Malware Detection in the AI Era: Attacks and Defenses on Machine Learning Classifiers

Dmitrijs Trizna

Security Researcher | Microsoft Corporation, University of Genova

Luca Demetrio

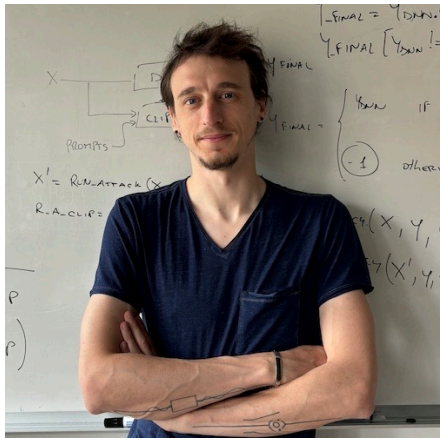
Assistant Professor | University of Genova

WhoAreWe

Security Researcher @ Microsoft
Tech-lead in M365/Azure back-end:
AI/ML for Linux & Kubernetes cyber-threat detection

Doctoral Researcher @ UNIGE
AI/ML based malware detectors

In Past:
- Red Teaming
- NATO events
- OSCP, SANS/GIAC certs
- …



Assistant Professor @ UNIGE

Doing research for fun and profit since 2019

Main contribution in offensive security
with Adversarial EXEmples

Reviewer for top-tier journals and conferences

Working in EU projects

elsa
European Lighthouse on Secure and Safe AI

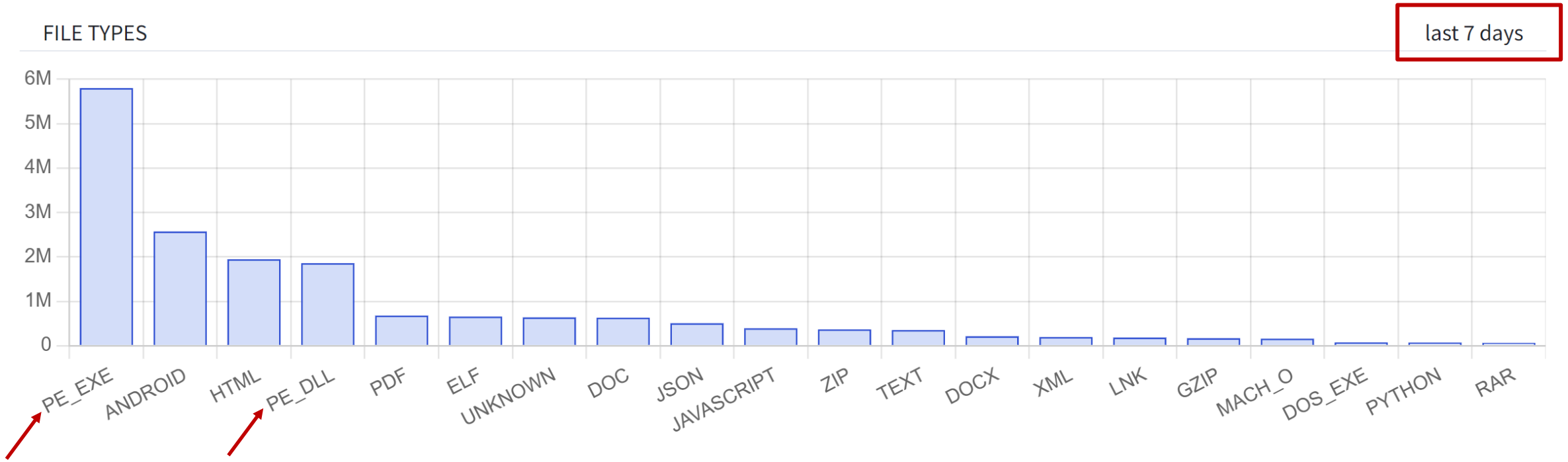
TESTABLE

Agenda

- Part 1: **Building** AI-Based Malware Detector
  - Intro and Threat Model
  - Static Malware Analysis with AI/ML
  - Dynamic Malware Analysis with AI/ML
- Part 2: **Attacking** AI-Based Malware Detector
  - Intro to Adversarial Machine Learning
  - Adversarial EXEmples
  - Defenses against Adversarial Manipulations

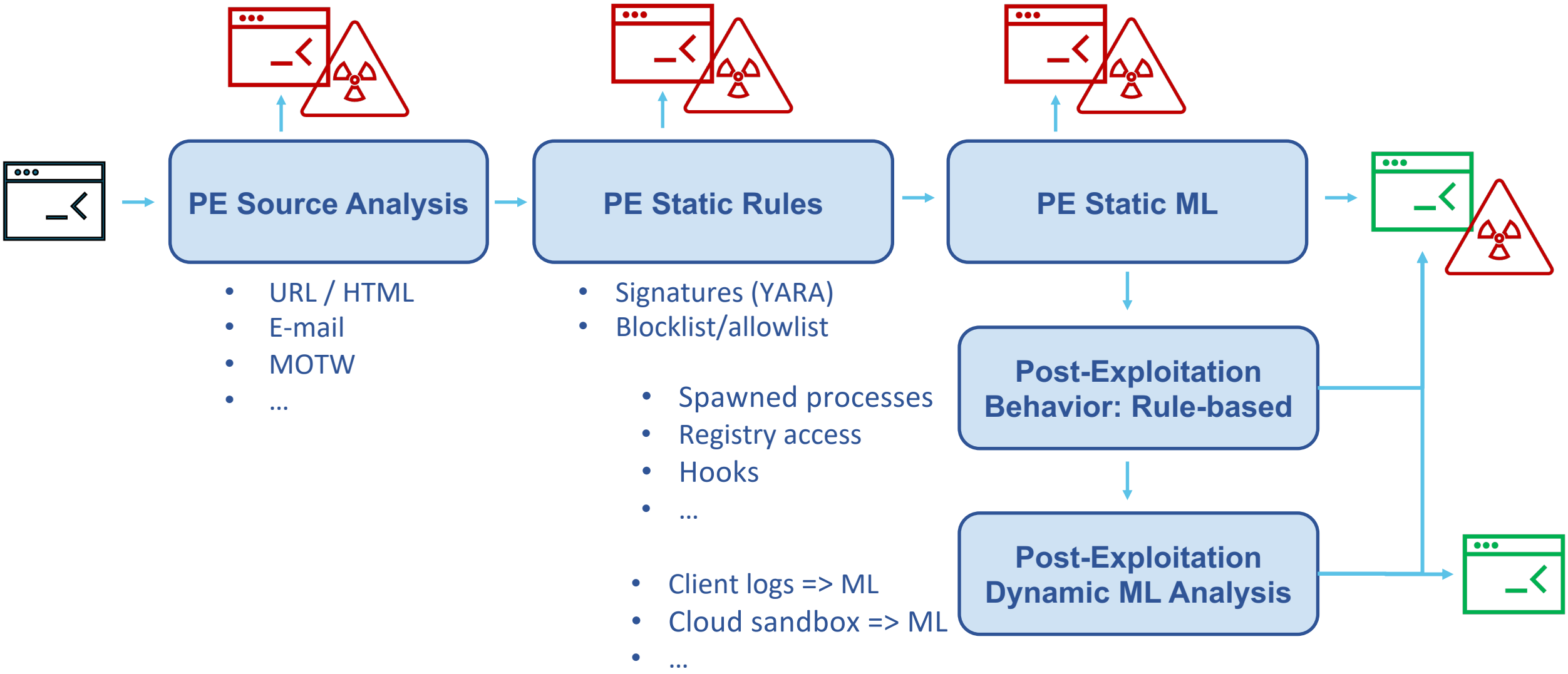
Threat Model

PE Files Are Still The Most Common Malware Type

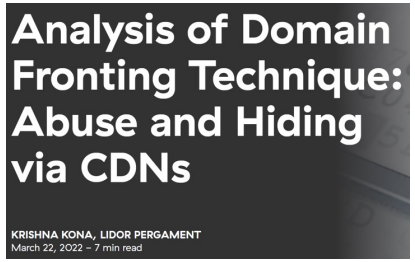


[Source: https://www.virustotal.com/gui/stats]

