CENTRE FOR SECURE INFORMATION TECHNOLOGIES

SECURING OUR DIGITAL TOMORROW

MALWARE AND MACHINE LEARNING

Dr. BooJoong Kang

Research Fellow , CSIT

23 Sept. 2020

QUEEN'S UNIVERSITY BELFAST

Who am I?

- BooJoong Kang (강부중)
- Research Fellow
- Centre of Secure Information Technologies
- Queen’s University Belfast, UK
- BS, MS, PhD from Hanyang University

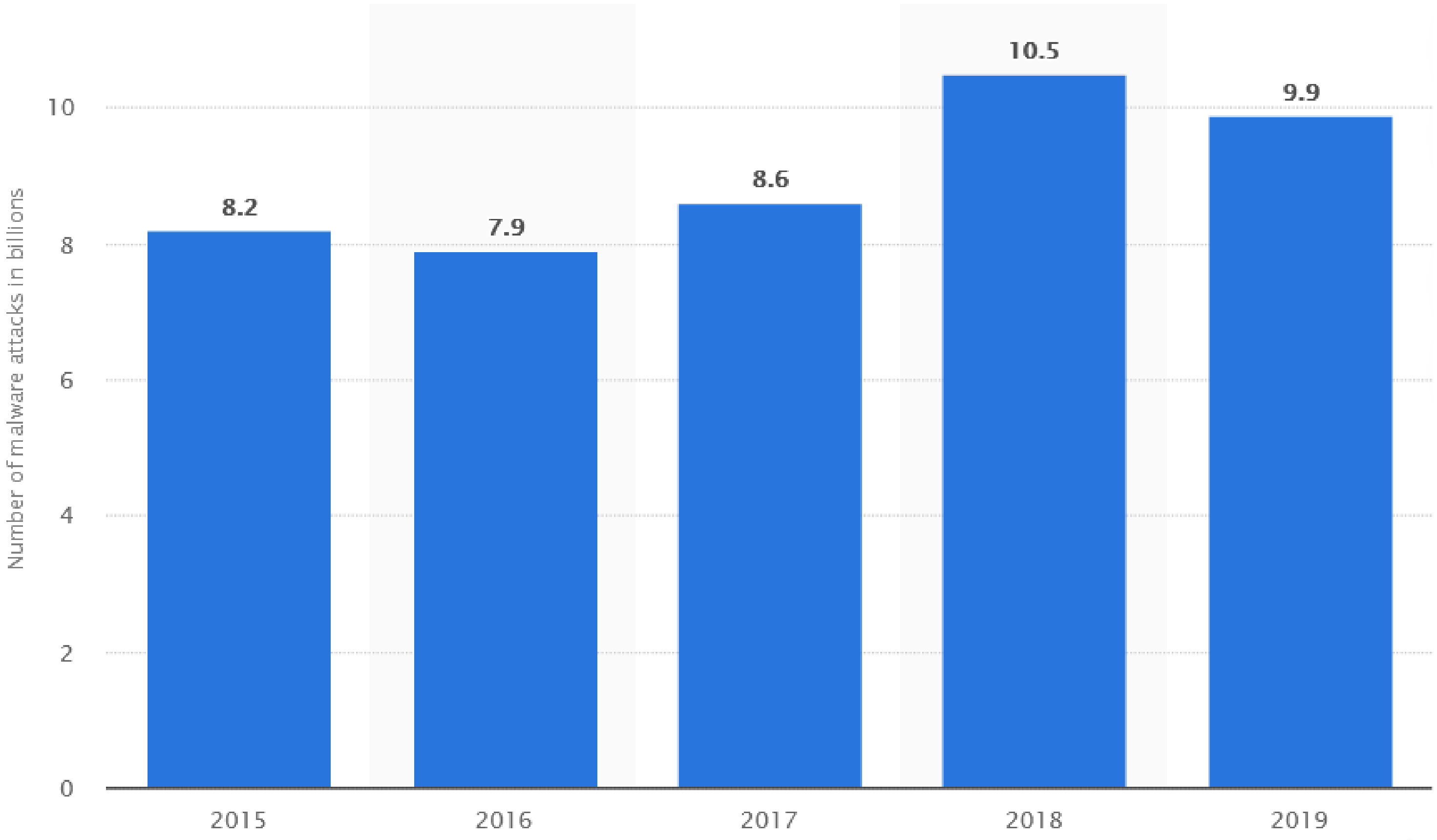
- Cybersecurity
  - Network Security and Malware Analysis

Outline

- Malware
- Malware Analysis
- Malware Detection
- ML-driven Malware Detection
- Adversarial Examples

Malware

- Malware (a portmanteau for malicious software) is any software intentionally designed to cause damage to a computer, server, client, or computer network



The first of many: Creeper

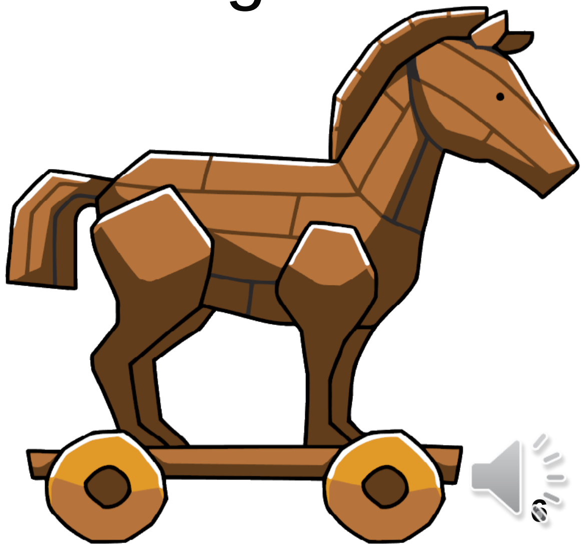
- Bob Thomas at BBN Technologies in 1971
- As a demonstration of mobile applications
  - Software that could automatically hop between computers on a network
- Find a computer, hop over to it and display “I’M THE CREEPER : CATCH ME IF YOU CAN.”

- Later, Thomas’s colleague Ray Tomlinson
- An update to the Creeper
  - Leave a copy of itself before moving onto the next one (self-replication)
- Invented another virus called the Reaper
  - Find infected computers and delete the Creeper

- Creeper: Worm
- Updated Creeper: Virus
- Reaper: Anti-Virus

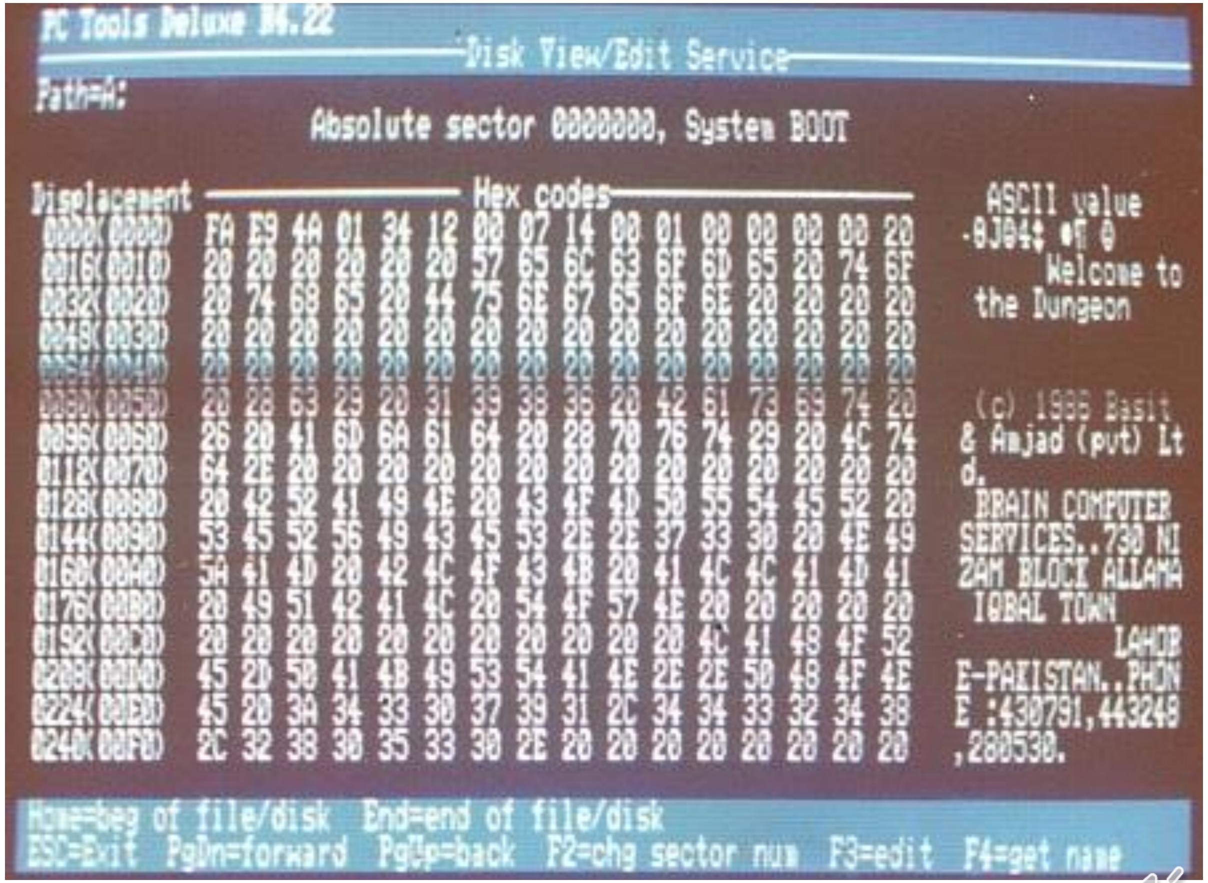
The first Trojan: ANIMAL

- John Walker, 1975
- "20 questions" program that guess the animal
- Become popular and start to occupy a lot of his time to copy it
- Subroutine, called PERVADE, saves copies of ANIMAL to any user-accessible directories that would, while the user was answering the questions
- Many of these directories were shared between offices
  - so the Trojan spread via that vector as well



