University of BRISTOL

CyBOK

Cyber Security Body of Knowledge

Malware and Attack Technologies

Dr Fabio Pierazzi

https://fabio.pierazzi.com

@fbpierazzi

bristol.ac.uk

University of BRISTOL

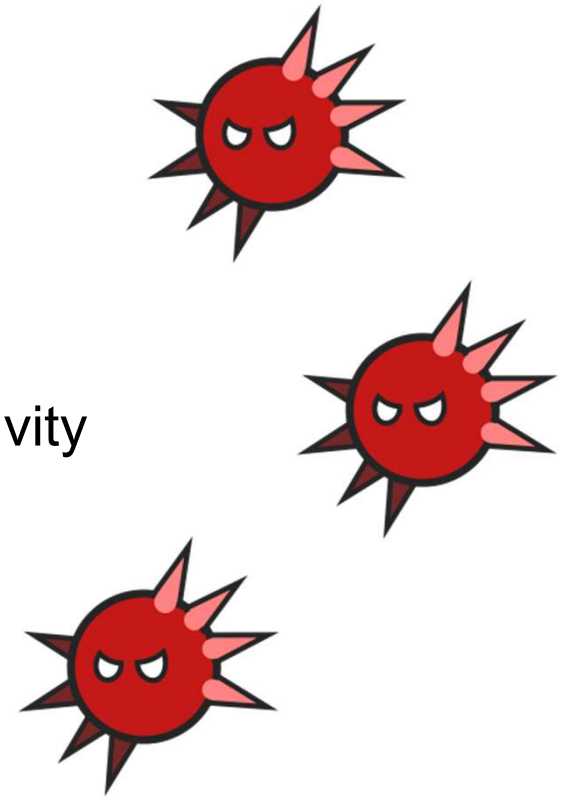
CyBOK

The CyBOK project would like to understand how the CyBOK is being used and its uptake. The project would like organisations using, or intending to use, CyBOK for the purposes of education, training, course development, professional development etc. to contact it at contact@cybok.org to let the project know how they are using CyBOK.

bristol.ac.uk

Malware

- Short for “Malicious Software”
- Any program that performs malicious activity
- Malware/Malicious code



Outline

- A Taxonomy of Malware
- Malicious Activities By Malware
- Malware Analysis
- Malware Detection
- Malware Response

University of BRISTOL

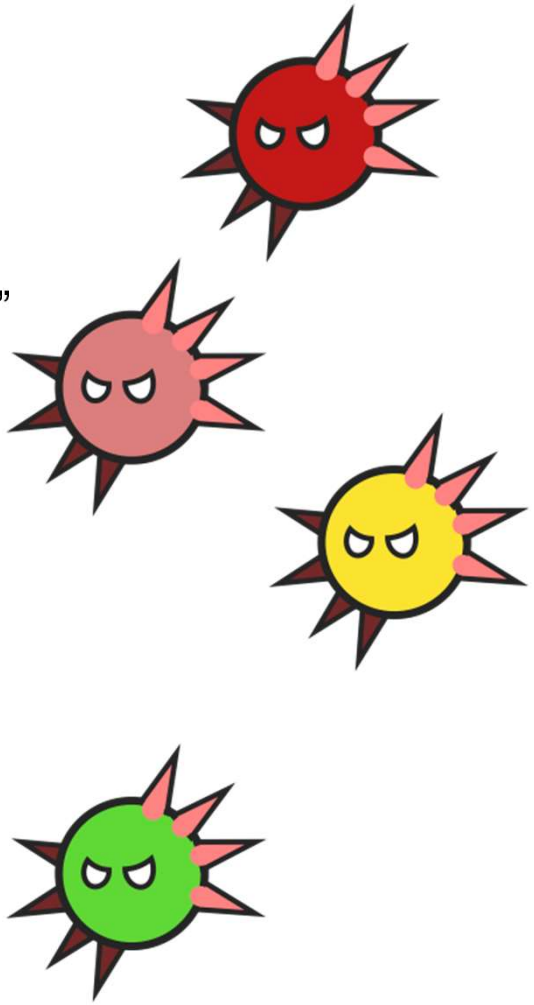
A Taxonomy of Malware

bristol.ac.uk

A Taxonomy of Malware

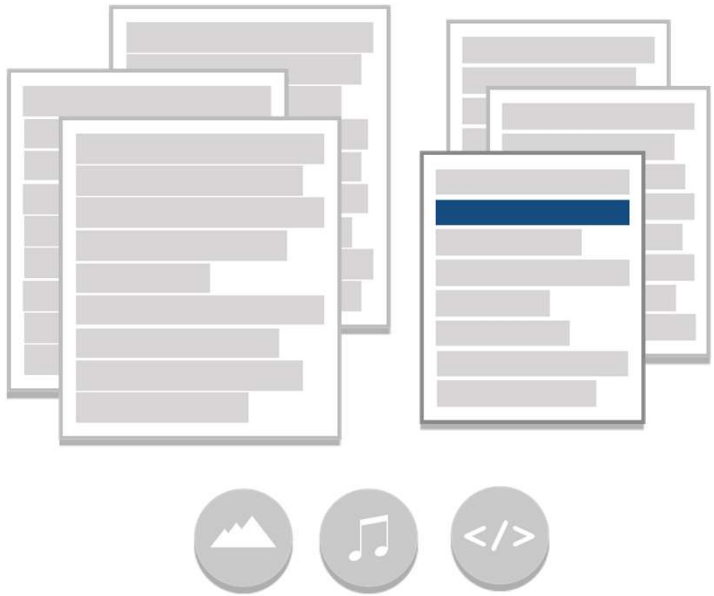
- Many types of malware
  - Often also “Potentially Unwanted Programs (PUPs)”
- Identify **common characteristics**
- Useful to develop **general countermeasures**

