Malware Analysis

Paolo Falcarin

Ca' Foscari University of Venice

Department of Enviromental Sciences, Informatics and Statistics

paolo.falcarin@unive.it

CM0626 – Software Security

22 April 2025

What is a malware ?

Ca' Foscari University of Venice

- A Malware is a set of instructions that run on your computer and make your system do something that an attacker wants it to do.
- The purpose is often to steal, damage or manipulate sensitive information relating to an individual

Malware = Malicious Software

Ca' Foscari University of Venice

- Set of instructions that cause a site's security policy to be violated
- Often leveraging an inadvertent flaw or vulnerability (design or implementation)
  - To propagate/install on target
  - To cause harm on target

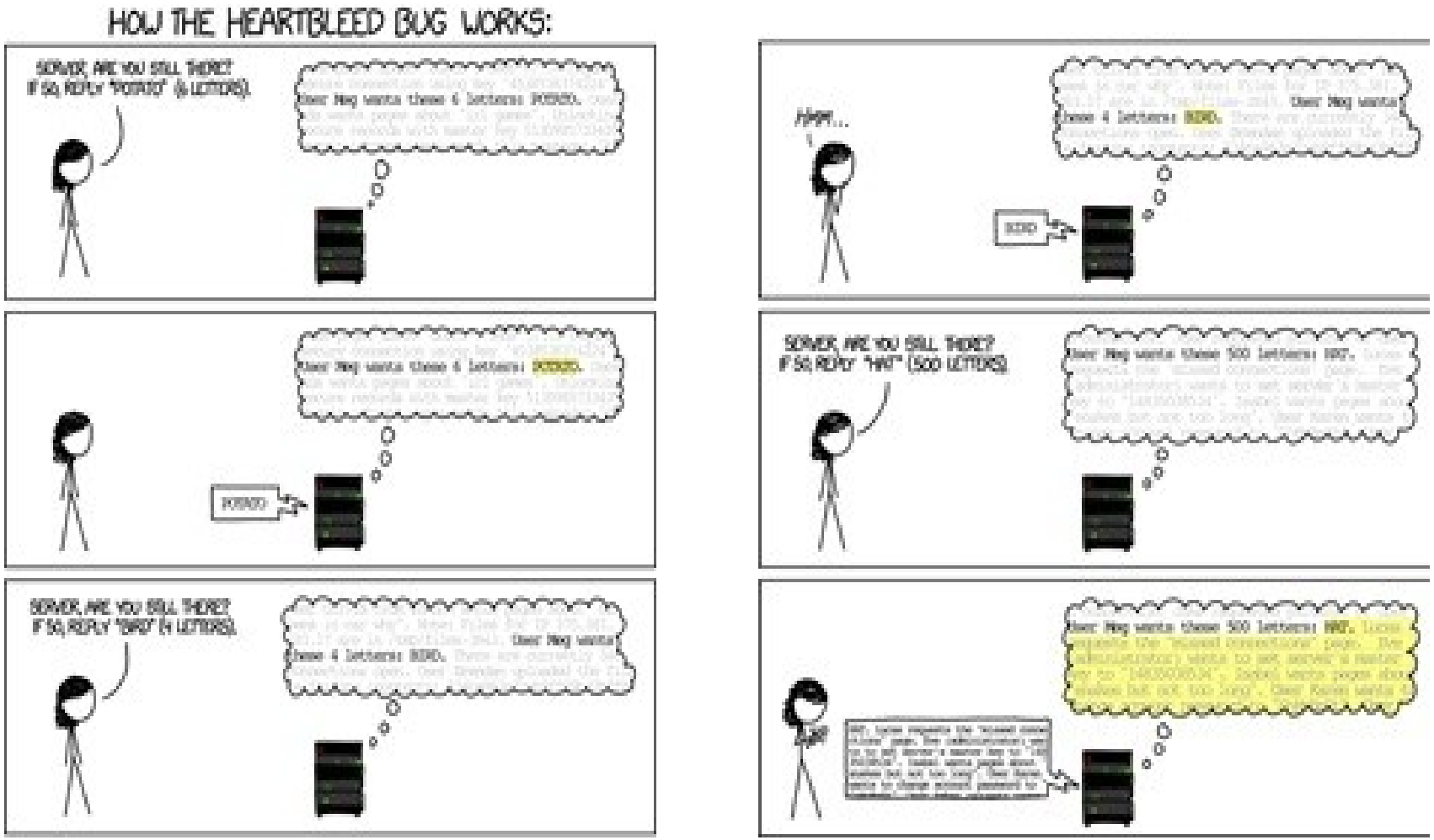
Software bugs: HeartBleed

Ca' Foscari University of Venice

- Buffer overread vulnerability in OpenSSL crypto library exploited by HeartBleed in 2014
  - http://heartbleed.com/
  - The Heartbleed bug allows anyone on the Internet to read the memory of the systems running the vulnerable versions of the OpenSSL software.
  - This compromises the secret keys and allows attackers to eavesdrop on communications, steal data and to impersonate services and users

Heartbleed

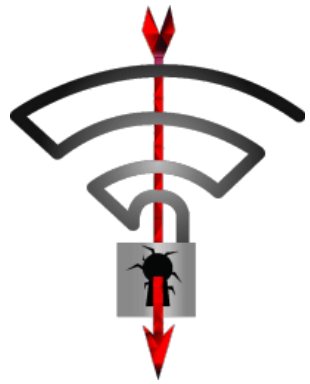
Ca' Foscari University of Venice



Design flaws: KRACK on WPA2

Ca' Foscari University of Venice

- KRACK attack on WPA2 wi-fi standard protocol in 2017
  - https://www.krackattacks.com/
- Attackers can use this novel attack technique to read information that was previously assumed to be safely encrypted.
- It can be abused to steal sensitive information
  - credit card numbers, passwords, chat messages, emails, photos, …
- The attack worked against all modern protected Wi-Fi networks.
  - Most of wifi devices should have been patched now…



What does malware do?

Ca' Foscari University of Venice

- Steal personal information
- Delete files
- Click fraud
- Steal software serial numbers or crypto keys
- Use your computer as relay for DoS attacks
- Cyber-weapon

Vulnerabilities

Ca' Foscari University of Venice

- Vulnerabilities in Operating Systems and widespread applications are exploited to attack a system
- Vulnerabilities Database: http://www.cve.mitre.org/
- CVE-2015-1641 Vulnerabilities in Microsoft Office could allow remote code execution.
- CVE-2015-2424 Microsoft Office Zero-Day CVE-2015-2424 leveraged by Tsar Team.
- CVE-2015-1701 Adobe & Windows Zero-Day exploits likely leveraged by Russia's APT28 in highly-targeted attack.

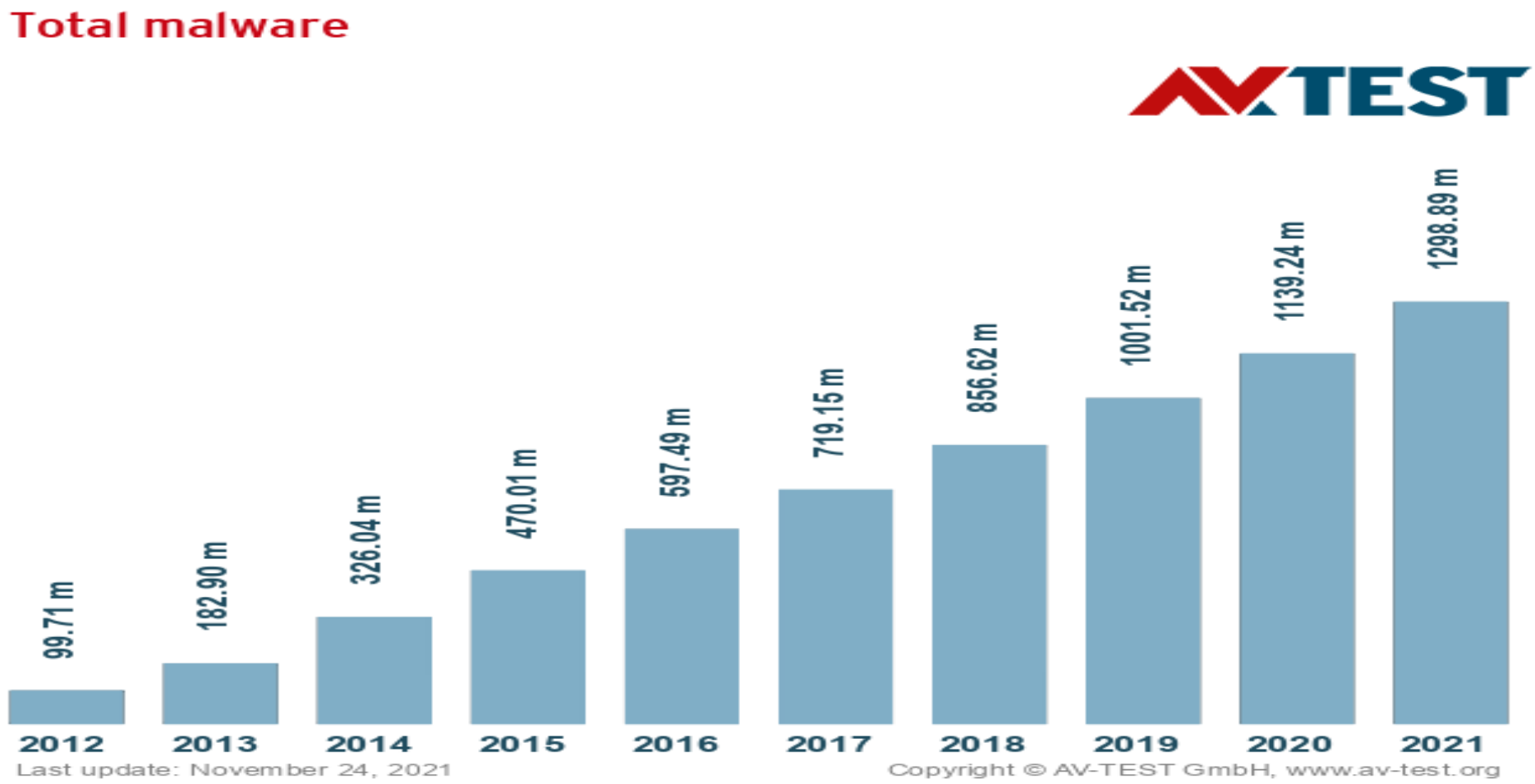
Zero Day Exploit

Ca' Foscari University of Venice

- An exploit takes advantage of a vulnerability to attack a sytem and enact malicious behaviour
- A Zero-Day exploit is based on a recently discovered vulnerability, that has no patch available yet
- Time between exploit discovery and wide activation shrinking
- Malware developer has trade-off
  - Big splash but faster discovery
  - Reduced attack rate but longer undiscovered

Number of malware signatures

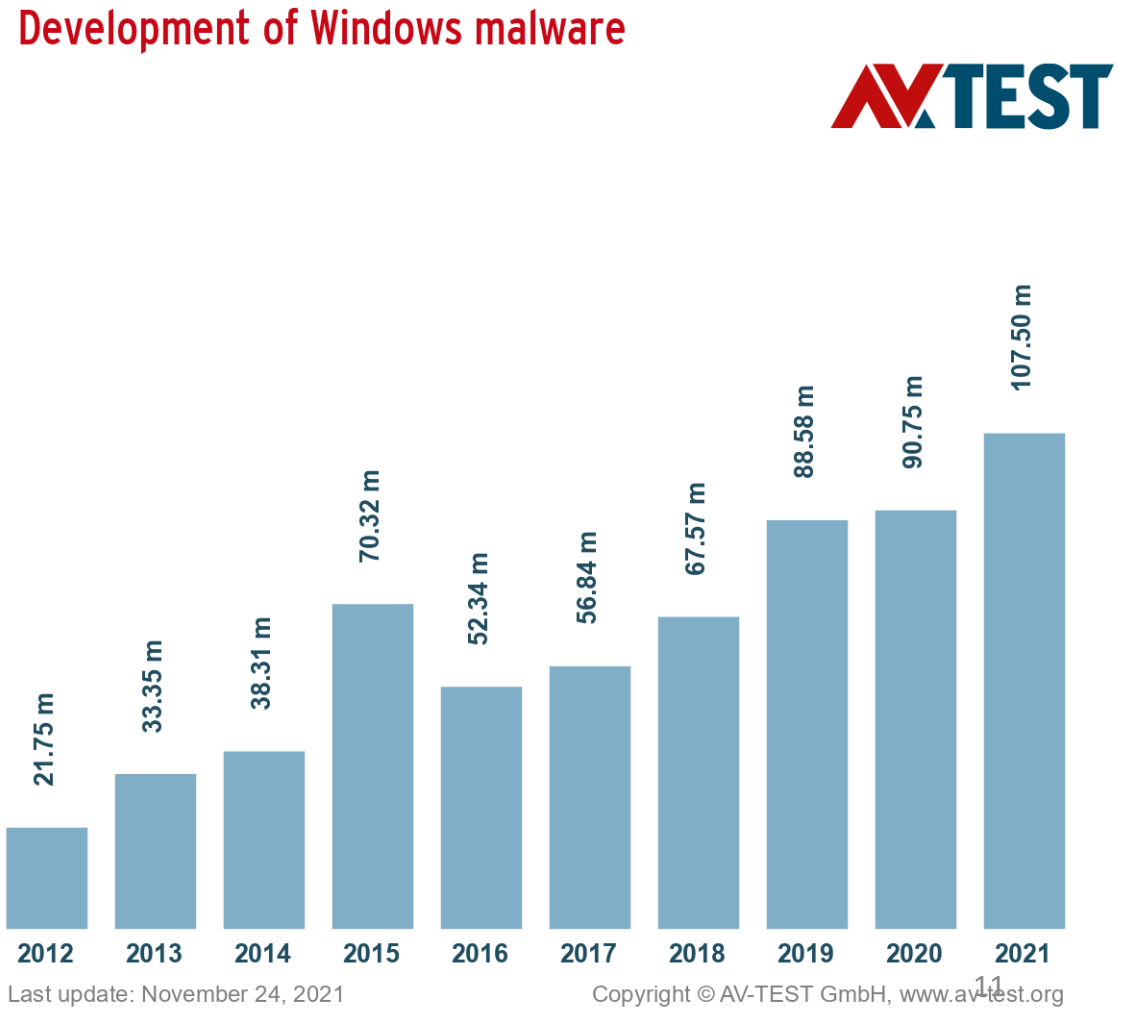
Ca' Foscari University of Venice



Malware in Numbers

Ca' Foscari University of Venice

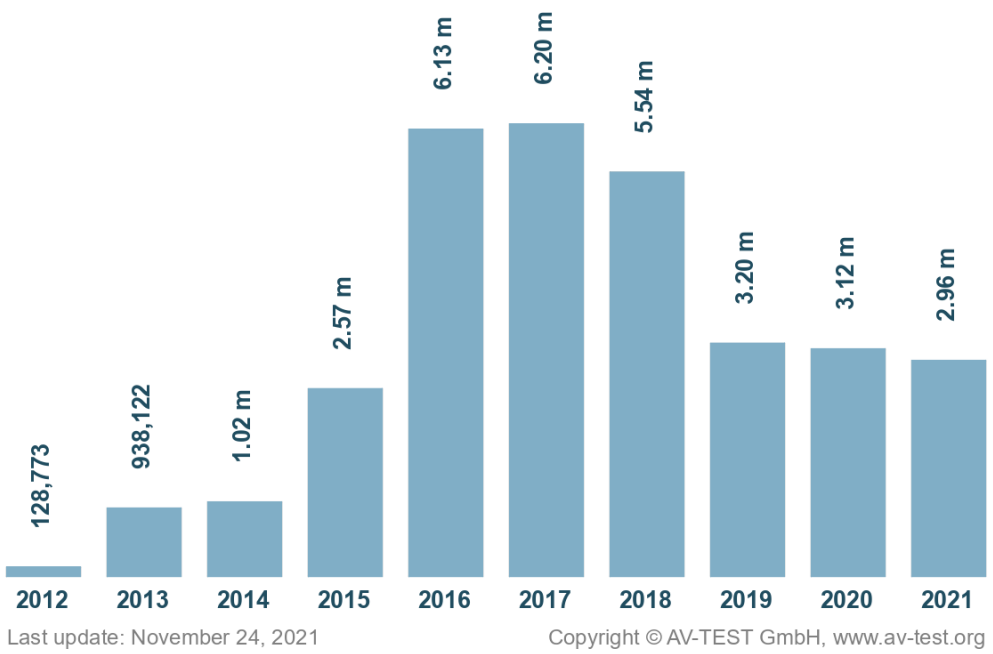
With new malware constantly being developed, the importance of protecting our systems against such threats is becoming increasingly important.



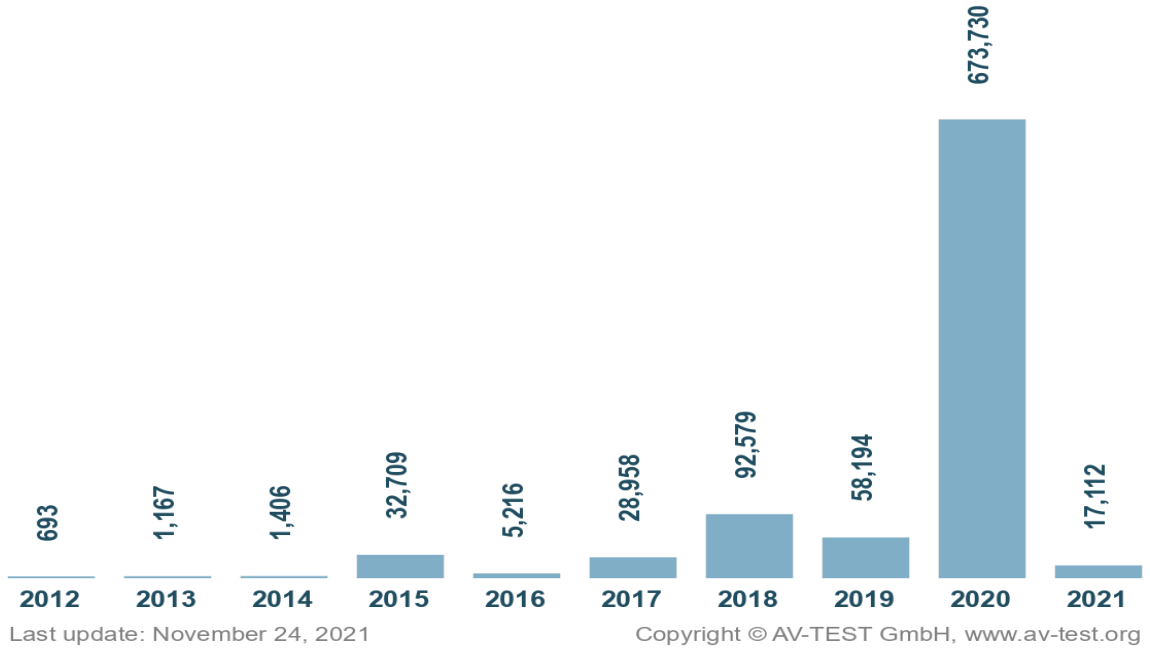
Mobile Malware

Ca' Foscari University of Venice

Development of Android malware



Development of MacOS malware



Malware examples

Ca' Foscari University of Venice

Malware examples on Unis

Ca' Foscari University of Venice

University Of East Anglia Suffers Second Data Breach

University of Greenwich students' data leaked online

University College London hit by ransomware attack

King's College London student data breach results in underaking

sky NEWS

theguardian

itv NEWS

EveningStandard.

In 2016/2017 there were 1,152 cyber attacks on UK Universities

How malware spreads?