Modern Malware Detector Stack



Holistic Evasion of Modern Stack



The Problem With YARA: Evading Elastic Security EDR with a NOP instruction

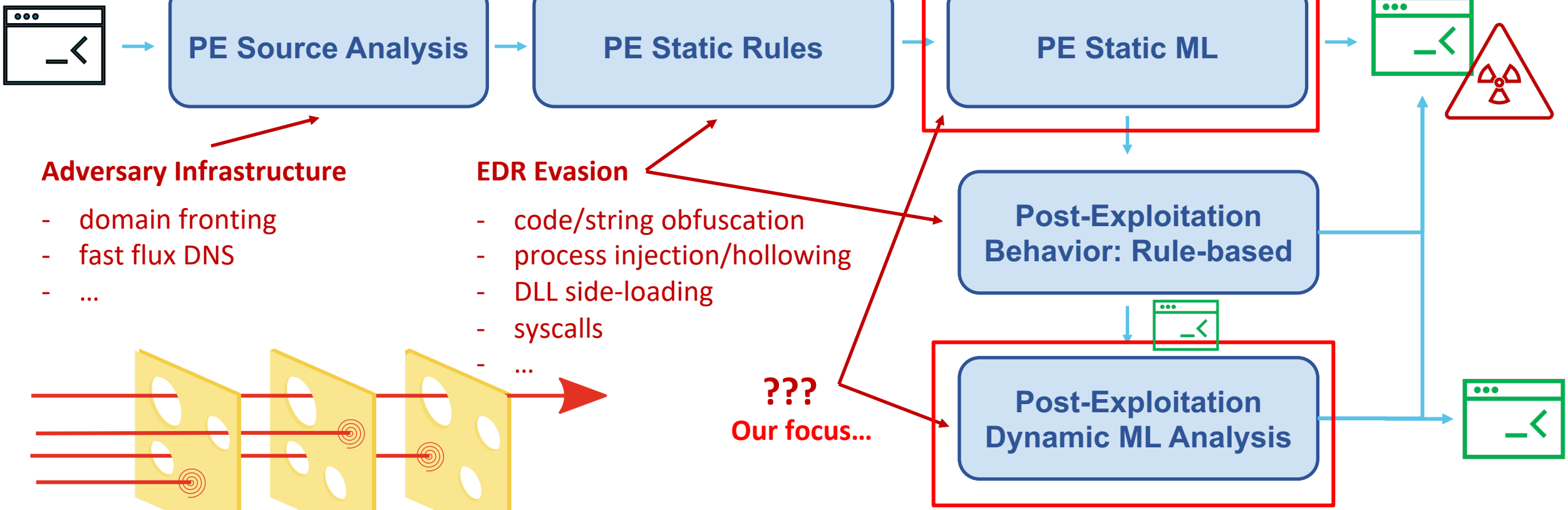
Feb. 10, 2024 | Categories: Research

edr evasion

Past year

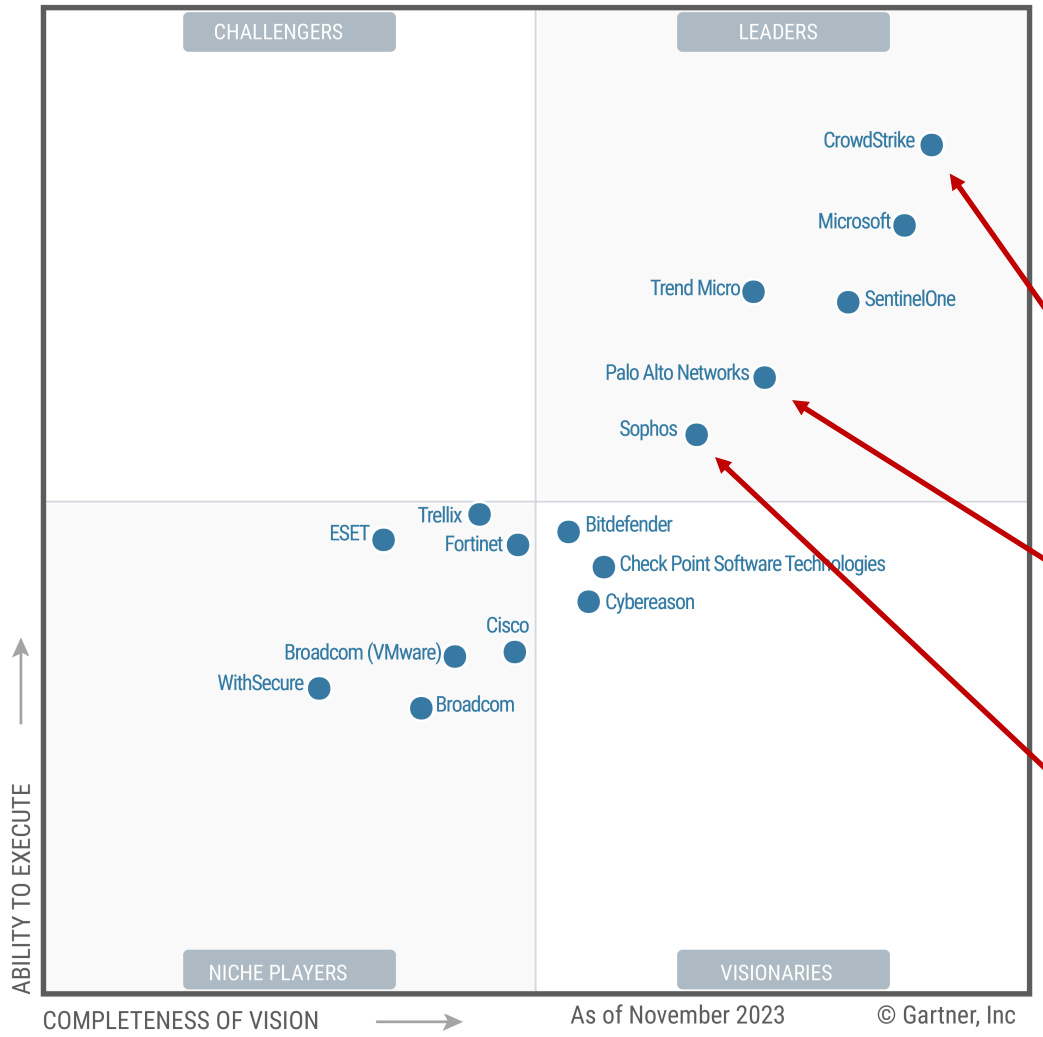
About 10,900 results (0.26 seconds)

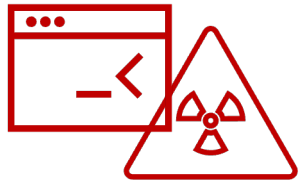
Evading Windows Defender Detections with a combination of MiniDumpWriteDump, Donut, and Process Injection Techniques



Attack Just on Static Module

Figure 1: Magic Quadrant for Endpoint Protection Platforms





Petya Ransomware

malicious
Threat Score: 100/100
AV Detection: 93%
Labeled As: Malware
#petyavirus #petya #apt
#blackenergy #quedagh
#ransomware #sandworm
#sandwormteam #temp.noble
#voodoobear #jigsaw #shamoon

PE Static ML

CrowdStrike Falcon
Static Analysis and ML
Error
× No Additional Data

Avira	✓
Sophos	✓
VirusBlokAda	✓
McAfee	✓

CMC	Undetected
Cylance	Undetected
DeepInstinct	Undetected
Elastic	Undetected
Jiangmin	Undetected
Kingsoft	Undetected
Lionic	Undetected
Palo Alto Networks	Undetected
Panda	Undetected

Source: Gartner (December 2023)

Limitations and Strenghts of Static Rules

YARA

YARA Basics

- Rules have straightforward syntax:
- Easy to use CLI tool:

```
yara <rule> <file>
```

- Features:

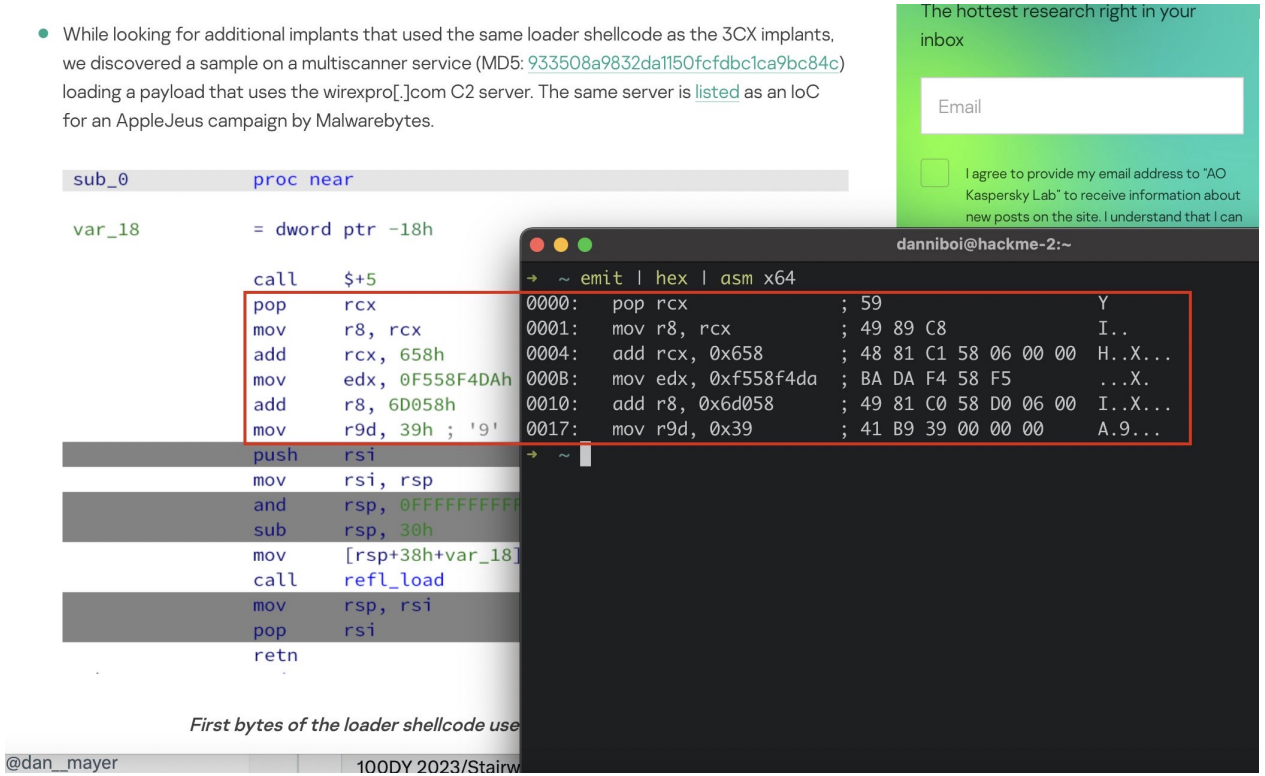
```
rule Example
{
  strings:
    $a = "This program cannot" xor
    $b = { 41 42 ( 43 | 44 )}

  condition:
    $a or $b
}
```

```
dtrizna@tanuki:/data/aponte/repos/mal-pipeline$ yara -X yara_rules/gen_
SUSP_XORed_URL_In_EXE /data/datasets/pe/pe_validation/malware/e430acd25
0x68650:$s1:xor(0x01,http://)
0x68964:$s1:xor(0x01,http://)
```

- Python API:

```python
def check_sample(self, sample_bytes, rules) -> yara.Match:
    rules = yara.compile(source=rules)
    self.matches = rules.match(data=sample_bytes)
    if self.matches:
        print(f"[+] Match found: {self.matches}")
```