The first PC virus: Brain

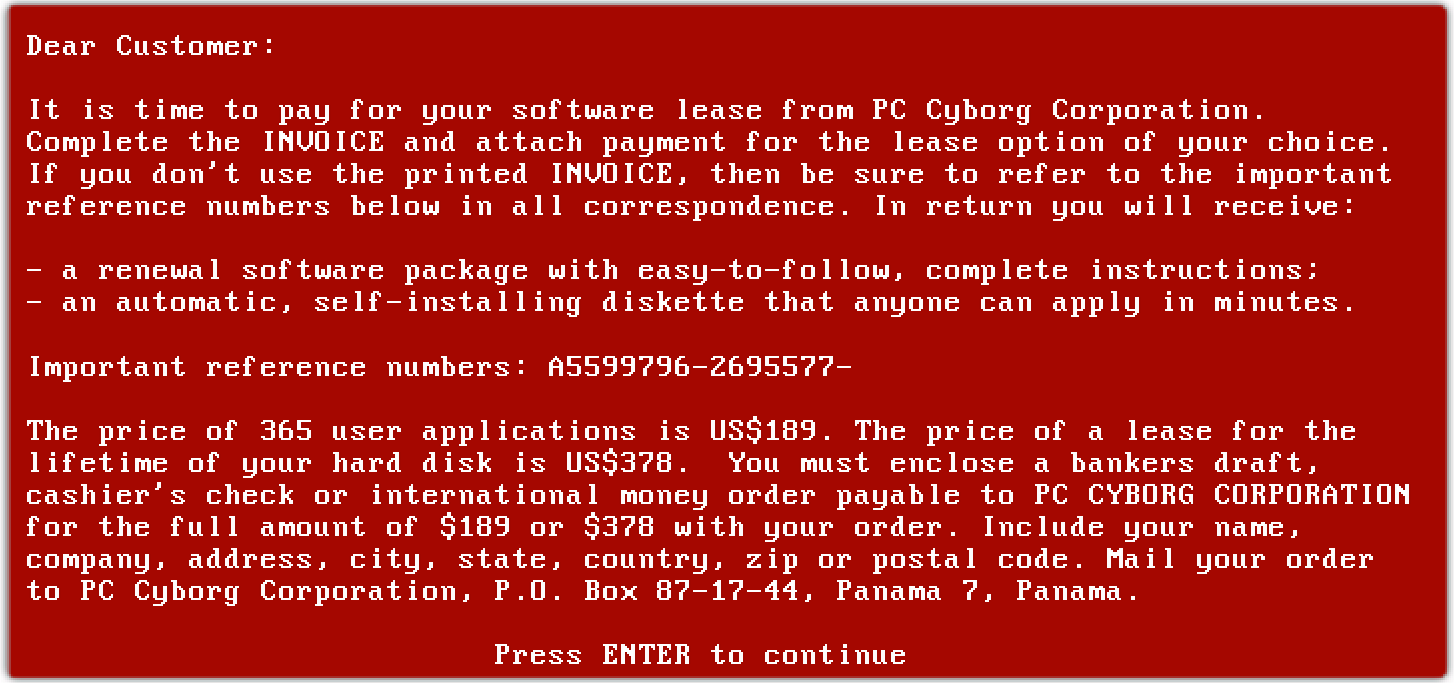
- Basit Farooq Alvi and Amjad Farooq Alvi, two brothers, in 1986
- Infect IBM PC, slow down floppy disks and take up a good chunk of memory
- Designed to protect their medical software from piracy
- A message in the code that included their address and phone numbers



```
PC Tools Deluxe R4.22
                        Disk View/Edit Service
Path=A:
                Absolute sector 0000000, System BOOT

Displacement ----------- Hex codes ------------------------        ASCII value
 0000(0000)  FA E9 4A 01 34 12 00 07 14 00 01 00 00 00 00 20   -8J04‡ ●¶ @
 0016(0010)  20 20 20 20 20 20 57 65 6C 63 6F 6D 65 20 74 6F         Welcome to
 0032(0020)  20 74 68 65 20 44 75 6E 67 65 6F 6E 20 20 20 20    the Dungeon
 0048(0030)  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
 0064(0040)  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
 0080(0050)  20 28 63 29 20 31 39 38 36 20 42 61 73 69 74 20    (c) 1986 Basit
 0096(0060)  26 20 41 6D 6A 61 64 20 28 70 76 74 29 20 4C 74   & Amjad (pvt) Lt
 0112(0070)  64 2E 20 20 20 20 20 20 20 20 20 20 20 20 20 20   d.
 0128(0080)  20 42 52 41 49 4E 20 43 4F 4D 50 55 54 45 52 20    BRAIN COMPUTER
 0144(0090)  53 45 52 56 49 43 45 53 2E 2E 37 33 30 20 4E 49   SERVICES..730 NI
 0160(00A0)  5A 41 4D 20 42 4C 4F 43 4B 20 41 4C 4C 41 4D 41   ZAM BLOCK ALLAMA
 0176(00B0)  20 49 51 42 41 4C 20 54 4F 57 4E 20 20 20 20 20    IQBAL TOWN
 0192(00C0)  20 20 20 20 20 20 20 20 20 20 20 4C 41 48 4F 52              LAHOR
 0208(00D0)  45 2D 50 41 4B 49 53 54 41 4E 2E 2E 50 48 4F 4E   E-PAKISTAN..PHON
 0224(00E0)  45 20 3A 34 33 30 37 39 31 2C 34 34 33 32 34 38   E :430791,443248
 0240(00F0)  2C 32 38 30 35 33 30 2E 20 20 20 20 20 20 20 20   ,280530.

Home=beg of file/disk  End=end of file/disk
ESC=Exit  PgDn=forward  PgUp=back  F2=chg sector num  F3=edit  F4=get name
```

The first ransomware: AIDS Trojan

- Joseph Popp in 1989
- Once installed, it counts all the times you booted up the computer
- Once you’d restarted 90 times
  - hide all files, rendering them inaccessible
  - demand you send a letter to an address with $189 to “renew your license.”
- He was eventually caught but let off the hook after being declared mentally unfit and agreeing to donate the profits to AIDS research

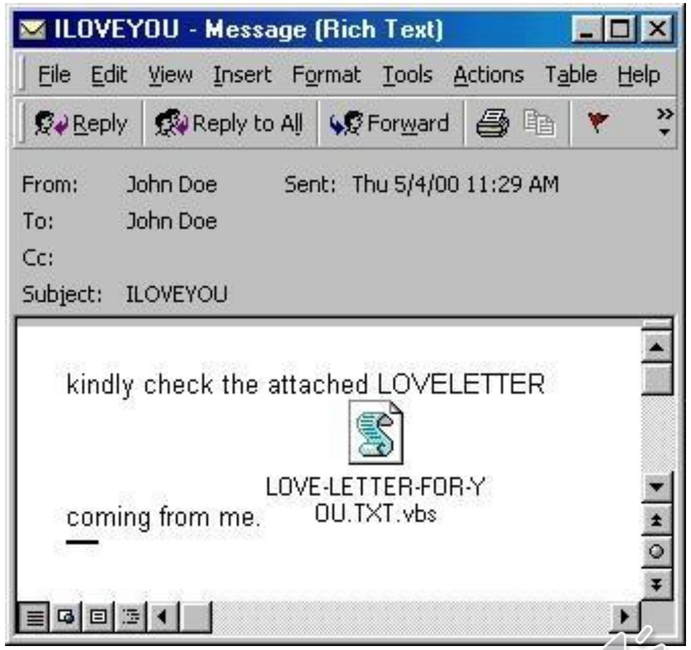


```
Dear Customer:

It is time to pay for your software lease from PC Cyborg Corporation.
Complete the INVOICE and attach payment for the lease option of your choice.
If you don't use the printed INVOICE, then be sure to refer to the important
reference numbers below in all correspondence. In return you will receive:

- a renewal software package with easy-to-follow, complete instructions;
- an automatic, self-installing diskette that anyone can apply in minutes.

Important reference numbers: A5599796-2695577-

The price of 365 user applications is US$189. The price of a lease for the
lifetime of your hard disk is US$378.  You must enclose a bankers draft,
cashier's check or international money order payable to PC CYBORG CORPORATION
for the full amount of $189 or $378 with your order. Include your name,
company, address, city, state, country, zip or postal code. Mail your order
to PC Cyborg Corporation, P.O. Box 87-17-44, Panama 7, Panama.

                          Press ENTER to continue
```

The first social engineering attack: Melissa

- In 1999, the first email-based viruses and the first to ever use social engineering
- Spread by sending emails to email contacts
  - The headline: "Important Message From ABC“
  - The body text: "Here is that document you asked for ... don't show anyone else ;-)“
- Infected DOC file includes a list of 80 pornographic websites as well as usernames and passwords
- Also other DOC files found on the computer, which have also been infected
  - classified or private files would be shared with friends, family, or work associates

- ILOVEYOU (Love Letter Worm) in 2000
- An email disguised as a love letter
- The attached text file overwrite files, steal user data such, then send itself to everyone on your email contact list
- Compromised an estimated 45 million computers around the world (about 10% of all connected computers) and caused over 8 billion dollars in damages



From: John Doe Sent: Thu 5/4/00 11:29 AM

To: John Doe

Cc:

Subject: ILOVEYOU

kindly check the attached LOVELETTER coming from me.

