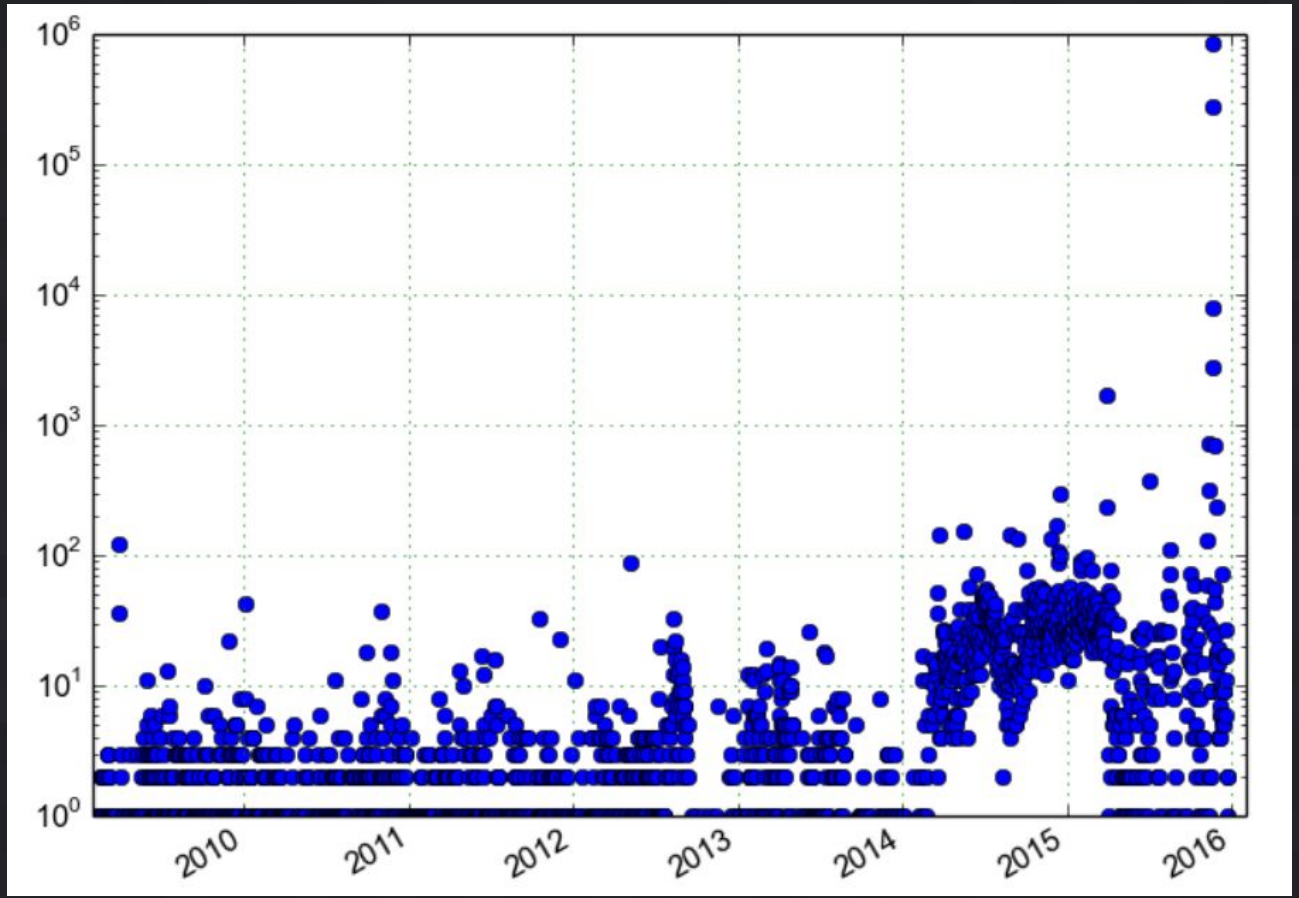


First look in VT



90% in VT

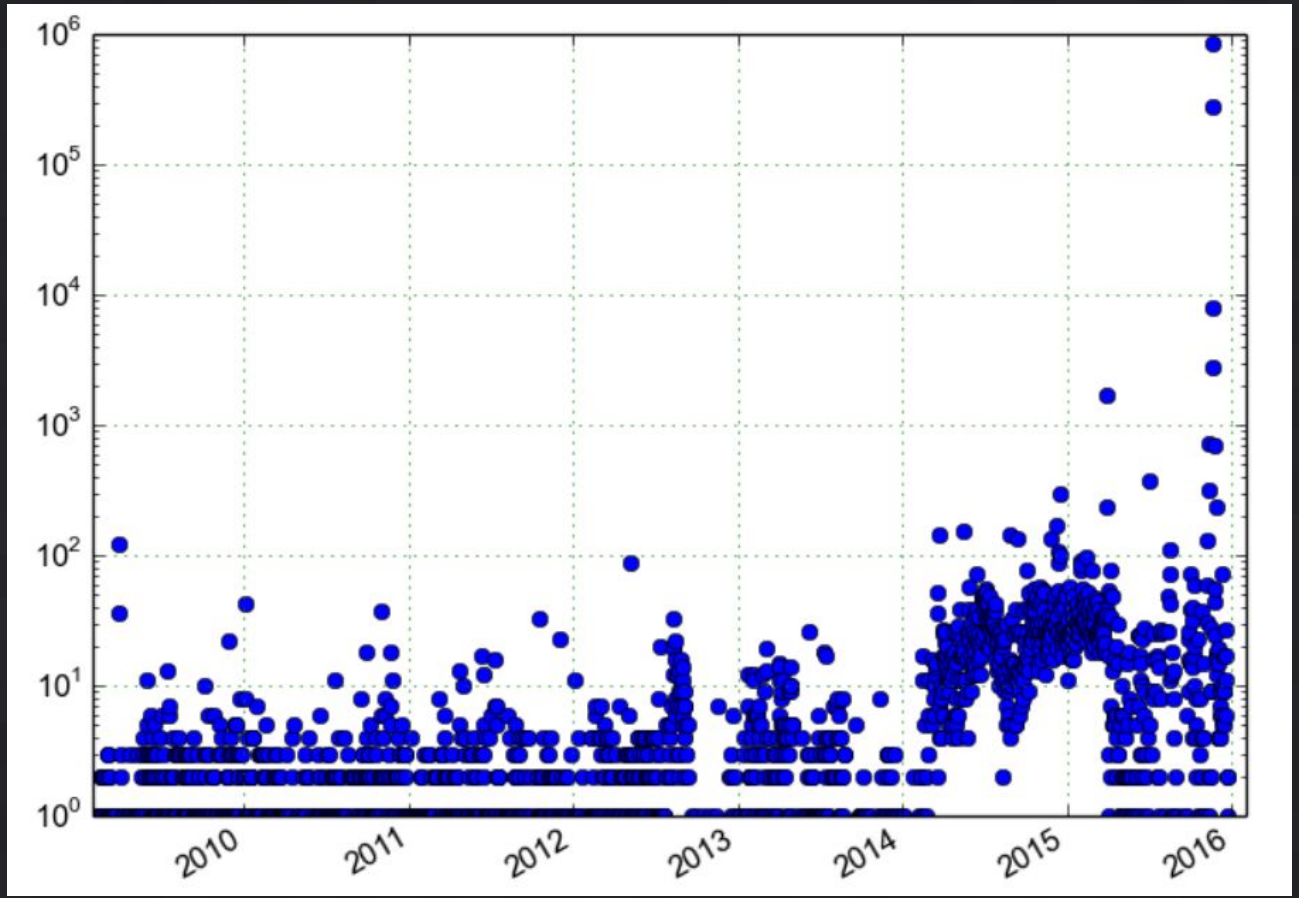
First look in VT



90% in VT

89% same/before day

First look in VT



90% in VT

89% same/before day

1,6% known before

Dataset

Dataset

160k Win32 EXE

440k Win32 DLLs

55k Android

4,3k Mach-O

4,5k ELF

Dataset

160k Win32 EXE

440k Win32 DLLs

55k Android

2,5k OOXML Word

4,3k Mach-O

47k PDF

4,5k ELF

5k MS Excel

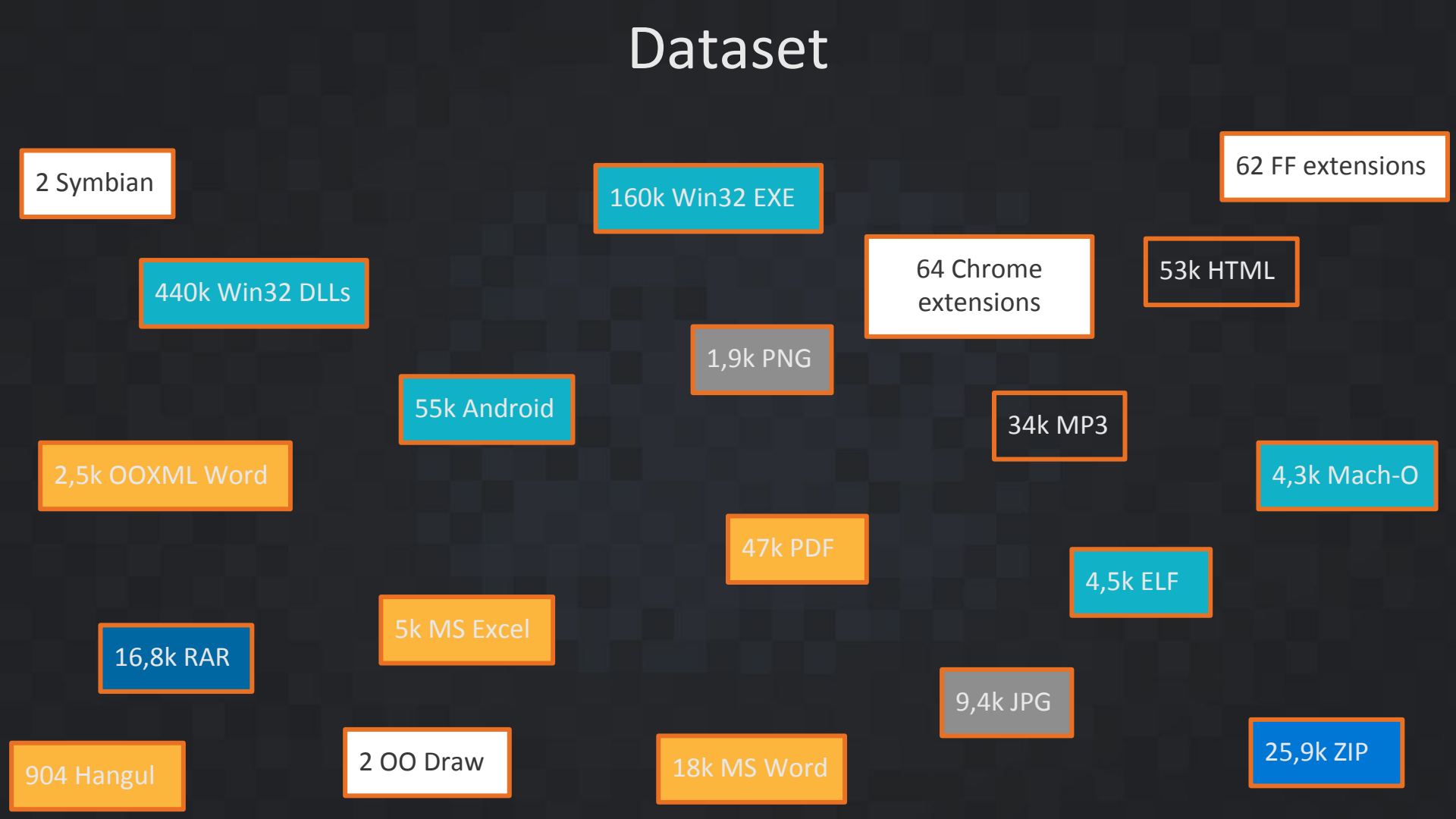
904 Hangul

18k MS Word

Dataset


160k Win32 EXE
53k HTML
440k Win32 DLLs
1,9k PNG
55k Android
34k MP3
2,5k OOXML Word
4,3k Mach-O
47k PDF
4,5k ELF
5k MS Excel
16,8k RAR
9,4k JPG
25,9k ZIP
904 Hangul
18k MS Word

Dataset


2 Symbian
160k Win32 EXE
62 FF extensions
440k Win32 DLLs
64 Chrome extensions
53k HTML
1,9k PNG
55k Android
34k MP3
2,5k OOXML Word
4,3k Mach-O
47k PDF
4,5k ELF
5k MS Excel
16,8k RAR
9,4k JPG
25,9k ZIP
904 Hangul
2 OO Draw
18k MS Word

Windows Executables

Windows Executables

Subsystem	DLLs	Executables
WINDOWS_GUI	66.327	162.327
EFI_BOOT_SERVICE_DRIVER	214.887	21.201
WINDOWS_CUI	139.246	10.285
EFI_RUNTIME_DRIVER	24.435	3215
NATIVE	92	888
EFI_APPLICATION	781	400
WINDOWS_CE_GUI	113	59
UNKNOWN	28	36
EFI_ROM	17	0
XBOX	3	0
Total	445.929	198.411

Windows Executables

Subsystem	DLLs	Executables
WINDOWS_GUI	66.327	162.327
EFI_BOOT_SERVICE_DRIVER	214.887	21.201
WINDOWS_CUI	139.246	10.285
EFI_RUNTIME_DRIVER	24.435	3215
NATIVE	92	888
EFI_APPLICATION	781	400
WINDOWS_CE_GUI	113	59
UNKNOWN	28	36
EFI_ROM	17	0
XBOX	3	0
Total	445.929	198.411

Windows Executables

Subsystem	DLLs	Executables
WINDOWS_GUI	66.327	162.327
EFI_BOOT_SERVICE_DRIVER	214.887	21.201
WINDOWS_CUI	139.246	10.285
EFI_RUNTIME_DRIVER	24.435	3215
NATIVE		38
EFI_APPLICATION	7	00
WINDOWS_CE_GUI	1	59
UNKNOWN	28	36
EFI_ROM	17	0
XBOX	3	0
Total	445.929	198.411

172,612 executables
subsystem 2 and
subsystem 3
13,7% of the dataset

Windows executables

- 60% of the samples have a size between 100K and 1M
- 98% x86_32, 1,8% x86_64, 0,01% ARM
- 51% of the samples with an entropy higher than 7
- 18,3% binaries are signed (11 with revoked certs)

172k samples are still too many

Sample ingestion pipeline

172k samples are still too many

We design a possible pipeline to process the samples

This pipeline is an instrument:

- Understand the distribution of samples
- Understand the challenges for a company
- Estimate the cost (computational and human)

Sample ingestion pipeline

Pipeline leverages de-facto malware analysis techniques