Ca' Foscari University of Venice

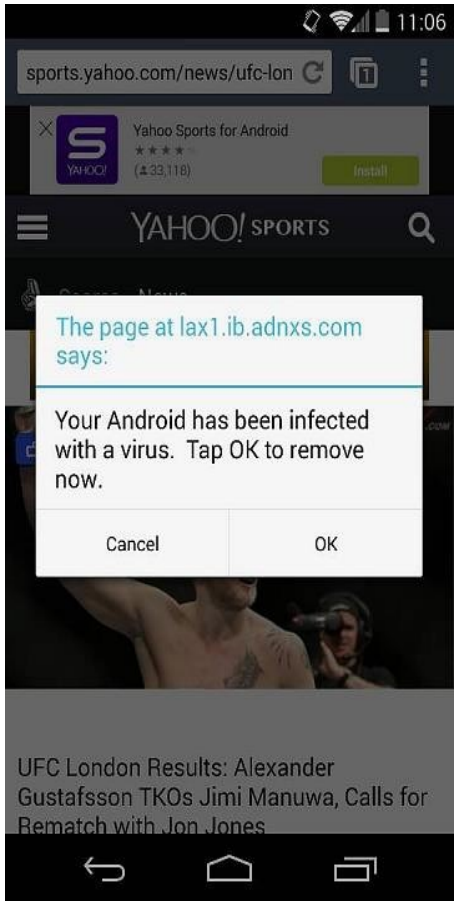
- Malicious URLs (Pop-ups)
- Via CD, USB etc…
- Social networks websites
- By downloading software that's infected
- Opening insecure email attachments or loading content stored from an unknown source
  - E.g. Macros in MS office

Mobile Malware

Ca' Foscari University of Venice

- Some Malware specifically targets mobile devices (smartphones, tablets, smart watches)
  - Android devices are most commonly attacked

- The results of mobile malware can lead to:
  - Identity theft
  - Financial loss
  - Data contained on the device being stolen/wiped



Adware

Ca' Foscari University of Venice

- Adware displays advertisements onscreen in the form of an internet banner or pop up
- The main function of adware is to provide revenue to organizations who distribute adware
- Once installed a user is tracked by cookies. This information can be obtained by a 3rd party without the knowledge of the user
- Adware may contain key-loggers that monitor the keys pressed on a keyboard in order to extract sensitive information such as usernames and account details

Example of Adware pop ups

Ca' Foscari University of Venice

What is a Ransomware?

Ca' Foscari University of Venice

- Ransomware is a kind of cyber attack that involves hackers taking control of a computer system and blocking access to it until a ransom is paid.
- Restricting the ability to carry out general activities on the system
- Encrypting files
- Disabling Apps
- In order to restore the functionality of the system a user is often required to pay money to the cybercriminal who sent the malware.
- http://www.bbc.co.uk/news/av/technology-35091536/what-is-ransomware

Ransomware

Ca' Foscari University of Venice

- Trj/SMSlock.A
- Russian ransomware
- April 2009

To unlock you need to send an SMS with the text4121800286to the number3649Enter the resulting code:Any attempt to reinstall the system may lead to loss of important information and computer damage

Ransomware: CryptoLocker

Ca' Foscari University of Venice

Latest example's of Ransomware

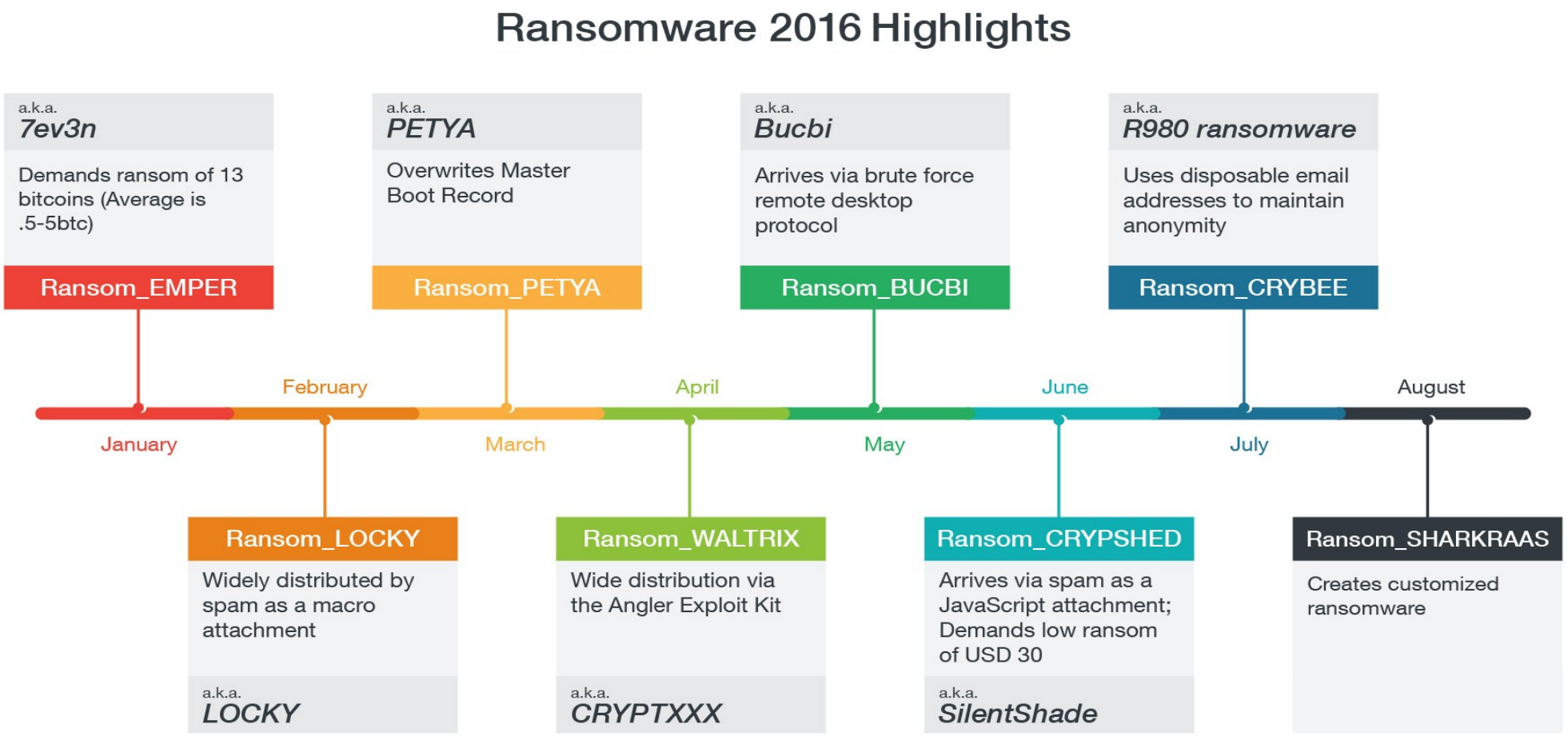
Ca' Foscari University of Venice

- A computer malware has spread across 150 countries
- The ransomware took over users' files, demanding $300 (£230) to restore them
- Approx. $70,000 (£54,000) had been paid in total in a bid to get any locked data released

Source: BBC NEWS

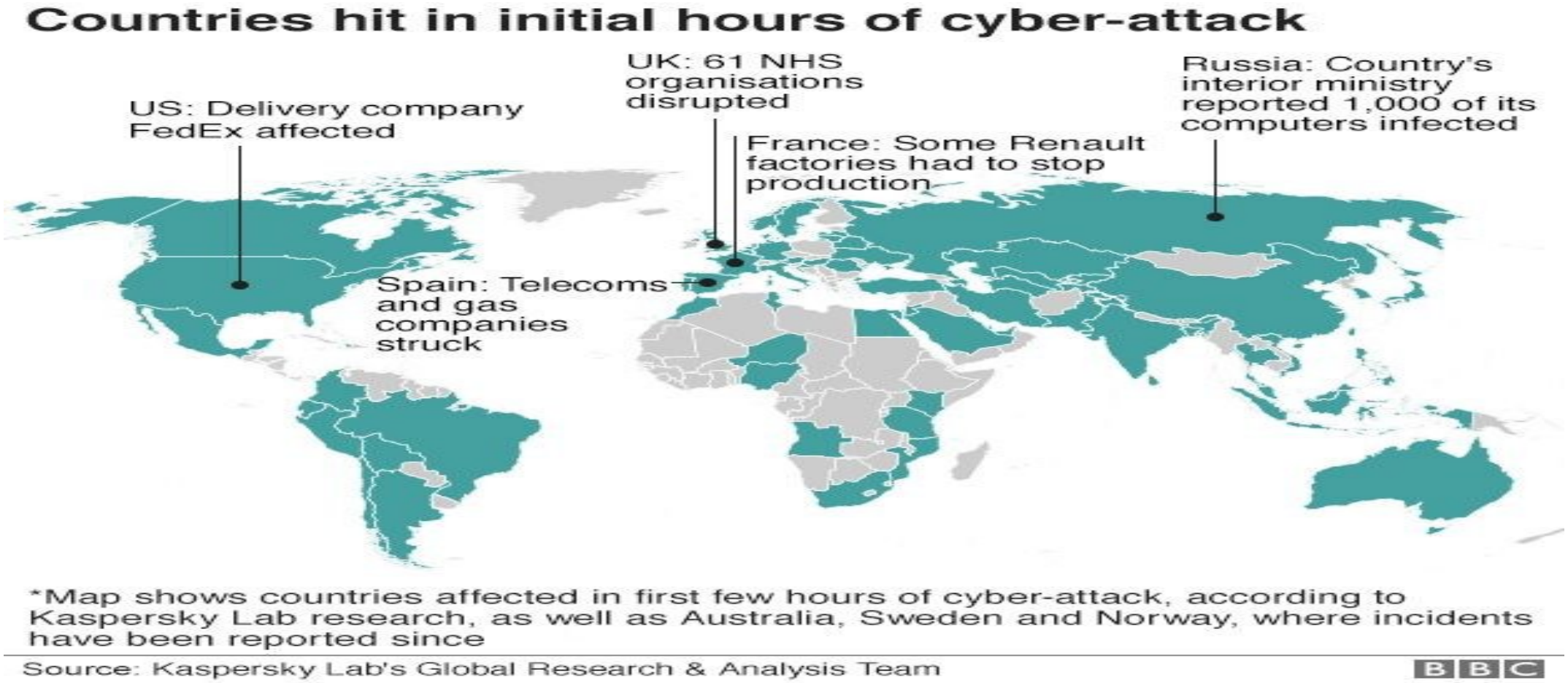
Ransomware Boom in 2016

Ca' Foscari University of Venice



Ransomware cyber-attacks 12th May 2017

Ca' Foscari University of Venice

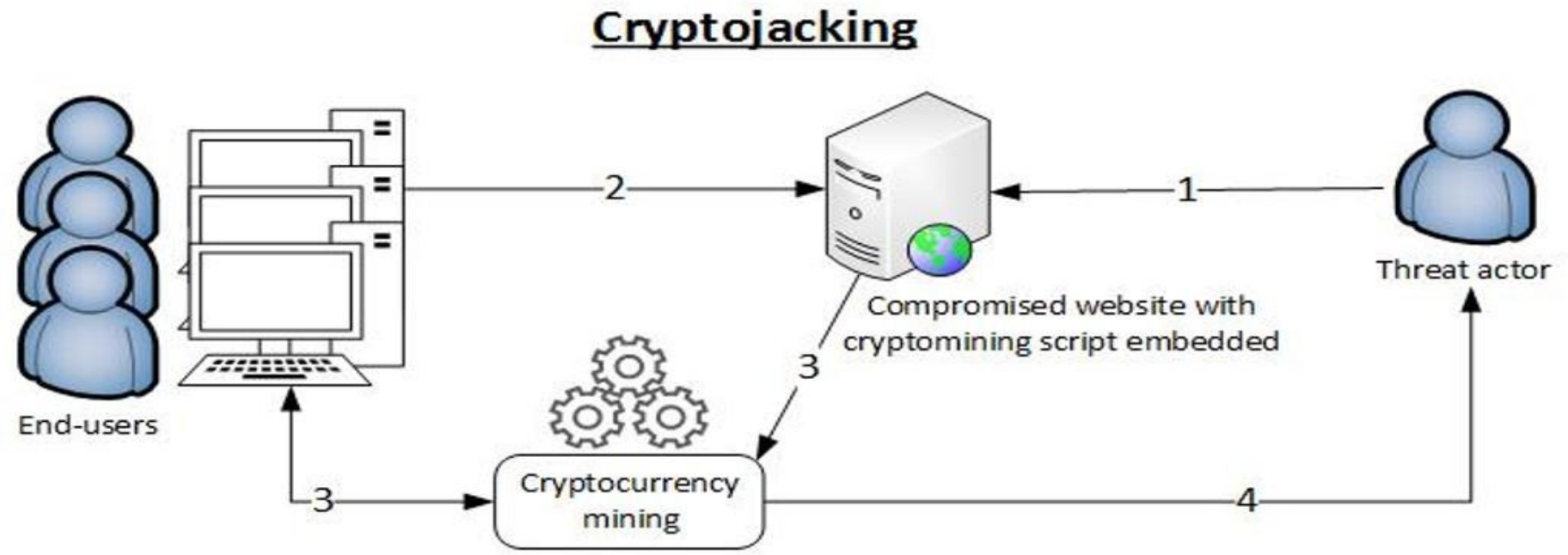


Malware Summary

Ca' Foscari University of Venice

Code type	Characteristics
Virus	Attaches itself to program and copies to other programs
Trojan Horse	Contains unexpected, additional funtionality
Logic Bomb	Triggers action when condition occurs
Time Bomb	Triggers action when specified time occurs
Trapdoor	Allows unauthorized access to functionality
Worm	Propagates copies of itself through a network
Rabbit	Replicates itself without limit to exhaust resources
Netbot	Trapdoor programs orchestrated through control channel (IF
Root Kit	Hooks standard OS calls to hide data

Cryptojacking



Steps

1. The threat actor compromises a website
2. Users connect to the compromised website and the cryptomining script executes
3. Users unknowingly start mining cryptocurrency on behalf of the threat actor
4. Upon successfully adding a new block to the blockchain, the threat actor receives a reward in cryptocurrency coins

Some Virus Type

Ca' Foscari University of Venice

- Polymorphic : uses a polymorphic engine to mutate while keeping the original algorithm intact (packer)
- Metamorphic : Change after each infection

Virus Parts

Ca' Foscari University of Venice

- Infection mechanism
  - How the virus moves from victim to victim
- Trigger
  - The condition that causes the payload to activate or be delivered
- Payload
  - The activity of the virus beyond the spreading
  - E.g., installing software, harvesting information

Virus Operation

Ca' Foscari University of Venice

- Virus Phases:
  - Dormant: Waiting on trigger event
  - Propagation: Replicating to programs/disks
  - Triggering: By event to execute payload
  - Execution: Executing payload
- Details usually Machine/OS specific
  - Exploits different features or weaknesses

Virus Pseudocode


```
beginvirus:
If spread-condition then begin
    For some set of target files do begin
        If target is not infected then begin
            Determine where to place virus instructions
            Copy instructions from beginvirus to endvirus into target
            Alter target to execute new instructions
If trigger pulled
    Perform some actions
Goto beginning of infected program
endvirus:
```

Virus Attachment

Ca' Foscari University of Venice

- A Virus can attach itself to a program or to data by
  - Appending itself to either the beginning or end of either source code or assembly, so it is activated when the program is run
  - Integrate itself into the program, spread out code
  - Compress original program so addition of virus does not change file system
  - Integrate into data: executable text macro, scripting
  - Macros and email attachments
- An activated virus may:
  - Cause direct or immediate harm
  - Run as a memory resident program (TSR, daemon, or service)
  - Replace or relocate boot sector programs, start at system start-up

Macros Viruses

Ca' Foscari University of Venice

- Macro code attached to some data file
  - Interpreted rather than compiled
  - Platform independent
  - Mobile code
- Interpreted by program using the file
  - E.g., Word/Excel macros
  - Esp. using auto command and command macros
  - Often automatically invoked
- Blurs distinction between data and program files making task of detection much harder
- Classic trade-off: ”ease of use” vs ”security”

Email Viruses

- Spread using email with attachment containing a macro virus
  - Melissa, LoveBug
- Triggered when user opens or executes attachment
  - Also when mail viewed by using scripting features in mail agent
  - Usually targeted at Microsoft Outlook mail agent and Word/Excel documents, Microsoft IIS

What is a trojan

Ca' Foscari University of Venice

A trojan describes the class of malware that appears to perform a desirable function but in fact performs undisclosed malicious functions that allow unauthorized access to the victim computer

Wikipedia

Trojan Horses

- Seemingly useful program that contains code that does harmful things
  - Perform both overt and covert actions
- Frequently embedded in applets or games, email attachments
- Trojan horse logins, spoof authentication or webpage forms

Malware Infection Methods

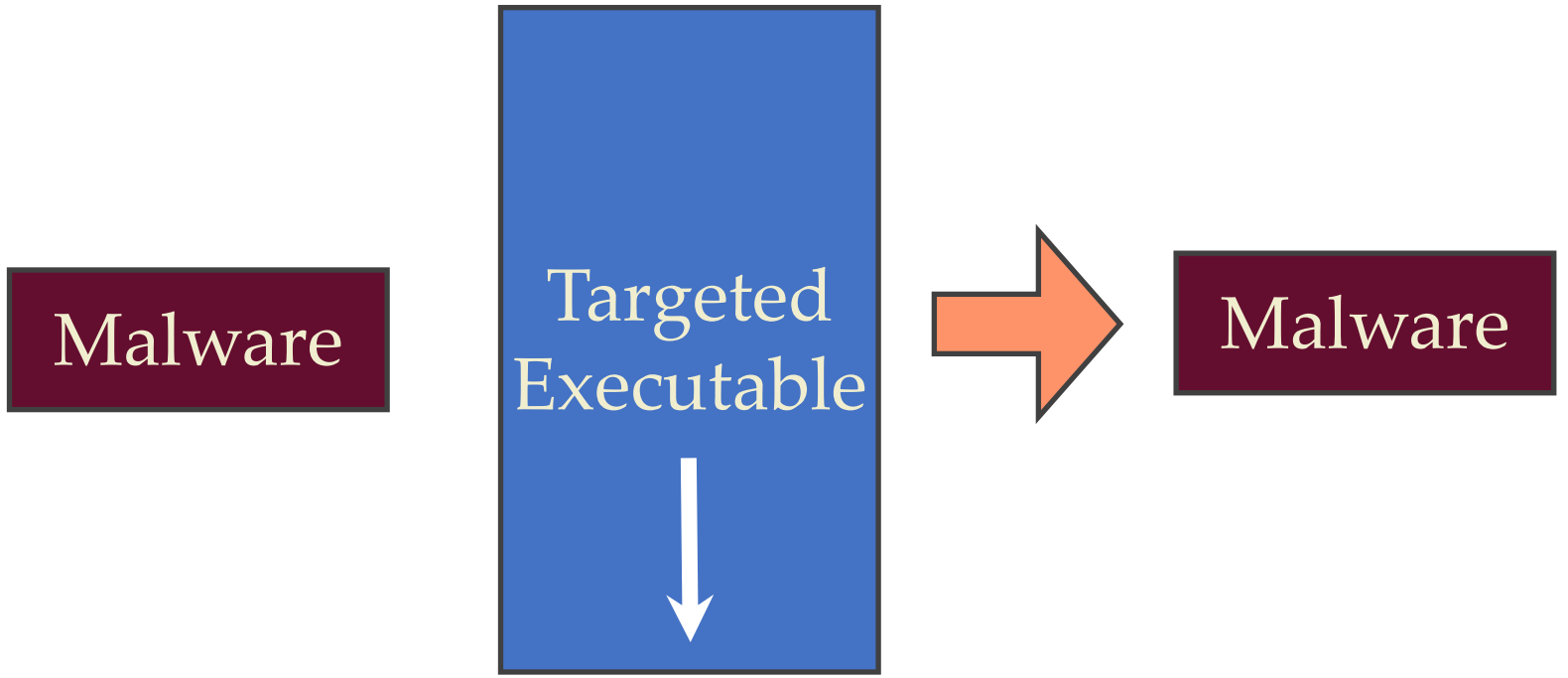
Infection Methods

Ca' Foscari University of Venice

- What to Infect
- Executable
- Interpreted file
- Kernel
- Service
- MBR
- Hypervisor