- We identify **six main dimensions**



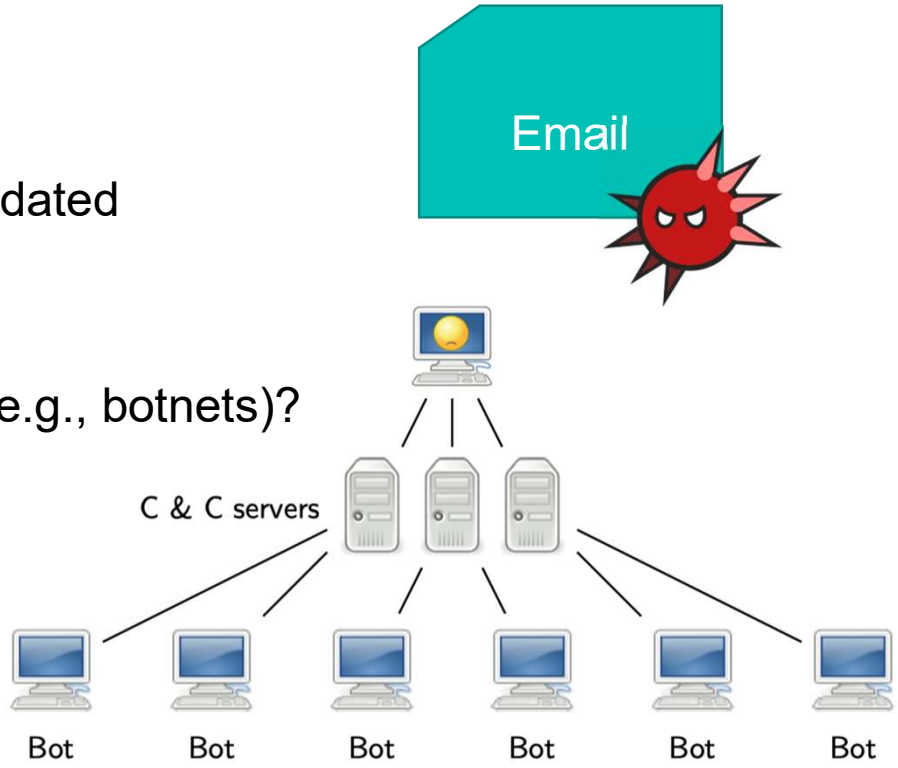
Malware Taxonomy: Dimensions

- **(1) Standalone vs Host-Program**
  - Is it an independent program, or is it part of another program?
- **(2) Persistent vs Transient**
  - Is it on filesystem, or just in memory?
- **(3) Layers of System Stack**
  - Firmware, boot sector, operating system kernel, drivers, APIs, user applications



Malware Taxonomy: Dimensions

- **(4) Auto-Spreading?**
  - Does it spread automatically, or does the infection happen through user actions? (e.g., emails)
- **(5) Dynamically Updates?**
  - Static (one-time) vs dynamically updated
- **(6) Coordinated?**
  - Is it part of a coordinated network (e.g., botnets)?



Taxonomy: Examples

	standalone or host-program	persistent or transient	layers of system stack	auto-spreading?	dynamically updatable?	coordinated?
viruses	host-program	persistent	firmware and up	Y	Y	N
malicious browser extensions	host-program	persistent	application	N	Y	Y
botnet malware	both	persistent	kernel and up	Y	Y	Y
memory-resident malware	standalone	transient	kernel and up	Y	Y	Y

University of BRISTOL

Malicious Activities By Malware

bristol.ac.uk

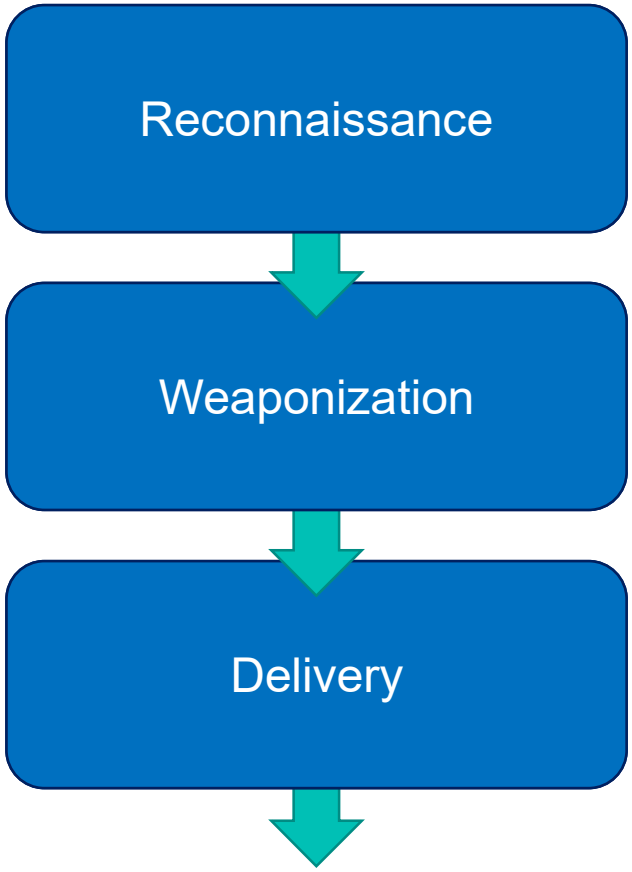
Malicious Activities

- Malware codifies malicious activities intended by an attacker

- Malware attacks require a multi-step approach
  - Example:
    - USB Key left in parking lot
    - Someone out of curiosity picks it up inserts it in a PC
    - USB Key contains auto-install malware
    - Exfiltrates information