static analysis
dynamic analysis
manual inspection

Sample ingestion pipeline

VirusTotal

How much can we trust these AVs?

- Time of last scan vs current detection
- AV configuration parameters might be different
- Different types of engines (some are ML, heuristic…)
- FP prone AVs?
- Inaccurate / generic labels

Sample ingestion pipeline

AV results after one year:

- 4,684 samples from 0 positives to 1+
- 2,281 from 1+ positives to 0
- A few samples removed from VT

3.5% of samples changed their disposition

Sample ingestion pipeline

AVClass[2] (state of the art for AV label aggregation)

69% of the samples classified into 1,057 families

allaple	54,097
virut	16,328
browsefox	7,400
outbrowse	4,600
installcore	2,395

49%

[2] https://github.com/malicialab/avclass

Sample ingestion pipeline

Dynamic analysis

- Extract additional information
- We leveraged a state of the art set up
- Internal to the company, we borrowed processing time
- Tuned and maintained: detonation, disarm anti-analysis, etc…

Sample ingestion pipeline

Dynamic analysis

- Part of the samples showed low / no activity
  - We ran those on a second sandbox

Sample ingestion pipeline

A stunning 19% of the samples did not show a meaningful activity

Table 7. Classification of Samples with No/Low Activity

	No activity	Low activity
GUI	599	270
Missing DLLs	3,814	599
Crash	0	723
Corrupted file	9,805	64
Total	14,218	1,656
Still Unexplained	10,159	6,499

Sample ingestion pipeline

This takes (in one single day)

- 17 GiB of space
- 55 VMs (5 minute per sample)

dedicated to samples that have a GUI, crash, missing dependencies, or are corrupted

Manual analysis experiments

How much manual analysis effort needed?

3 different experiments

- High priority group
- Samples with low / no activity
- 64 bit binaries

These groups sum up to 24k binaries

Sampled files from each of those groups

Manual analysis experiments

Experiment configuration:

- Analysts with 2 to 6 years of experience
- Asked these questions:
  - GW/MW?
  - Class (keylogger, RAT, botnet) and family?
  - How much time did it take?
  - Which approach did you use?
    - Blackbox
    - Manual
  - Would you need a deeper manual analysis?

Manual analysis experiments

High priority group

- Extracted several samples per cluster and singleton files
  - 52% / 43.2% labelled malicious (5% margin of error)
  - ~3% / ~5% required manual analysis
  - Malware type and family, 5% better for clustered samples vs singleton samples.