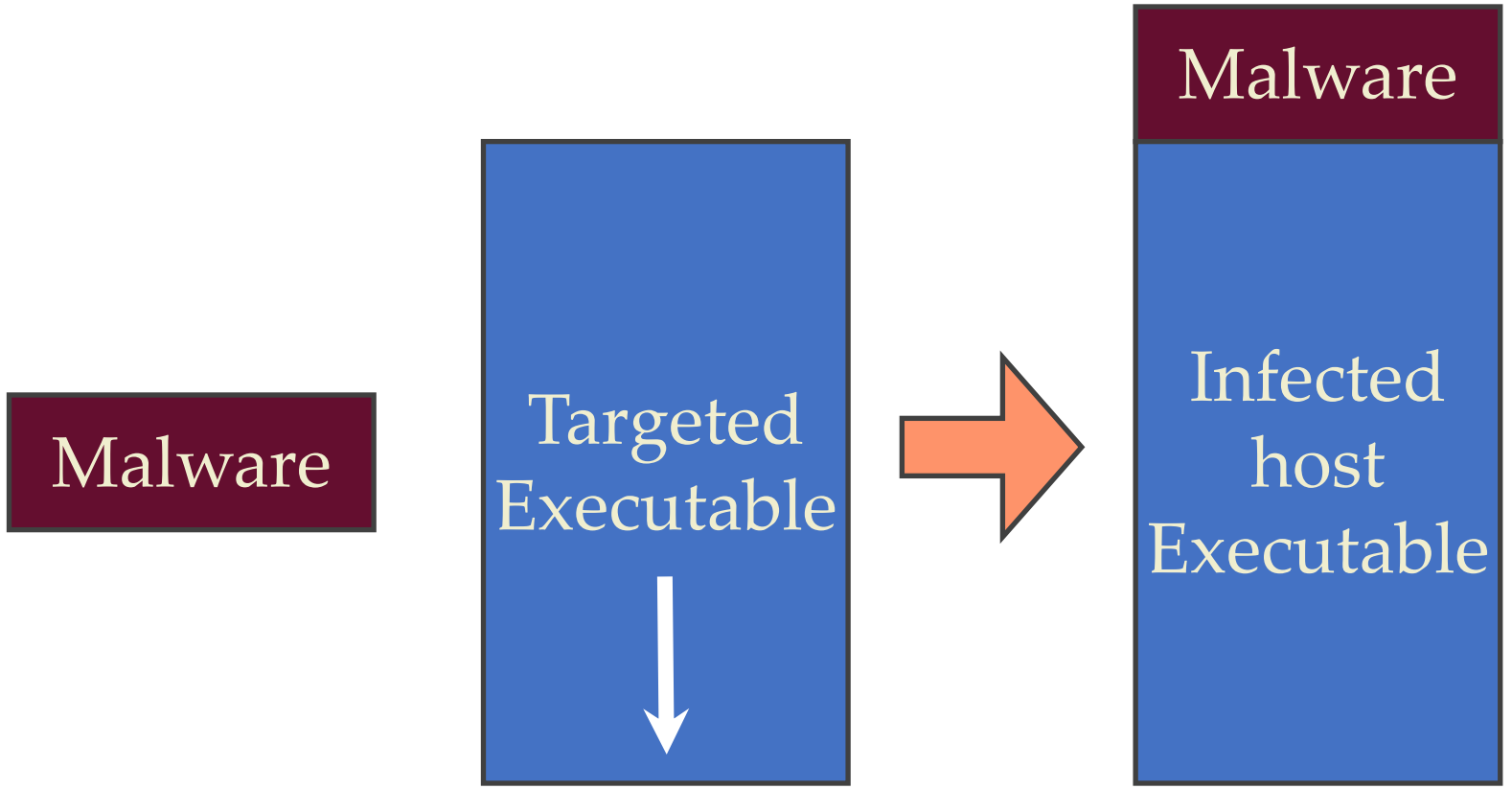
Overwriting malware

Ca' Foscari University of Venice



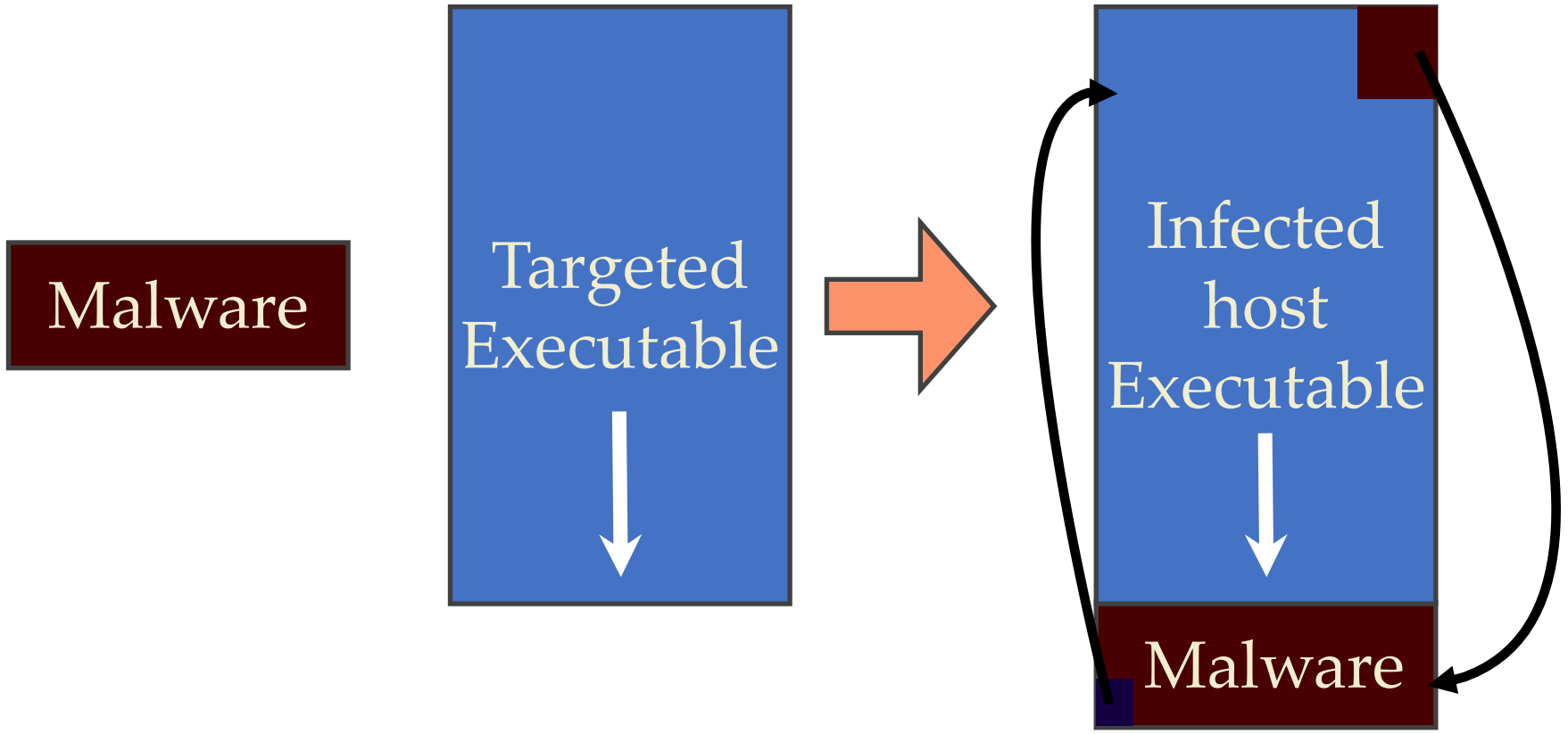
Prepending malware

Ca' Foscari University of Venice



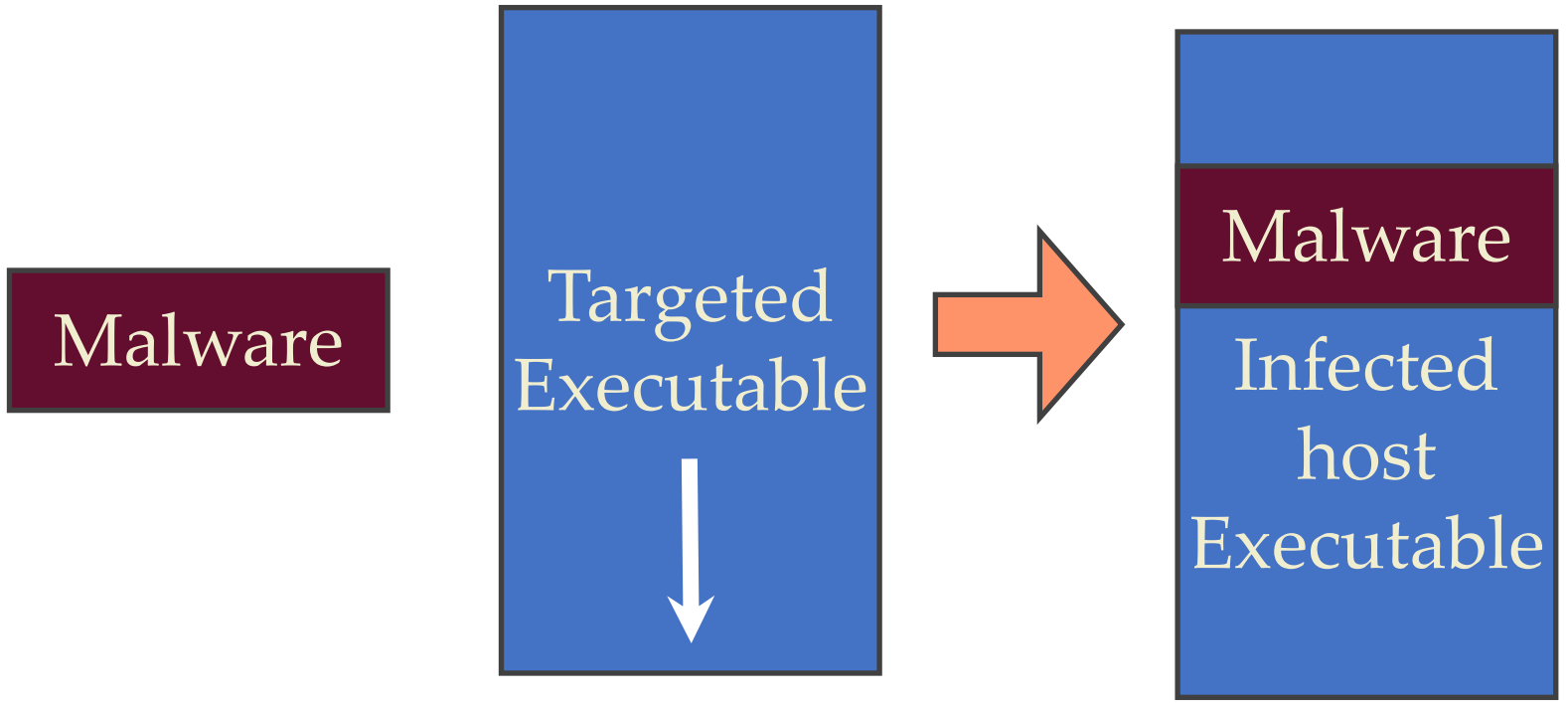
Appending malware

Ca' Foscari University of Venice



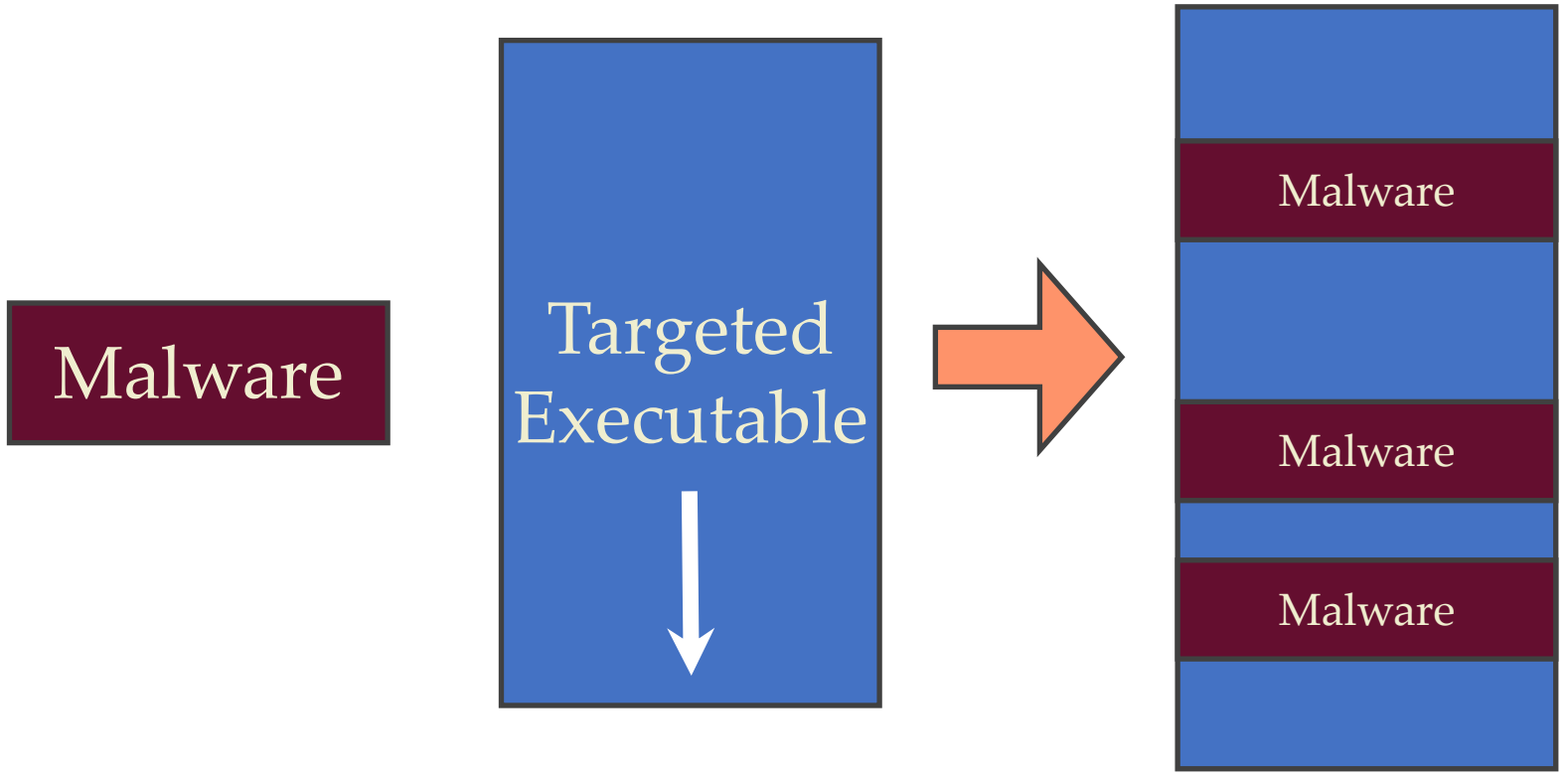
Cavity malware

Ca' Foscari University of Venice



Multi-Cavity malware

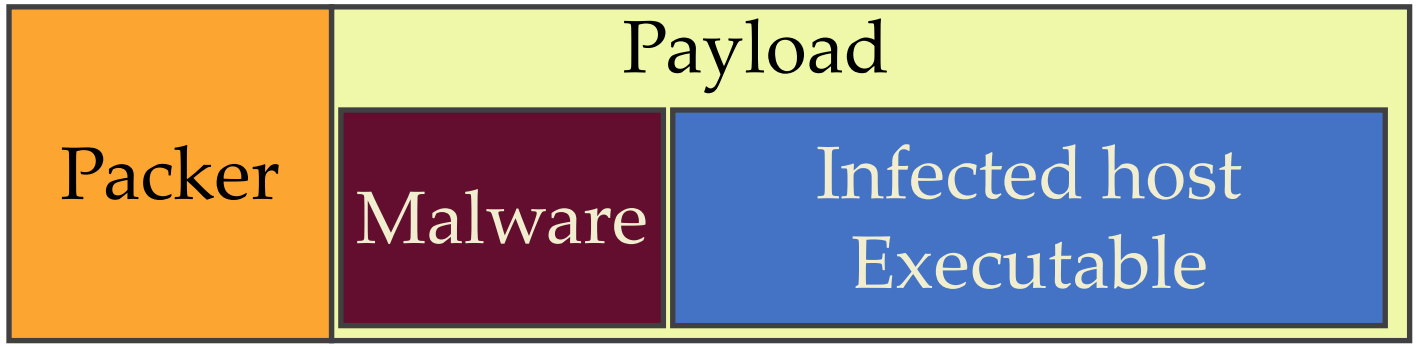
Ca' Foscari University of Venice



Packers

Ca' Foscari University of Venice

- Code eventually decrypted in memory at some point



Packer functionalities

Ca' Foscari University of Venice

- Compress
- Encrypt
- Randomize (polymorphism)
- Anti-debugging technique
- Add junk code
- Anti-VM
- Virtualization

Macro virus

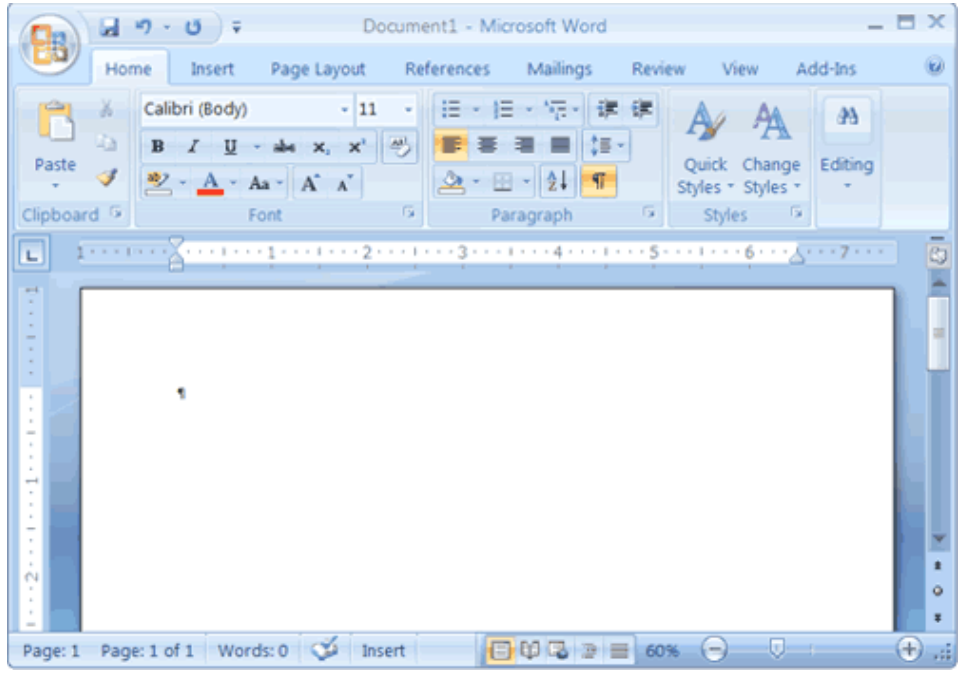
Ca' Foscari University of Venice

- Use the builtin script engine
- Example of call back used (word)
  - AutoExec()
  - AutoClose()
  - AutoOpen()
  - AutoNew()

Document based malware

Ca' Foscari University of Venice

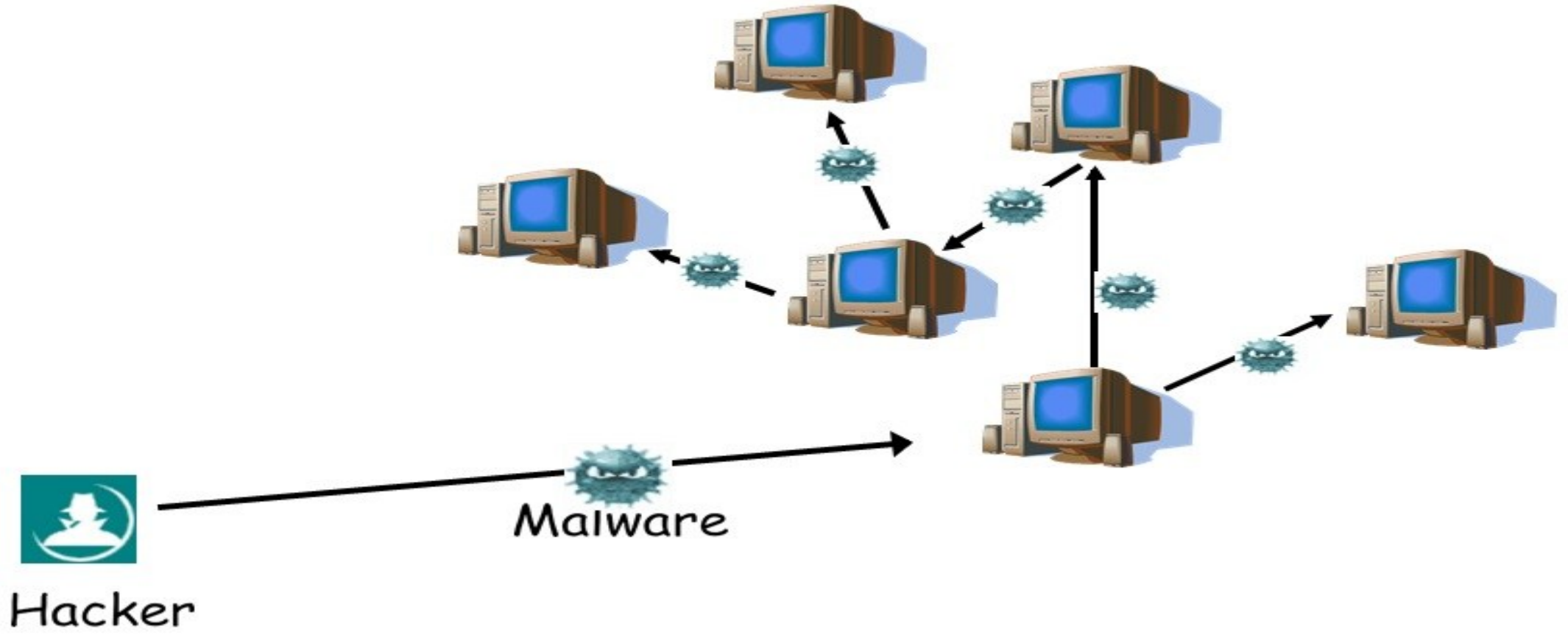
- MS Office
- Open Office
- Acrobat
- Flash
- Ransomware often is a MS-Word macro