- **Cyber Kill Chain model**

The Cyber Kill Chain

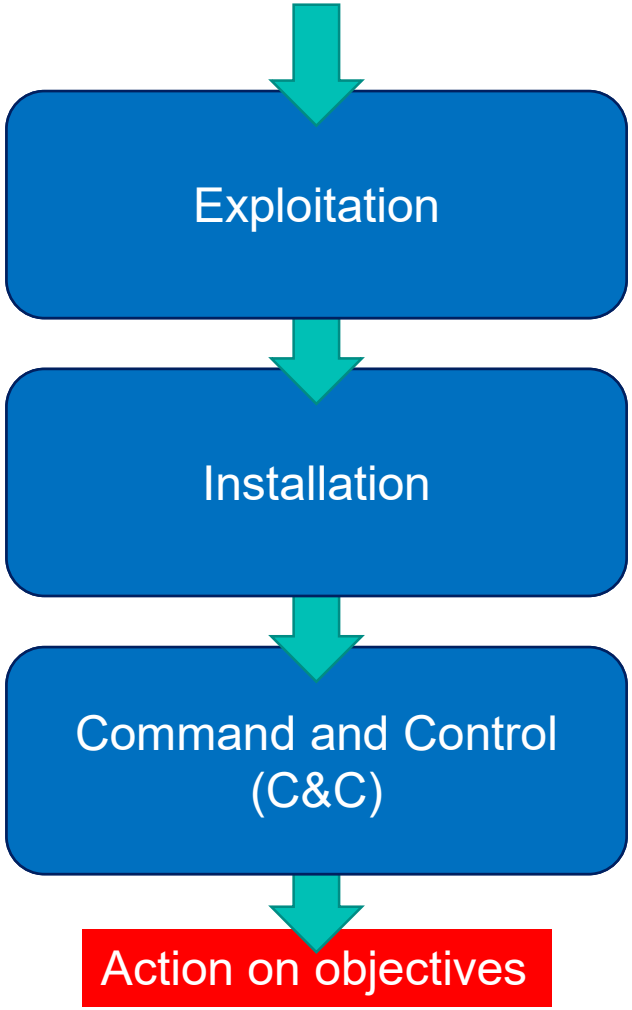


Gather information

Prepare exploits into malicious payload

Send malicious payload

The Cyber Kill Chain



Execute exploit on victim system

Install permanently on target system

Establish remote command connection

Underground Eco-system

- Organized Cyber-crime
  - Support full malware lifecycle
    - Development
    - Deployment
    - Operations
    - **Monetization**
  - Specialized roles
    - Plausible deniability
    - More effective malware
- 9-to-5 malware development
- Underground markets

Action Objectives

- Toolkits
  - Easy-to-use automated tools (e.g., keyloggers)
- Campaigns
  - Large-scale attacks (e.g., botnet)
  - Long-running
- Advanced Persistent Threats
  - Target specific organization
  - Low and slow
  - Lateral Movement and Data Exfiltration

Malware Analysis

bristol.ac.uk

Why Malware Analysis?

Benefits include:

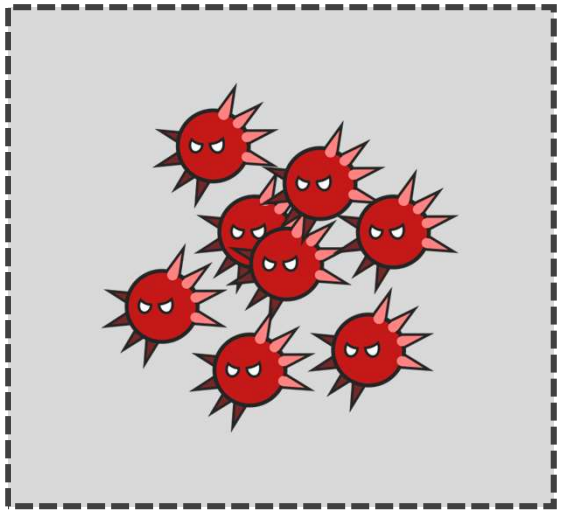
- Understand intended malicious activities
- Useful to attribution
- Understand and predict trends/scope of malware attacks

Three typical **phases**:

1. Understand Malware Format
2. Static Analysis
3. Dynamic Analysis

Acquiring Malware Data

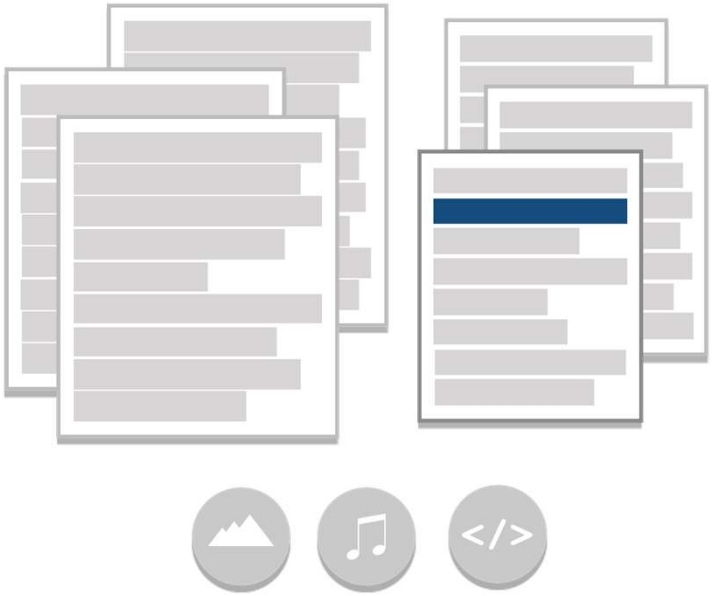
- Network/Host Sensors in Protected Environment
  - Capture malware “live”
- Malware Collection Efforts
  - E.g., researchers
- Threat Intelligence Exchange
- Legal/Ethical Responsibilities
  - Avoid damage
  - Share responsibly



Static Analysis

- Examine malware code without executing it
  - Binary code
  - Intermediate code (e.g., bytecode)
  - Source code