LOVE-LETTER-FOR-YOU.TXT.vbs

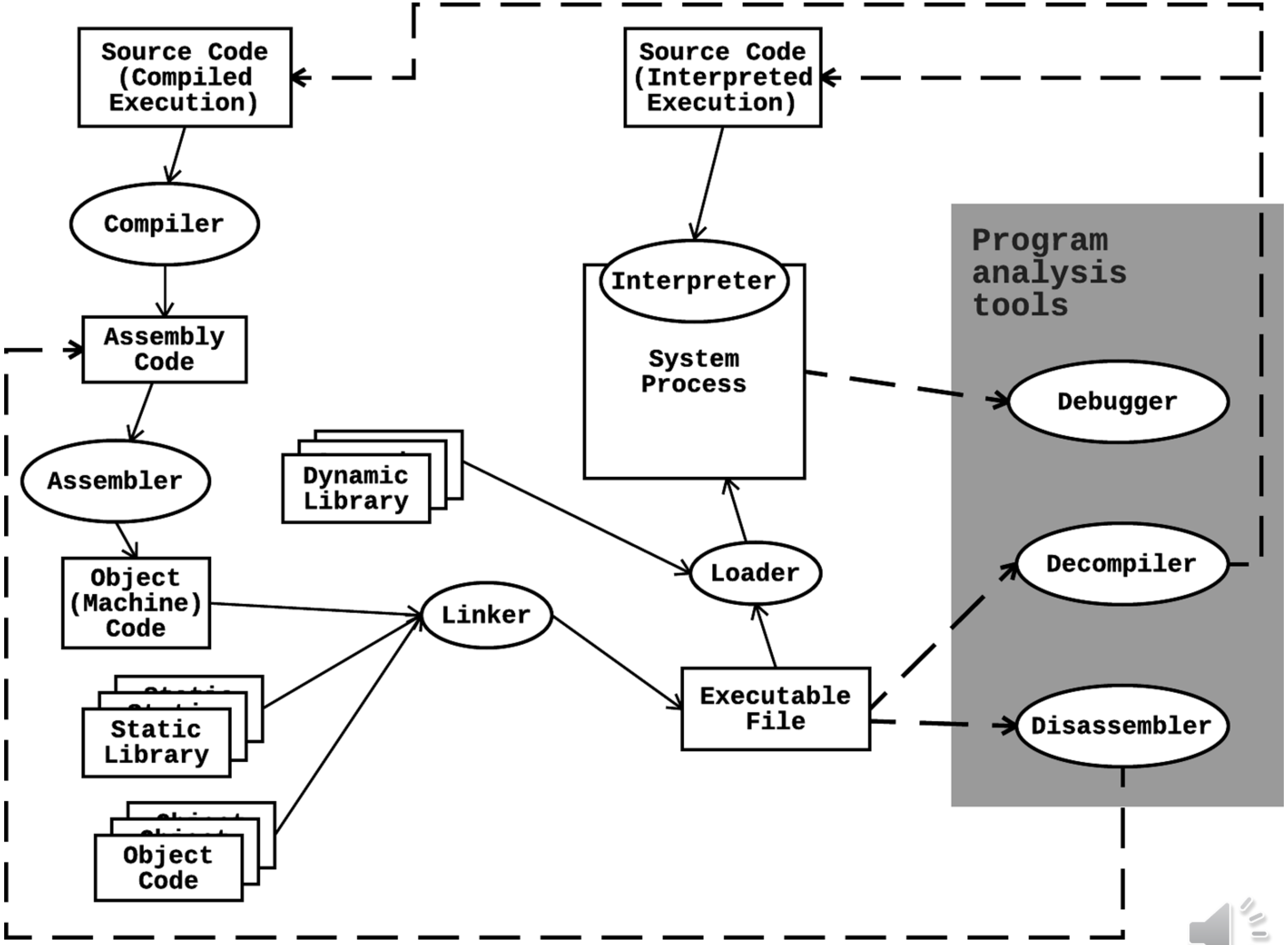
Malware in the 21st Century

- Between 2000 and 2010, malware grew significantly, both in number and in how fast infections spread
  - A dramatic increase in malware toolkits, including the now infamous Sony rootkit and Crimeware kits
  - SQL injection attacks became a leading threat, claiming popular victims
    - SQL Slammer (2003) infected nearly 75,000 computers in ten minutes
  - Conficker Worm (2008) caused some of the worst damage seen since Slammer appeared in 2003
- Between 2010 and the present time, significant evolution in the sophistication of malware
  - Organized crime and state sponsors upped the game dramatically with large, well-funded development teams
  - Developing advanced malware with evasion tactics that outsmart conventional anti-malware systems
  - Infiltrating factories and military systems became a common reality, and the monetization of malware grew rapidly with dramatic growth in ransomware and other illegal schemes
  - Stuxnet Worm (2010) was designed with the express purpose of attacking Iran’s nuclear program and included the ability to impact hardware as well as software
    - one of the most resource-intensive bits of malware created to date
  - Zeus Trojan (2011) is one of the most successful pieces of botnet software in the world, impacting millions of machines
  - WannaCry Ransomware (2017) locked people out of their data and demanded they pay a ransom or lose everything
    - Affected at least 150 countries, including hospitals, banks, telecommunications companies, warehouses, and many other industries

CENTRE FOR SECURE INFORMATION TECHNOLOGIES

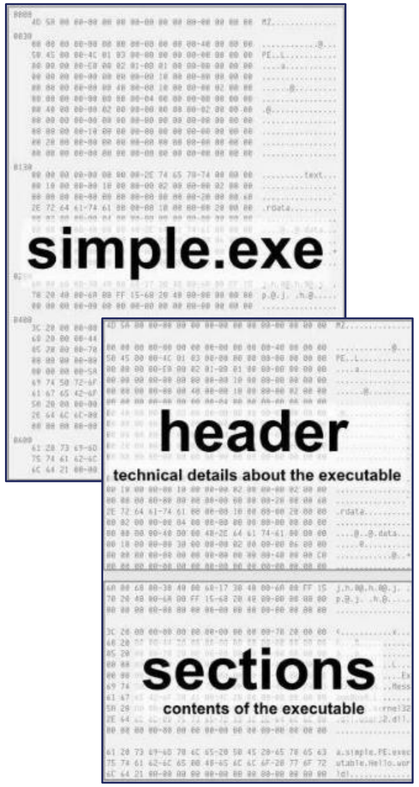
Malware Analysis

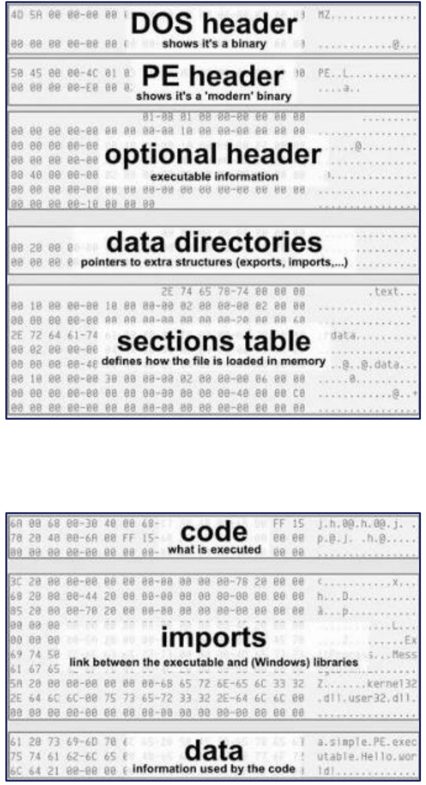
- To understand
  - How it works
  - How to identify it
  - How to defeat or eliminate it