Malware Analysis

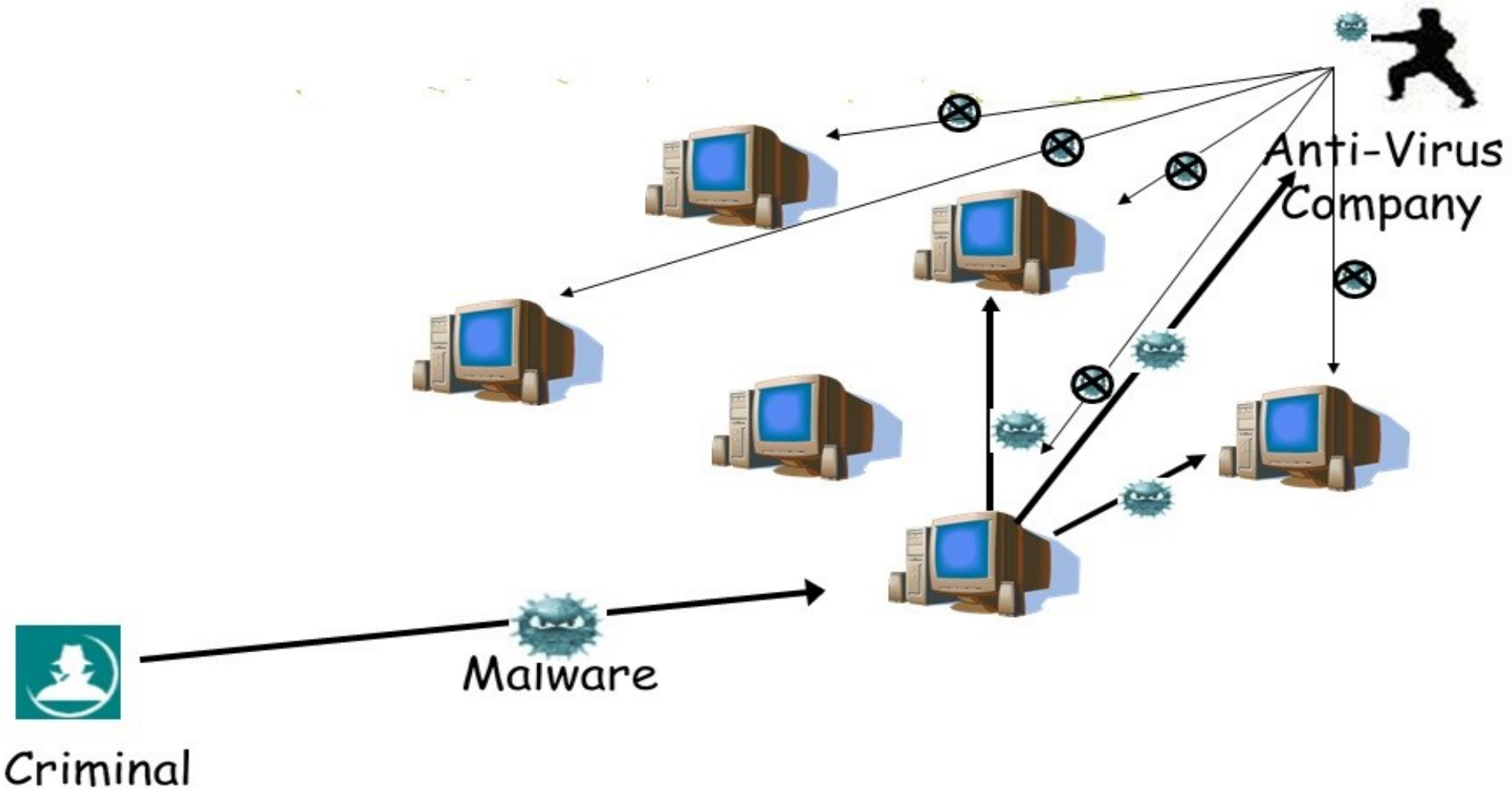
Malware Spread

Ca' Foscari University of Venice

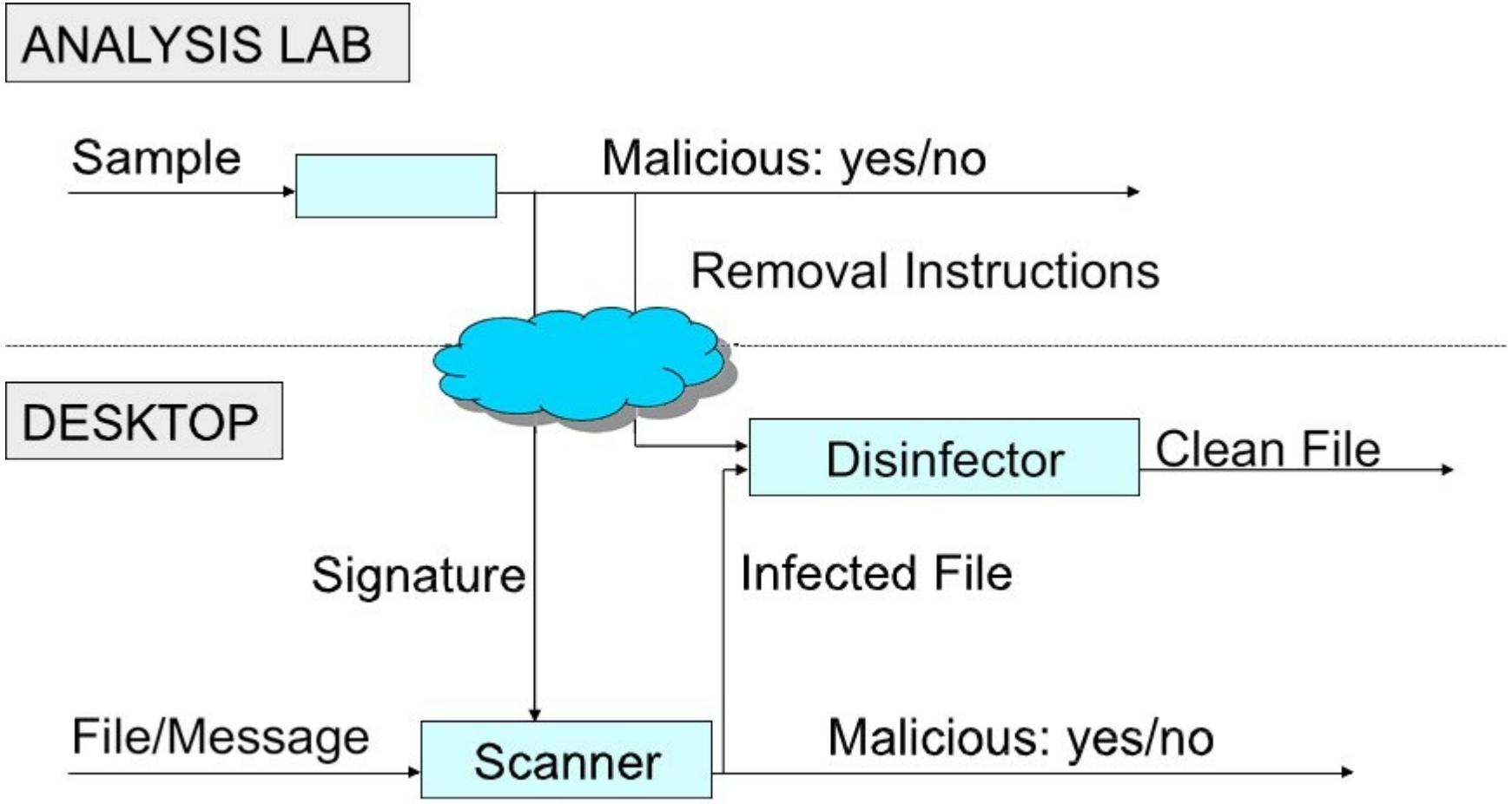


How Malware is detected

Ca' Foscari University of Venice



AV Detection Process

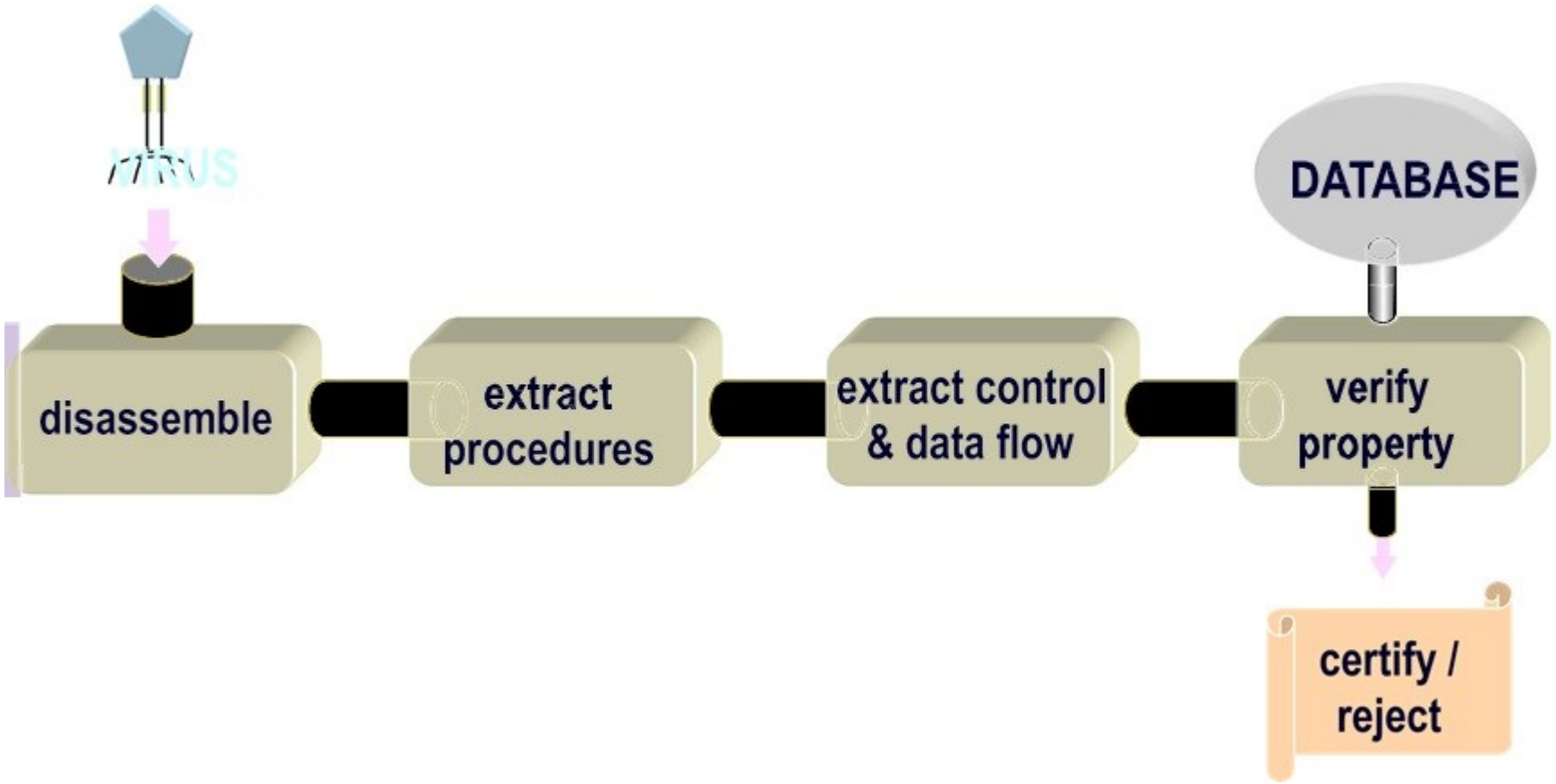


Signature Scanning

- Early viruses had characteristic code patterns known as signatures
- "first generation" scanner created a database of patterns, search files for patterns (McAfee)
- Use data-mining, learning, feature extraction etc. to look for disguised or obfuscated patterns
- Can only scan for known signatures
- Static AV scanners
  - Strings
  - Regular expressions
  - Static behaviour analyser
- Dynamic AV scanners
  - Behaviour Monitors

Typical Analysis Pipeline

Ca' Foscari University of Venice



Static Signature

- Hex strings from virus variants

```
67 33 74 20 73 38 6D 35 20 76 37 61
67 36 74 20 73 32 6D 37 20 76 38 61
67 39 74 20 73 37 6D 33 20 76 36 61
```

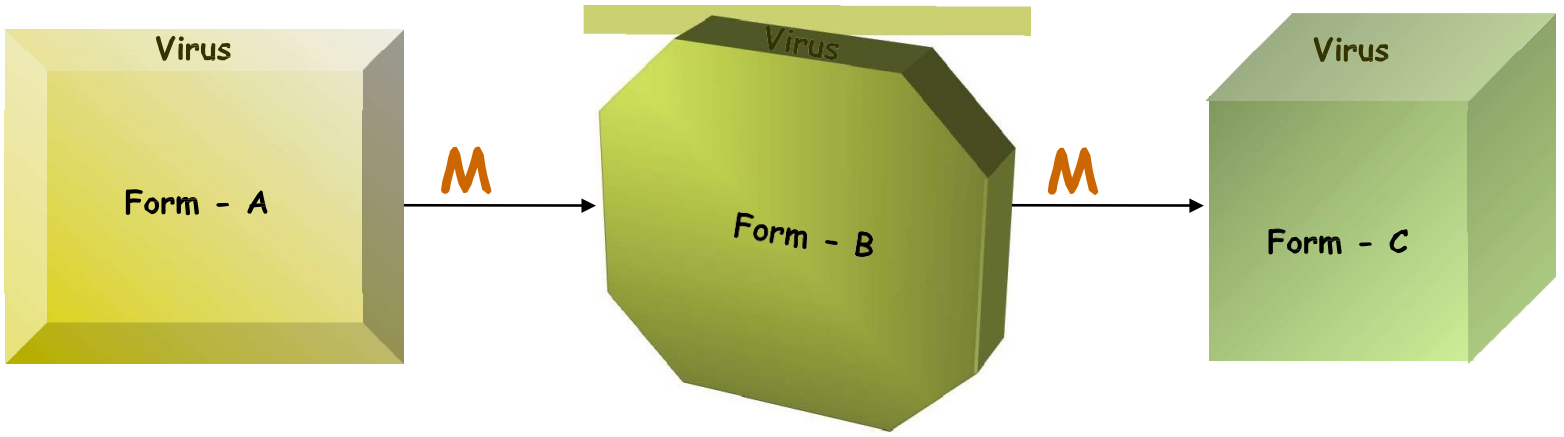
- Hex string for detecting virus

```
67 ?? 74 20 73 ?? 6D ?? 20 76 ?? 61
?? = wildcard
```

Signature Avoiding Viruses

Ca' Foscari University of Venice

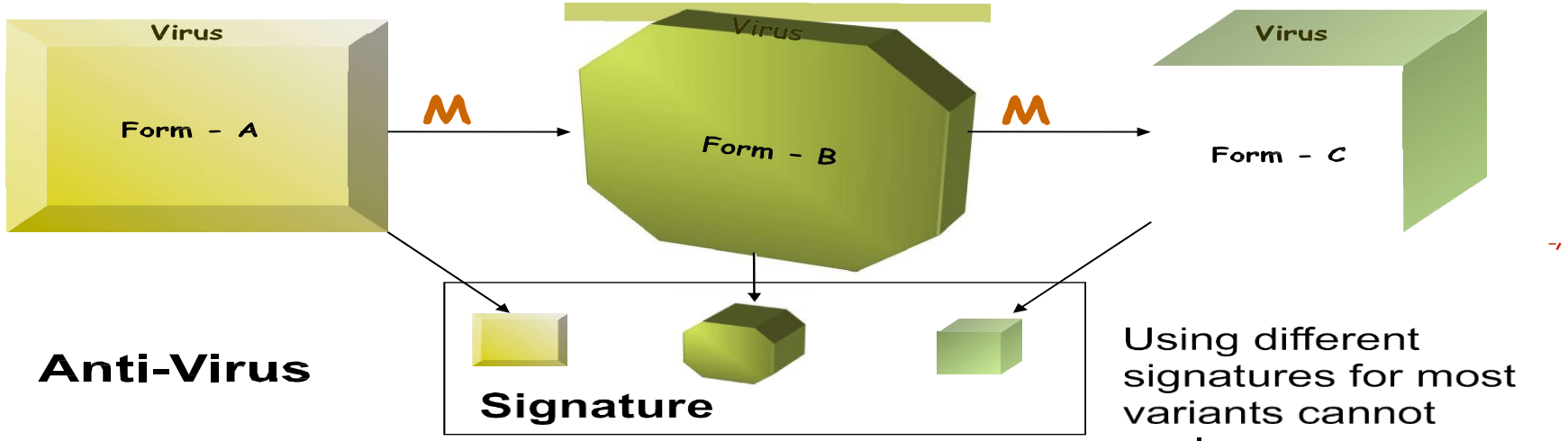
- Polymorphic Virus produces varying but operationally equivalent copies of itself
  - Use alternative but equivalent instructions
  - Gets around signature scanners. Whale virus, 32 variants
- Stealth Virus actively tries to hide all signs of its presence
  - A virus can intercept calls to read a file and return correct values about file sizes etc. Brain Virus



Another Signature Avoiding Virus

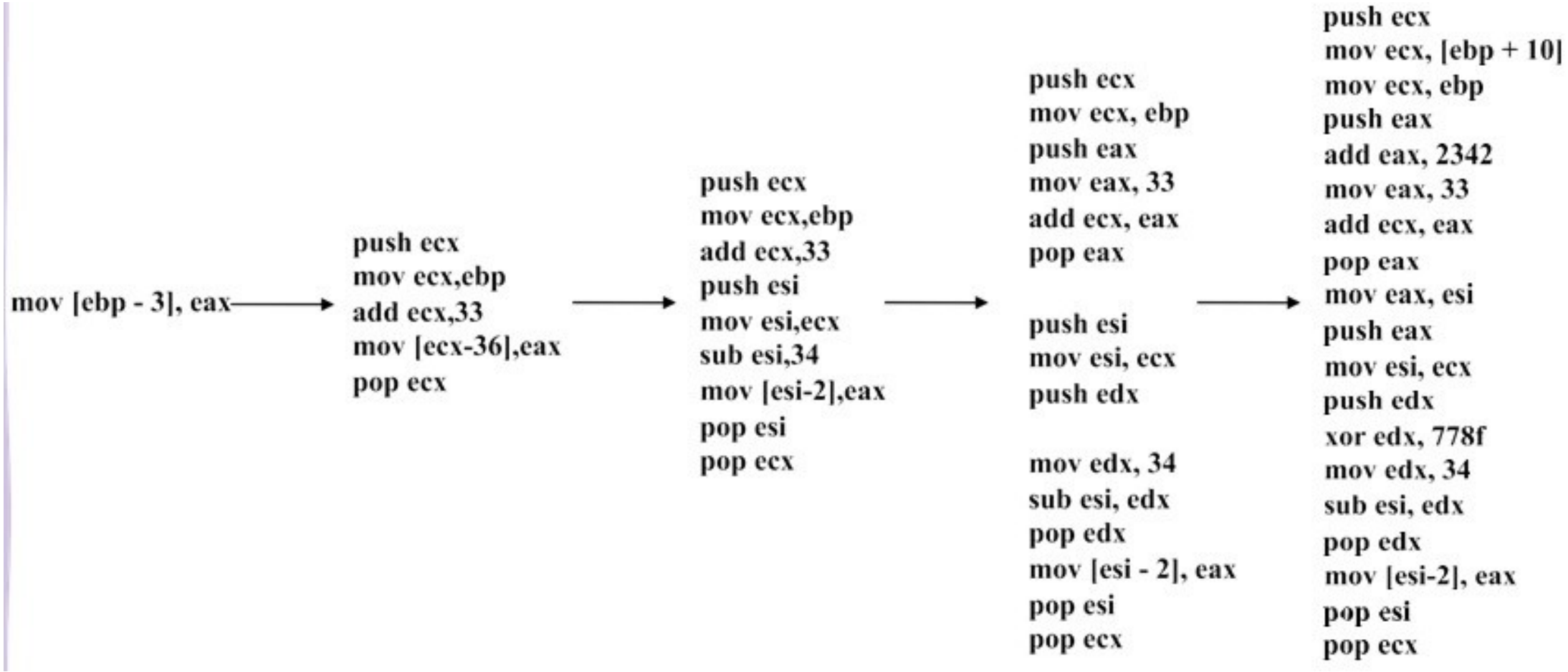
Ca' Foscari University of Venice

- Encrypted Virus stores bulk of self encrypted
  - Small decrypt routine in clear
  - Key stored in clear
- Metamorphic virus: mutates with every infection
  - Similar to polymorphic
  - But this is a complete rewrite



Metamorphism Example

Ca' Foscari University of Venice



Notable Metamorphic Viruses

- Win32.Evol (July 2000)
  - Swaps Instructions with equivalents
  - Insert Junk code between essential instructions
- Win32.Bugbear.B (June 2003)
  - Encrypted body with polymorphic decryptor
  - Spreads via email and shared folders
  - Disarms popular antivirus
  - Install keylogger, backdoor

Other Virus Scanners

- Second generation
  - Use heuristics rather than direct signatures
    - Look for code fragments like encrypt/decrypt loops
  - Use integrity checks to track changes for good binary and configuration file
  - Detect change by comparing checksum

Other Virus Scanners

Ca' Foscari University of Venice

- Third generation
  - Track virus by actions rather than code
  - Identify/notify/prevent anomalous behaviour
    - E.g., installing device driver after visiting a web site
  - E.g, Cisco Security Agent.
  - Host based intrusion detection.
  - Behaviour blocking software.