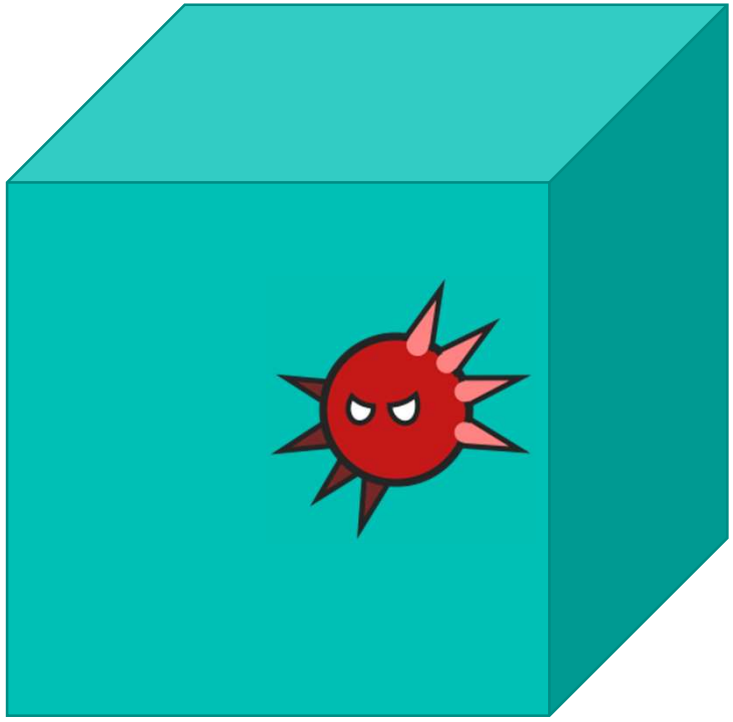
Program Analysis



- Pros
  - Better coverage of behaviors
  - Safe
- Limitations
  - Overestimates behaviors
  - Weak to obfuscation

Dynamic Analysis

- Monitors runtime behavior of malware execution to identify malicious behaviours
  - Typically from a stack layer lower than the malware itself
- Pros
  - You see actual behavior
  - (Relatively) language-independent
- Limitations
  - Underestimates behavior (coverage, triggers)
  - Safety and Live-Environment



Other Analysis Techniques

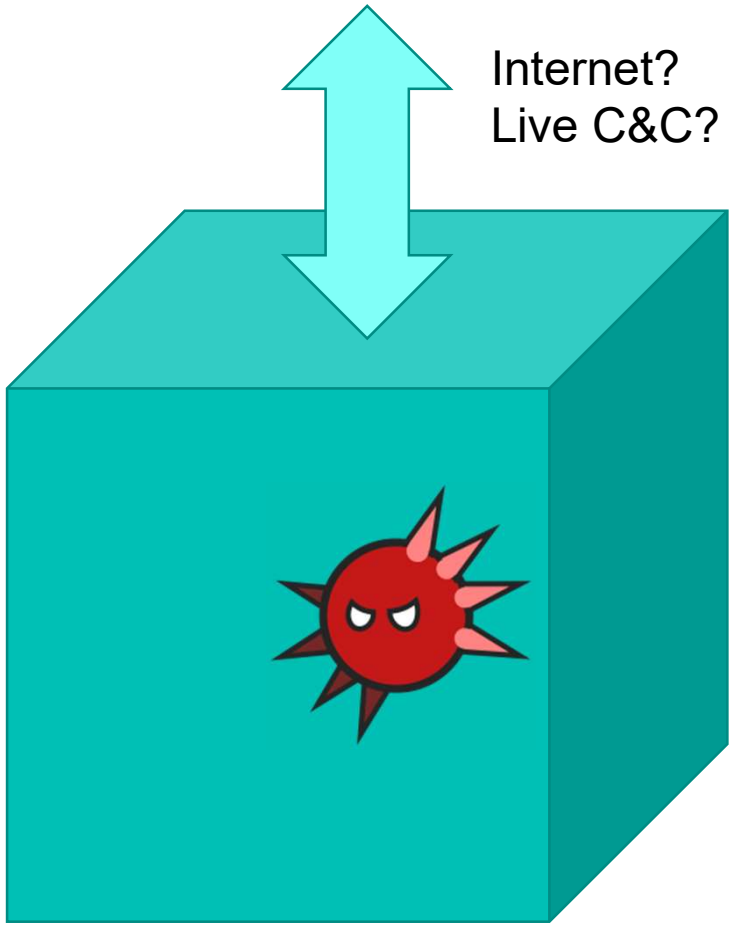
- Fuzzing
  - Randomised input to programs
  - To discover vulnerabilities/bugs/crashes
  - Also trigger malware behavior
  - Limitation: code coverage

- Symbolic Execution
  - It treats variables and equations as symbols and formulas that can potentially express all program paths
  - Limitation: Convergence (needs to execute end-to-end, one at a time)

- Concolic Execution
  - Combines Concrete and Symbolic Execution.
  - Online Concolic Execution: forking on all feasible branches

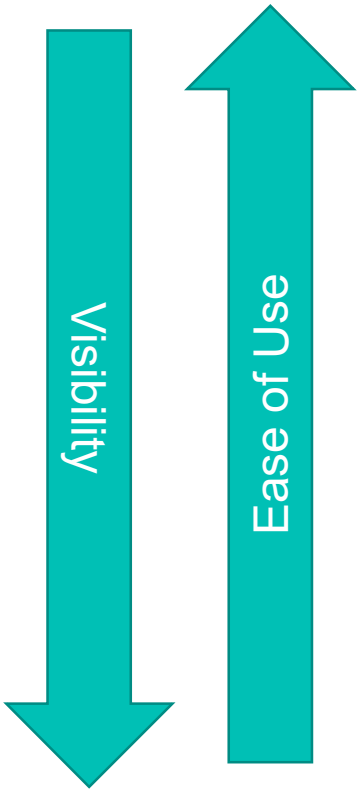
Analysis Environments

- Dedicated environment for Dynamic Analysis
  - Results/Cost Trade-Off.
- Cost
  - Analysis Time
  - Manual Human Effort
- Other aspects:
  - Safety
  - Live-environment