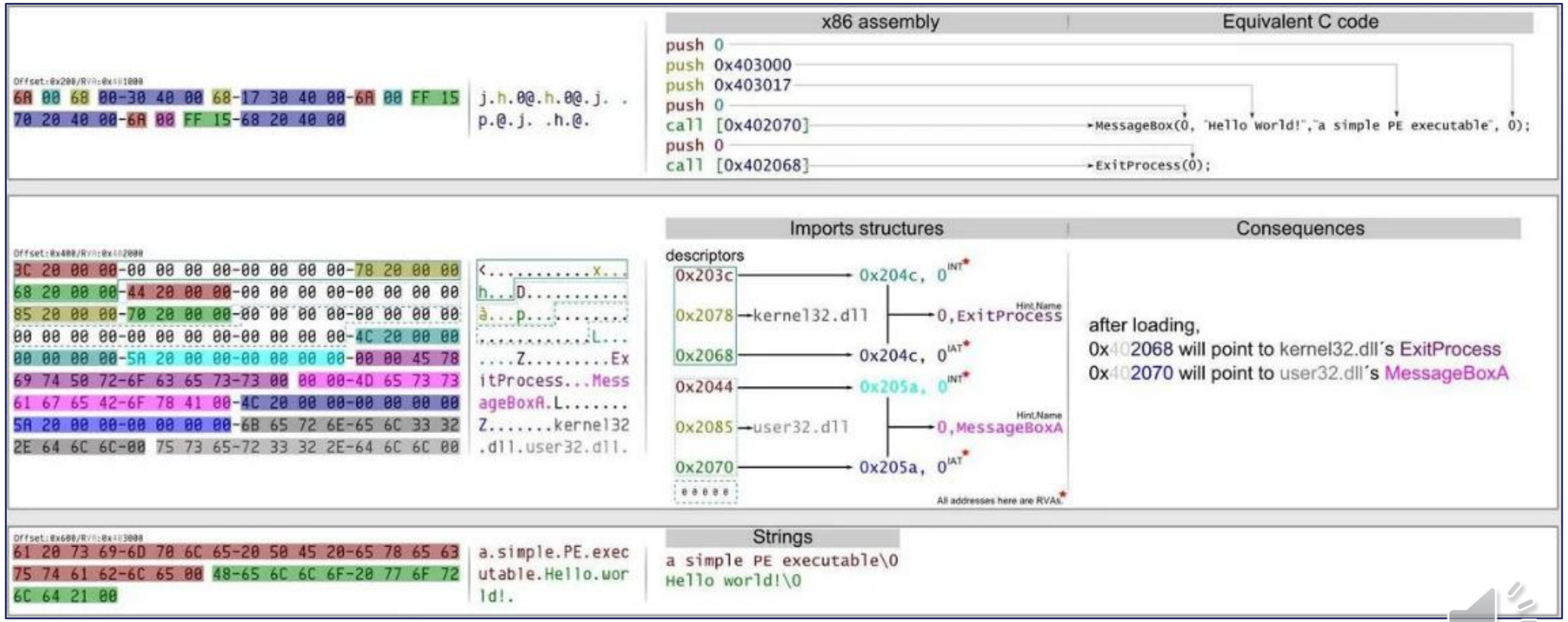


Static Analysis

- Static Analysis examines the file itself
  - Reverse-engineering with a disassembler or a decompiler


simple.exe
header
technical details about the executable
sections
contents of the executable


DOS header
shows it's a binary
PE header
shows it's a 'modern' binary
optional header
executable information
data directories
pointers to extra structures (exports, imports,...)
sections table
defines how the file is loaded in memory
code
what is executed
imports
link between the executable and (Windows) libraries
data
information used by the code



```
Offset:0x200/RVA:0x401000
6A 00 68 00-30 40 00 68-17 30 40 00-6A 00 FF 15   j.h.0@.h.0@.j. .
70 20 40 00-6A 00 FF 15-68 20 40 00               p.@.j. .h.@.
```

x86 assembly	Equivalent C code

`push 0`
`push 0x403000`
`push 0x403017`
`push 0`
`call [0x402070]`
	`MessageBox(0, "Hello world!","a simple PE executable", 0);`

`push 0`
`call [0x402068]`
	`ExitProcess(0);`

```
Offset:0x400/RVA:0x402000
3C 20 00 00-00 00 00 00-00 00 00 00-78 20 00 00   <...........x...
68 20 00 00-44 20 00 00-00 00 00 00-00 00 00 00   h...D...........
85 20 00 00-70 20 00 00-00 00 00 00-00 00 00 00   à...p...........
00 00 00 00-00 00 00 00-00 00 00 00-4C 20 00 00   ............L...
00 00 00 00-5A 20 00 00-00 00 00 00-00 00 45 78   ....Z.........Ex
69 74 50 72-6F 63 65 73-73 00 00 00-4D 65 73 73   itProcess...Mess
61 67 65 42-6F 78 41 00-4C 20 00 00-00 00 00 00   ageBoxA.L.......
5A 20 00 00-00 00 00 00-6B 65 72 6E-65 6C 33 32   Z.......kernel32
2E 64 6C 6C-00 75 73 65-72 33 32 2E-64 6C 6C 00   .dll.user32.dll.
```

after loading,
0x402068 will point to kernel32.dll´s ExitProcess
0x402070 will point to user32.dll´s MessageBoxA

```
Offset:0x600/RVA:0x403000
61 20 73 69-6D 70 6C 65-20 50 45 20-65 78 65 63   a.simple.PE.exec
75 74 61 62-6C 65 00 48-65 6C 6C 6F-20 77 6F 72   utable.Hello.wor
6C 64 21 00                                       ld!.
```

Strings

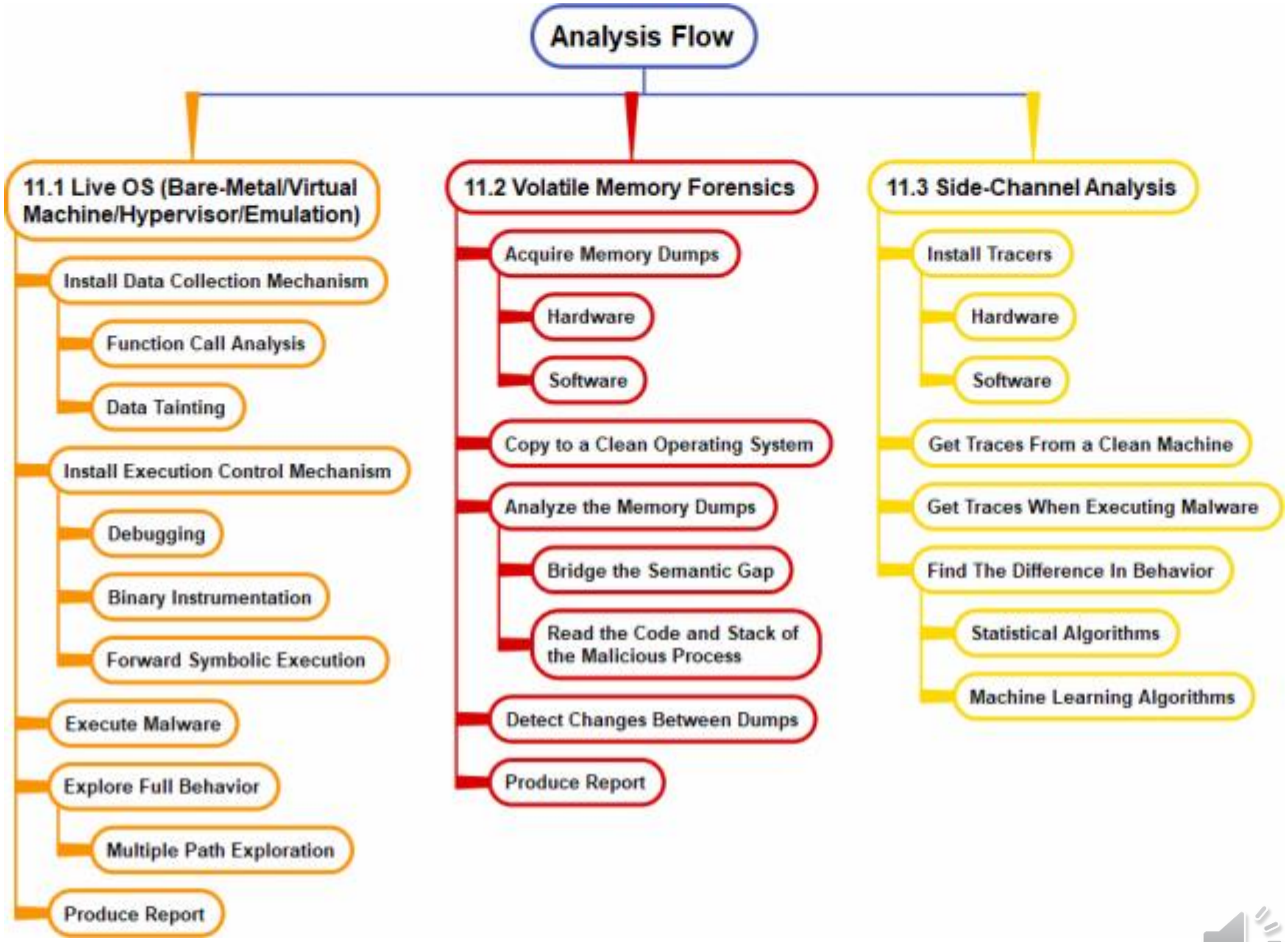
```
a simple PE executable\0
Hello world!\0
```