Other Virus Scanners

- Fourth Generation
  - Use multiple techniques
    - Scanning
    - Access control
    - Behavioural analysis
    - Intercept system calls
  - Analyze program behavior (heuristics)
    - Network access
    - File open
    - Attempt to delete file
    - Attempt to modify the boot sector

Sandbox analysis

Ca' Foscari University of Venice

- Running the executable in a VM
- Observe it
  - File activity
  - Network
  - Memory
- Dealing with a packer
  - Launch the exe in a VM
  - Wait until it is unpacked
  - Dump the memory

Impossibility result

- P is a perfect detection program
- V is a virus
- V can call P
  - if P(V) = true -> halt
  - if P(V) = false -> spread

- It is not possible to build a perfect virus/malware detector (Cohen)

Anti-Antivirus techniques

Ca' Foscari University of Venice

1. Attacking Integrity Checkers

- Intercept open() system calls: open a non-infected backup of the file instead
- Restore system to the original state after the attack
- Infect the system before checksums are computed

2. Attacking Behaviour Monitor

- Act benign if running in emulator
- Obfuscation and redundant/useless system calls

Open Source toolsets

Ca' Foscari University of Venice

- SysAnalyzer
  - Automated malcode analysis system (not a sandbox!)
- Malcode Analyst Pack
  - suite of tools useful for malcode analysts
- VirtualBox
  - x86 and AMD64/Intel64 virtualization product
- BeaEngine
  - disassembler library x86 x86-64 (IA32 and Intel64)
- Libemu
  - x86 shellcode emulation

Databases/Tools

- RE-Google IDA plugin
  - Queries Google Code for information about the functions contained in a disassembled binary
- ClamAV
  - Open source (GPL) antivirus engine
- Malware lookup services
  - VirusTotal, The Malware Hash Registry
  - ThreatExpert, McAfee SiteAdvisor
- Utilities and Assessment Tools
  - McAfee Free Tools, Collaborative RCE Tool Library

Extensible analysis frameworks

Ca' Foscari University of Venice

- Cuckoo Sandbox
  - Modular malware analysis system
- Zero Wine Malware Analysis Tool
  - Research project to dynamically analyse the behaviour of malware
- Malheur
  - Automatic analysis of malware behaviour
- Radare2
  - Open source tools to disassemble, debug, analyse, manipulate binary files

Recent Trends in Malware

Current AV technology

Ca' Foscari University of Venice

- Human Intensive
  - Analysts specialized in specific attacks and technologies (e.g. packers)
  - Knowledge resident in specialists
- High workload
  - Spyware – may have hundreds programs
  - About 8 samples per analyst per day
- Attack: Overwhelm Analysts!
  - Timeframe: 5 minutes to several weeks to write a virus signature
  - Several hours for testing false positives/negatives

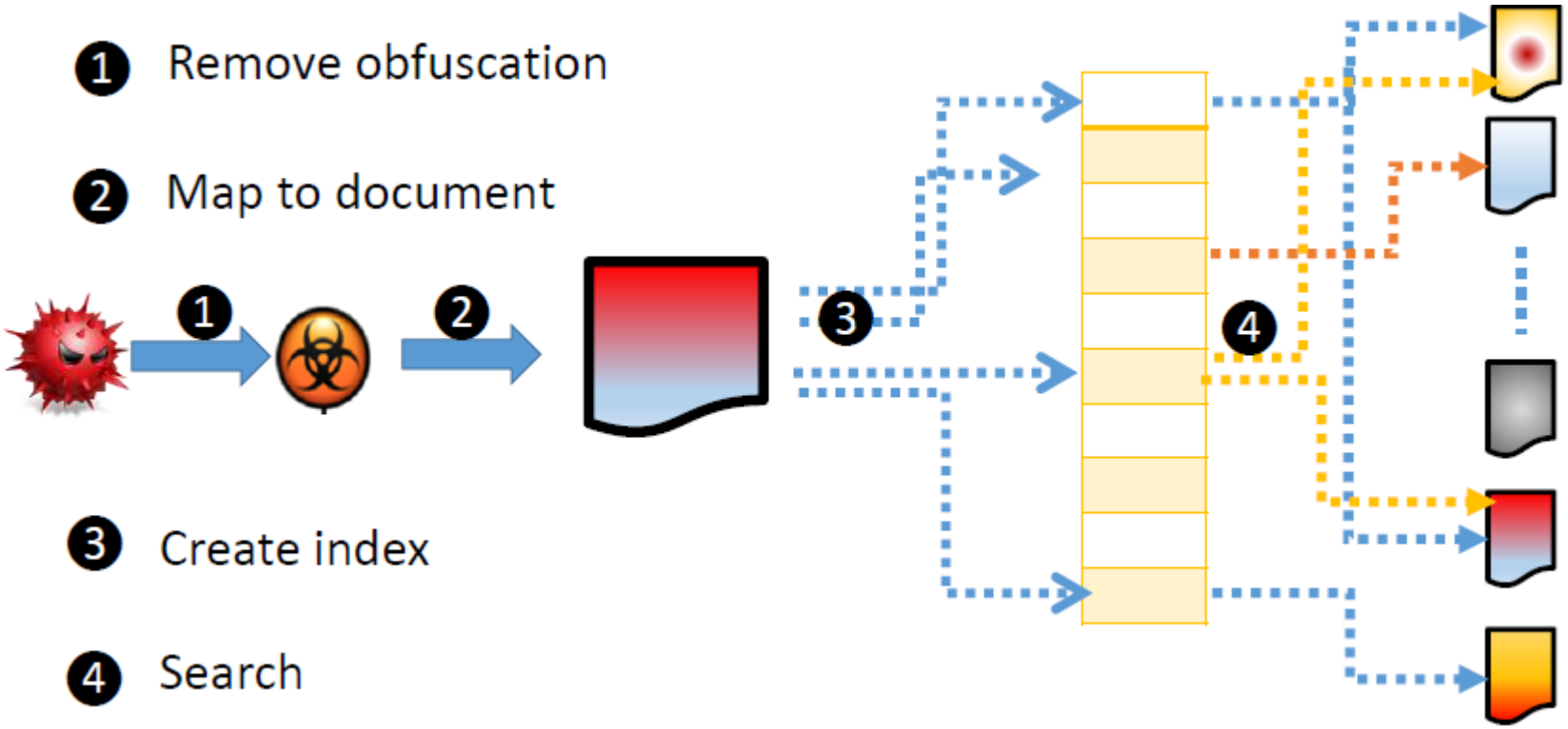
Signature Updates and Automation

Ca' Foscari University of Venice

- More than 1.5 Billion virus definition updated every day
- Up to 60 Terabytes of data sent every day
- Employ incremental update technologies and compression
  - 85% reduction
- Automation
  - Automatic filtering of customers' submissions (95%)
- Analysis
  - Auto-replication of threats in VMs
  - Auto-fingerprint generation
- Quality Assurance: automated parallel testing
  - huge set of files for false-positive testing

A “Google” for Malware

Ca’ Foscari University of Venice



- Miles, Craig, Arun Lakhotia, Charles LeDoux, Aaron Newsom, and Vivek Notani. "VirusBattle: State-of-the-art malware analysis for better cyber threat intelligence." In *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, pp. 1-6. IEEE, 2014.

Malware as Document

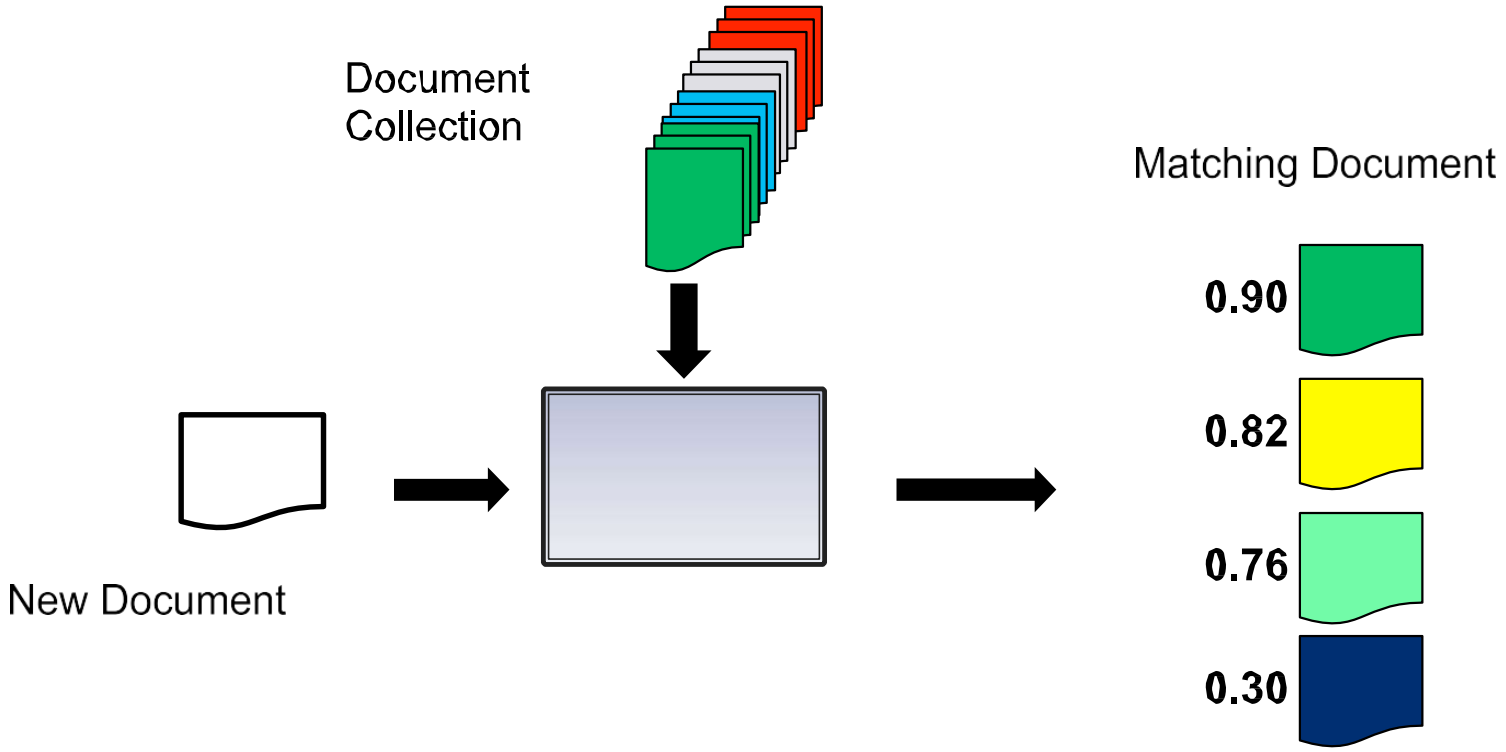
Ca' Foscari University of Venice

1. Model Documents
   - Bag of features model
   - Ex: k-consecutive words
2. Define Similarity Function
3. Choose/Create Algorithm

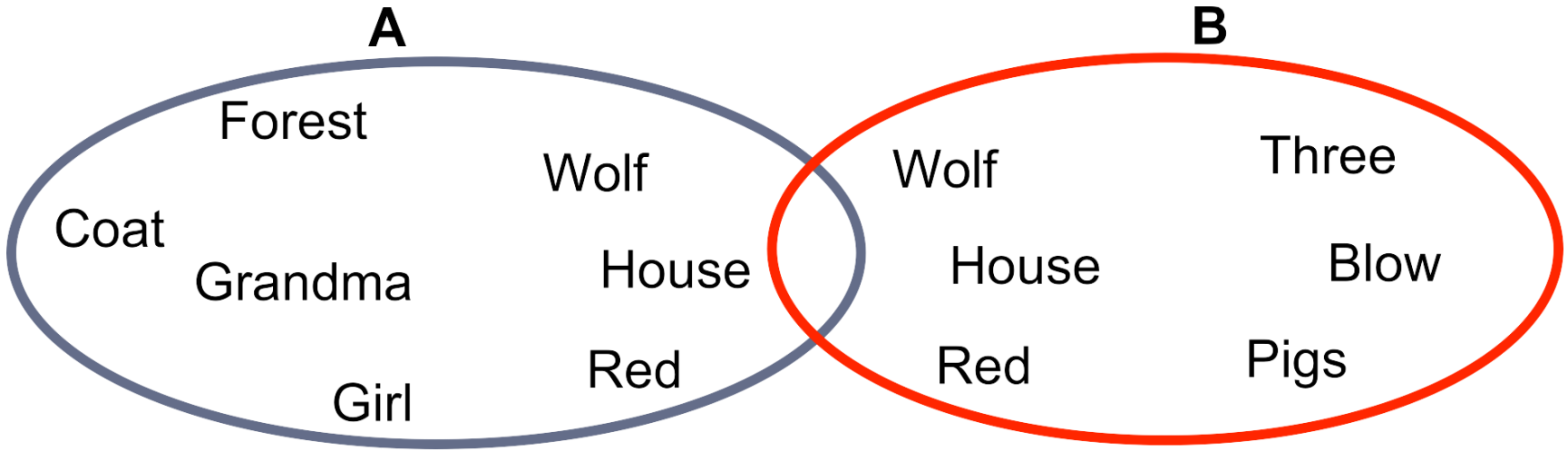
Info Retrieval: Nearest Match

Ca' Foscari University of Venice

- Nearest Match (unsupervised)



Define Similarity Function



$$\text{Similarity}(A,B) = |A \cap B| / |A \cup B|$$

$$= 3 / 10$$

$$= 0.3$$

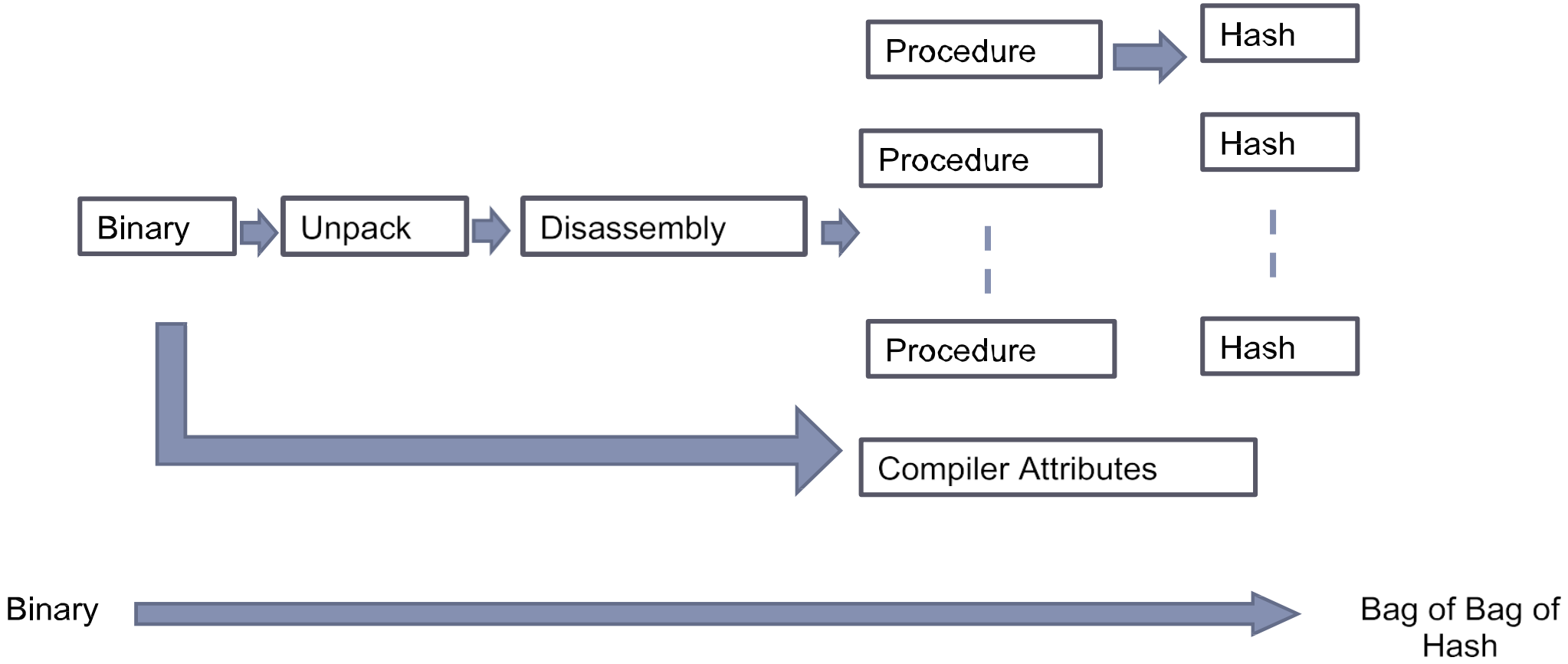
Choose/Create Algorithm

Ca' Foscari University of Venice

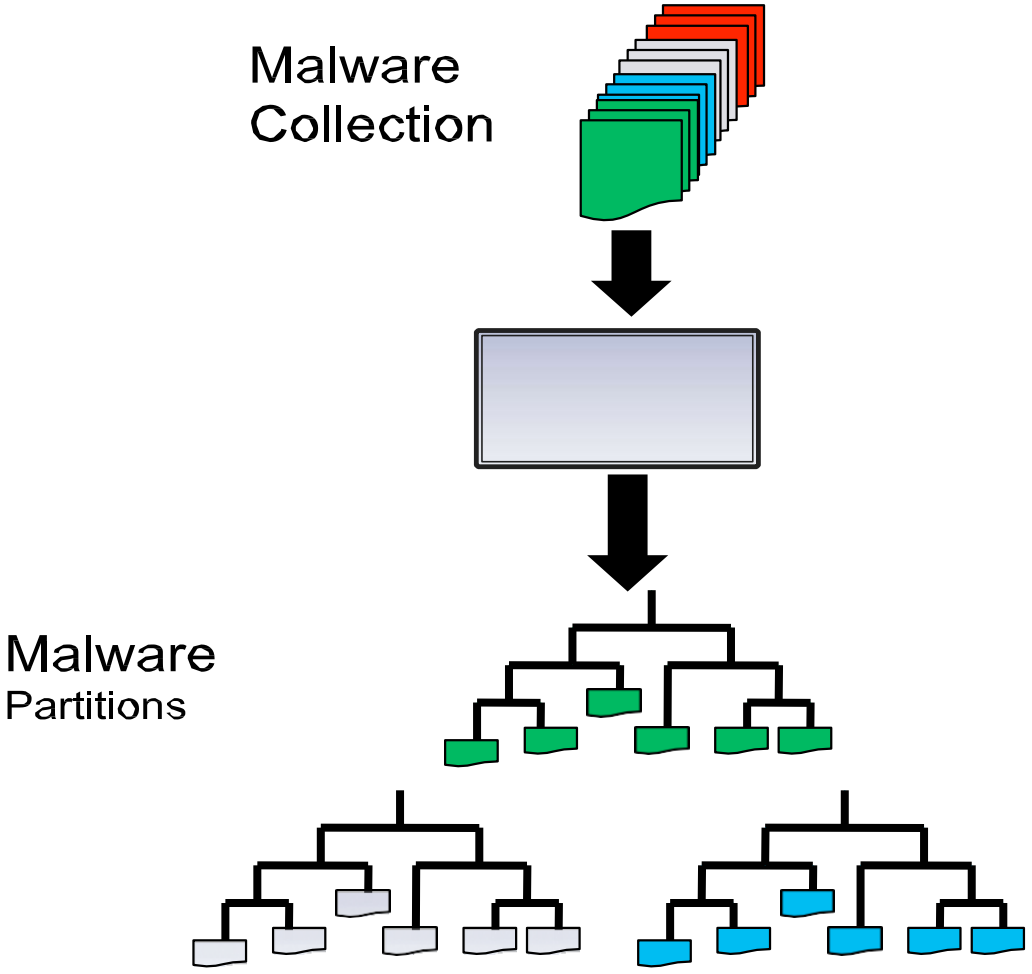
- Supervised Learning
  - Neural Networks
  - Bayesian Statistics
  - Inductive Learning
  - Support Vector Machines
  - Regression
- Unsupervised Learning
  - K-Means Clustering
  - Hierarchical Clustering
  - K-Nearest Neighbour
- Semi-supervised
  - Used some labels to seed clusters