Common Environments

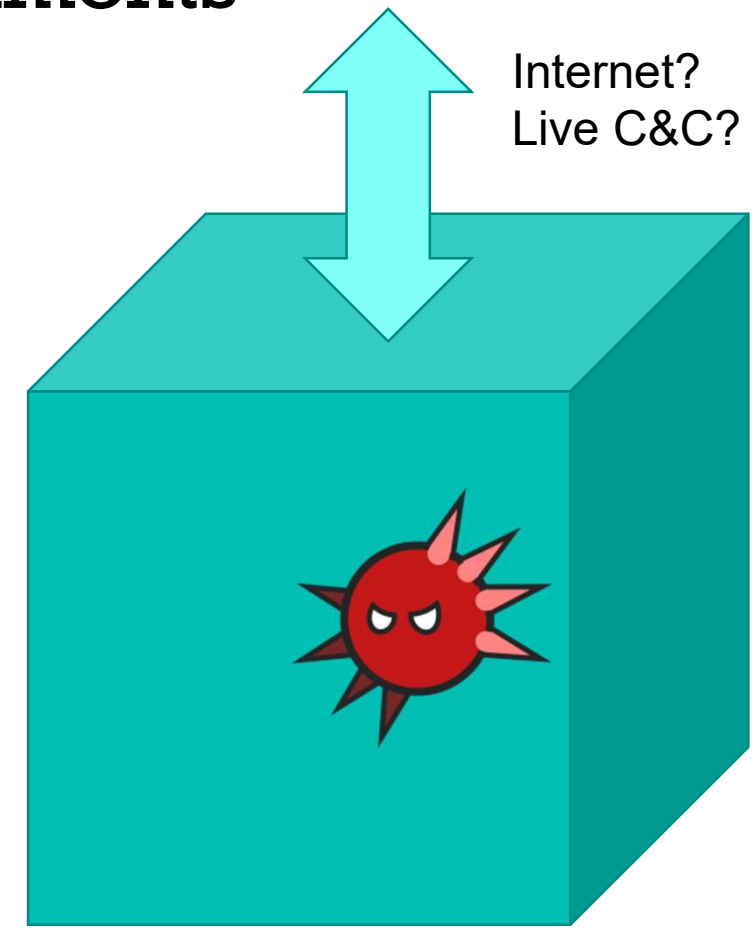
- Machine Emulator
  - Code-based architecture emulation
- Type 2 Hypervisor
  - Runs in host OS, provides virtualisation service for hardware
- Type 1 Hypervisor
  - Runs directly on system hardware
- Bare-metal machine
  - No virtualisation



Safety and Live-Environments

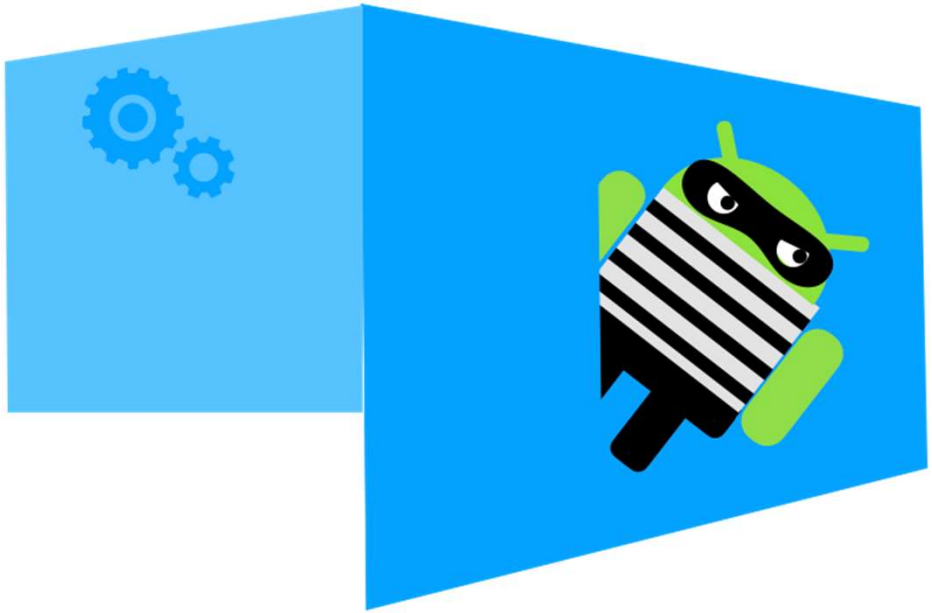
- Safety
  - The malware may do malicious actions. Can we allow it?
    - Also legal implications
  - Virtualized network environments

- Live-environment
  - Does the malware exhibit intended malicious behaviors?
    - Network connectivity
    - Real users on the machine
    - C&C/update servers online



Anti-Analysis and Evasion Techniques

- Evading Analysis Methods
  - Anti-disassembly
  - Obfuscation
    - Packing
    - Control-flow obfuscation
    - Code emulation
- Identifying the Analysis Environments
  - Red-pill testing (e.g., discrepancies CPU instructions)
  - Live-environment
    - User events, logs, history
    - Installed applications



Malware Detection

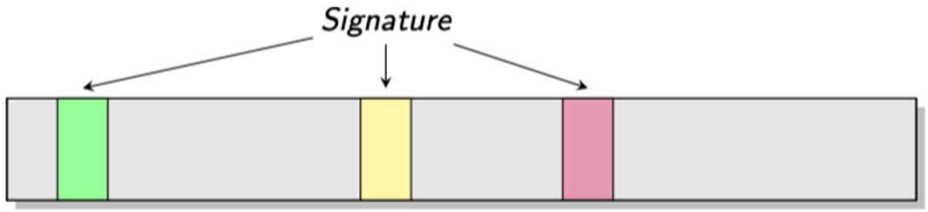
bristol.ac.uk

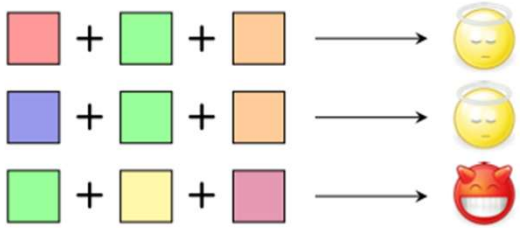
Malware Detection Objective

Identify the presence of a malware

- Network-level (e.g., distribution)
- Host-level (e.g., stored/in-memory)