Features from Static Analysis

- Strings
- Byte-patterns
- Code
  - Functions
    - Function Call Graph
  - Instructions
    - Opcode
    - Control Flow Graph
  - APIs
- Combinations of Features can characterise programs

Dynamic Analysis

- Examine the runtime behaviour of programs
  - the execution and effects of programs with VMs or Debuggers



Analysis Flow

- 11.1 Live OS (Bare-Metal/Virtual Machine/Hypervisor/Emulation)
  - Install Data Collection Mechanism
    - Function Call Analysis
    - Data Tainting
  - Install Execution Control Mechanism
    - Debugging
    - Binary Instrumentation
    - Forward Symbolic Execution
  - Execute Malware
  - Explore Full Behavior
    - Multiple Path Exploration
  - Produce Report
- 11.2 Volatile Memory Forensics
  - Acquire Memory Dumps
    - Hardware
    - Software
  - Copy to a Clean Operating System
  - Analyze the Memory Dumps
    - Bridge the Semantic Gap
    - Read the Code and Stack of the Malicious Process
  - Detect Changes Between Dumps
  - Produce Report
- 11.3 Side-Channel Analysis
  - Install Tracers
    - Hardware
    - Software
  - Get Traces From a Clean Machine
  - Get Traces When Executing Malware
  - Find The Difference In Behavior
    - Statistical Algorithms
    - Machine Learning Algorithms

Features from Dynamic Analysis

- Executed Code
  - Function Calls
    - Called FCG
  - Executed Instructions
    - Executed CFG
  - API Calls
- Memory/Register Usages
- Network Traffic
- Any other behaviours

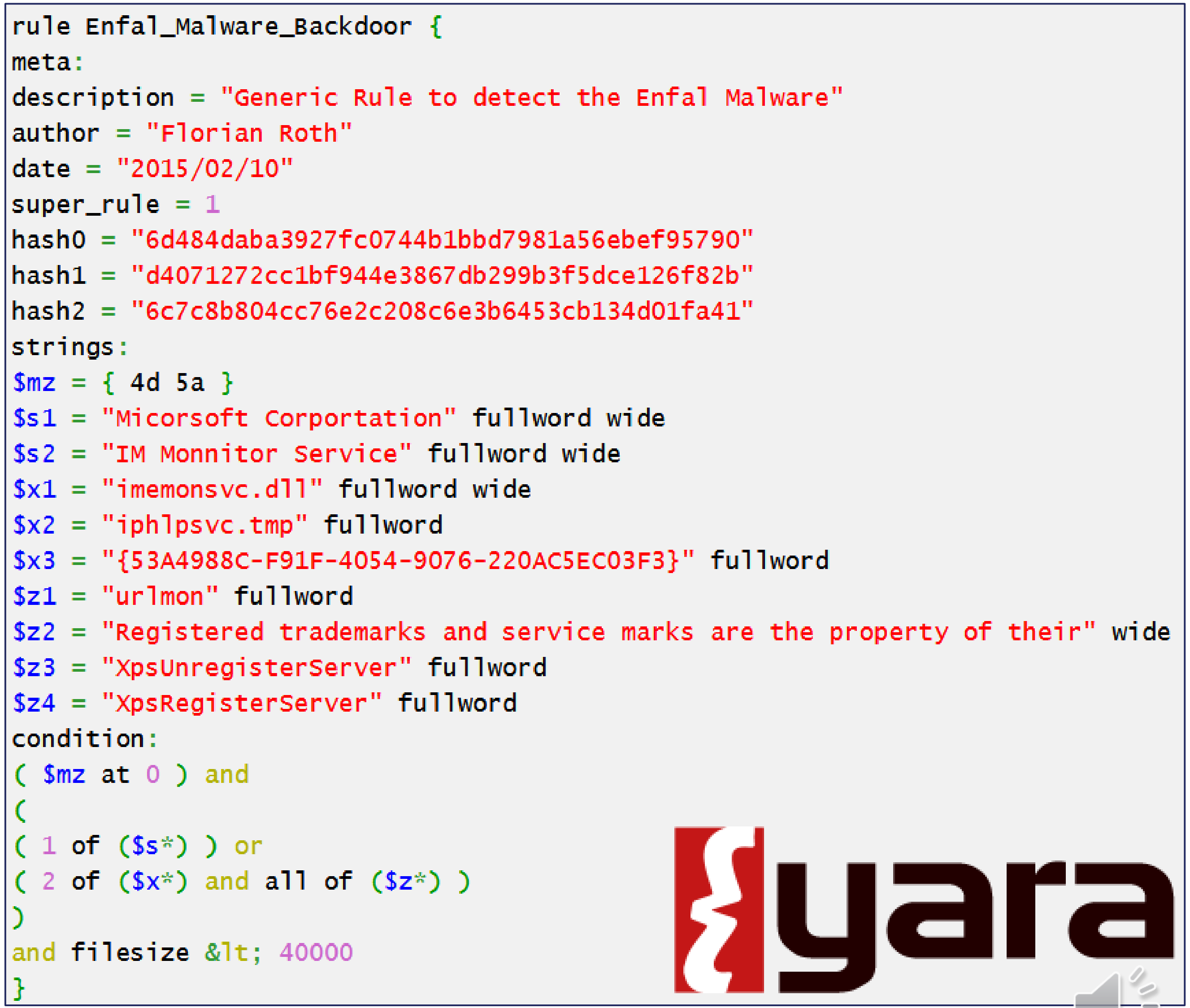
Malware Detection

- Identify Unique Properties of Malware from Static/Dynamic Analysis
- Confusion Matrix in Malware Detection
  - Low False Positive is more important in Malware Detection

	Malware	Benign
Detected as Malware	True Positive	False Positive
Detected as Benign	False Negative	True Negative

Malware Signatures

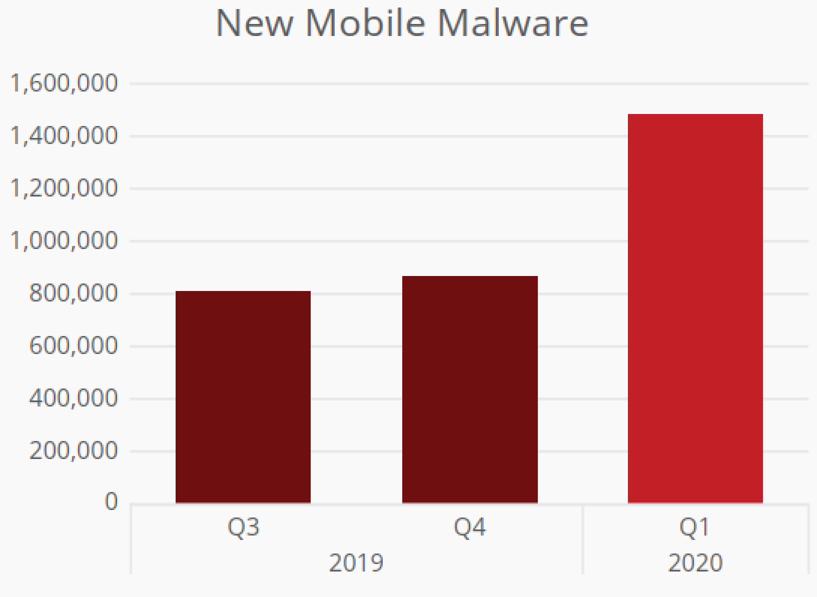
- Unique Properties of Malware
  - Hash
  - Strings
  - Byte-patterns
- Fast & Low False Positives
- Easy to Evade
  - a padding yields a different hash
  - variants, zero-day malware
    - continuous signature updates