Malware as Document

Ca' Foscari University of Venice



Partition collection

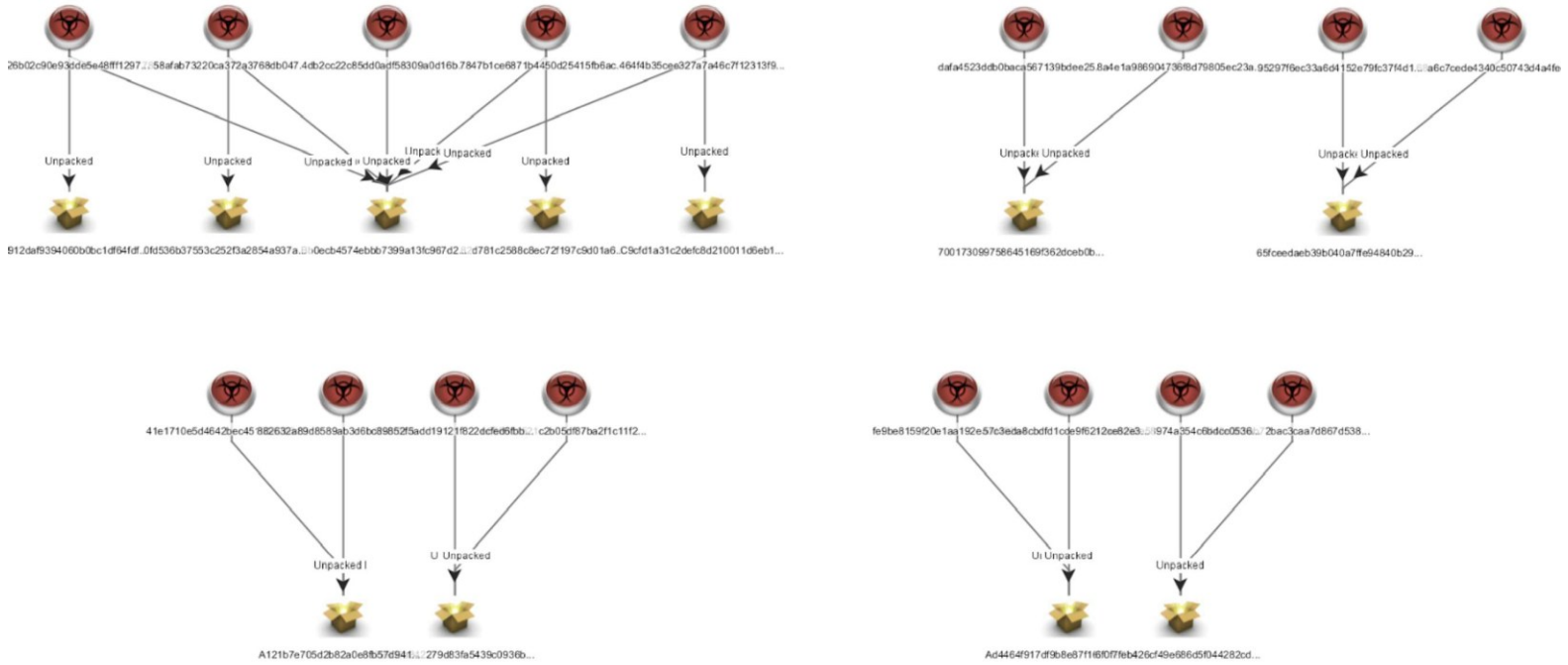


Unpacking

Ca' Foscari University of Venice

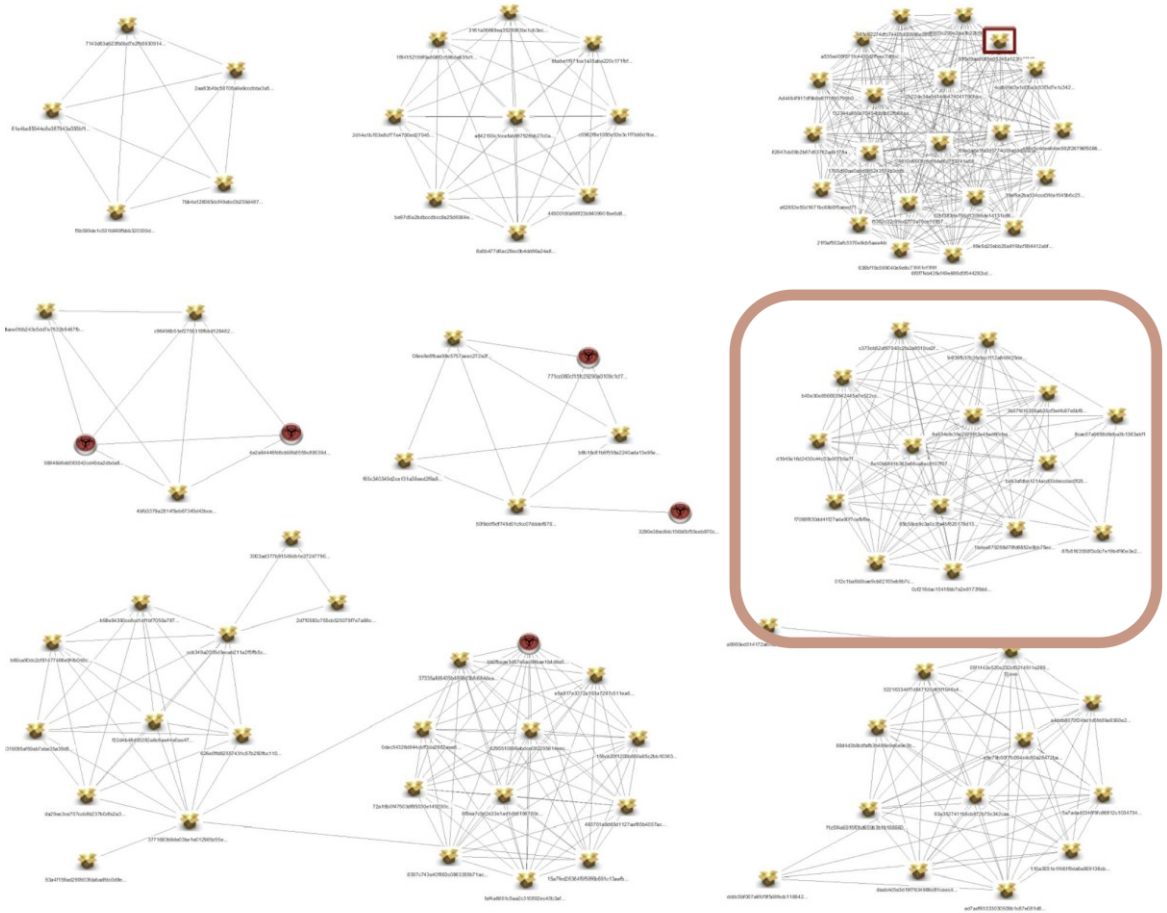
- Run the program in a VM and watch its execution
- Determine when it's completed unpacking
- Create an executable from memory image

Similar Binaries after Unpacking



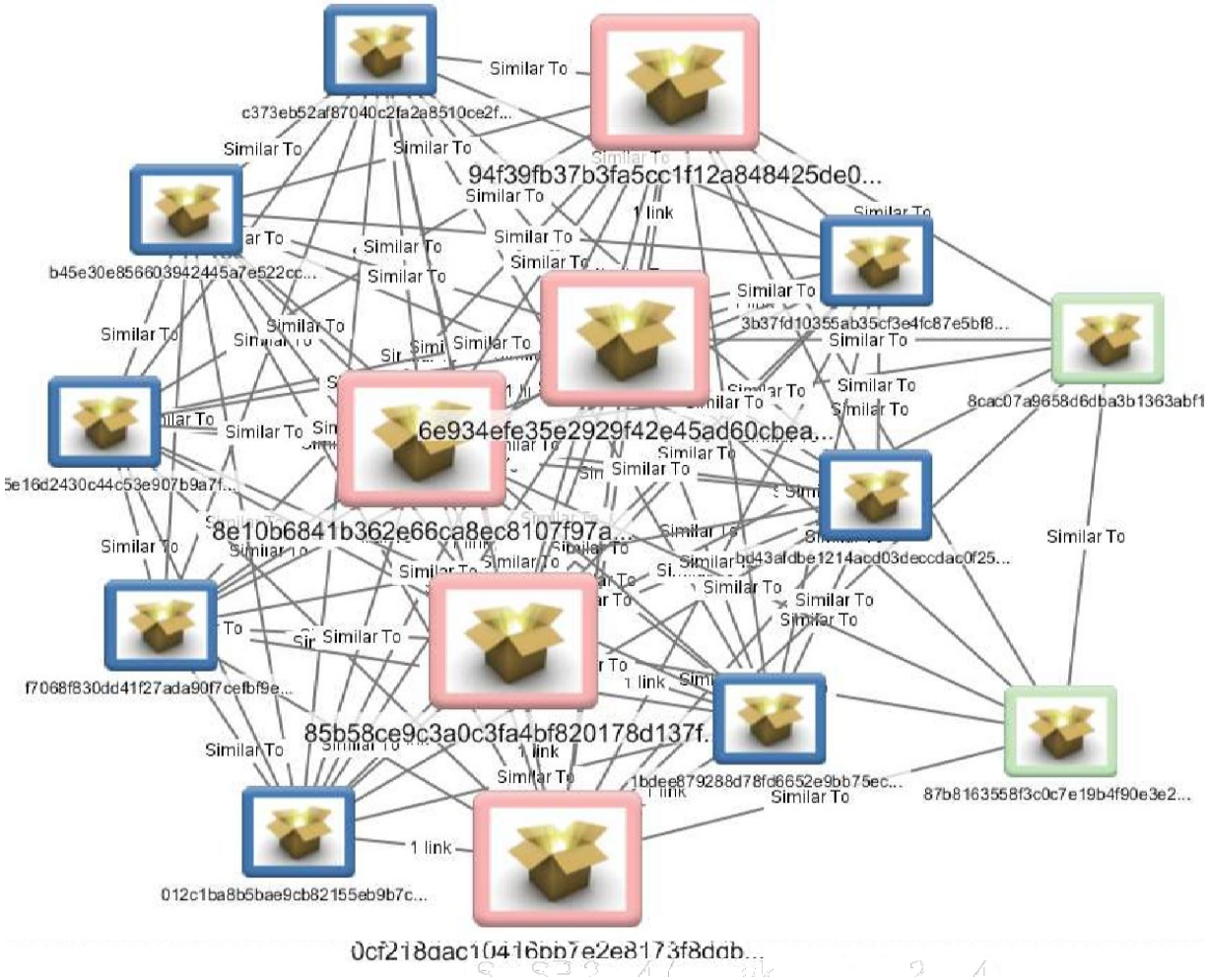
Different Binaries mapped to same MD5 after unpacking

Clusters found



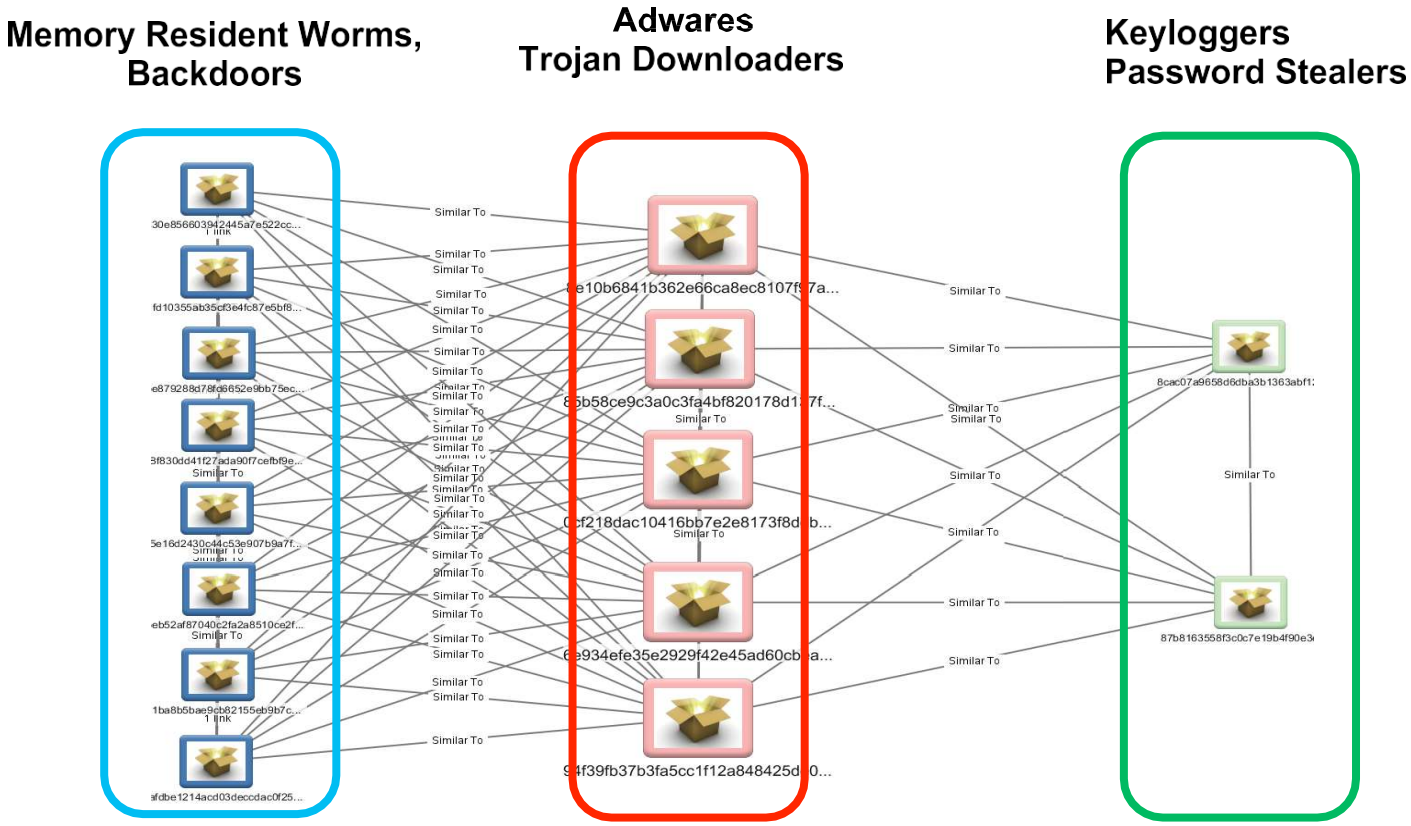
Complete Subgraphs

Ca' Foscari University of Venice



Reorganize

Ca' Foscari University of Venice

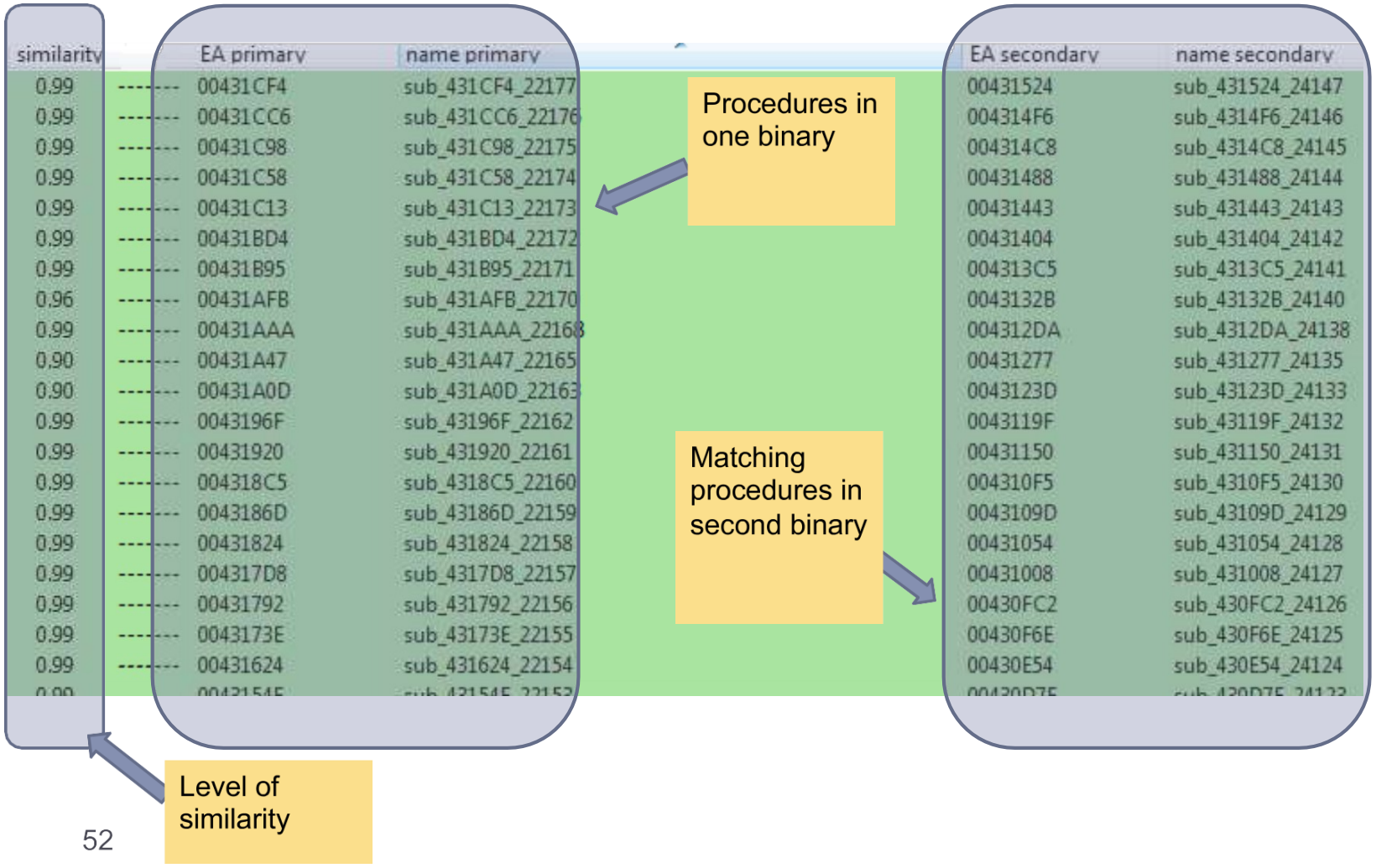


Validating Clusters using Bindiff

Ca' Foscari University of Venice

- Select a pair of similar binaries (ex. matched by VirusBattle)
- Perform side-by-side comparison using BinDiff
  - BinDiff is an interactive tool for comparing too binaries
  - VirusBattle helps in locating similar binaries in a large collection