YARA Is Awesome: Do's

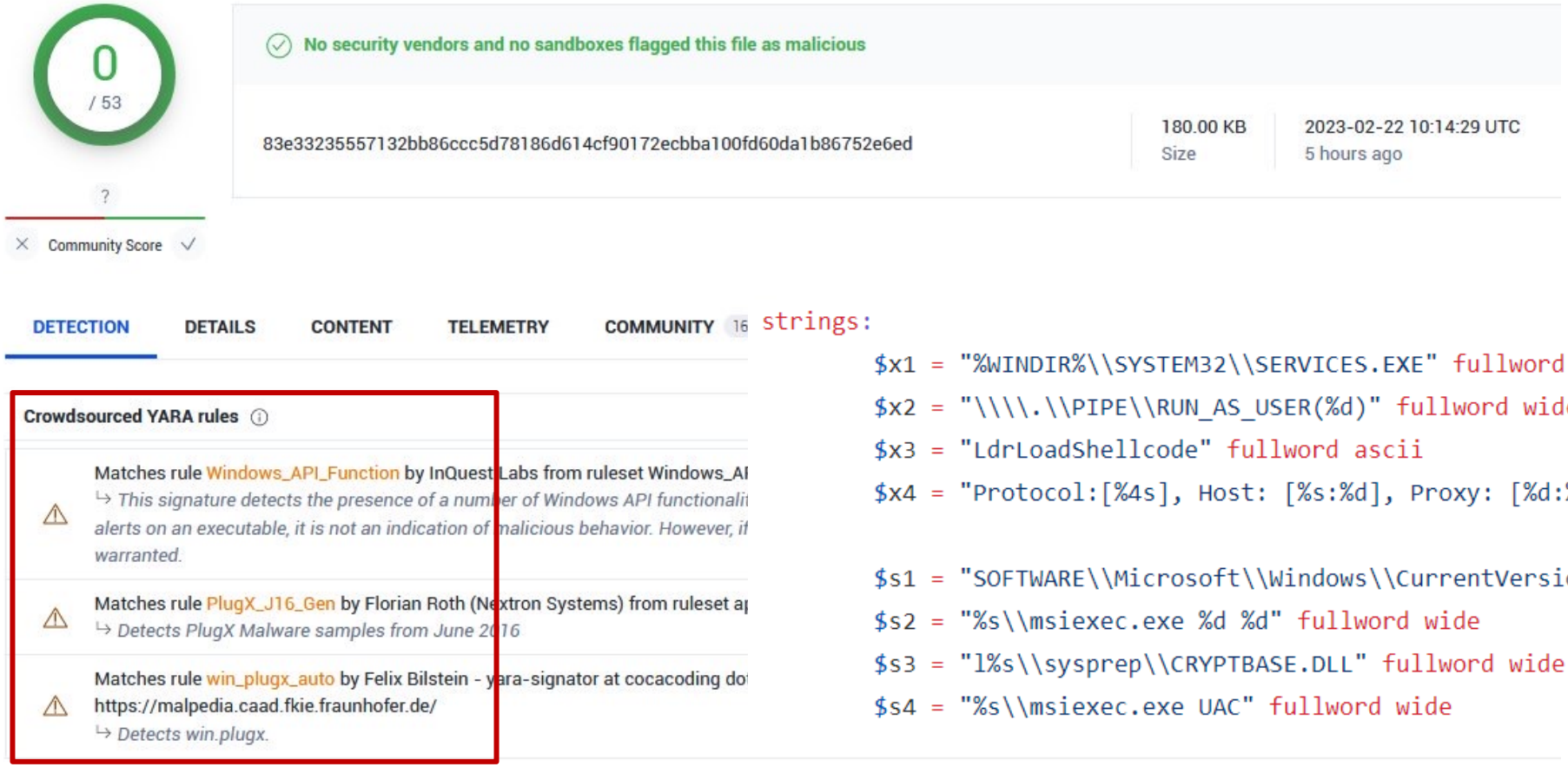



```
rule dev_3CX_Gopuram_AppleJeus_Shellcode_Hunt
{
    meta:
        author = "Daniel Mayer (daniel@stairwell.com)"
        description = "A rule for Gopuram shellcode based off of a screenshot of ida in Kaspersky
        version = "1.0"
        date = "2023-04-03"
        reference1="https://securelist.com/gopuram-backdoor-deployed-through-3cx-supply-chain-at
    strings:
        $test_hunt = {
            59                  // pop rcx
            49 89 C8            // mov r8, rcx
            48 81 C1 58 06 00 00 // add rcx, 0x658
            BA DA F4 58 F5      // mov edx, 0xf558f4da
            49 81 C0 58 D0 06 00 // add r8,0x6d058
            41 B9 39 00 00 00   // mov r9d, 0x39
            }

    condition:
        all of them
}
```

[Source:https://securelist.com/gopuram-backdoor-deployed-through-3cx-supply-chain-attack/109344/]

[Source:https://github.com/MayerDaniel/100DY_2023/blob/main/dan/dev_3CX_Gopuram_AppleJeus_Shellcode_Hunt.yara]

YARA Is Awesome: Do's



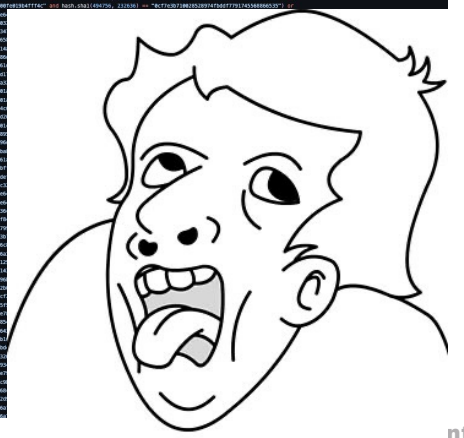

```
strings:
        $x1 = "%WINDIR%\\SYSTEM32\\SERVICES.EXE" fullword wide
        $x2 = "\\\\.\\PIPE\\RUN_AS_USER(%d)" fullword wide
        $x3 = "LdrLoadShellcode" fullword ascii
        $x4 = "Protocol:[%4s], Host: [%s:%d], Proxy: [%d:%s:%d:%s:%s]" fullword ascii

        $s1 = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\User Age
        $s2 = "%s\\msiexec.exe %d %d" fullword wide
        $s3 = "l%s\\sysprep\\CRYPTBASE.DLL" fullword wide
        $s4 = "%s\\msiexec.exe UAC" fullword wide

condition:
        ( uint16(0) == 0x5a4d and filesize < 600KB and ( 1 of ($x*) or 4 of ($s*) ) ) or ( 8 of them )
```

[Source: https://github.com/Neo23x0/signature-base/blob/6b8e2a00e5aafcfcfc767f3f53ae986cf81f968a/yara/apt_win_plugx.yar#L10]

YARA Is Awesome: Do's and Don'ts

```
rule XProtect_MACOS_6eaea4b
{
    meta:
        description = "MACOS.6eaea4b"
    condition:
        Macho and filesize < 10MB and (
            hash.sha1(0, 454544) == "8bda23d6fe3c5f
            hash.sha1(0, 478384) == "eff0e86a0c1fdb
            hash.sha1(0, 888976) == "3bddee4293c423
            hash.sha1(0, 465728) == "ab859e350bca96
            hash.sha1(0, 462816) == "175a12023d4de5
            hash.sha1(0, 465536) == "c91343995496fc
            hash.sha1(0, 465536) == "231b970b66af08
            hash.sha1(0, 888976) == "674493bd15f6df
            hash.sha1(0, 482496) == "7034b366281882
            hash.sha1(0, 482416) == "f2a6ca3b9ebcfa
            hash.sha1(0, 465936) == "f058b8f68f2e30
            hash.sha1(0, 482608) == "cadcf5a2618893
            hash.sha1(0, 922656) == "b6cd41a0b199a1
            (hash.sha1(23728, 267226) == "e15f33dc0
            (hash.sha1(21520, 269322) == "920c2ba31
            (hash.sha1(45744, 277834) == "1812f9c0c
            (hash.sha1(45440, 278122) == "572efd909
            (hash.sha1(45456, 278106) == "01677a475
            (hash.sha1(36704, 270282) == "02f12d6ef
            (hash.sha1(53088, 270282) == "02f12d6ef
            hash.sha1(0, 478384) == "dd7e5f9407f670
```

```
rule XProtect_MACOS_6eaea4b
{
    meta:
        description = "MACOS.6eaea4b"
    condition:
        Macho and filesize < 10MB and (
            hash.sha1(0, 454544) == "8bda23d6fe3c5f61bbe035b3b3955c128fe5fd0c" or
            hash.sha1(0, 478384) == "eff0e86a0c1fdb31442b3b27ae275265144b22ec" or
            hash.sha1(0, 888976) == "3bddee4293c423dcf791872e214c364b89df558b" or
            hash.sha1(0, 465728) == "ab859e350bca96ed8ab4d3ee87ecbdaad42cbf76" or
            hash.sha1(0, 462816) == "175a12023d4de5d0b2cb484fa6b22f4a579c59b0" or
            hash.sha1(0, 465536) == "c91343995496fc20c853d177411338cfe954994f" or
            hash.sha1(0, 465536) == "231b970b66af08780b6fbaf07367d1c8d73d7f8e" or
            hash.sha1(0, 888976) == "674493bd15f6df947d6a32d42ffdd800197a05a9a" or
            hash.sha1(0, 482496) == "7034b366281882f3839089dbc99dde1c409db2d1" or
            hash.sha1(0, 482416) == "f2a6ca3b9ebcfab66eda50621dbf1bb1e52d3e07" or
            hash.sha1(0, 465936) == "f058b8f68f2e306ca00f3c43b485536ec9efa13a" or
            hash.sha1(0, 482608) == "cadcf5a2618893e06477dde8162a651c0b971ad7" or
            hash.sha1(0, 922656) == "b6cd41a0b199a131572e9185805a523a4af285b5" or
            (hash.sha1(23728, 267226) == "e15f33dc0ab40e560b25a2548fa76f98c46d7a64" and hash.sha1(492920, 235168) == "5134edfe096a6ff12f803
            (hash.sha1(21520, 269322) == "920c2ba31e95917aa2aae5e0b9f60c62034e913b0" and hash.sha1(490900, 237060) == "0bd8de35f2a5924eda465
            (hash.sha1(45744, 277834) == "1812f9c0cc91ac3f1b21549ba3345bd388687be8" and hash.sha1(522084, 238540) == "a7634e3dcd5541fa5f735
            (hash.sha1(45440, 278122) == "572efd9090128cb2ba7cf03a1ef95852b8803d61" and hash.sha1(521784, 238816) == "a229148416fa671c5ee1a
            (hash.sha1(45456, 278106) == "01677a4751a36b7c2e85350a0325450d06ddf94b" and hash.sha1(521784, 238816) == "a1bf16e40febb8fae0d61
            (hash.sha1(36704, 270282) == "02f12d6efddb9915ef5e48dc8672e8c49eaf695d" and hash.sha1(508868, 235172) == "0cdb4314624dd23045bf4
            (hash.sha1(53088, 270282) == "02f12d6efddb9915ef5e48dc8672e8c49eaf695d" and hash.sha1(525252, 235172) == "0cdb4314624dd23045bf4
            hash.sha1(0, 478384) == "dd7e5f9407f670a8ee04ba4b326c70c409db4871" or
            hash.sha1(0, 474224) == "166f3d5be9cde70c3bf0a22fbb8365d13d81ca34" or
            hash.sha1(0, 450256) == "e6beeb6b32a140904a648fca9dab614d73dcd94c" or
            hash.sha1(0, 474224) == "09b03db91357d5a067439d101e81c163f4eba4b0" or
            hash.sha1(0, 888864) == "e3861456673963ebfedc99665868106ec00e23607" or
            hash.sha1(0, 922480) == "bd13d22095d377938c50088e559fa3079143cb0f2" or
            (hash.sha1(26160, 264618) == "25cb0ea0b706034409c7439ada832e141a9099cf" and hash.sha1(495056, 232672) == "bdc7c63c90390e7d737c0
            (hash.sha1(24704, 266058) == "e522e55f91c3fe14079fa142b0ffdd1a929657c9" and hash.sha1(493628, 234068) == "aaa386881e9f0c210e8c3
            (hash.sha1(47920, 275578) == "e29a36bf609f5c1700c91261574bf83757f5d6cc" and hash.sha1(524636, 235708) == "f87ee1e0488fbf3c64ab9
            (hash.sha1(47616, 275866) == "e2dbe92730b3e06937d5270b21abd8151ba3a504" and hash.sha1(524348, 235988) == "d43f3b412debf84206efc
            (hash.sha1(47632, 275850) == "5fb180aefdfa3a3c7163bd37fac7b8eb193e5286" and hash.sha1(524352, 235984) == "0a59d04c27ca3761e7133
            (hash.sha1(22208, 268314) == "757f1e6b691c2e91f5b9fcebacf35edfcc9ce315" and hash.sha1(493612, 233732) == "639f0af300aed26656ac2
            (hash.sha1(22208, 268314) == "757f1e6b691c2e91f5b9fcebacf35edfcc9ce315" and hash.sha1(493612, 233732) == "639f0af300aed26656ac2
            hash.sha1(0, 454448) == "e4b84e22214062b57a3f3a81fba5d4ddd163b0bb" or
            hash.sha1(0, 474144) == "5c448f6272d63a57c0e7965d09bd93e23a15ee86" or
            hash.sha1(0, 888864) == "a15f39ce5007e25e742d071d15c8e38658165e5a" or
            hash.sha1(0, 922384) == "cbf08fae71fcd46cc852fad75026854660c40e168" or
            (hash.sha1(25584, 264906) == "38ae77158e1ce3079a36303bf45d46246befc753" and hash.sha1(494844, 232940) == "dddb9c37fa39a49c7e17f
            (hash.sha1(24128, 266330) == "da85d3675bbd891f6e7d0269173243adaa1300ff" and hash.sha1(493444, 234284) == "d0fa0e947ab4d51278bc3
            (hash.sha1(46928, 276282) == "51b695a80c74a0d30d4c614a8ecc605457bb7adc" and hash.sha1(524176, 236216) == "05806c64f585306b25a75
            (hash.sha1(46624, 276570) == "326e55ae93aeced73be9bf830a574a4ea551b231" and hash.sha1(523872, 236496) == "55efadfed76814a75f11e
            (hash.sha1(46656, 276554) == "77b726c10e6d381456704be730c76b5963ff9625" and hash.sha1(523876, 236492) == "4420fede63ed01df9fd20
            (hash.sha1(37392, 269162) == "77756fb4720bc7ea364a947df659495a473ff15d" and hash.sha1(509160, 234424) == "dfdab2704a010782a4b0c
            (hash.sha1(53776, 269162) == "77756fb4720bc7ea364a947df659495a473ff15d" and hash.sha1(525544, 234424) == "dfdab2704a010782a4b0c
            hash.sha1(0, 450112) == "21b63689d192a7d1309d98afa35d42f695098d7a" or
            hash.sha1(0, 474048) == "509dba18a168fdeecf990704741e14cb17b2a31e" or
```