- Cross-checked verdicts for clusters
  - 86% verdicts were consistent

Manual analysis experiments

64bit files (2,603 samples)

- 82% have 0 positives
  - From 101 selected files only 11 should require further inspection.
- For the rest
  - 67% considered benign

Manual analysis experiments

- Estimation: ~27k samples either require interaction, crashed, corrupted, missing dependencies
  - 100 VMs per day if ran on a sandbox

- Between 30 sec and 90 min to inspect the info / samples
  - Estimation: 900 hours to take a cursory look at the 24k unknown samples.

Takeaways

Takeaways

1. Complete analysis: 600 machines (5 min/sample)
2. Community info: only 3.5% of changed verdicts
3. Automated pipeline reclassified 16% of samples
4. Manual inspection of remaining 15% would take >100 person-days

Takeaways

6. But only 5% of samples marked as requiring additional manual inspection

   Substitute decision process by ML?

7. Up to 16% of resources consumed by samples that do not run properly

More info

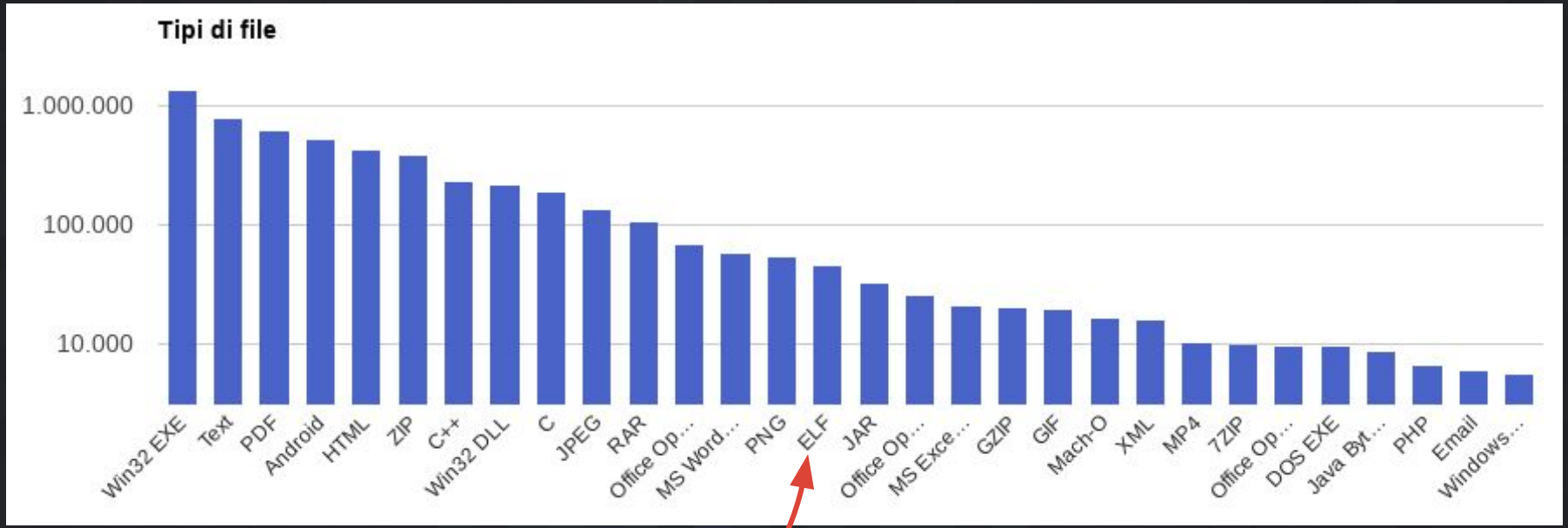
Link to the paper:

"A Close Look at a Daily Dataset of Malware Samples"
ACM Transactions on Privacy and Security

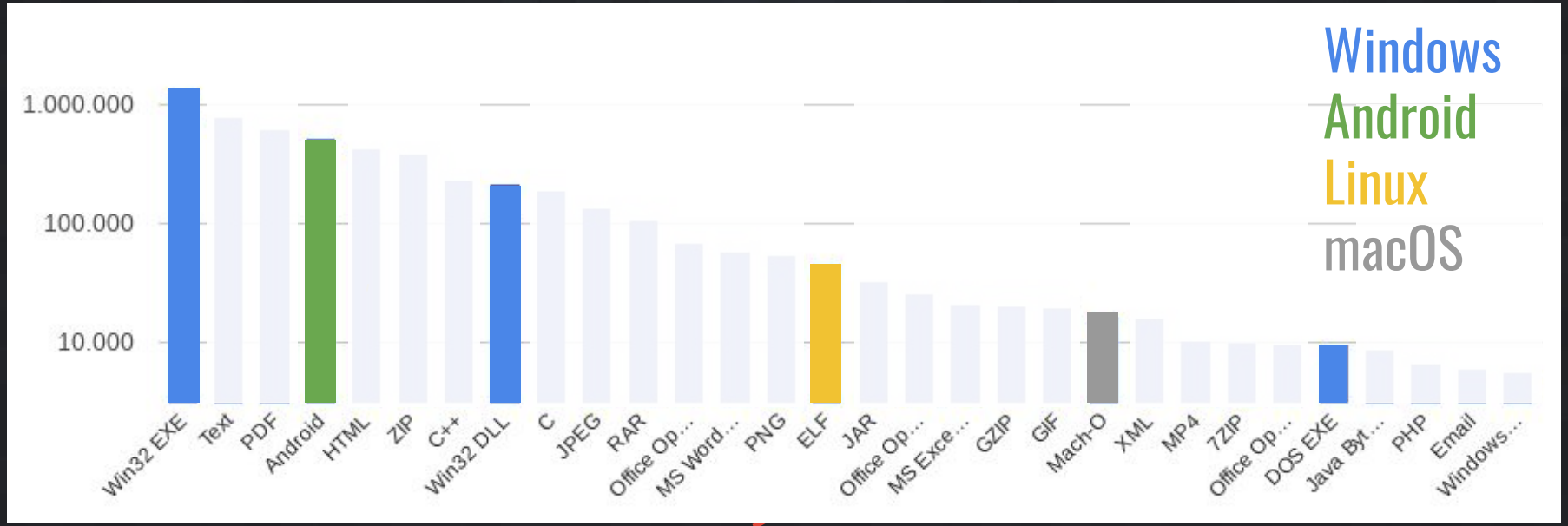
http://s3.eurecom.fr/docs/tops19_dailymalware.pdf