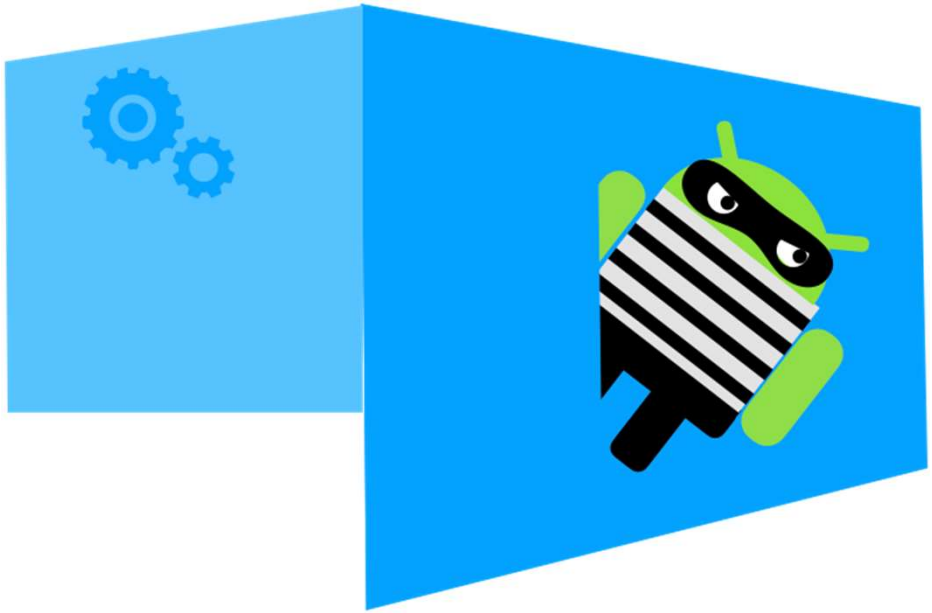
- AVs or IDS may prevent it → **Indicators of Compromise (IoCs)**
  - features and artifacts of the malware





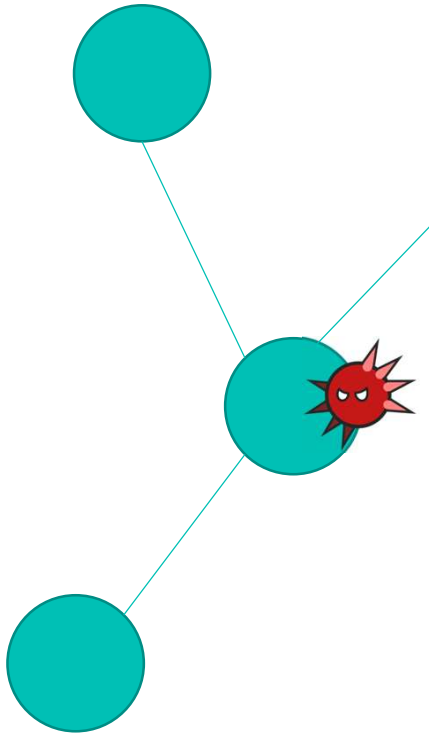
Evasion and Countermeasures

- Evading signature-based misuse detection
  - Packing
  - Polymorphism
  - Update routine
- Heuristics (e.g., high entropy) may lead to false alarms



Detection of Malware Attacks

- **Solution**: Aim to detect malicious activities in general
  - Anomaly detection
- Network-based monitoring
  - Network events
  - Domain names
  - Temporal activities
- Host-based monitoring
  - File system
  - Processes
  - System Calls