[Source: https://github.com/SentineLabs/XProtect-Malware-Families/blob/779534f1c966611e07d9d993906ffee3c57b5974/2192.yara#L2438]

Malware Is Too Variable For Manual Human Labor

[Source: “Evolution of Malware Prevention”, https://info.microsoft.com/rs/157-GQE-382/images/Windows%2010%20Security%20Whitepaper.pdf]

Static Malware Analysis With Machine Learning

Part 1:

Feature Engineering

Windows PE File Format

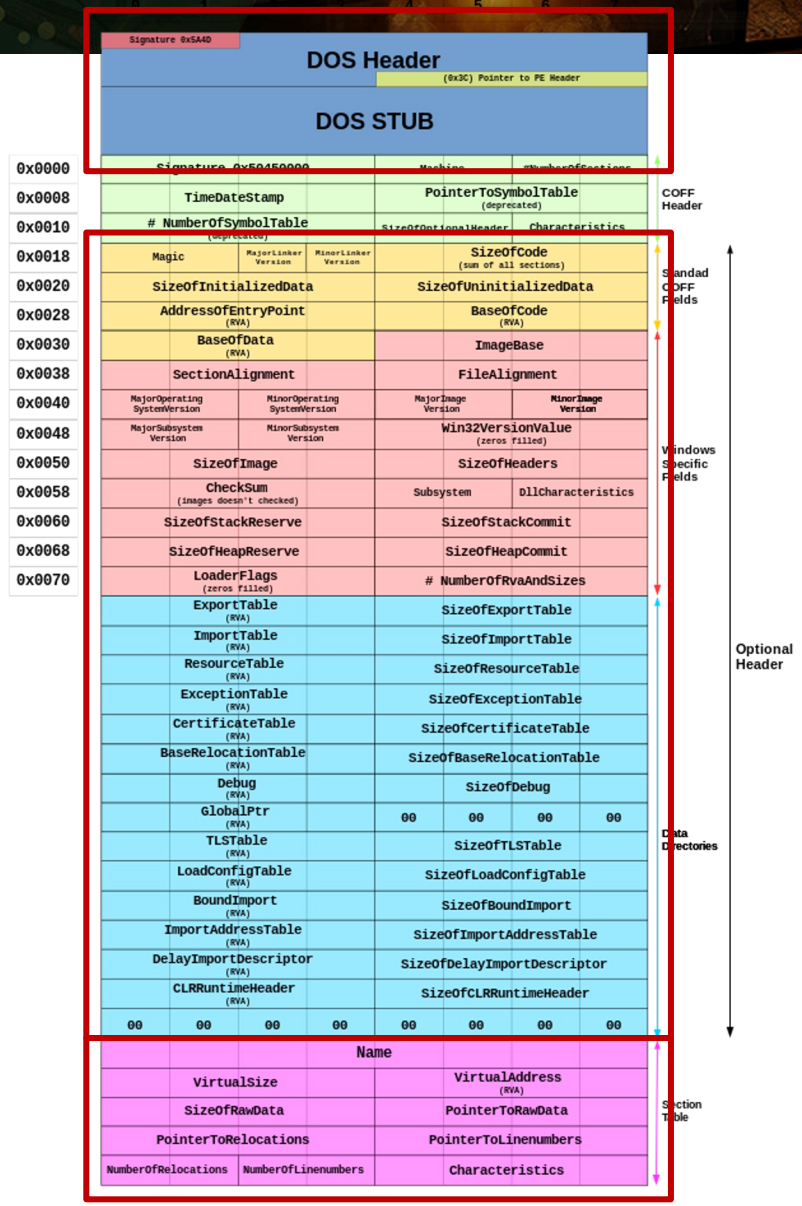
Format adapted for “modern” programs
(from Windows NT 3.1 on)

Before there were other formats, one is the DOS
(kept for retro compatibility)

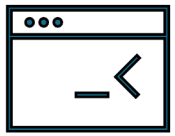
PE Header
Real metadata of the program
Describes general information of the file

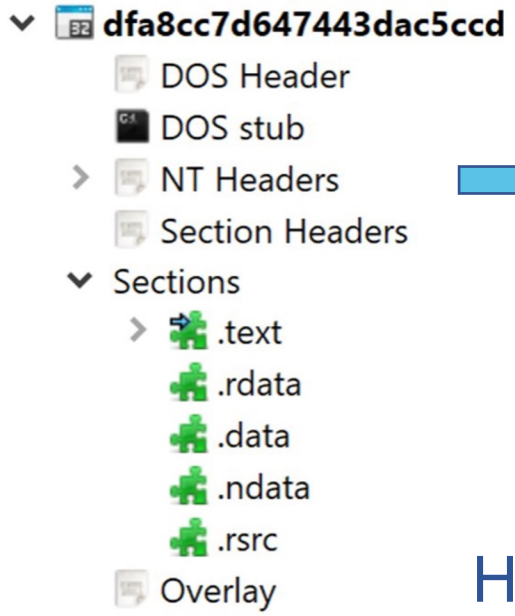
Optional Header
Spoiler: not optional at all :)
Instructs the loader where to find each object inside the file

Section Table and Sections
Describes where to find code, initialized data, resources, etc. to the loader
Examples: code is “.text”, read-only data is “.rodata”, resources are “.rsc”, and counting



Static Malware Classification with ML Models





Static features

(imports, exports, sections, …)

Supervised ML

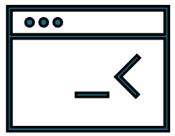
p(malicious) = 0.951

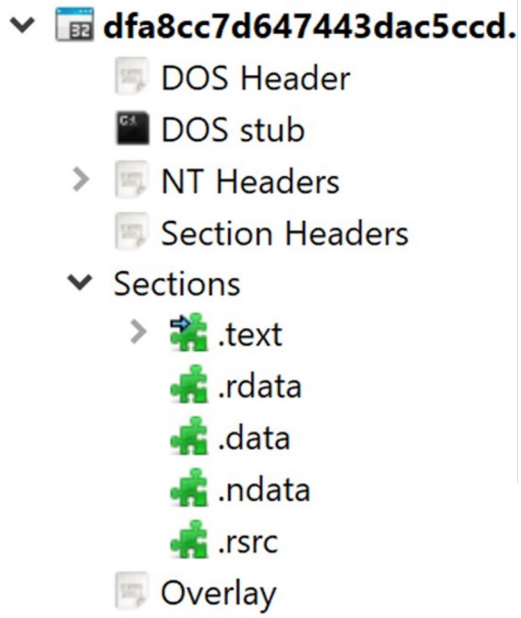
Human or "greedy" labels:

Malicious

Benign

Static Malware Classification with ML Models






```
71  class ByteEntropyHistogram(FeatureType):
72      ''' 2d byte/entropy histogram based loosely on (Saxe and Berlin, 2015).
73      This roughly approximates the joint probability of byte value and local entropy.
74  125  class SectionInfo(FeatureType):
75  126      ''' Information about section names, sizes and entropy.  Uses hashing trick
76  127      to summarize all this section info into a feature vector.
77  128      '''
78  129  195  class ImportsInfo(FeatureType):
    130  196      ''' Information about imported libraries and functions from the
    131  197      import address table.  Note that the total number of imported
         198      functions is contained in GeneralFileInfo.
         199      '''
```

Lab 1:

Static Malware Detection using ML with Gradient Boosted Decision Trees (GBDT)

aka.ms/malware-lab1

Static Malware Analysis With Machine Learning

Part 2:

Byte Level Analysis with Neural Networks

Development of AI and Neural Nets

1999: SIFT — 2006: SURF

Pre- “Deep Learning”: Feature Engineering Era

Computer Vision (CV):

- SIFT
- SURF

Development of AI and Neural Nets

1999: SIFT — 2006: SURF

2000: HMM for POS
2001: CRF for NER

Pre- “Deep Learning”: Feature Engineering Era

Computer Vision (CV):

- SIFT
- SURF

Natural Language Processing (NLP):

- HMM & POS tags
- CRFs & NER tags

Back in 2000 , People Magazine PUBLISHER highlighted Prince Williams' PERSON style who at the time was a little more fashion-conscious , even making fashion statements at times .