LINUX MALWARE

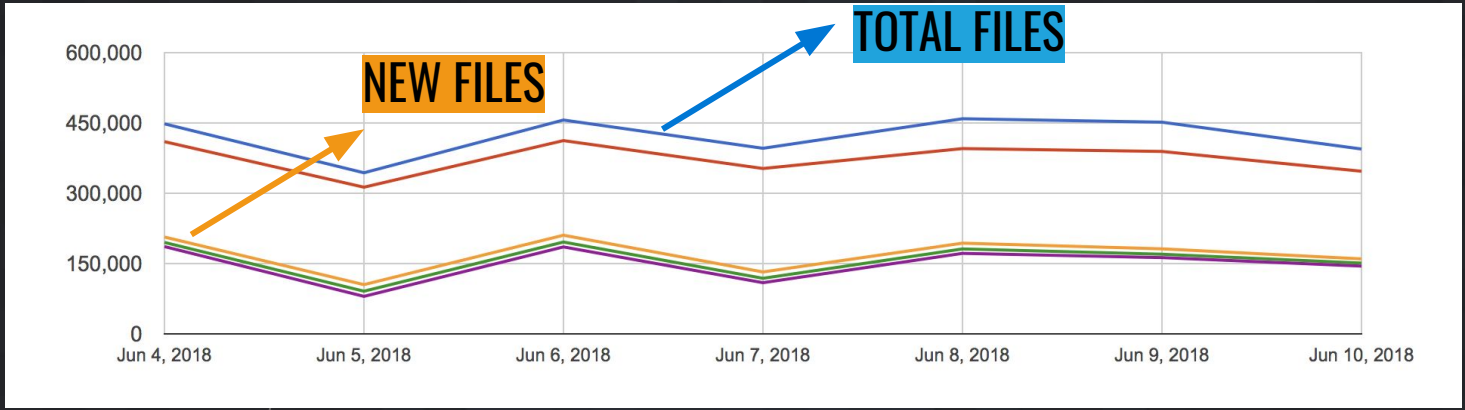
Linux Malware



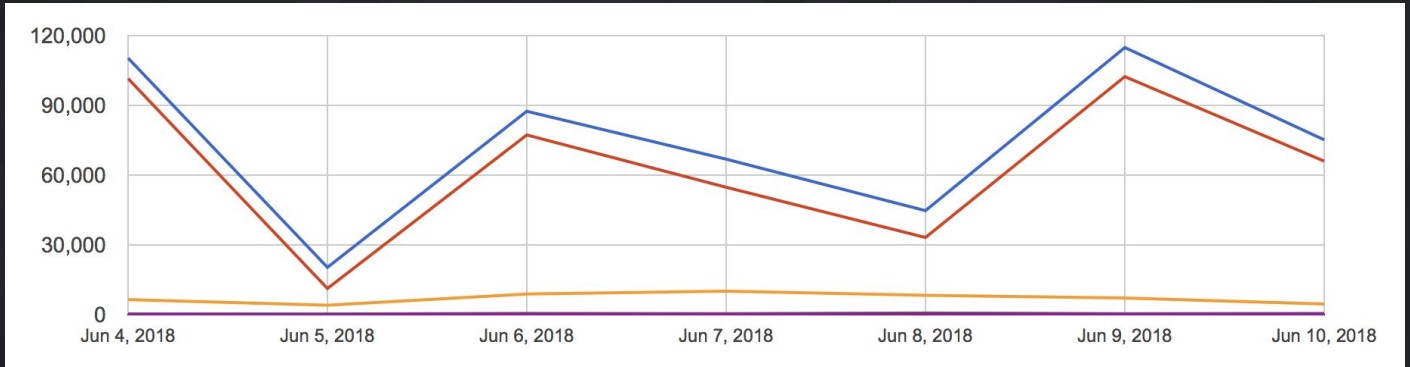
Linux Malware



Linux Malware

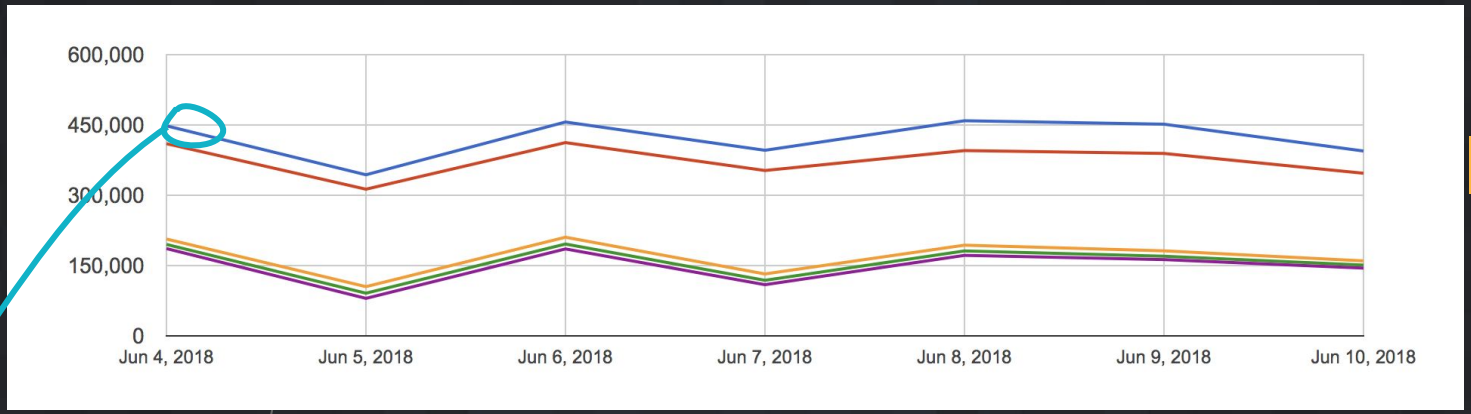


PE FILES



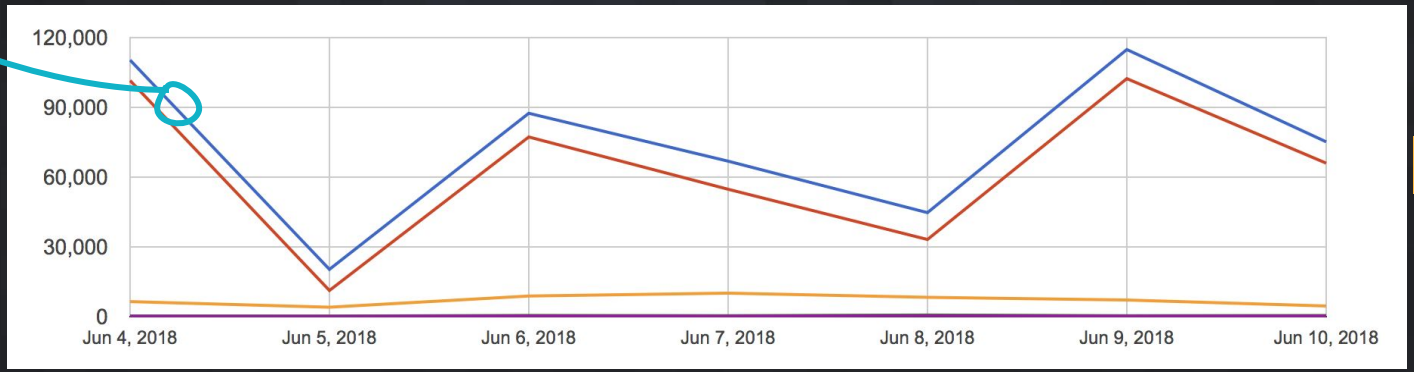
ELF FILES

Linux Malware



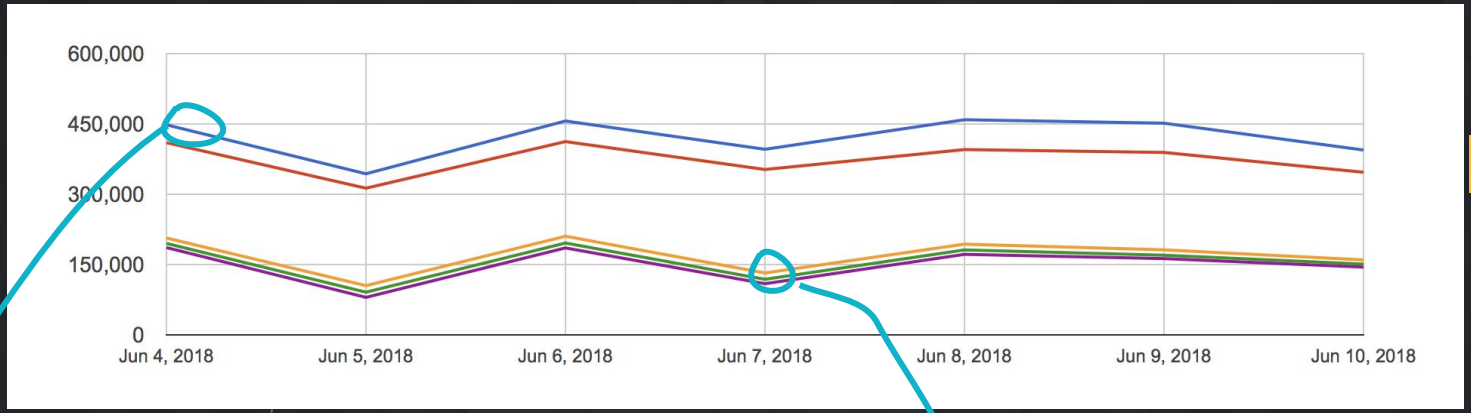
PE FILES

5x



ELF FILES

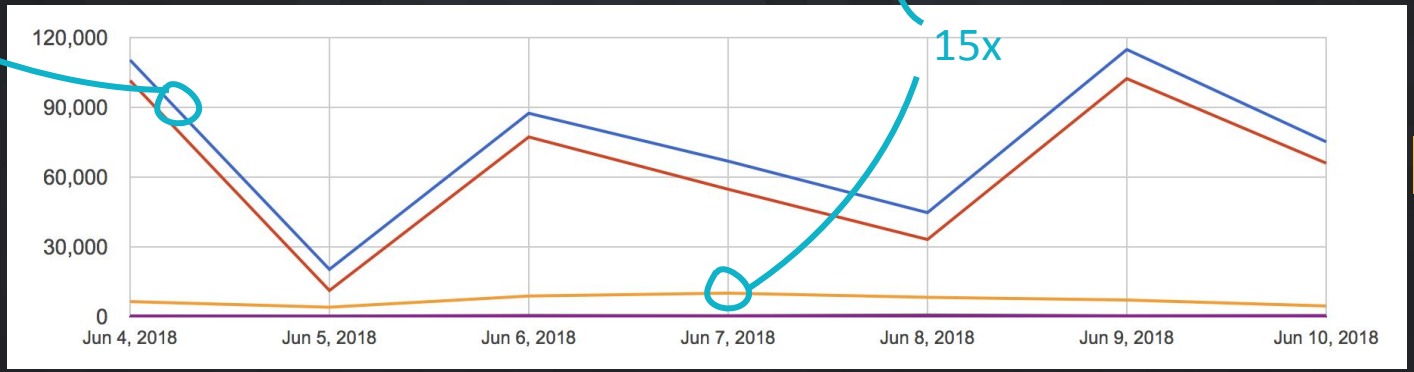
Linux Malware



PE FILES

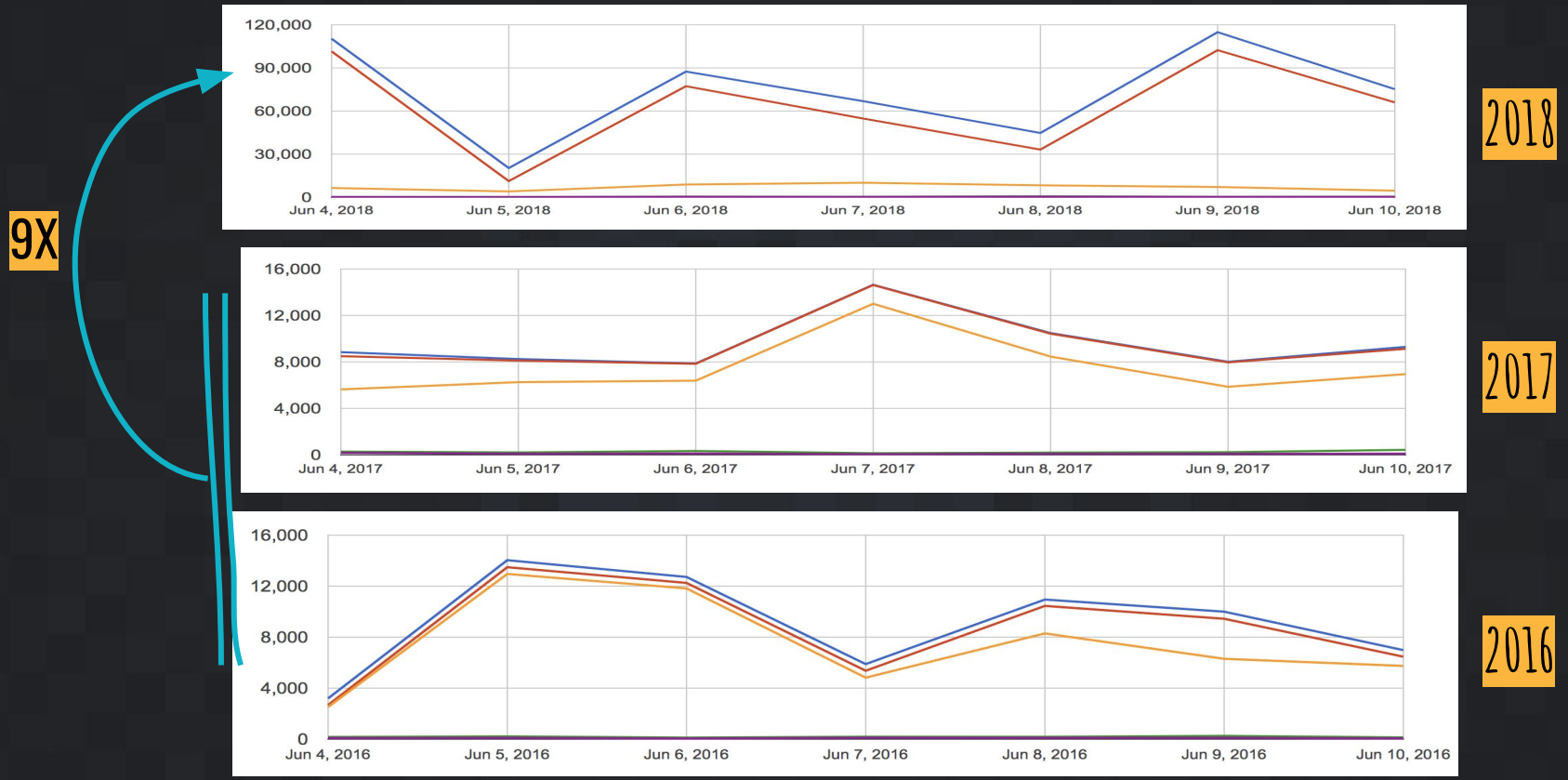
5x

15x



ELF FILES

Linux Malware



Diversity

- Devices (Servers, desktops, routers, cameras, printers, etc)
- Architectures (Intel, AMD, MIPS, PPC, ARM, etc)
- Operating systems (Linux, FreeBSD, Android, Solaris, etc)

Dataset

- Samples collected for 1 year
- 200 selected samples per day
- Final dataset of 10k ELF binaries

Persistence

ELF BINARIES ADOPTING PERSISTENCE STRATEGIES

Path	Samples w/o root	Samples w/ root
`/etc/rc.d/rc.local`	-	1393
`/etc/rc.conf`	-	1236
`/etc/init.d/`	-	210
`/etc/rcX.d/`	-	212