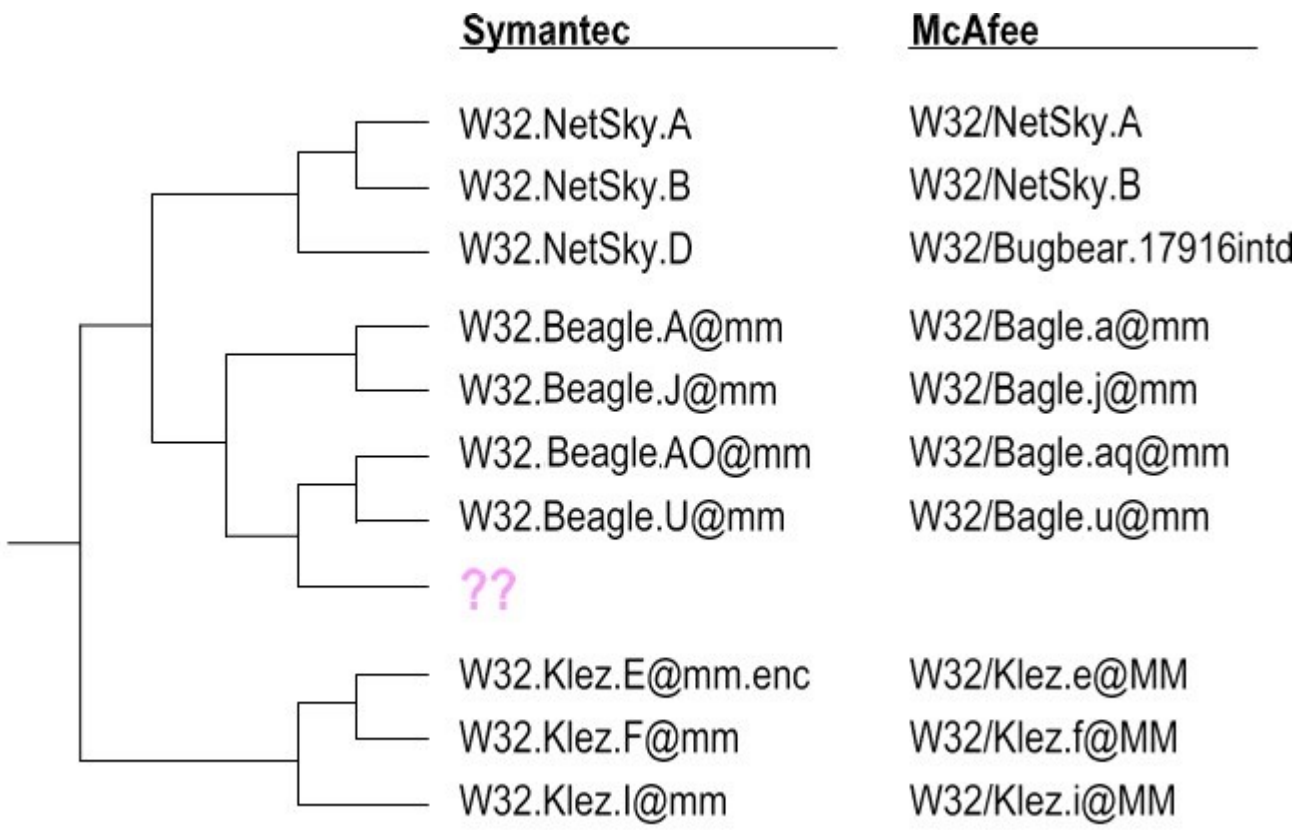
Investigate matches in two binaries



52

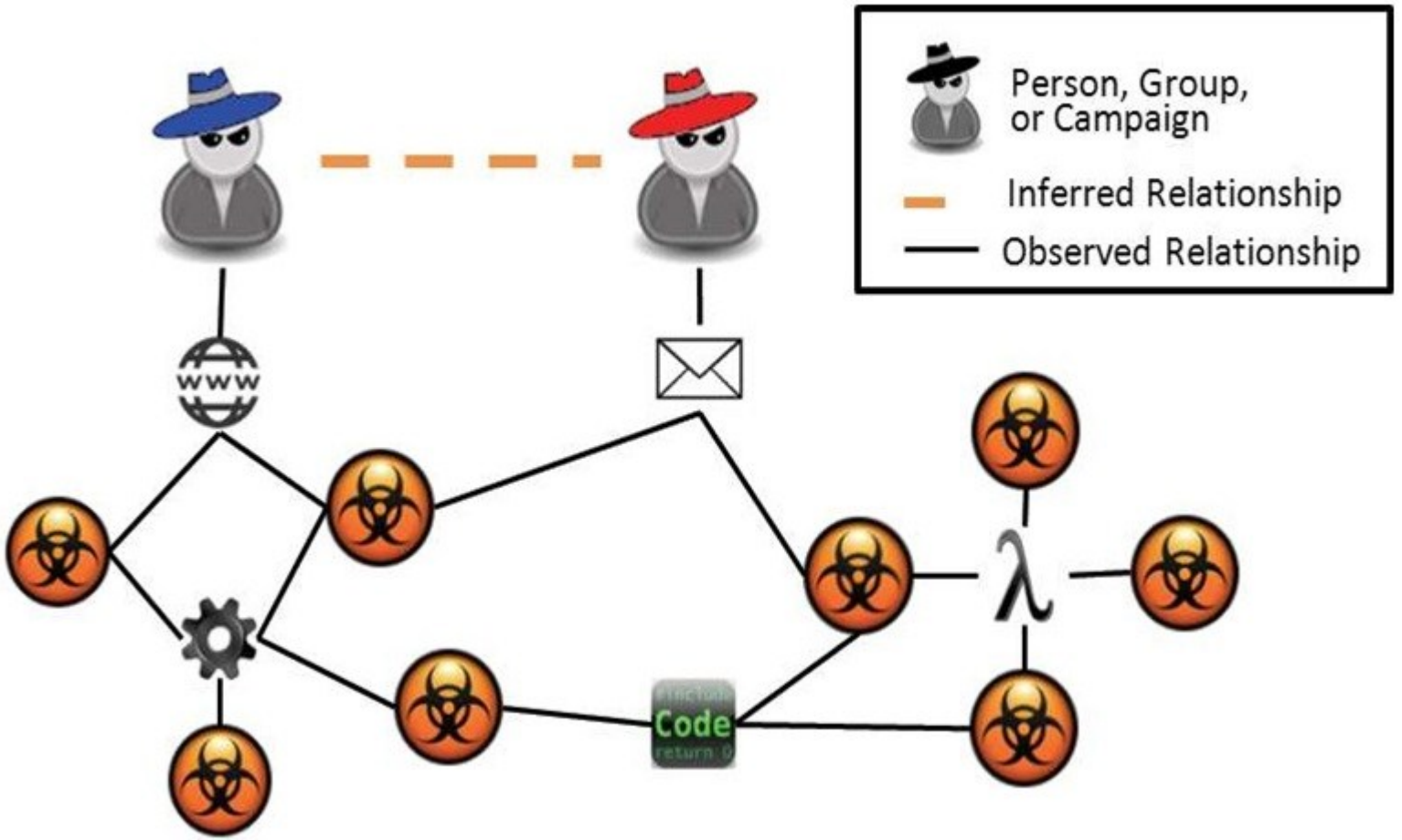
Virus Phylogeny

Ca' Foscari University of Venice



Connecting Actors from Malware

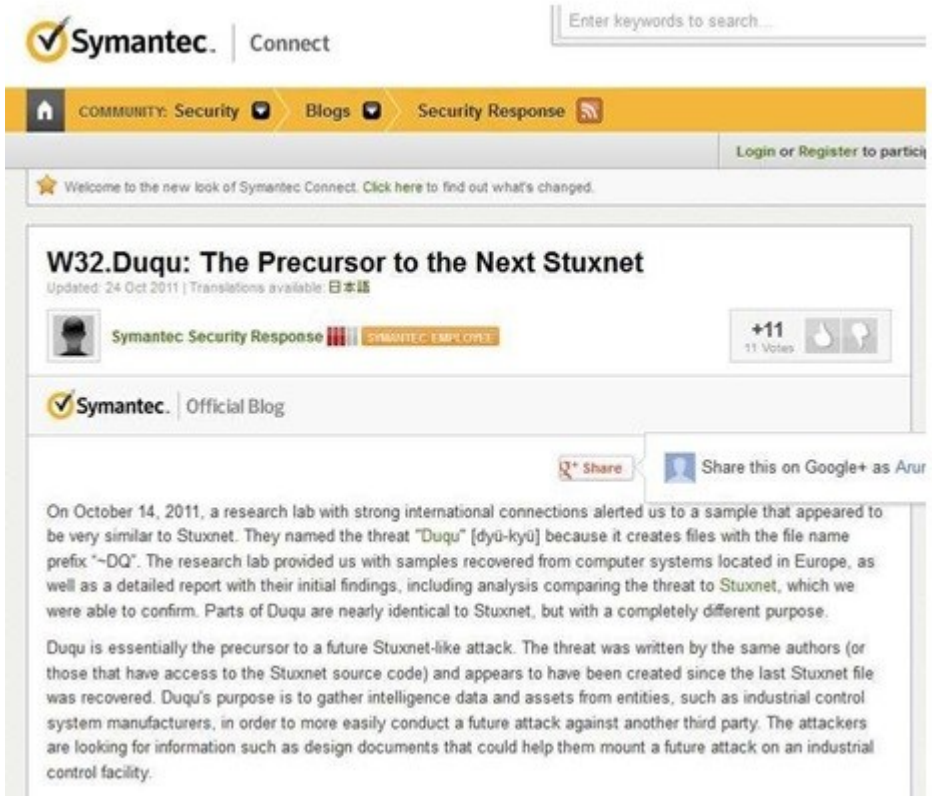
Ca' Foscari University of Venice



Code connects Actors

Ca' Foscari University of Venice

- Stuxnet, Duqu come from the same factory
- They were written on the same platform by the same group of programmers
- …linked specific portions of code



Summary

Ca' Foscari University of Venice

- Malware Variant generation process
- Managing very large collection of malware
  - Use information retrieval
  - Derive features from semantics
  - Normalize representation to enable string comparison
- Connect Actors through shared code

Memory based attacks

Buffer overflow

Ca' Foscari University of Venice

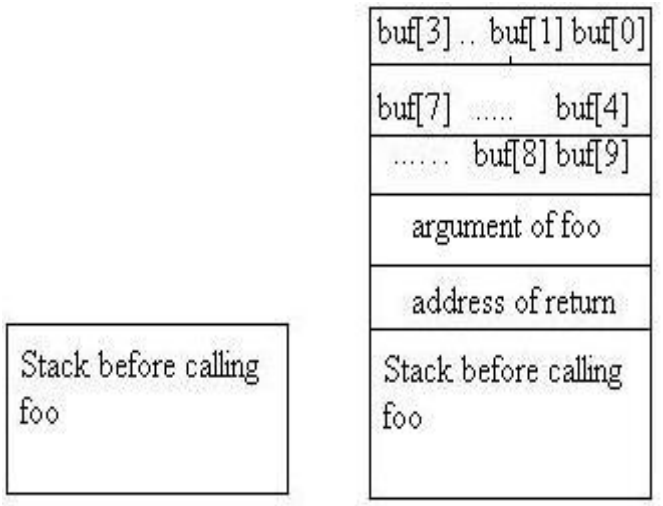
- The buffer overflow attack uses input to a poorly implemented application, typically with root / admin privileges.
- The buffer overflow attack results from input that is longer than the developer intended.
- The input contains binary code that can be run in the stack memory

Stack Memory

- A push stores a new data item on top of the stack, a pop removes the top item.
- The stack is used to store local variables and the return address of a function.
- The calling function pushes the return statement onto the stack, containing the return address
- Then the called function pushes zeroes on the stack to store its local variable.
- Since foo has a variable buf, this means there will be space for 10 characters allocated.

```
int main(int argc, char* argv[]) {
    foo(argv[1]);
                    return 0;
}
```

```
void foo(const char* input) {
    char buf[10];
    printf("Hello World\n");
}
```



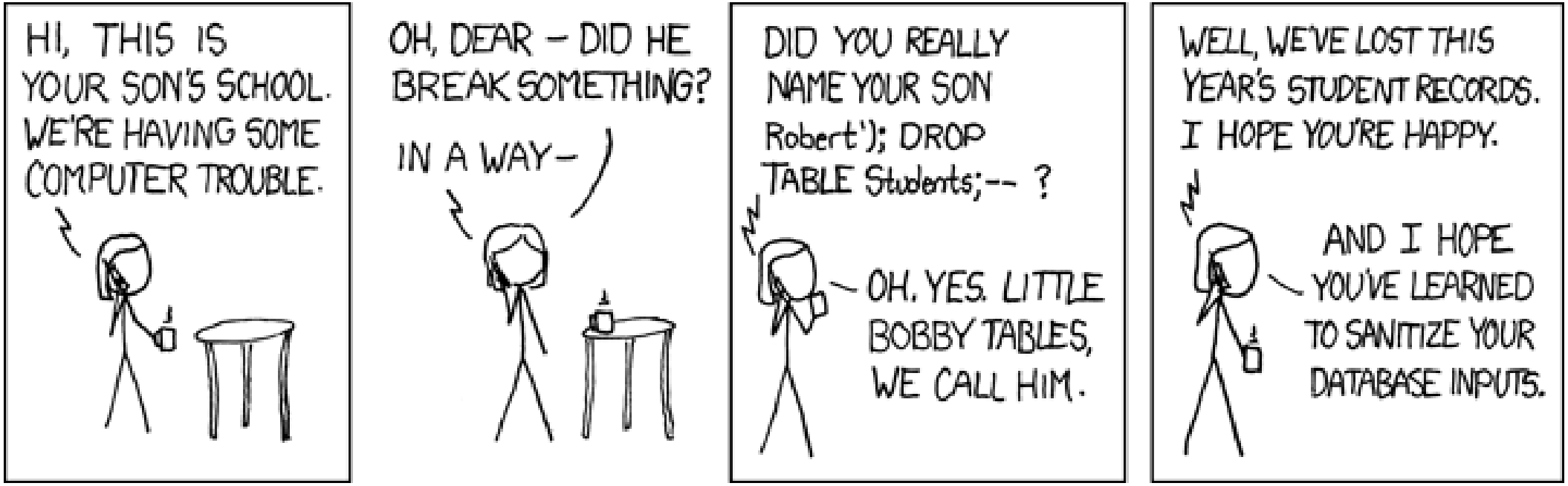
Buffer Overflow: Causes and Cures

Ca' Foscari University of Venice

- Typical memory exploit involves code injection
  - Put malicious code at a predictable location in memory, usually masked as data
  - Trick vulnerable program into passing control to it
    - Overwrite saved EIP, function callback pointer, etc.
- Idea: prevent execution of untrusted code
  - Make stack and other data areas non-executable
    - Note: messes up useful functionality (e.g., ActionScript)
  - Digitally sign all code
  - Ensure that all control transfers are into a trusted, approved code image

SQL injection vulnerability

Ca' Foscari University of Venice



W⊕X

Ca' Foscari University of Venice

- "Write xor eXecute"
- Mark all writeable memory locations as non-executable
  - Example: Microsoft's DEP (Data Execution Prevention)
  - This blocks all code injection exploits
- Hardware support
  - AMD "NX" bit, Intel "XD" bit (in post-2004 CPUs)
  - Makes memory page non-executable
- Widely deployed
  - Windows (since XP SP2), Linux (via PaX patches), OpenBSD, OS X (since 10.5)

W^X history

Ca' Foscari University of Venice

- W^X was first implemented in OpenBSD 3.3, released May 2003.
- 2004, Microsoft introduced a similar feature called DEP (Data Execution Prevention) in Windows XP
- Similar features are available for other operating systems, including the PaX and Exec Shield patches for Linux, and NetBSD's implementation of PaX.
- 2012 Microsoft extended from userland to the Windows kernel on the x86 and ARM architectures
- 2015, W^X was added in the OpenBSD kernel on the AMD64 architecture.
- 2016, W^X was fully implemented on NetBSD's AMD64 kernel and partially on the i386 kernel.
- 2016, Firefox's virtual machine for JavaScript also implements the W^X policy.
- From 2021, .NET 6.0 uses W^X

What Does W⊕X Not Prevent?

- Can still corrupt stack …
  - … or function pointers or critical data on the heap
- As long as "saved EIP" points into existing code, W⊕X protection will not block control transfer
- This is the basis of return-to-libc exploits
  - Overwrite saved EIP with address of any library routine, arrange memory to look like arguments
- Does not look like a huge threat
  - Attacker cannot execute arbitrary code
  - … especially if system() is not available

return-to-libc

- Overwritten saved EIP need not point to the beginning of a library routine
- Any existing instruction in the code image is fine
  - Will execute the sequence starting from this instruction
- What if instruction sequence contains RET?
  - Execution will be transferred… to where?
  - Read the word pointed to by stack pointer (ESP)
    - Guess what? Its value is under attacker's control!
  - Use it as the new value for EIP
    - Now control is transferred to an address of attacker's choice!
  - Increment ESP to point to the next word on the stack

ret2libc

Ca' Foscari University of Venice

- In a standard stack-based buffer overflow, an attacker writes their shellcode into the vulnerable program's stack and executes it on the stack.
- In newer systems the program's stack is protected (NX bit is set) and attackers can no longer execute their shellcode from the vulnerable program's stack.
  - return-to-libc technique enables attackers to bypass the NX bit protection
  - subvert program's execution flow by re-using existing executable code from the standard C library shared object (/lib/i386-linux-gnu/libc-*.so)
  - that is already loaded and mapped into the vulnerable program's virtual memory space, similarly like ntdll.dll is loaded to all Windows programs.

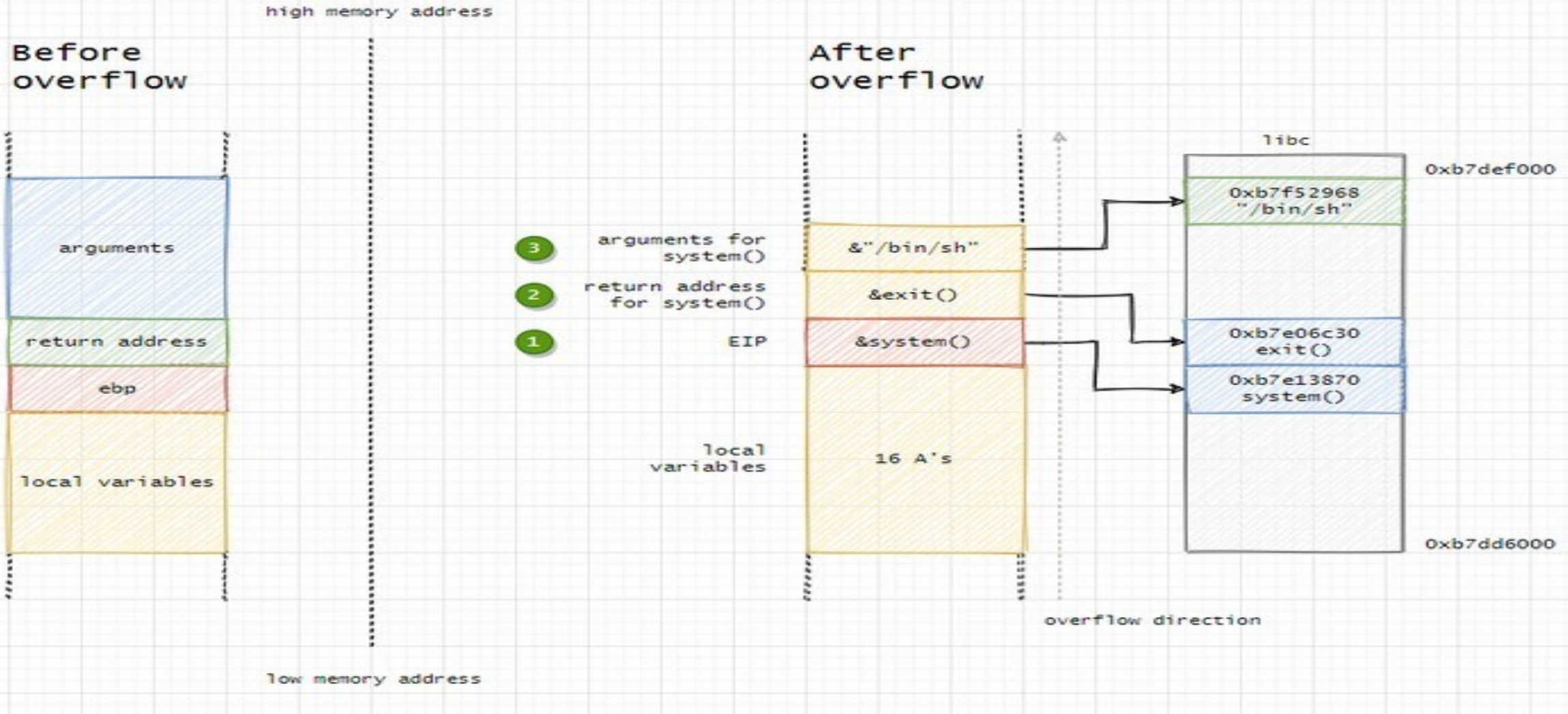
ret2libc

Ca' Foscari University of Venice

- In Stack-based overflow with no stack protection, attack overwrites the return address of the function with address of the shellcode
- ret2libc is similar but the return address is overwritten with a memory address that points to the function *system(const char *command)* that is in the libc library,
- When the overflowed function returns, the vulnerable program is forced to jump to the system() function and execute the shell command that was passed to the system() function as the *command argument as part of the supplied shellcode

Stack memory layout of the 32-bit vulnerable program when using ret-to-libc technique

Ca' Foscari University of Venice



Chaining RETs for Fun and Profit

Ca' Foscari University of Venice

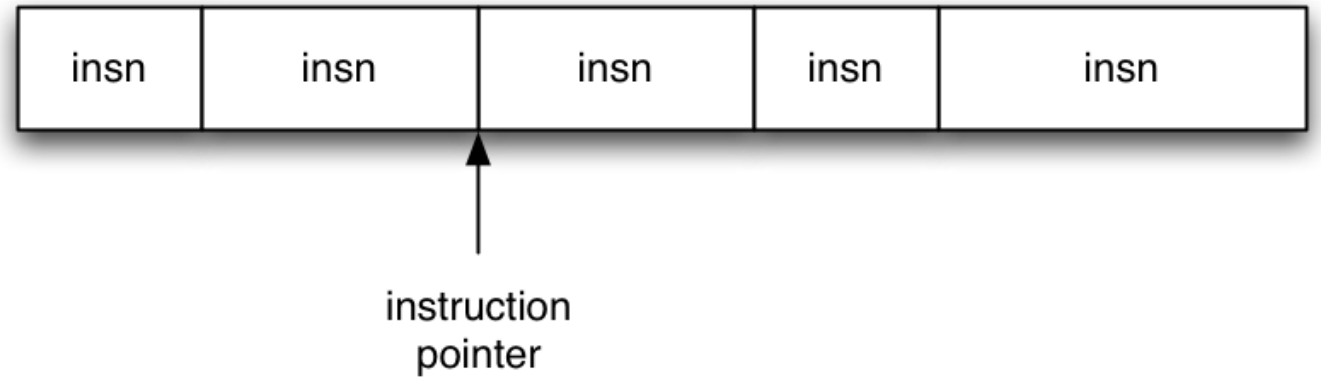
- Can chain together sequences ending in RET
  - Krahmer, "x86-64 buffer overflow exploits and the borrowed code chunks exploitation technique" (2005)
- What is this good for?
- Answer: everything
  - Turing-complete language
  - Build "gadgets" for load-store, arithmetic, logic, control flow, system calls
  - Attack can perform arbitrary computation using no injected code at all!