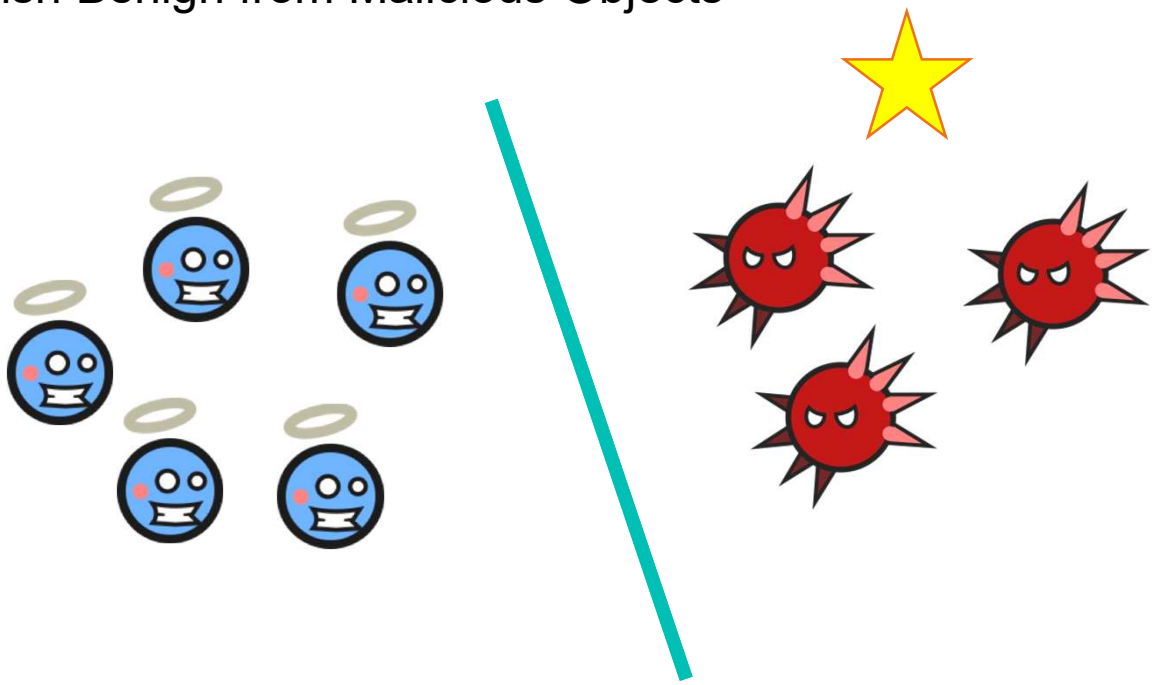


Prevent infection or detect it during execution

Example: Ransomware
Anomalous encryption operations

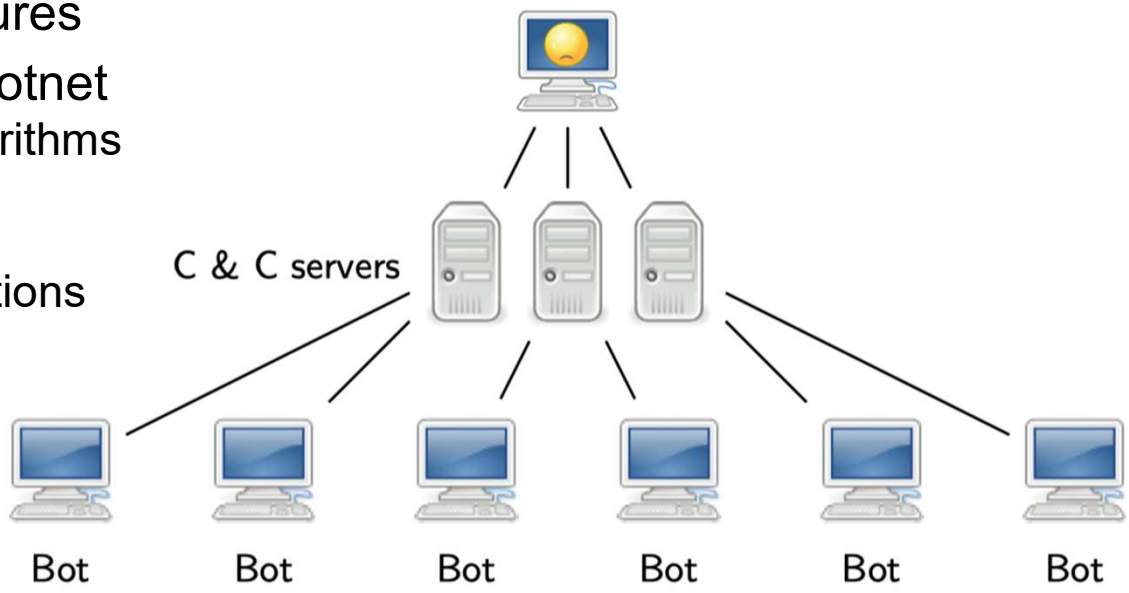
ML-based Security Analytics

- Machine Learning
  - Static and Dynamic Features
  - Build Models from Data
  - Distinguish Benign from Malicious Objects



ML-based Malware Detection

- Static and Dynamic Features
- Successful application: Botnet
  - Domain Generation Algorithms (DGA)
  - Fast-flux operations
  - C&C Server communications



- ML Limitations:
  - Challenging Feature Engineering
  - Adversarial attacks
  - Costly labeling for malware

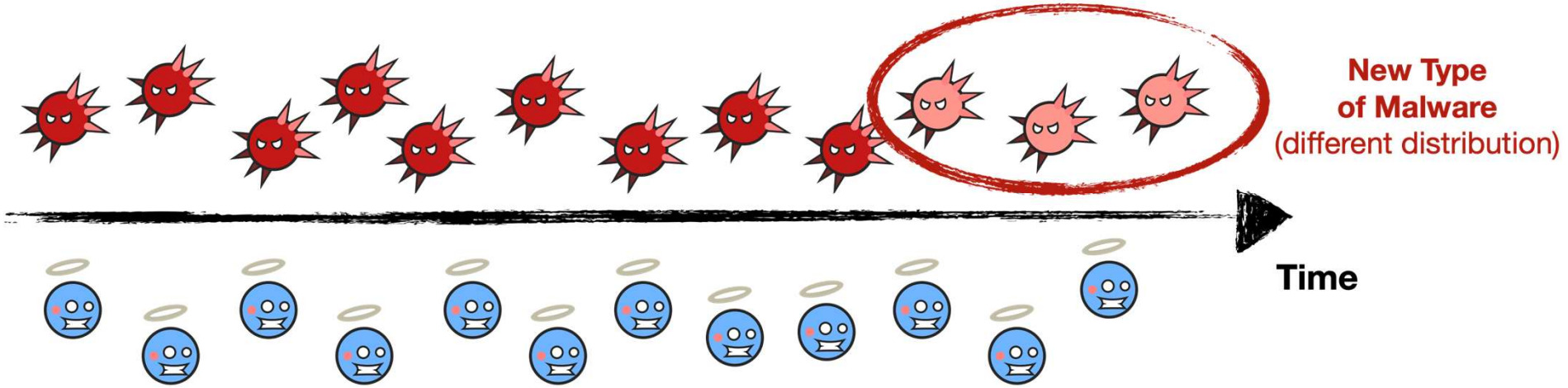
Deep Learning: poor explainability

Evasion of ML-based Malware Detection

- Mimicry Attack
  - Evasion
  - Recreate normal behavior (e.g., sequence of system calls)
  - Polymorphic blending attack of network traffic
- Targeted noise injection
  - Poisoning: “garbage in, garbage out”

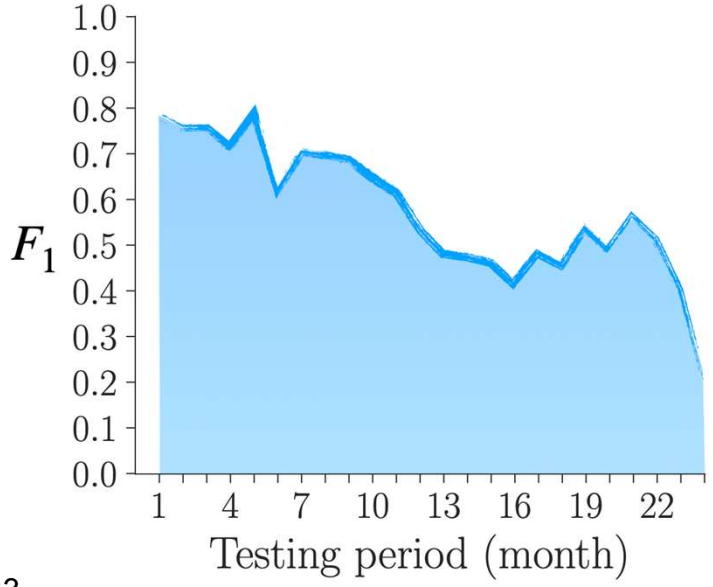
Partial countermeasures
- Ensemble of models
- Robust optimization
- Forget learning of old samples (against poisoning)