`/etc/rc.local`	-	11
`systemd` service	-	2
`~/.bashrc`	19	8
`~/.bash_profile`	18	8
X desktop autostart	3	1
`/etc/cron.hourly/`	-	70
`/etc/crontab`	-	70
`/etc/cron.daily/`	-	26
`crontab` utility	6	6
File replacement	-	110
File infection	5	26
Total	1644 (21.10%)	

Evasion

ELF PROGRAMS SHOWING EVASIVE FEATURES

Type of evasion	Samples	Percentage
Sandbox detection	19	0.24%
Processes enumeration *	259	3.32%
Anti-debugging	63	0.81%
Anti-execution	3	0.04%
Stalling code	0	-

* Not used for evasion but candidate behavior

Sandbox detection

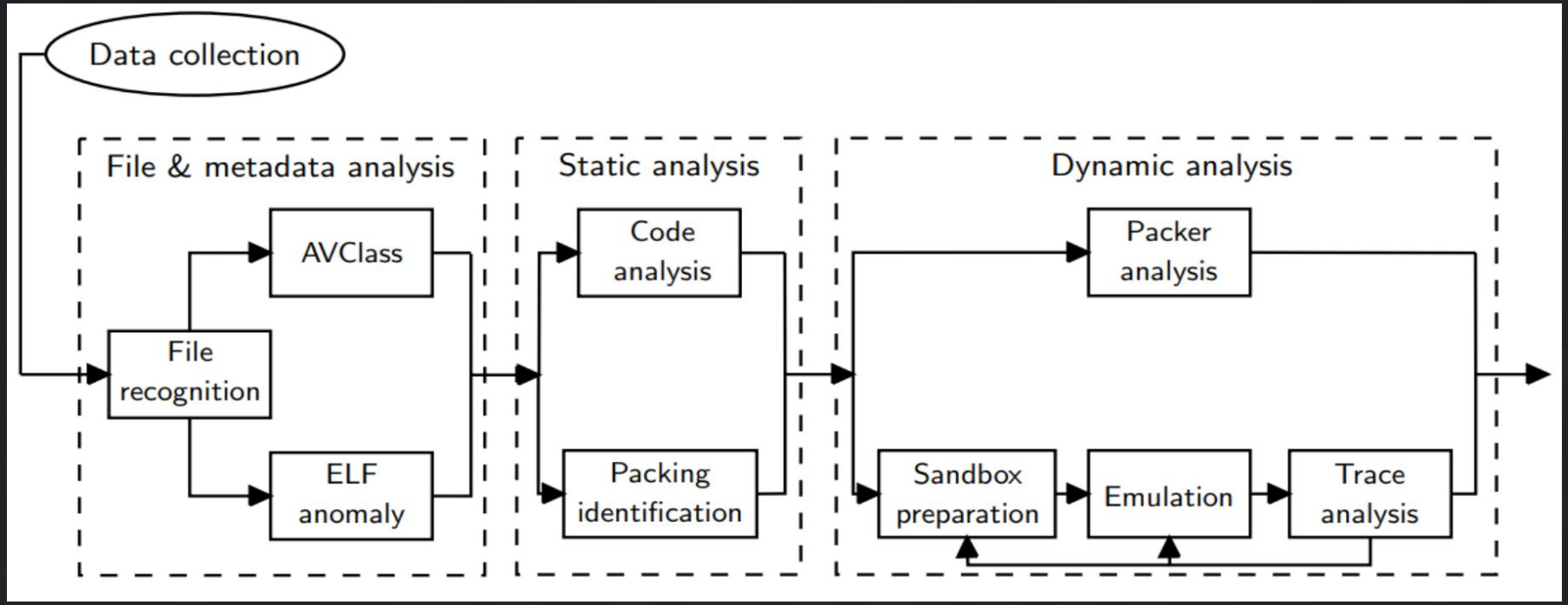
FILE SYSTEM PATHS LEADING TO SANDBOX DETECTION

Path	Detected Environments	#
/sys/class/dmi/id/product_name	VMware/VirtualBox	18
/sys/class/dmi/id/sys_vendor	QEMU	18
/proc/cpuinfo	CPU model/hypervisor flag	1
/proc/sysinfo	KVM	1
/proc/scsi/scsi	VMware/VirtualBox	1
/proc/vz and /proc/bc	OpenVZ container	1
/proc/xen/capabilities	XEN hypervisor	1
/proc/<PID>/mountinfo	chroot jail	1