Now-a-days the prince mainly wears navy COLOR suits ITEM (sometimes double-breasted DESIGN) , light blue COLOR button-ups ITEM with classic LOOK pointed DESIGN collars PART , and burgundy COLOR ties ITEM .

But who knows what the future holds ...

Duchess Kate PERSON did wear an Alexander McQueen BRAND dress ITEM to the wedding OCCASION in the fall of 2017 SEASON .

Development of AI and Neural Nets

1989: CNNs
(LeCun et al.)

1999: SIFT

2006: SURF

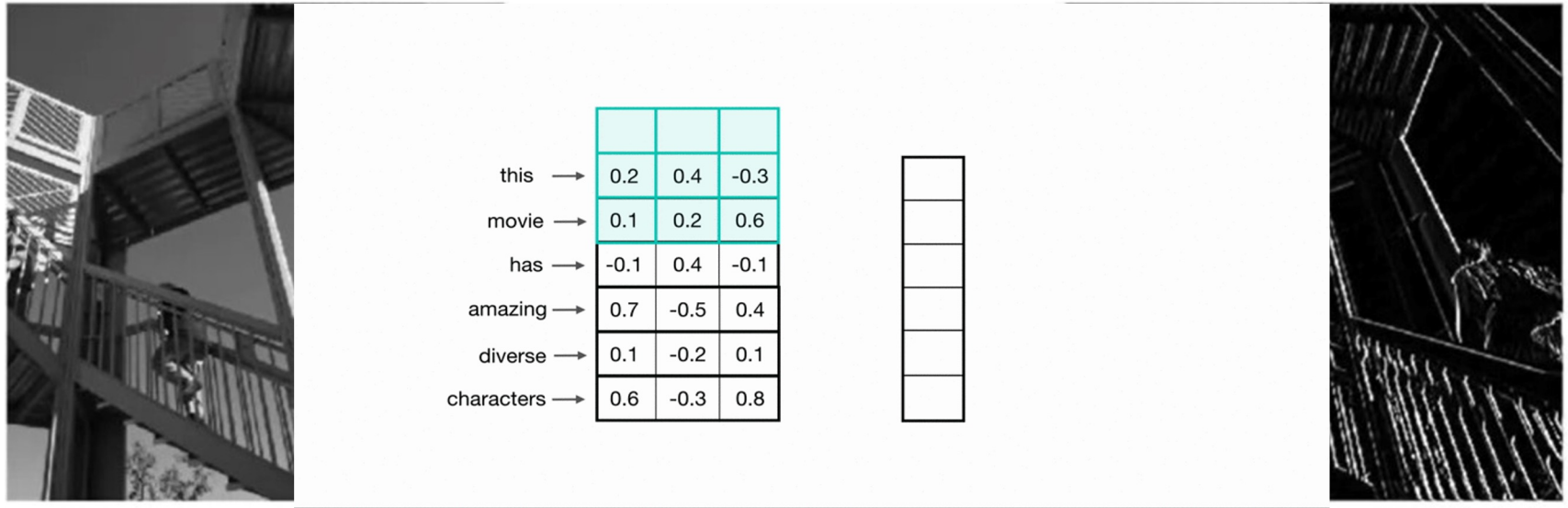
2000: HMM for POS
2001: CRF for NER

2010-2017:
CNNs and RNNs dominate CV and NLP

2010s Deep Learning allows to scale up -- Neural Nets “learn” low level features

Computer Vision (CV):

- Convolutional Neural Networks (CNNs)



Development of AI and Neural Nets

1989: CNNs
(LeCun et al.)

1997: LSTM
(Hochreiter and Schmidhuber)

1999: SIFT

2000: HMM for POS
2001: CRF for NER

2006: SURF

2010-2017:
CNNs and RNNs dominate CV and NLP

2010s Deep Learning allows to scale up -- Neural Nets “learn” low level features

Computer Vision (CV):

- Convolutional Neural Networks (CNNs)

NLP:

- Recurrent Neural Networks (RNNs)

CNNs for Malware Detection

1989: CNNs (LeCun et al.)

1997: LSTM (Hochreiter and Schmidhuber)

1999: SIFT

2000: HMM for POS

2001: CRF for NER

2006: SURF

2010-2017: CNNs and RNNs dominate CV and NLP

2017: CNN for Malware (MalConv)

Malware Detection by Eating a Whole EXE

Edward Raff[1,3,4], Jon Barker[2], Jared Sylvester[1,3] Robert Brandon[1,3,4]
Bryan Catanzaro[2], Charles N
[1]Laboratory for Physical Sciences, [2]NVIDIA, [3]Booz Allen Hamilton,
{edraff,jared,rbrandon}@lps.umd.edu, {jbarker,bcatanzaro}@

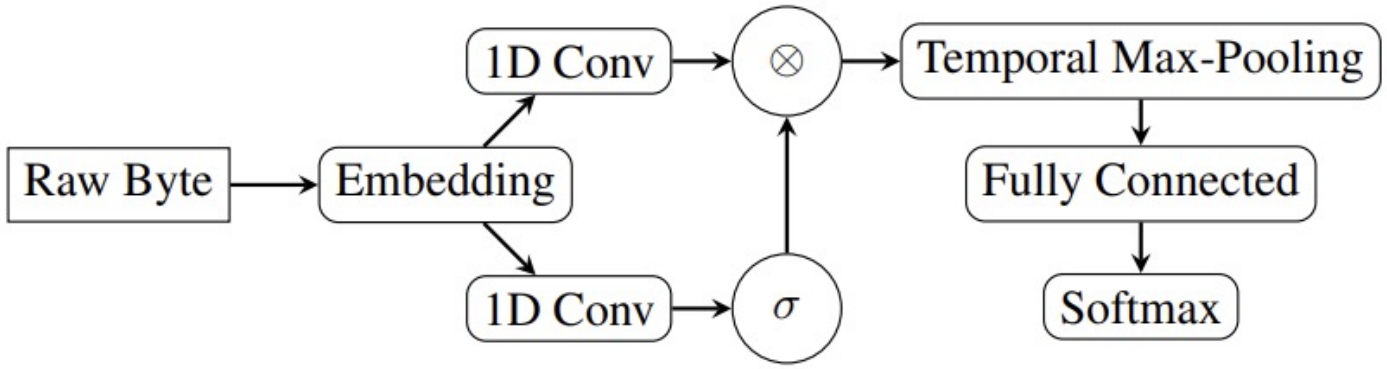


Figure 1. Architecture diagram of MalConv model.

[Source: https://arxiv.org/pdf/1710.09435]

Lab 2:

Feature Extraction from Byte Level Data with Neural Networks and PyTorch

aka.ms/malware-lab2

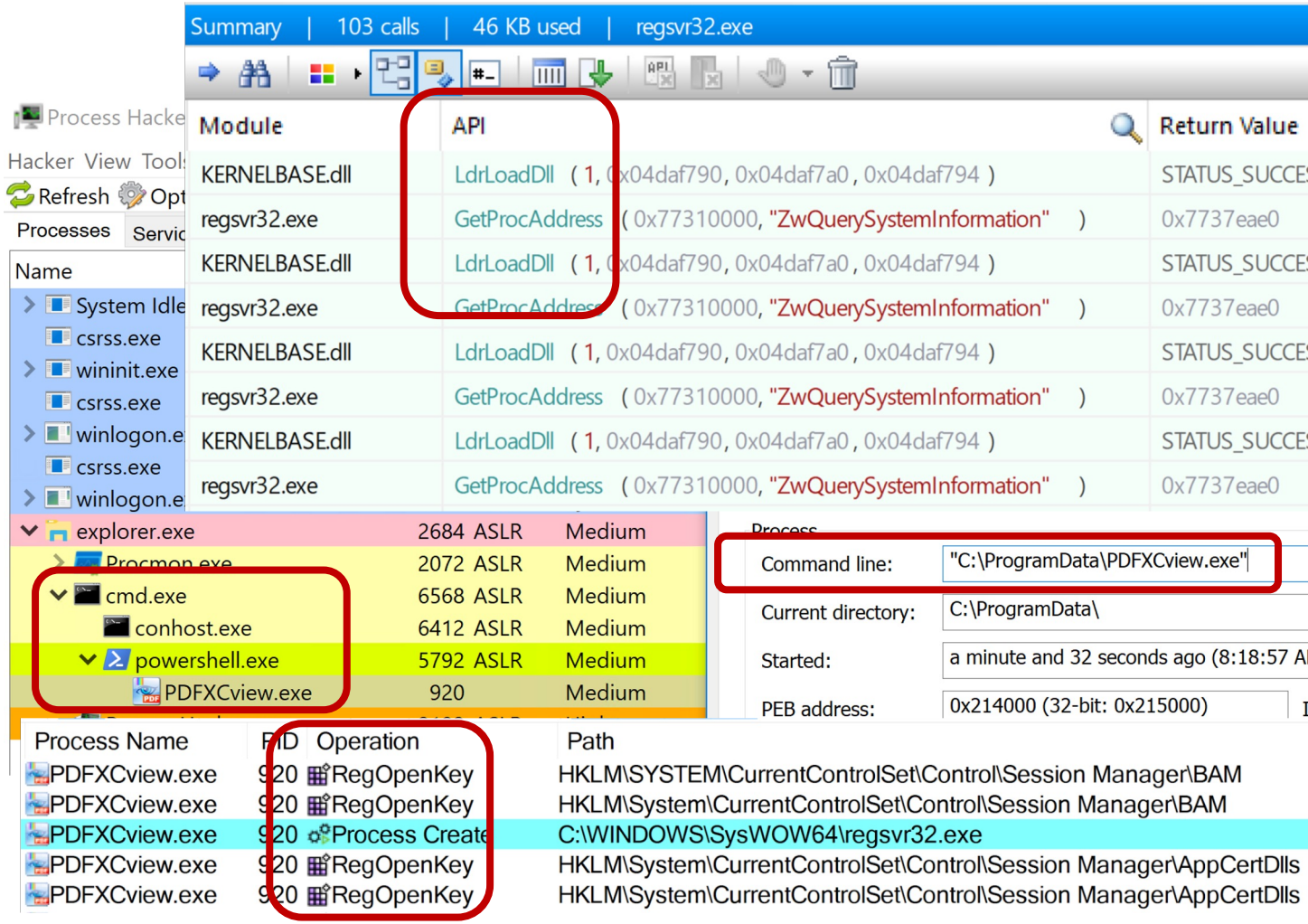
Dynamic Malware Analysis

Behavioral Properties

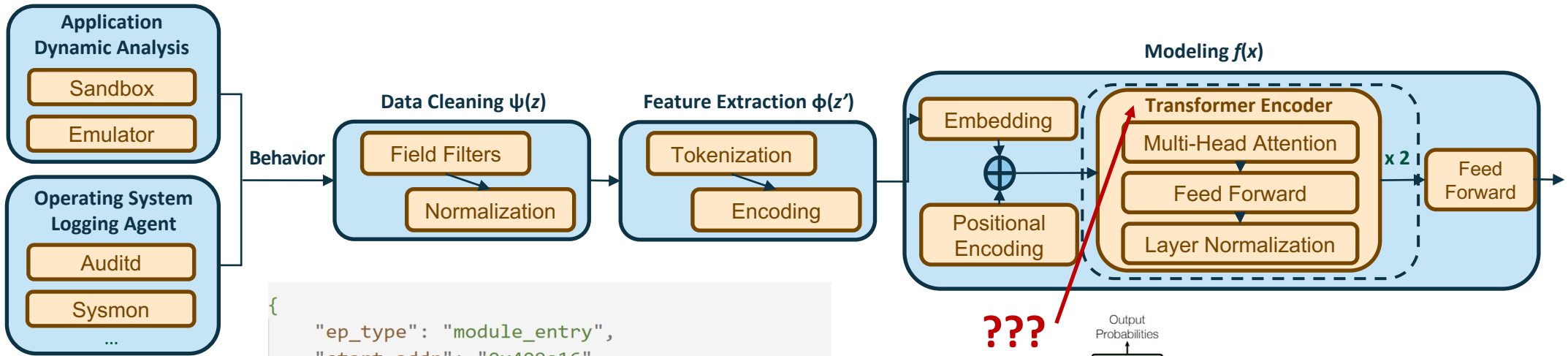
- dfa8cc7d647443dac5ccd.
  - DOS Header
  - DOS stub
  - NT Headers
  - Section Headers
  - Sections
    - .text
    - .rdata
    - .data
    - .ndata
    - .rsrc
  - Overlay

→ **Sandbox** →

→ **Emulator** →



Machine Learning Pipeline for Dynamic Analysis: Nebula



???

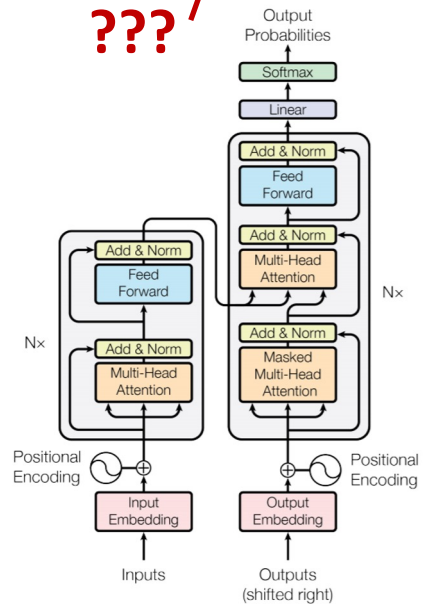
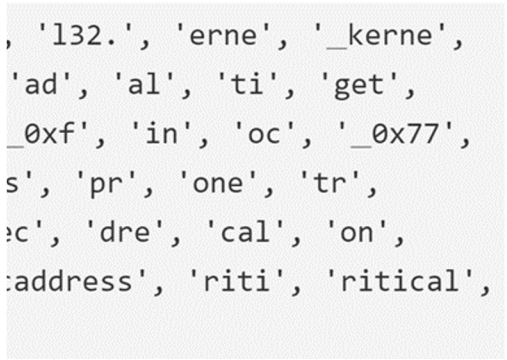


Figure 1: The Transformer - model architecture.



CNNs for Malware Detection

1989: CNNs
(LeCun et al.)

1997: LSTM
(Hochreiter and Schmidhuber)

1999: SIFT

2000: HMM for POS
2001: CRF for NER

2006: SURF

2010-2017:
CNNs and RNNs dominate CV and NLP

2017:
"Attention is All You Need"

Google introduces the "Transformer" model

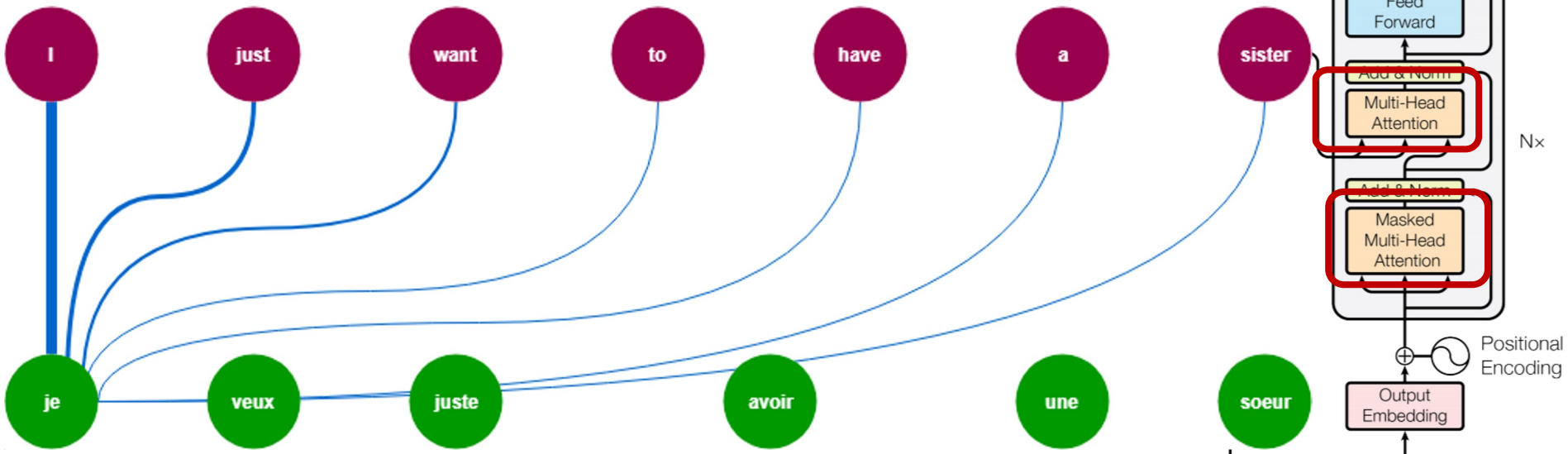
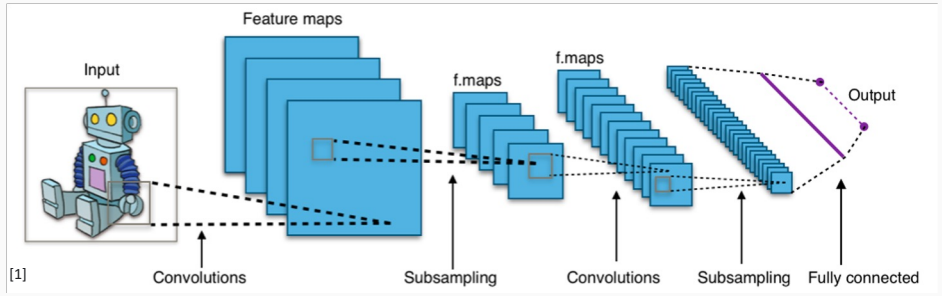


Figure 1: The Transformer - model architecture.

Computer Vision

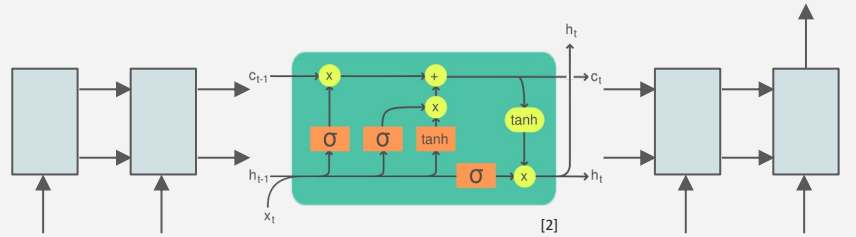
Convolutional NNs (+ResNets)



[1]

Natural Lang. Proc.

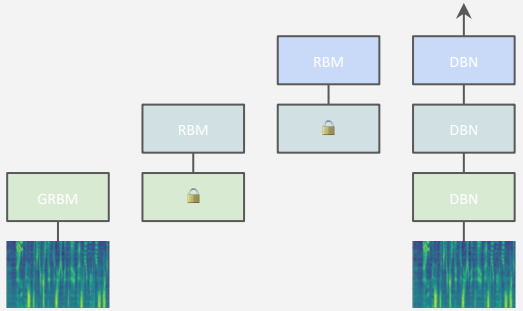
Recurrent NNs (+LSTMs)



h e l l o

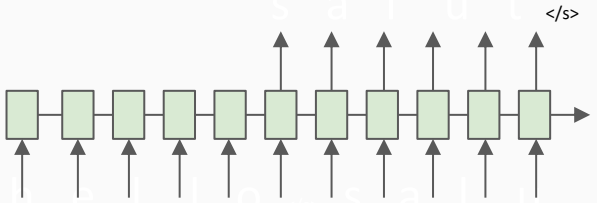
Speech

Deep Belief Nets (+non-DL)



Translation

Seq2Seq



RL

BC/GAIL

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \quad (17)$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

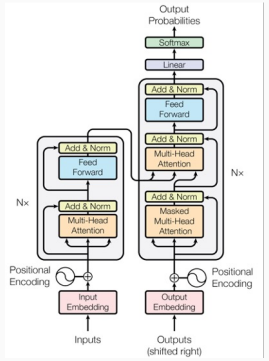
$$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta), \quad \text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \quad (18)$$

6: **end for**

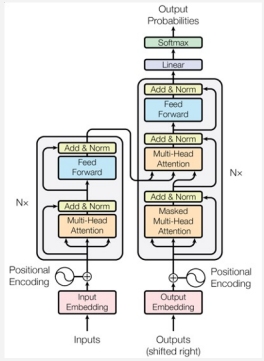
[1] CNN image CC-BY-SA by Aphex34 for Wikipedia https://commons.wikimedia.org/wiki/File:Typical_cnn.png
[2] RNN image CC-BY-SA by GChe for Wikipedia https://commons.wikimedia.org/wiki/File:The_LSTM_Cell.svg

[Source: Lucas Beyer, "Transformers."]