Concept Drift



- Malware evolves rapidly

- Solutions:
  - Active Learning
  - Online Learning



University of BRISTOL

Malware Response

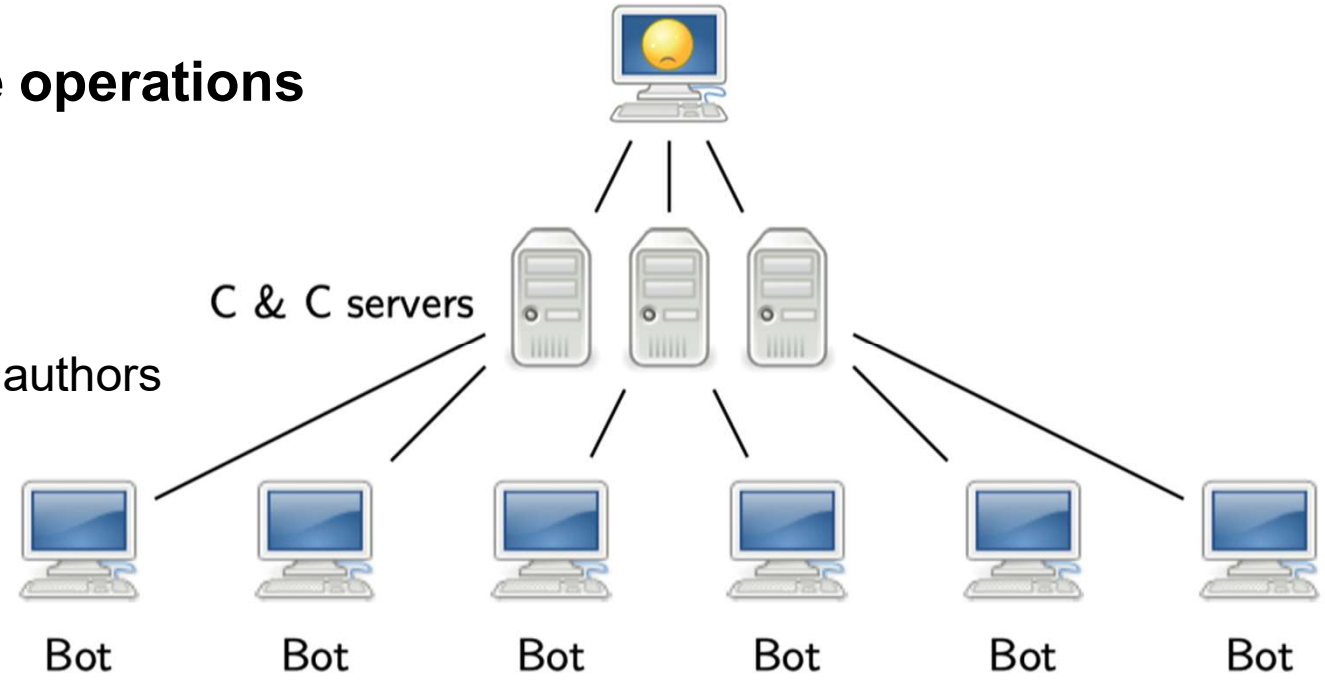
bristol.ac.uk

Malware Response

- After detection and possible mitigation, what can we do?

- **Disrupt malware operations**
  - Takedown

- **Attribution**
  - Identify malware authors



Disrupt Malware Operations

- Malware is often resilient and has contingency plans
  - Identify C&C mechanisms
  - Reverse-engineer DGA algorithms
  - Identify P2P backup networks
- Takedown the network by taking control of the infrastructure

Possible legal implications to be verified before proceeding

Attribution

- Law Enforcement wants to identify actual malicious actors

- Some hints for attribution:
  - WHOIS records
  - Coding style
  - Linguistic features
  - Control flow graphs

- Limitations:
  - WHOIS records often anonymized
  - Toolkits are often reused
  - False flags to deceive attribution

Right direction:
Integrate multiple streams and sources of data and evidence.

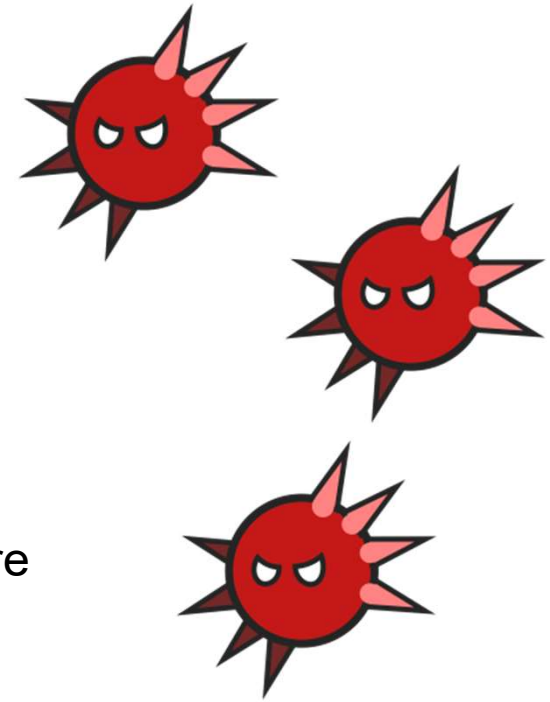
University of BRISTOL

Conclusion

bristol.ac.uk

Conclusion

- Attackers use malware to carry out malicious activities on their behalf
  - Different layers of system stack
  - Possibly support infrastructure
    - botnets
  - Wants to avoid detection and attribution
    - Detecting analysis environment
    - Obfuscation



- Defenders should work on
  - Analysis environments transparent to malware
  - Specialized program analysis algorithms
  - ML-based techniques
  - Malware response strategies (e.g., takedown)

University of BRISTOL

CyBOK

Cyber Security Body of Knowledge

Malware and Attack Technologies

Dr Fabio Pierazzi

https://fabio.pierazzi.com

@fbpierazzi

bristol.ac.uk