Dynamic Analysis

- Based on Qemu to support different architectures
- Syscalls and APIs tracing
  - Kprobes and uprobes based on Systemtap
- Five architectures supported with different endianess and ABIs
- Powered by Docker and BuildRoot
- Report generation

Pipeline

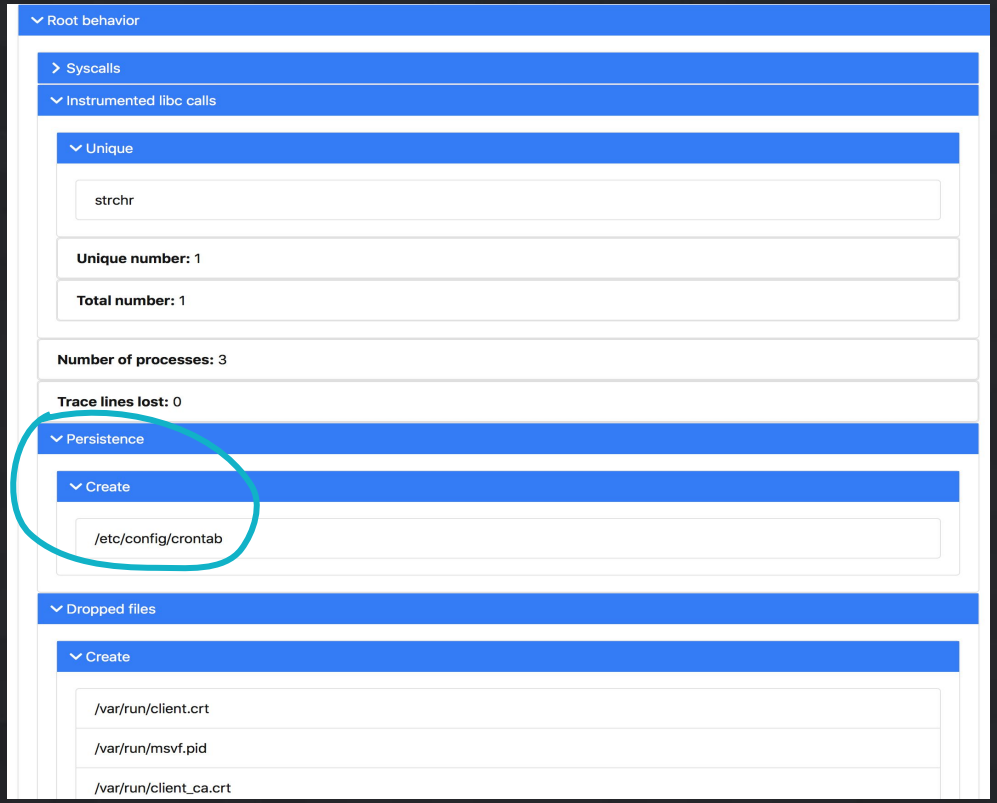

Data collection
File & metadata analysis
AVClass
File recognition
ELF anomaly
Static analysis
Code analysis
Packing identification
Dynamic analysis
Packer analysis
Sandbox preparation
Emulation
Trace analysis

Padawan

- Framework processing data in parallel
- Comprise several analysis modules
- Concept of workers and scheduler
- Distribute the load

Report

Persistence



SHA256: 0e0094d9bd396a6594da8e21911a3982cd737b445f591581560d766755097d92

https://blog.talosintelligence.com/2018/05/VPNFilter.html

Report

IMAGE DOWNLOAD ATTEMPTS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.122.3	192.168.122.1	DNS	75	Standard query 0x1480 A photobucket.com
2	0.037730	192.168.122.1	192.168.122.3	DNS	91	Standard query response 0x1480 A photobucket.com A 209.17.68.100
3	0.039265	192.168.122.3	209.17.68.100	TCP	74	34348 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294929456 TSecr=0 WS=128
4	0.184414	209.17.68.100	192.168.122.3	TCP	74	80 → 34348 [SYN, ACK] Seq=0 Ack=1 Win=4356 Len=0 MSS=1452 TSval=2386541997 TSecr=4294929456 SACK_PERM=1
5	0.185304	192.168.122.3	209.17.68.100	TCP	66	34348 → 80 [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSval=4294929492 TSecr=2386541997
6	0.186094	192.168.122.3	209.17.68.100	HTTP	221	GET /user/nikkireed11/library HTTP/1.1
7	0.332951	209.17.68.100	192.168.122.3	TCP	66	80 → 34348 [ACK] Seq=1 Ack=156 Win=4511 Len=0 TSval=2386542145 TSecr=4294929492
8	0.443091	209.17.68.100	192.168.122.3	HTTP	755	HTTP/1.1 301 Moved Permanently (text/html) (text/html)
9	0.444377	192.168.122.3	209.17.68.100	TCP	66	34348 → 80 [ACK] Seq=156 Ack=690 Win=30316 Len=0 TSval=4294929557 TSecr=2386542255
10	7.443637	192.168.122.3	209.17.68.100	TCP	66	34348 → 80 [FIN, ACK] Seq=156 Ack=690 Win=30316 Len=0 TSval=4294931307 TSecr=2386542255
11	7.444080	192.168.122.3	192.168.122.1	DNS	81	Standard query 0xe1a8 A s1268.photobucket.com

1449	123.950056	192.168.122.3	192.168.122.1	DNS	73	Standard query 0x6ad6 A toknowall.com
1450	123.989082	192.168.122.1	192.168.122.3	DNS	89	Standard query response 0x6ad6 A toknowall.com A 188.165.218.31
1451	123.991109	192.168.122.3	188.165.218.31	TCP	74	42546 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294960443 TSecr=0 WS=128
1452	124.027092	188.165.218.31	192.168.122.3	TCP	74	80 → 42546 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1452 SACK_PERM=1 TSval=4143280679 TSecr=4294960443 WS=128
1453	124.028423	192.168.122.3	188.165.218.31	TCP	66	42546 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=4294960452 TSecr=4143280679
1454	124.029547	192.168.122.3	188.165.218.31	HTTP	220	GET /manage/content/update.php HTTP/1.1
1455	124.066083	188.165.218.31	192.168.122.3	TCP	66	80 → 42546 [ACK] Seq=1 Ack=155 Win=15616 Len=0 TSval=4143280718 TSecr=4294960453

SHA256: 0e0094d9bd396a6594da8e21911a3982cd737b445f591581560d766755097d92

https://blog.talosintelligence.com/2018/05/VPNFilter.html

Report

mkdir("/var/run/d6097e942dd0fdc1fb28ec1814780e6ecc169ec6d24f9954e71954eedbc4c70em", 0770) = 0
mkdir("/var/run/d6097e942dd0fdc1fb28ec1814780e6ecc169ec6d24f9954e71954eedbc4c70ew", 0770) = 0

open("/proc/mtd", O_RDONLY) = -2 (ENOENT)

connect(3, {AF_INET, 127.0.0.1, 9050}, 16) = -111 (ECONNREFUSED)

SHA256: d6097e942dd0fdc1fb28ec1814780e6ecc169ec6d24f9954e71954eedbc4c70
https://blog.talosintelligence.com/2018/05/VPNFilter.html

Padawan

https://padawan.s3.eurecom.fr

Understanding Linux Malware

Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, Davide Balzarotti

IEEE Symposium on Security & Privacy 2018

MANUAL ANALYSIS

Motivation

- Manual and highly technical activity
- Tedious and error-prone task
- Technical expertise has a huge variance