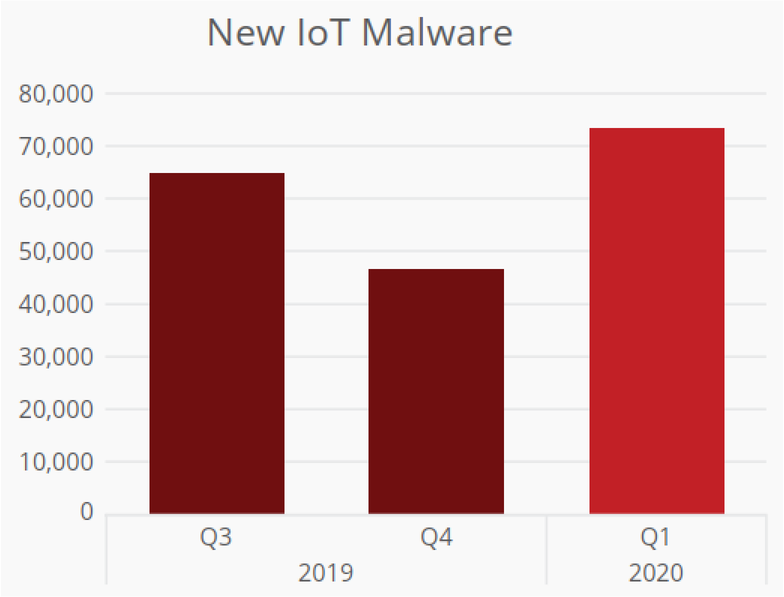


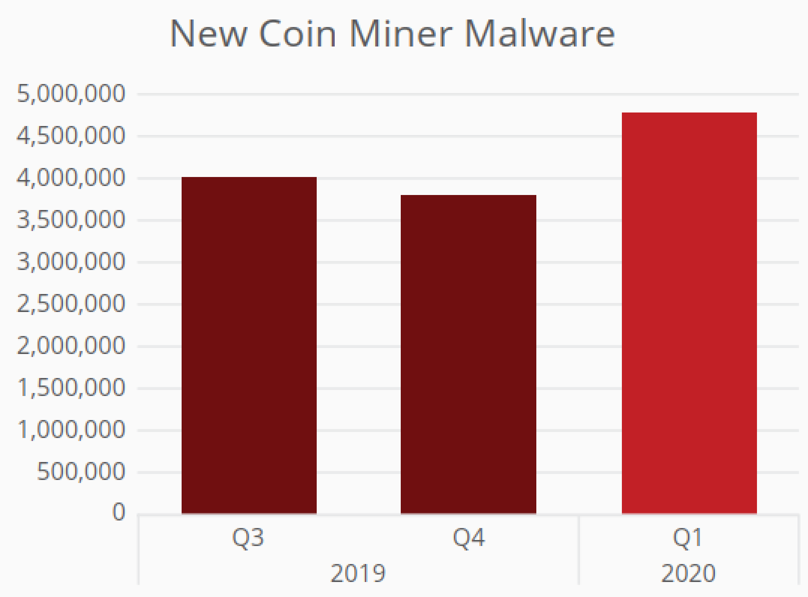
```
rule Enfal_Malware_Backdoor {
meta:
description = "Generic Rule to detect the Enfal Malware"
author = "Florian Roth"
date = "2015/02/10"
super_rule = 1
hash0 = "6d484daba3927fc0744b1bbd7981a56ebef95790"
hash1 = "d4071272cc1bf944e3867db299b3f5dce126f82b"
hash2 = "6c7c8b804cc76e2c208c6e3b6453cb134d01fa41"
strings:
$mz = { 4d 5a }
$s1 = "Micorsoft Corportation" fullword wide
$s2 = "IM Monnitor Service" fullword wide
$x1 = "imemonsvc.dll" fullword wide
$x2 = "iphlpsvc.tmp" fullword
$x3 = "{53A4988C-F91F-4054-9076-220AC5EC03F3}" fullword
$z1 = "urlmon" fullword
$z2 = "Registered trademarks and service marks are the property of their" wide
$z3 = "XpsUnregisterServer" fullword
$z4 = "XpsRegisterServer" fullword
condition:
( $mz at 0 ) and
(
( 1 of ($s*) ) or
( 2 of ($x*) and all of ($z*) )
)
and filesize &lt; 40000
}
```

Why ML?

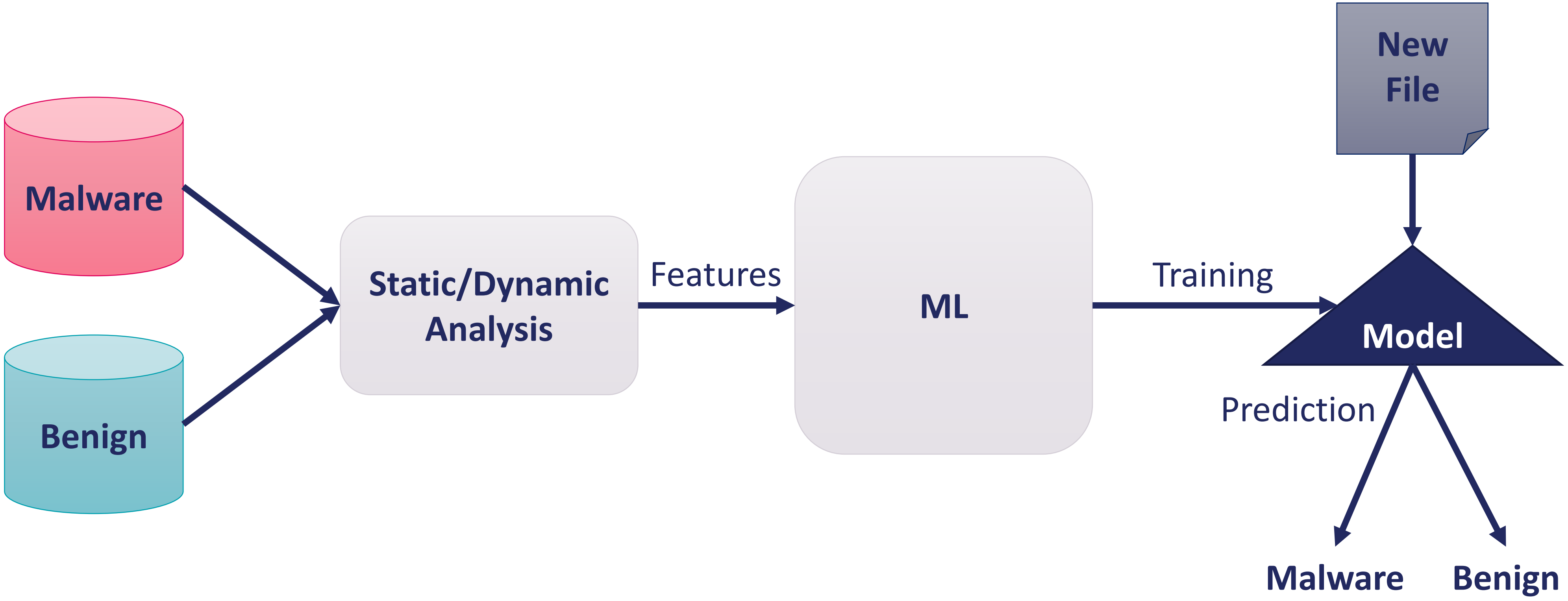
- Manual analysis for signatures can’t cope with the number of new malware