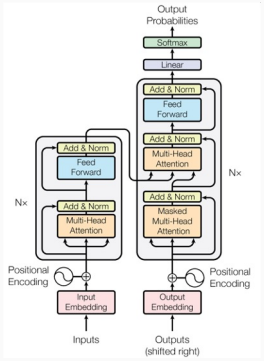
Computer Vision



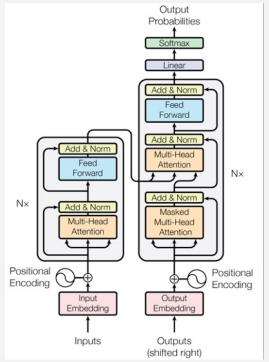
Natural Lang. Proc.



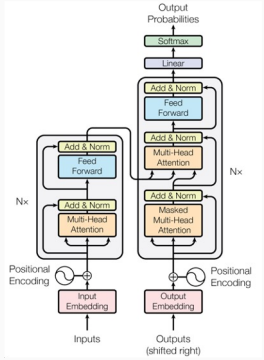
Reinf. Learning



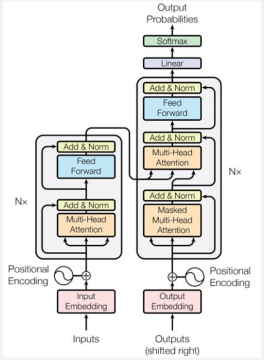
Speech



Translation



Graphs/Science



[Source: Lucas Beyer, "Transformers."]

Lab 3:

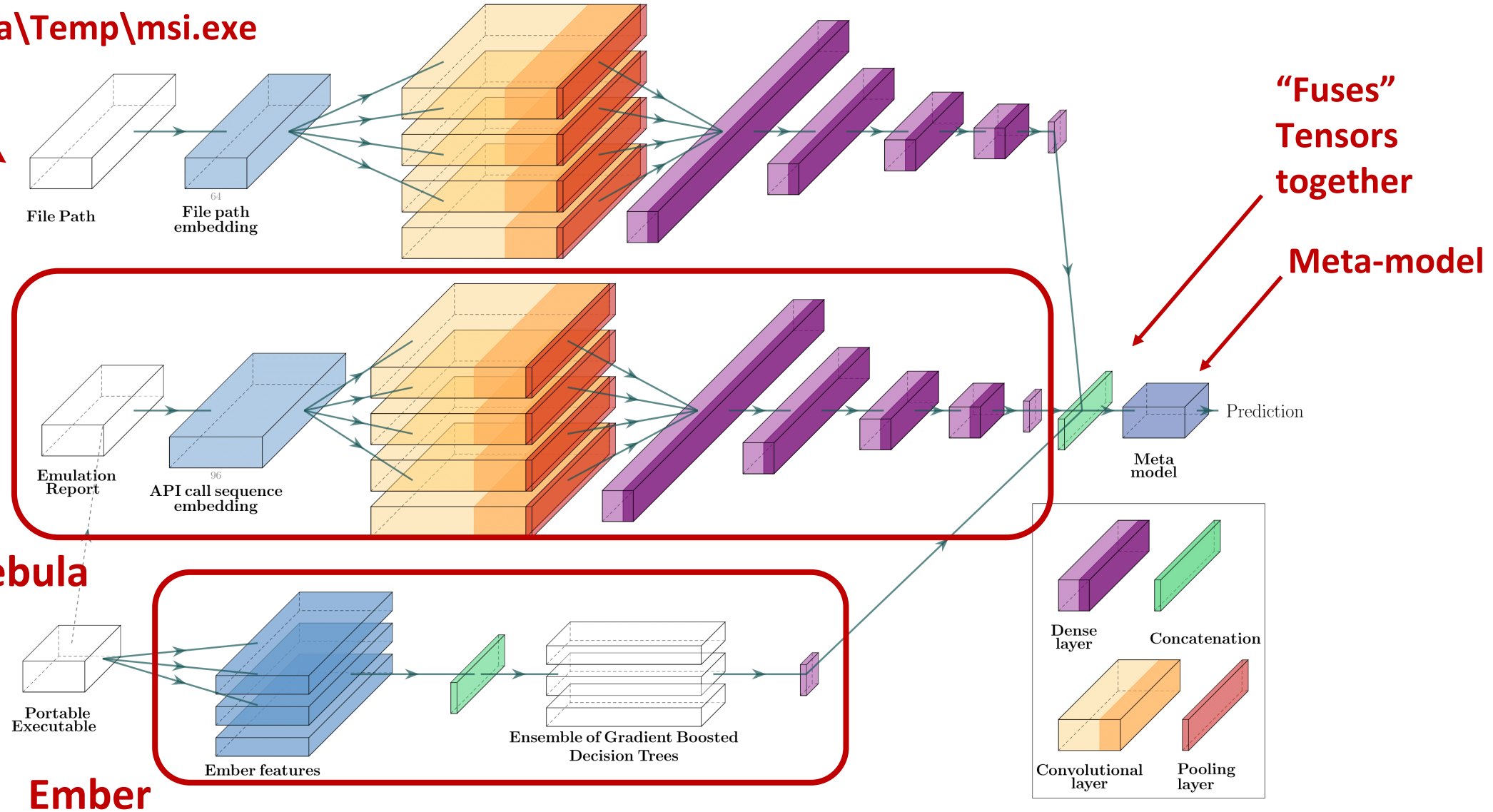
Dynamic Malware Analysis using Transformer Neural Network

aka.ms/malware-lab3

Hybrid Malware Analysis

End-to-End Monolithic Model

C:\...\AppData\Temp\msi.exe



"Fuses" Tensors together

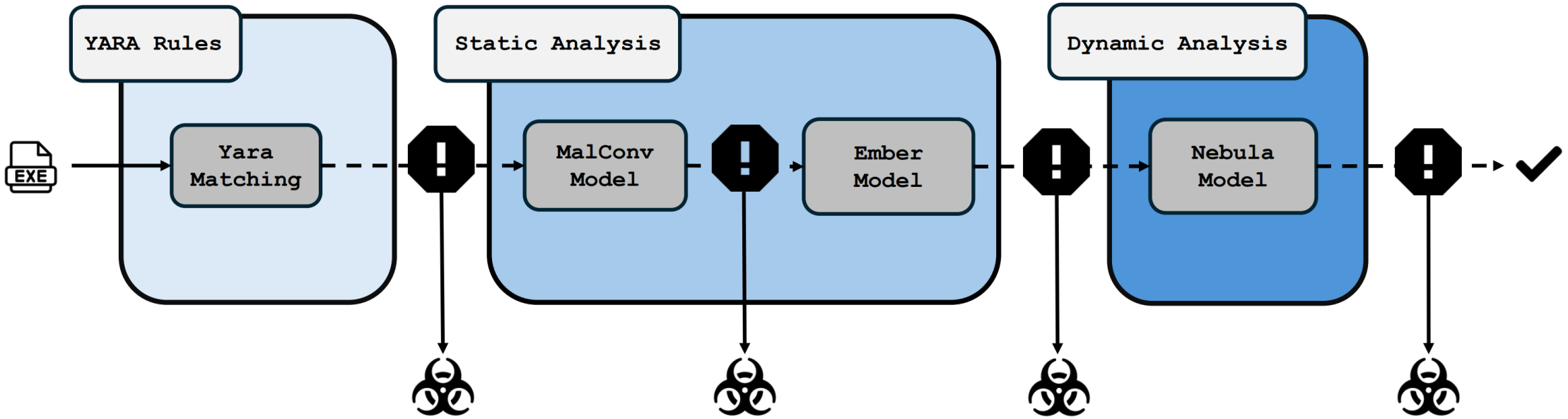
Meta-model

~Nebula

Ember

[Source: https://arxiv.org/abs/2208.12248]