Challenges

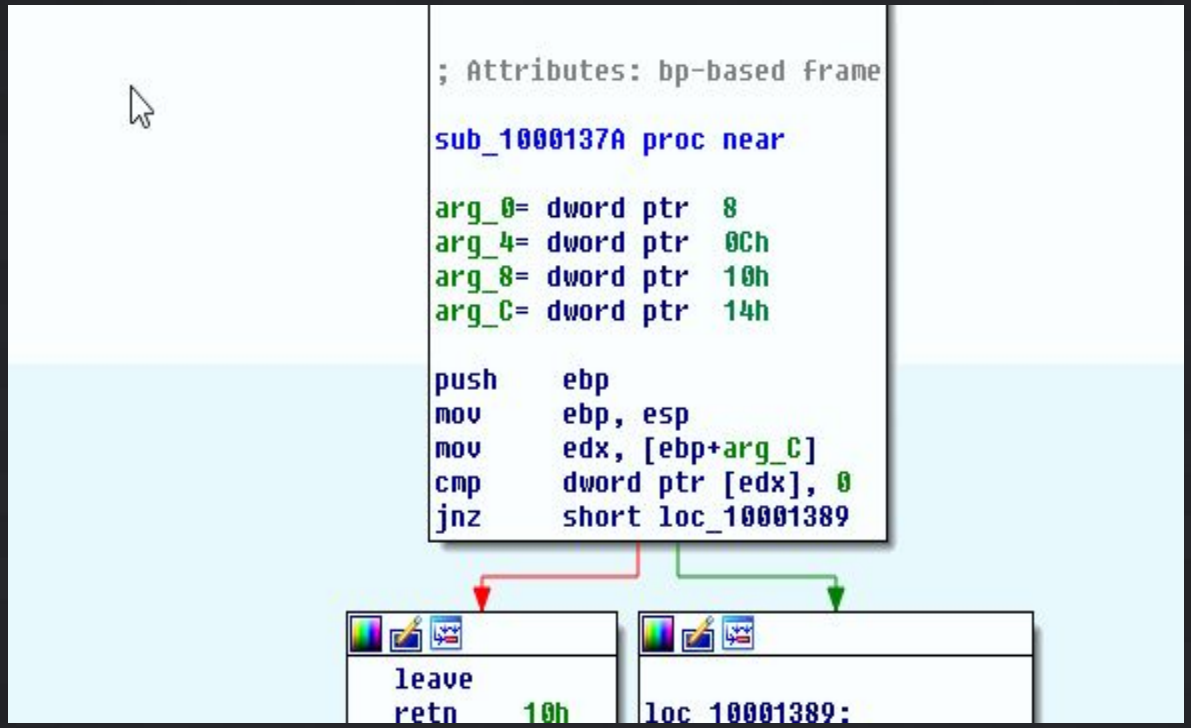
- Disarm the samples (anti-debugging, anti-vm)
- Unpacking and get the final and juicy payload
- Improve static analysis (cope with anti-disass techniques)
- Identify, document specific routines and algorithms (compression, crypto, etc)
- Filter out known libraries
- Identify custom versions of known functions

FIRST

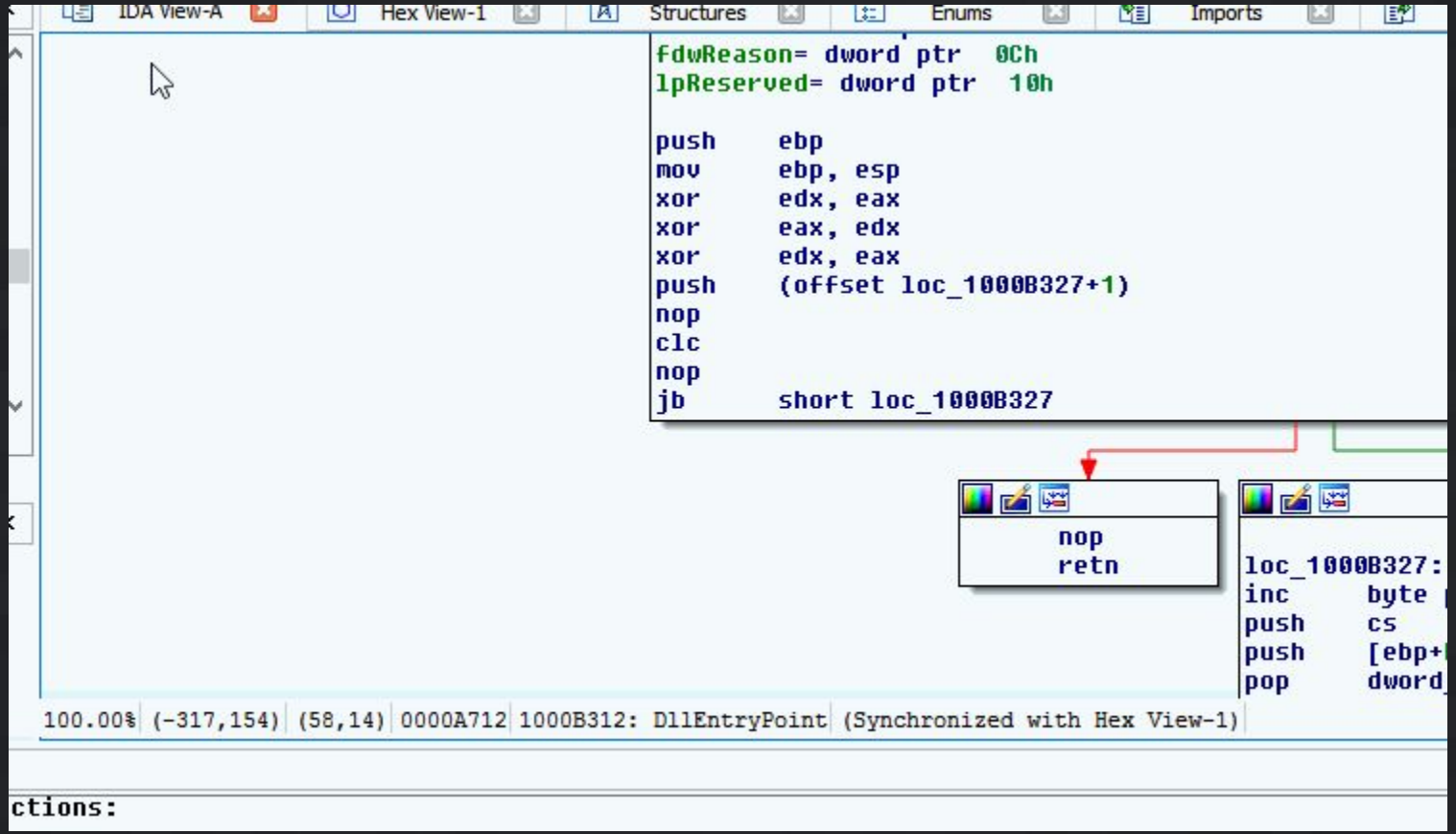
- Function Identification and Recover Signature Tool (FIRST)
- IDA Python plugin developed by Angel Villegas
- Avoid duplicate efforts
- 3 engines at the moment:
  - Exact, Mnemonic and Mask hashing
  - Recently committed Fcatalog support
- Plugins also available for R2 and Ghidra (under dev)
- CLI client will be released soon
- Backend available

https://first.talosintelligence.com/

FIRST


; Attributes: bp-based frame
sub_1000137A proc near
arg_0= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h
arg_C= dword ptr 14h
push ebp
mov ebp, esp
mov edx, [ebp+arg_C]
cmp dword ptr [edx], 0
jnz short loc_10001389
leave
retn 10h
loc_10001389:

FIRST


IDA View-A
Hex View-1
Structures
Enums
Imports
fdwReason= dword ptr 0Ch
lpReserved= dword ptr 10h
push ebp
mov ebp, esp
xor edx, eax
xor eax, edx
xor edx, eax
push (offset loc_1000B327+1)
nop
clc
nop
jb short loc_1000B327
nop
retn
loc_1000B327:
inc byte
push cs
push [ebp+
pop dword
100.00% (-317,154) (58,14) 0000A712 1000B312: DllEntryPoint (Synchronized with Hex View-1)
ctions:

Advocate

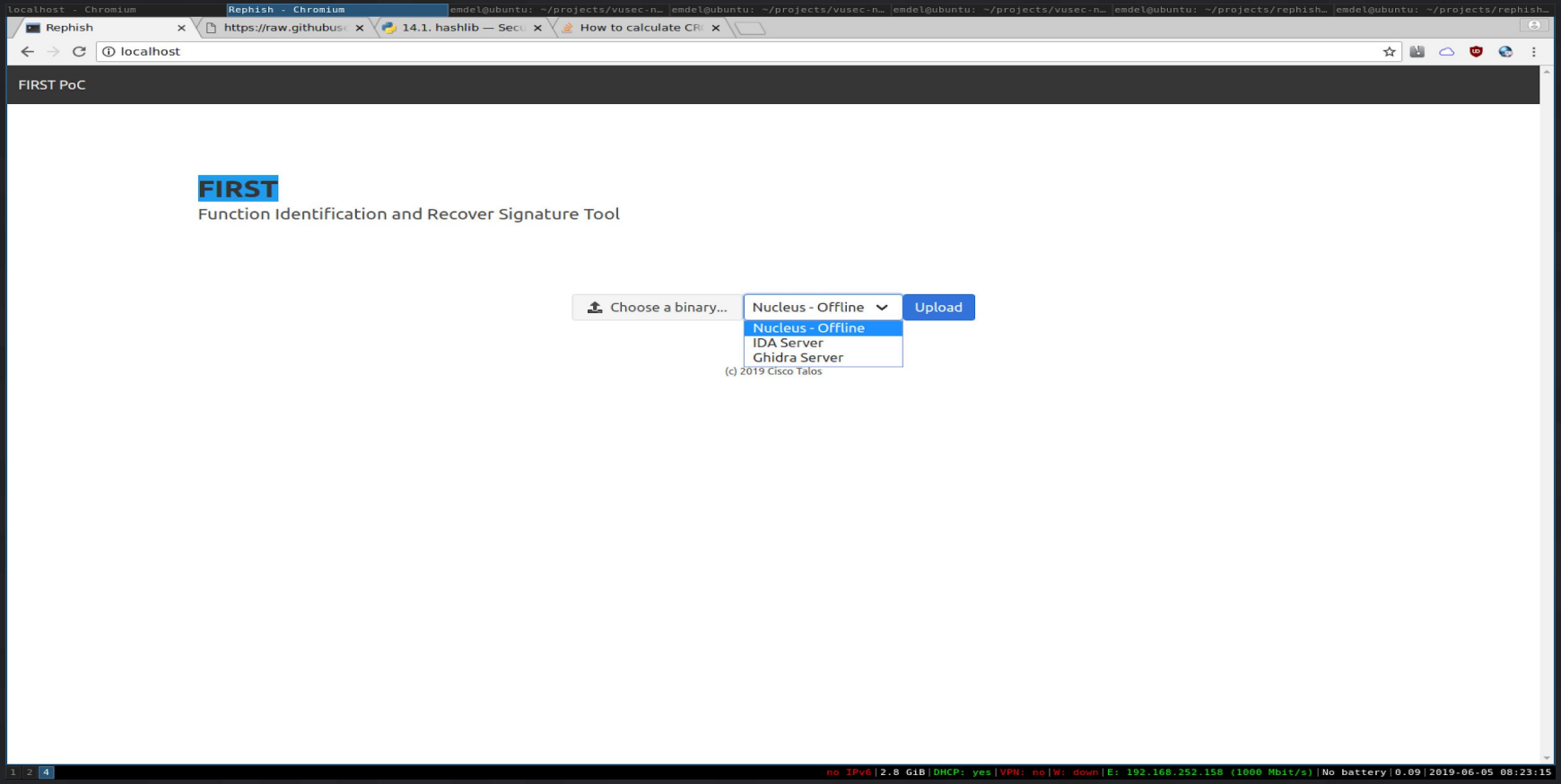
- Dockerize your code for automation
- Create REST APIs
- Create a web UI

Docker + Flask

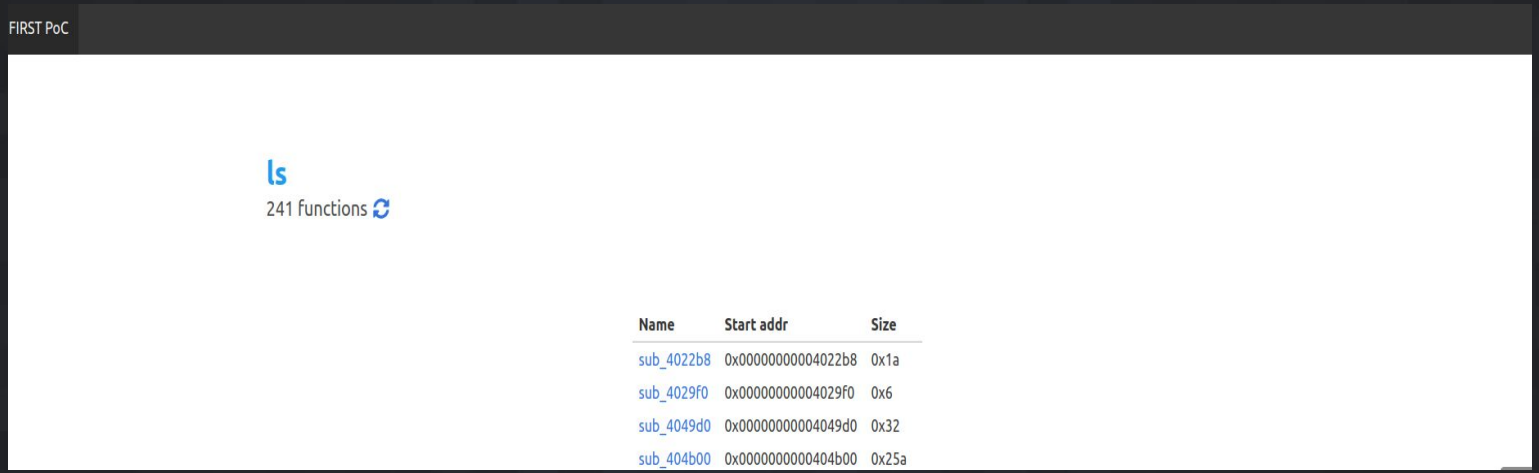
Example - IDA

- Configure IDA
- Dockerize IDA7.*
- Export the IDB
- Have a client based on python-idb

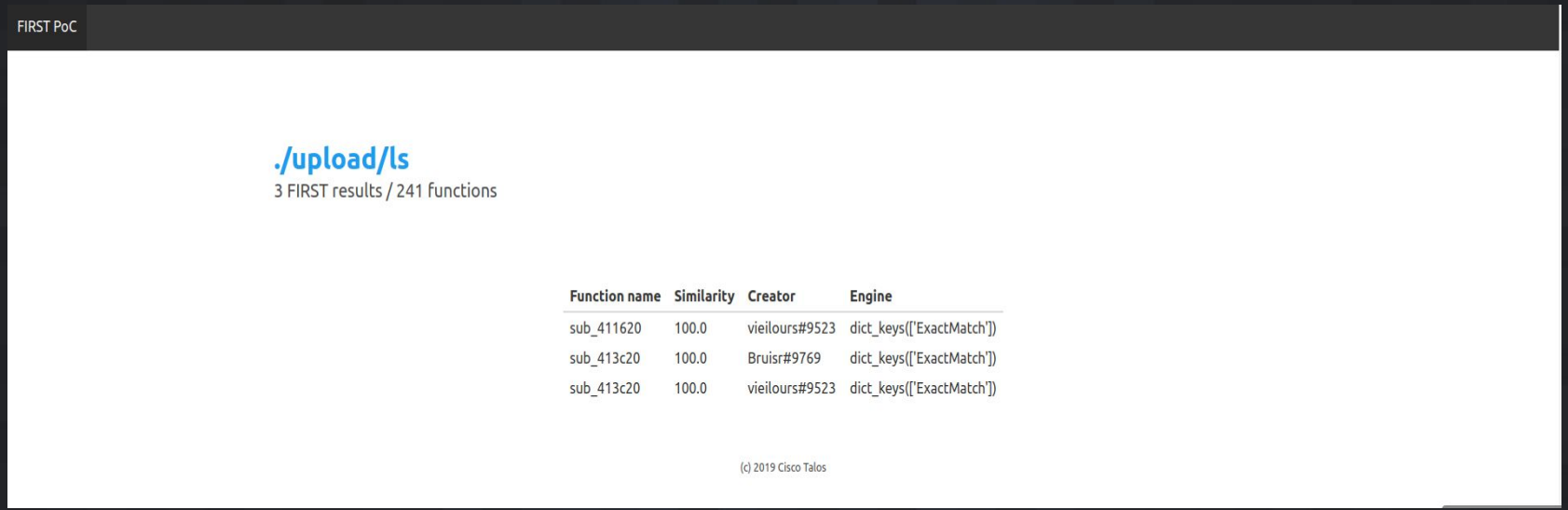
Example - Web UI



Example - Web UI



Example - Web UI



Scriptable sandbox

- PyREBox from my colleague Xabier Ugarte Pedrero
- Python scriptable sandbox
- Based on QEMU
- Automate any kind of task

Scriptable sandbox

Scriptable sandbox

Select the file(s) to submit

Browse... No files selected. Upload

Submit as a single task | Preserve file names

Submission options

Select analysis time: 5 min.

Select analysis VM: Windows XP SP3, 32 bit

Unpacker | ISFB-Gozi | Qakbot | Locky | RAT decoders

Generic-unpacker. No configuration options available.

Email: magrazia@cisco.com
Twitter: @emd3l

TALOSINTELLIGENCE.COM

blog.talosintelligence.com

@talossecurity