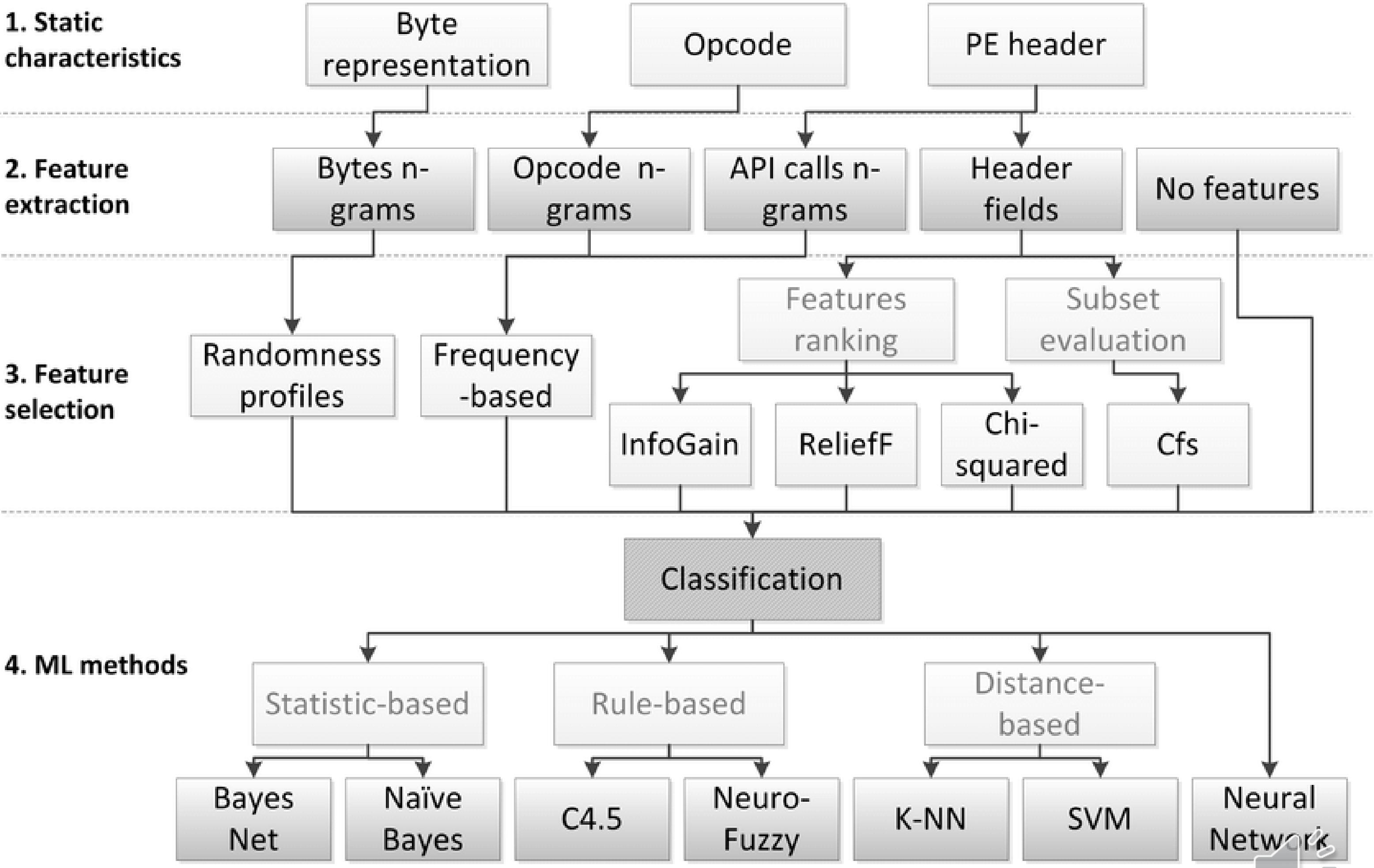


ML-driven Malware Detection



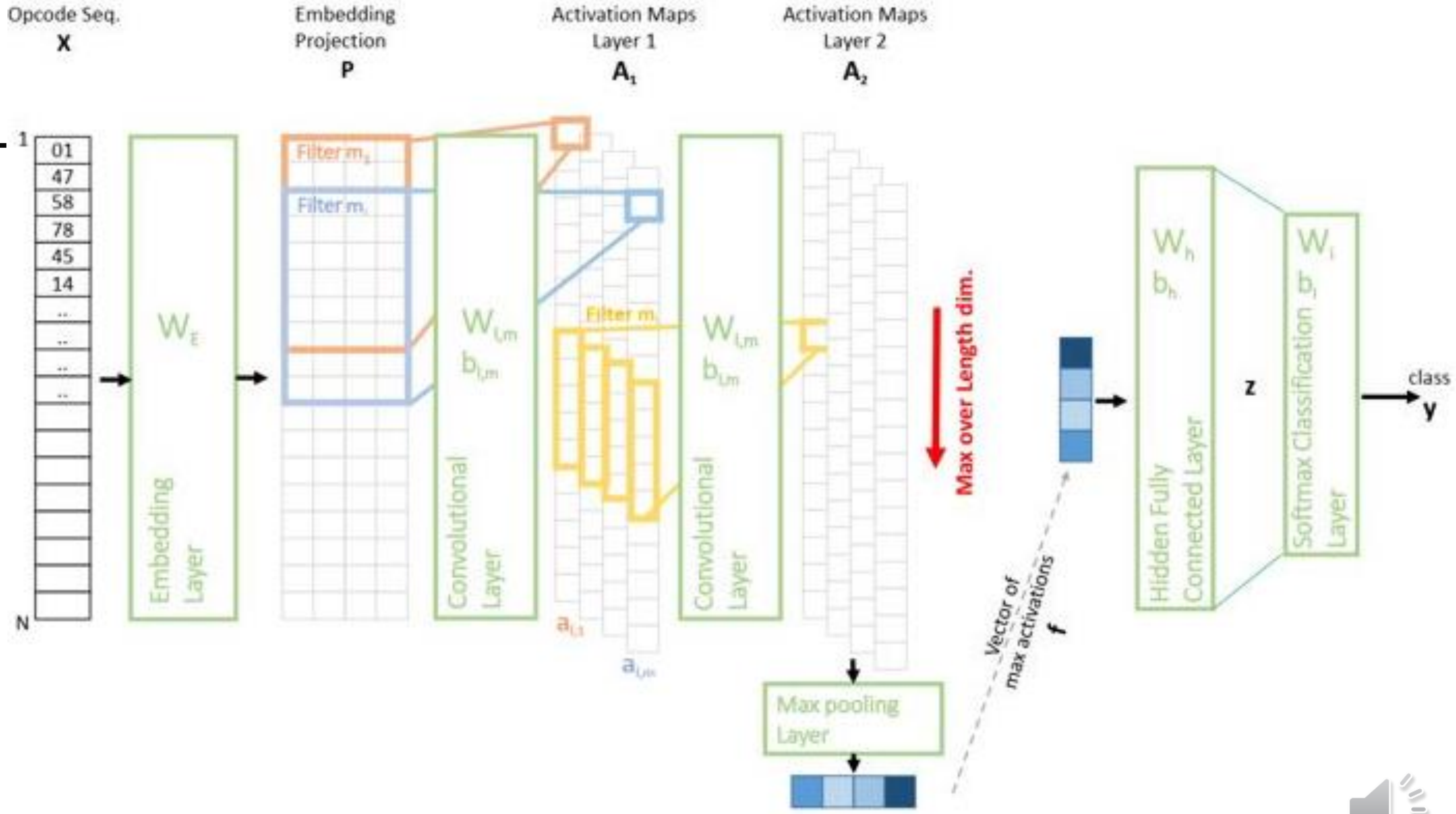
ML with Sequence Data

- N-grams
  - unigram of “COLD”
    - C, O, L, D
  - bigram of “COLD”
    - CO, OL, LD
  - n-gram (n=4) of “COLD”
    - COLD
- Word2vec
  - n-grams into a numerical vector
  - existence or frequency



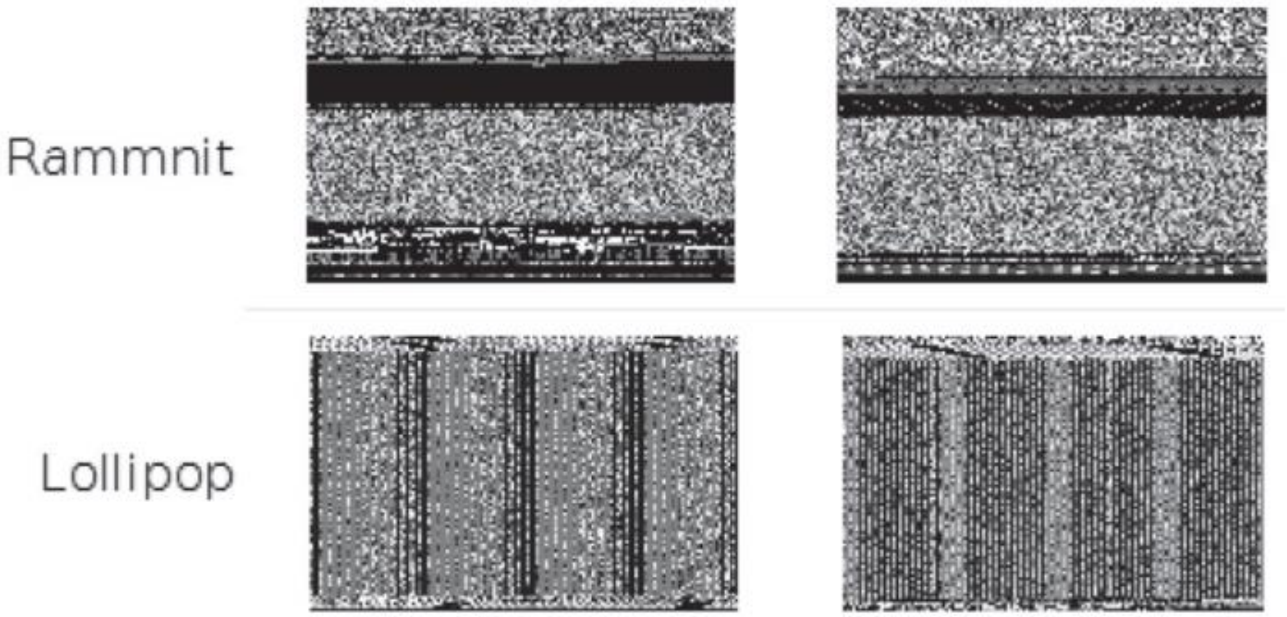
Deep Learning in Malware Detection

- Feature Extraction and Selection by DL
- No need to enumerate n-grams of Opcode sequences