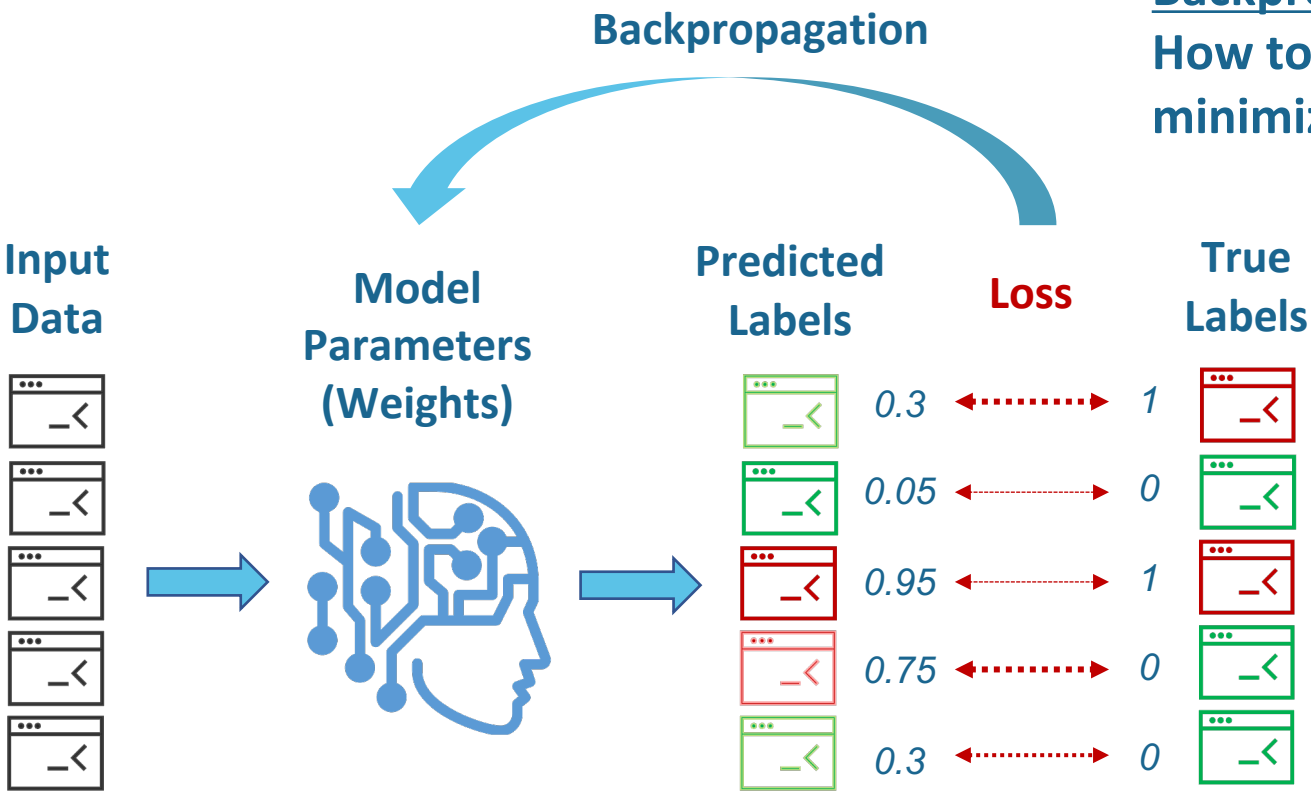
Modular Sequential Processing



[Source: https://arxiv.org/abs/2405.14478]

Training Machine Learning Model

Optimization: Loss



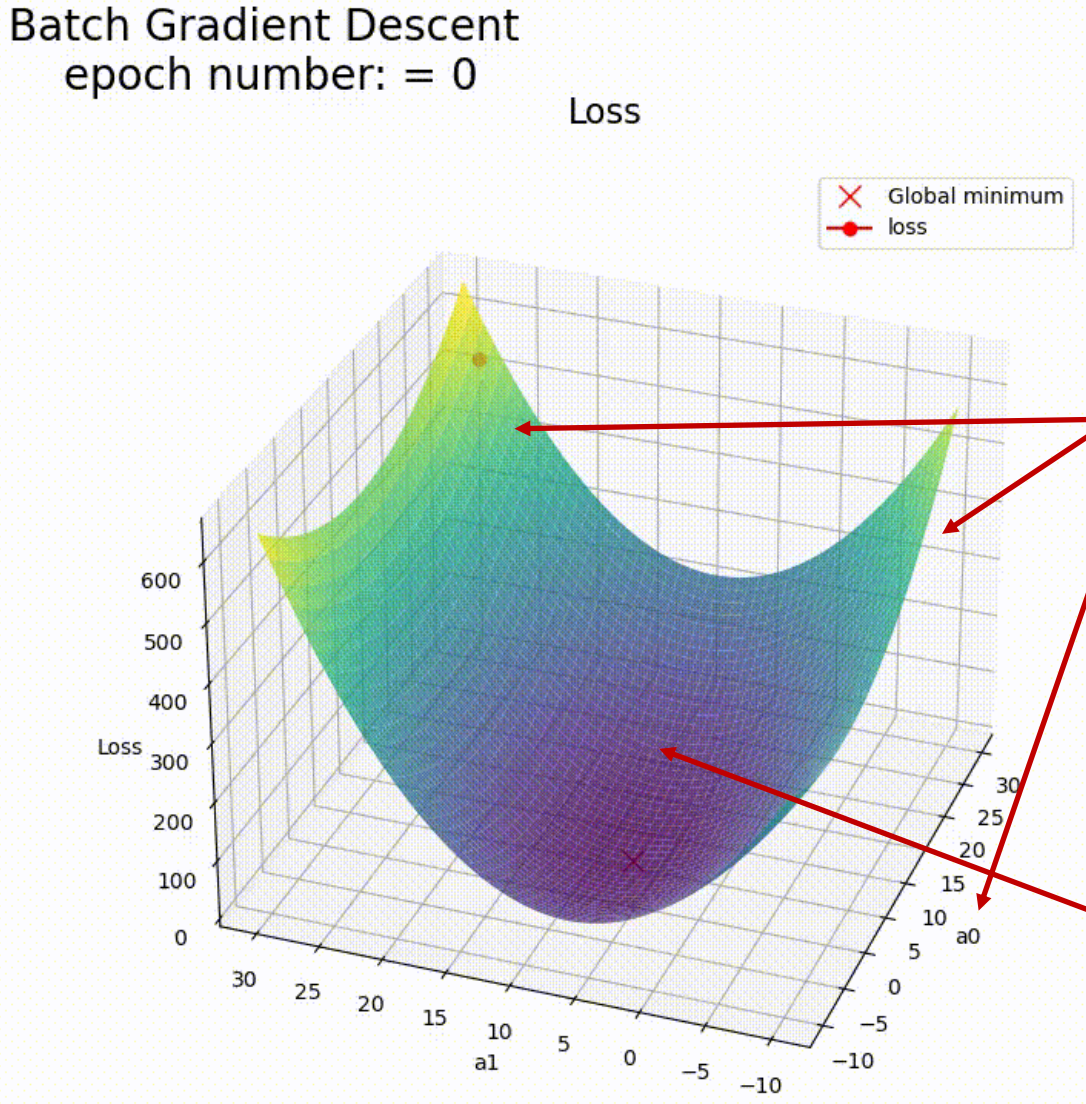
Backprop Intuition:
How to update model parameters to minimize loss during next pass?

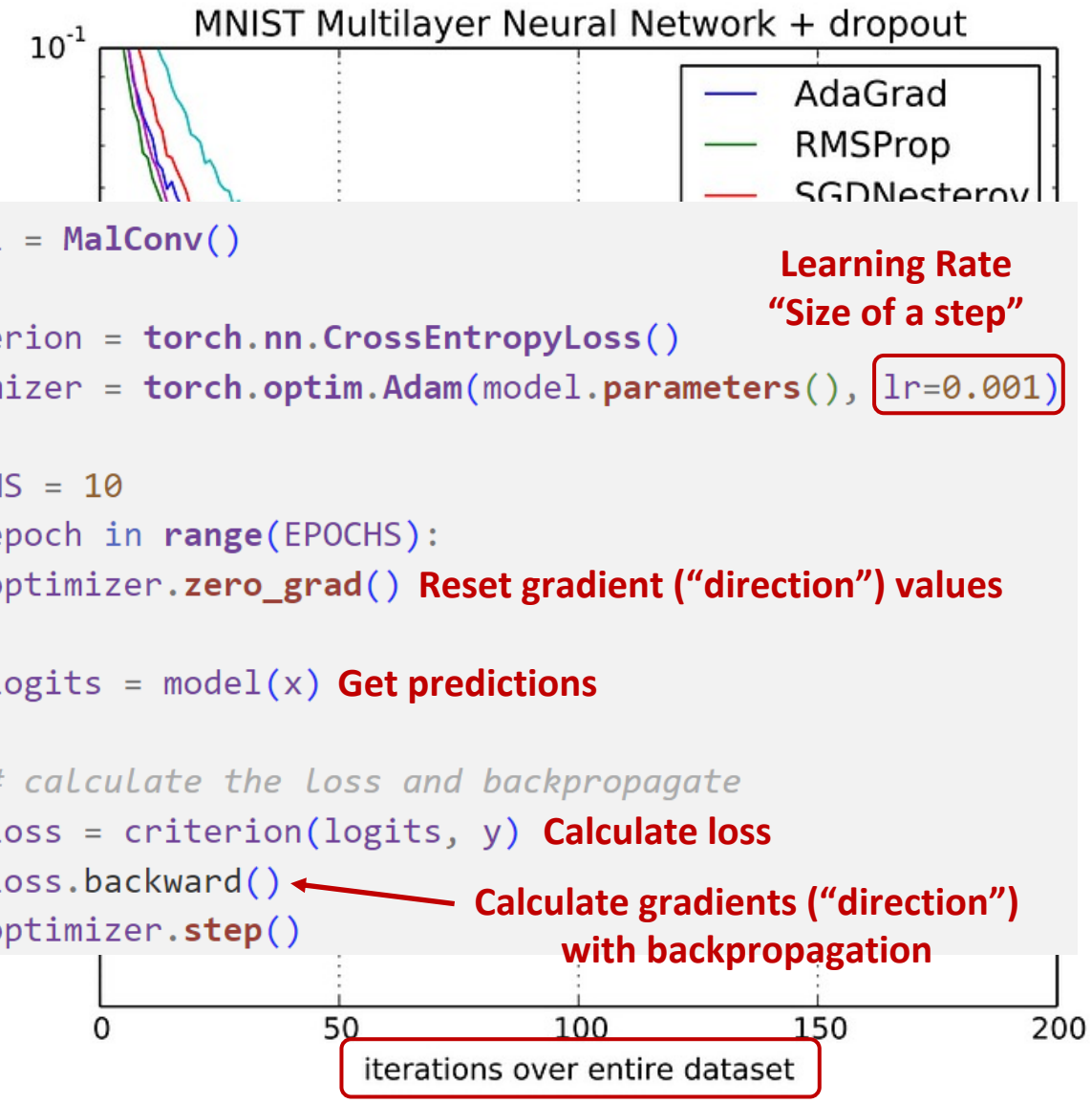
```
criterion = torch.nn.CrossEntropyLoss()

preds = torch.Tensor([[0.3, 0.05, 0.95, 0.75, 0.3]])
labels = torch.Tensor([[1, 0, 1, 0, 0]])

loss = criterion(preds, labels)
loss.item()
```

✓ 0.0s

```
3.0192079544067383
```

Optimization: Gradient Descend






```
model = MalConv()

criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

EPOCHS = 10
for epoch in range(EPOCHS):
    optimizer.zero_grad()

    logits = model(x)

    # calculate the loss and backpropagate
    loss = criterion(logits, y)
    loss.backward()
    optimizer.step()
```

Learning Rate
"Size of a step"

Reset gradient ("direction") values

Get predictions

Calculate loss

Calculate gradients ("direction") with backpropagation

[Source of Images]

Misuse this process? Sure!

Gradient -free and -full Attacks!

Part 2: **Attacking** AI-Based Malware Detector

- Intro to Adversarial Machine Learning
- Adversarial EXEmples
- Defenses against Adversarial Manipulations

Part 1 Survey

aka.ms/malware-survey1