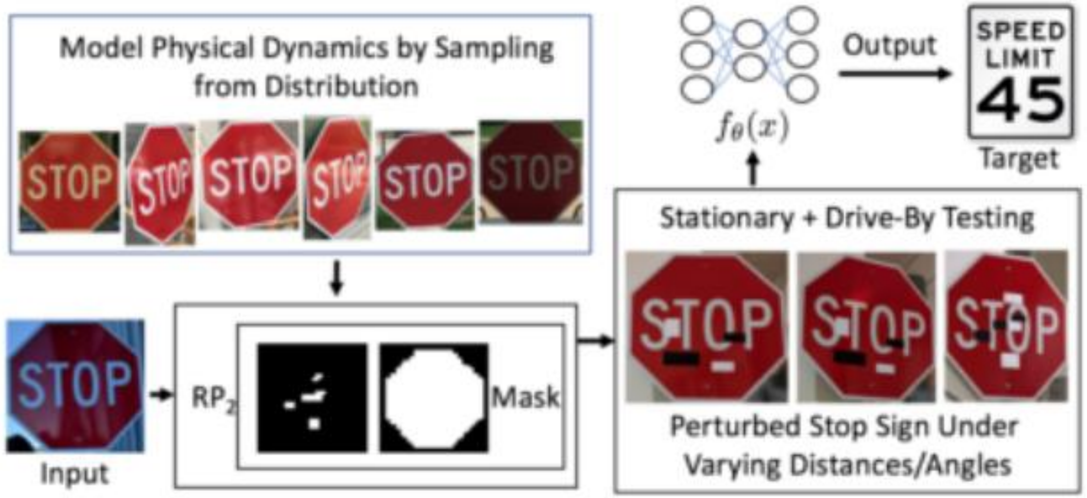
Malware Visualisation

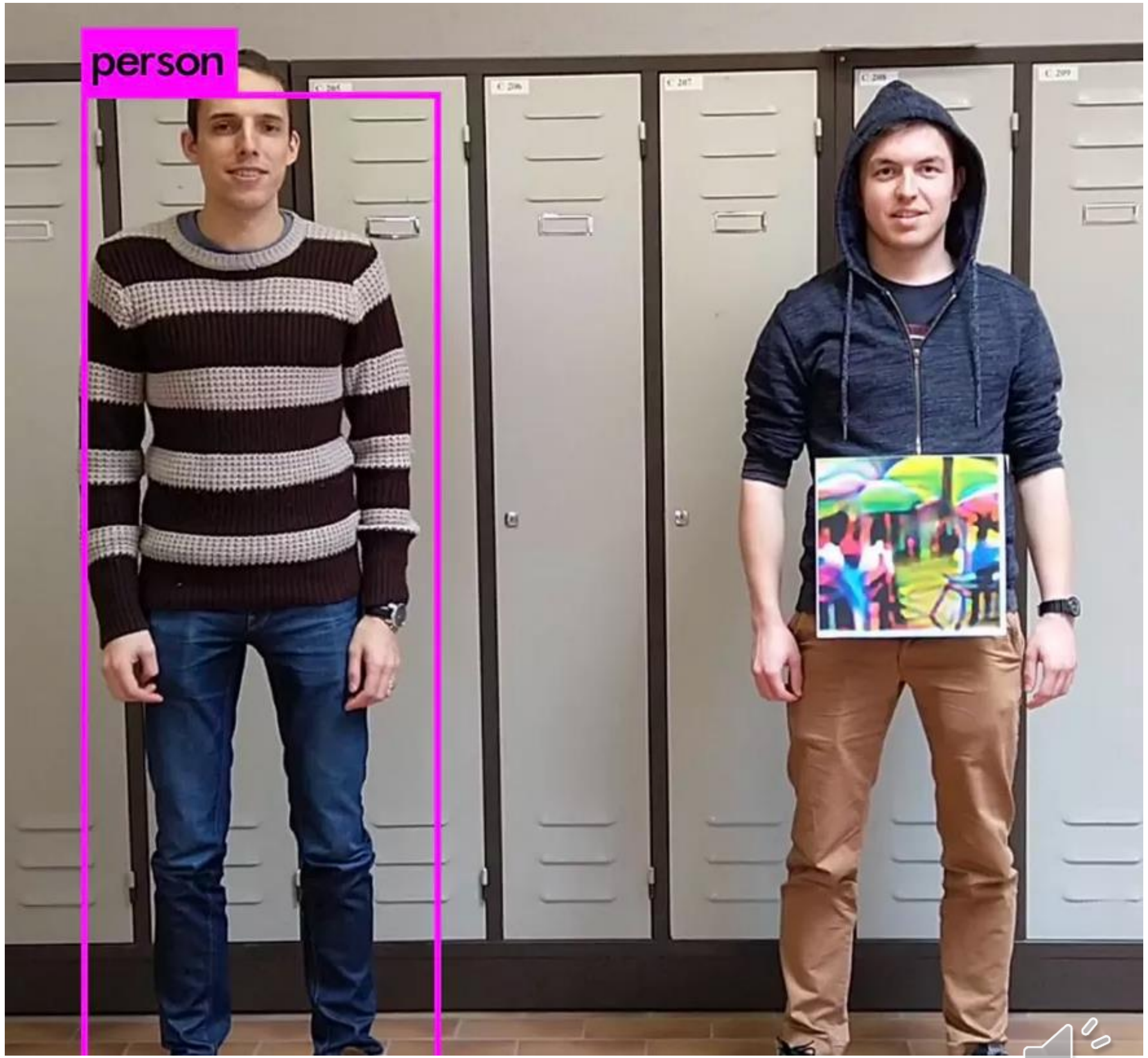
- Malware’s binary content as a gray-scale image
  - every byte as one pixel in an image
  - reused samples
- Some Drawbacks
  - need to select an image width
  - non-existing spatial correlations between pixels in different rows
  - suffer from code obfuscation techniques
    - encryption/compression will change the bytes



Adversarial Examples

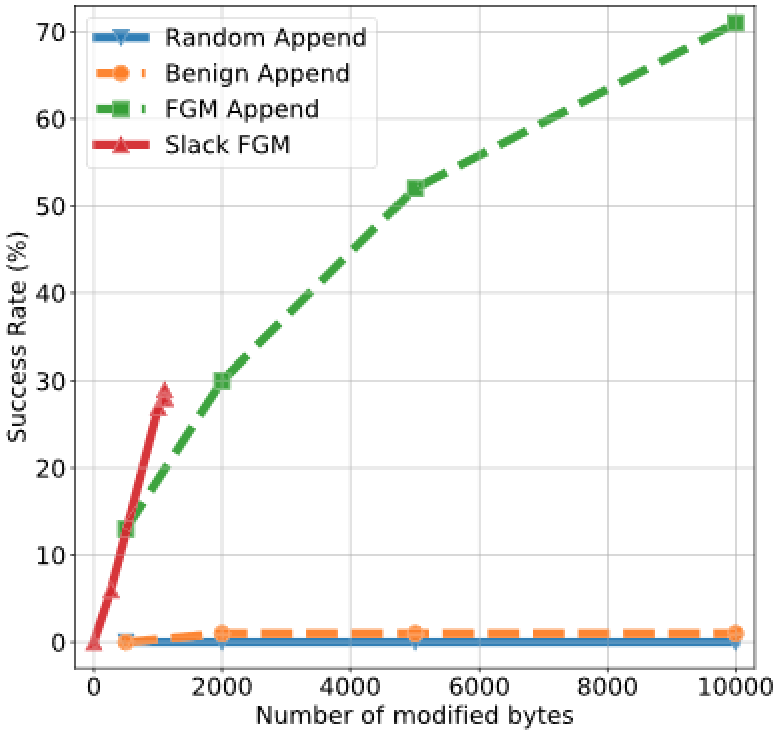
- Attempt to fool models


Model Physical Dynamics by Sampling from Distribution
STOP
Input
RP_2
Mask
Stationary + Drive-By Testing
Perturbed Stop Sign Under Varying Distances/Angles
$f_\theta(x)$
Output
SPEED LIMIT 45
Target



Adversarial Examples in Malware Detection

- Append Random, Benign, FGM bytes
  - might exceed the model’s maximum size
- Slack FGM
  - unused space between sections
    - misalignments between the virtual addresses and the multipliers over the block sizes on disk



Summary

- Malware
- Static/Dynamic Analysis
- Malware Detection
- ML-driven Malware Detection
- Adversarial Examples

References

- A Brief History of Computer Viruses
  - https://www.avg.com/en/signal/history-of-viruses
- A Brief History of Malware — Its Evolution and Impact
  - https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/
- #11: Machine Learning and Security
  - https://www.oreilly.com/library/view/machine-learning-and/9781491979891/ch04.html
- #12: PE101
  - http://pe101.corkami.com/
- #14: Dynamic Malware Analysis in the Modern Era
  - https://dl.acm.org/doi/10.1145/3329786
- #17: Yara: the pattern matching swiss knife
  - https://virustotal.github.io/yara/
- #18 McAfee Labs COVID-19 Threats Report
  - https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-july-2020.pdf

References

- #20: Machine Learning Aided Static Malware Analysis: A Survey and Tutorial
  - https://arxiv.org/abs/1808.01201
- #21: Deep Android Malware Detection
  - https://dl.acm.org/doi/abs/10.1145/3029806.3029823
- #22: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges
  - https://doi.org/10.1016/j.jnca.2019.102526
- #23: Robust Physical-World Attacks on Deep Learning Visual Classification
  - https://doi.org/10.1109/CVPR.2018.00175
- #23: Fooling automated surveillance cameras: adversarial patches to attack person detection
  - https://arxiv.org/pdf/1904.08653.pdf
  - https://www.youtube.com/watch?v=MIbFvK2S9g8
- #24: Exploring Adversarial Examples in Malware Detection
  - https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8844597

B.KANG@QUB.AC.UK

QUEEN'S
UNIVERSITY
BELFAST

ESTD 1845

CSIT

CENTRE
FOR SECURE
INFORMATION
TECHNOLOGIES