Hovav Shacham. 2007. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In Proceedings of the 14th ACM conference on Computer and communications security (CCS '07).

Ordinary Programming

Ca' Foscari University of Venice

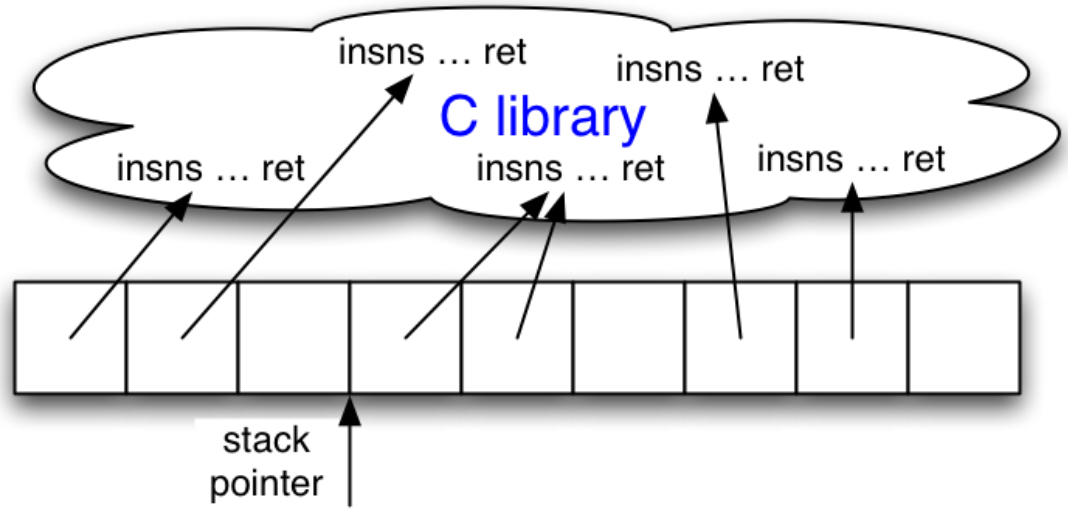
- Instruction pointer (EIP) determines which instruction to fetch and execute
- Once processor has executed the instruction, it automatically increments EIP to next instruction
- Control flow by changing value of EIP



Return-Oriented Programming

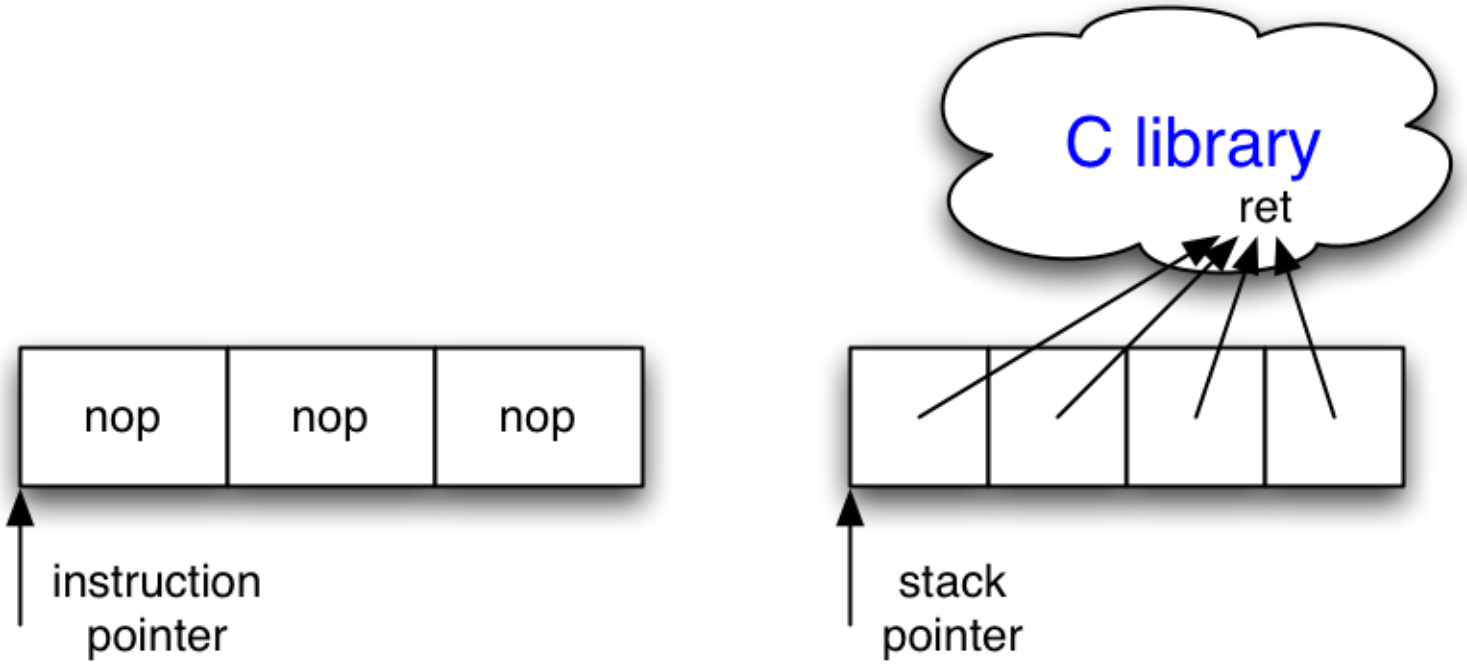
Ca' Foscari University of Venice

- Stack pointer (ESP) determines which instruction sequence to fetch and execute
- Processor doesn't automatically increment ESP
  - But the RET at end of each instruction sequence does



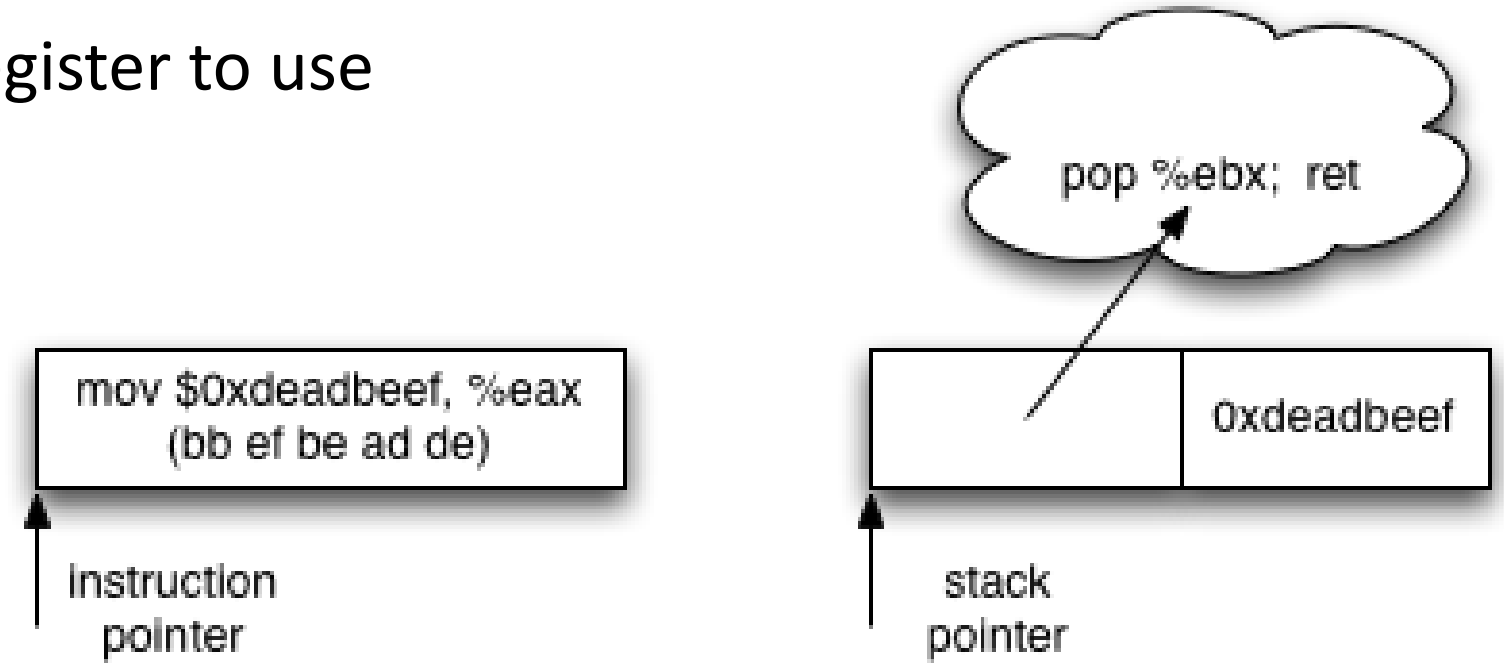
No-ops

- No-op instruction does nothing but advance EIP
- Return-oriented equivalent
  - Point to return instruction
  - Advances ESP



Immediate Constants

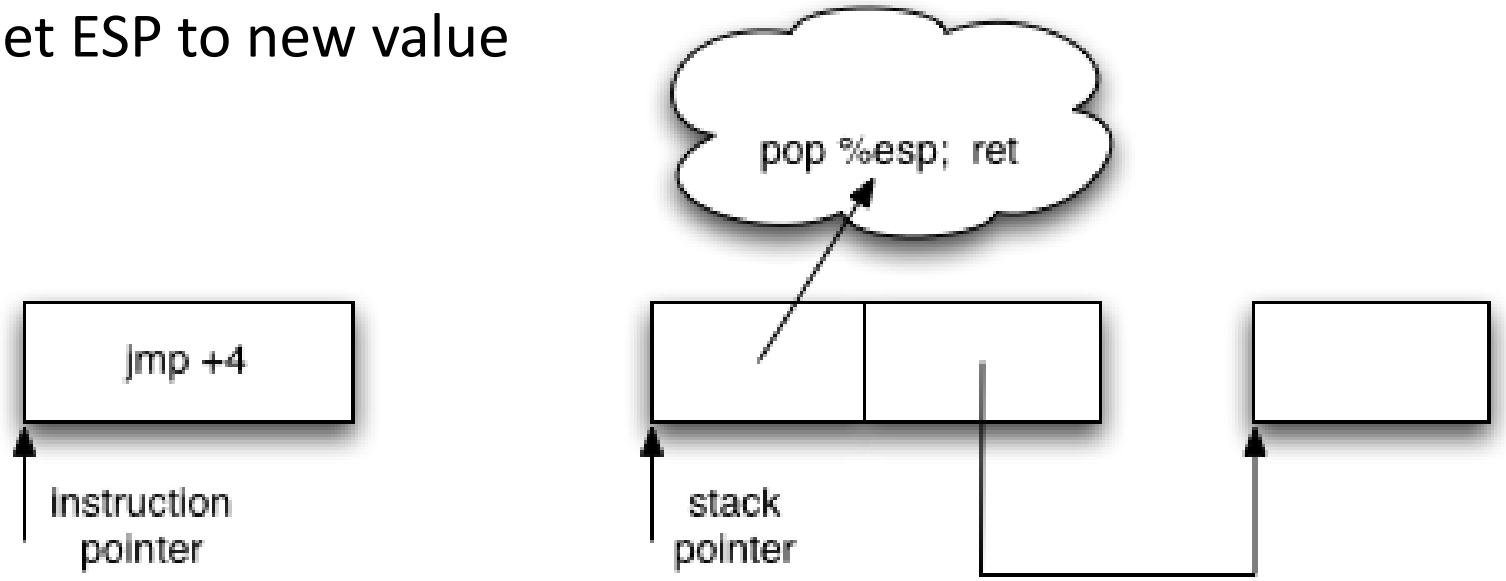
- Instructions can encode constants
- Return-oriented equivalent
  - Store on the stack
  - Pop into register to use



Control Flow

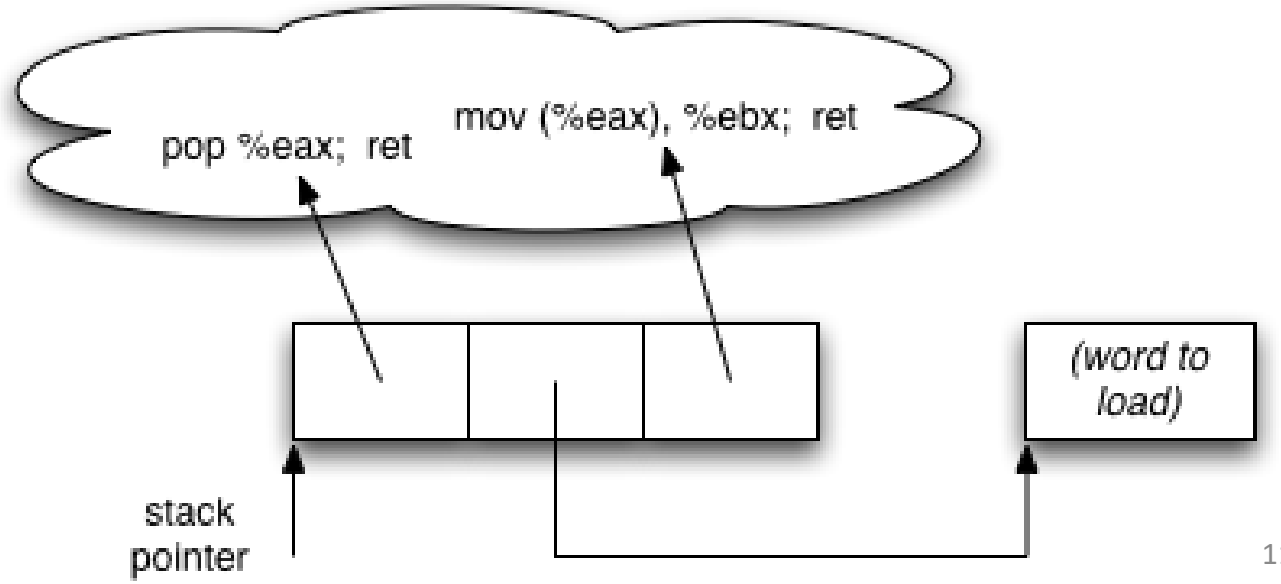
Ca' Foscari University of Venice

- Ordinary programming
  - (Conditionally) set EIP to new value
- Return-oriented equivalent
  - (Conditionally) set ESP to new value



Gadgets: Multi-instruction Sequences

- Sometimes more than one instruction sequence needed to encode logical unit
- Example: load from memory into register
  - Load address of source word into EAX
  - Load memory at (EAX) into EBX



Gadget Design

Ca' Foscari University of Venice

- Testbed: libc-2.3.5.so, Fedora Core 4
- Gadgets built from found code sequences:
  - Load-store, arithmetic & logic, control flow, syscalls
- Found code sequences are challenging to use!
  - Short; perform a small unit of work
  - No standard function prologue/epilogue
  - Messy interface, not an API
  - Some convenient instructions not always available

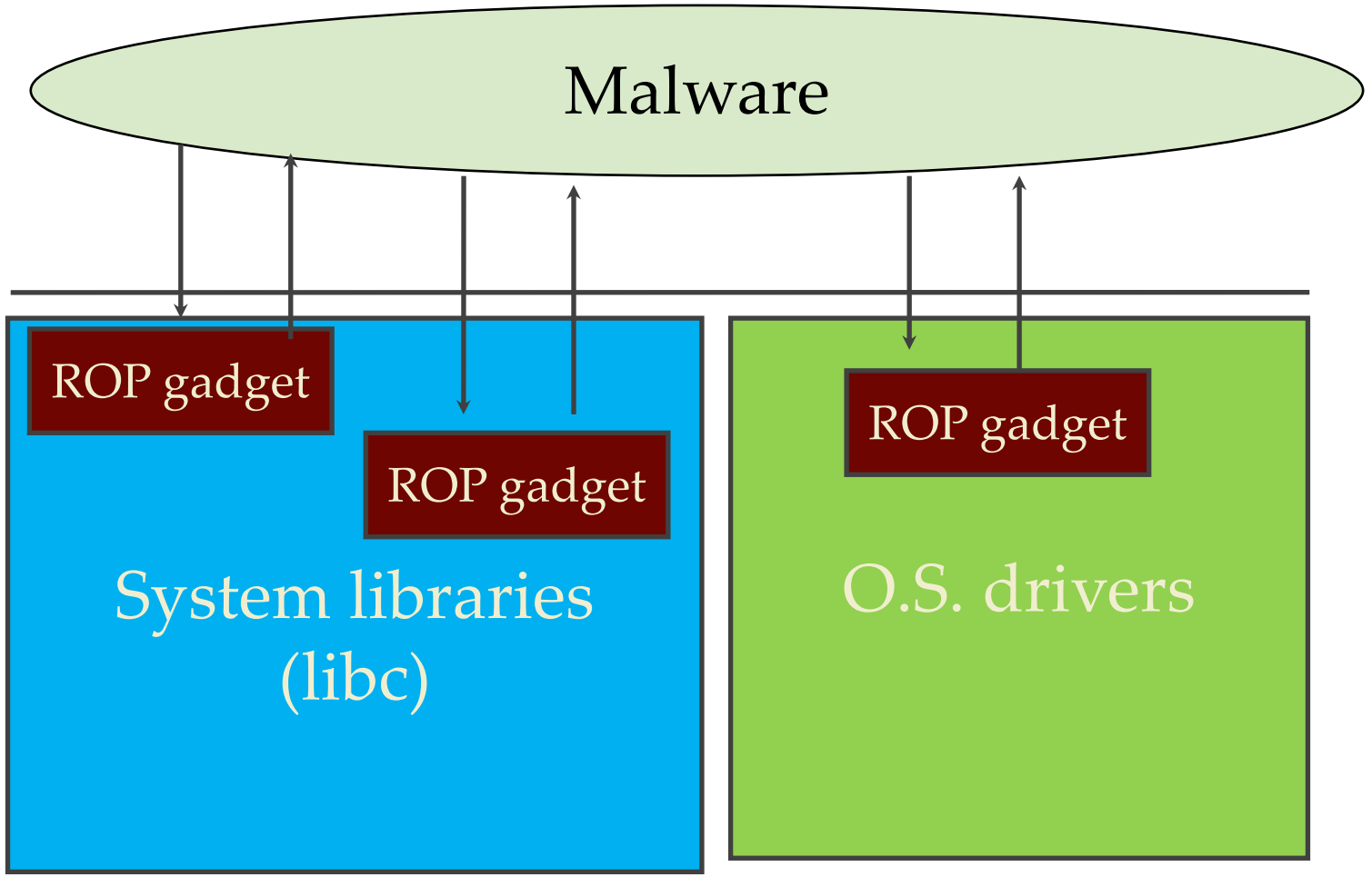
ROP gadgets

Ca' Foscari University of Venice

- Return-oriented programming (ROP) is a computer security exploit technique that allows an attacker to execute code in the presence of security defences such as non-executable memory and code signing.
- In this technique, an attacker gains control of the call stack to hijack program control flow and then executes carefully chosen machine instruction sequences, called "gadgets".
- Each gadget typically ends in a return instruction and is located in a subroutine within the existing program and/or shared library code.
- Chained together, these gadgets allow an attacker to perform arbitrary operations on a machine employing defences that thwart simpler attacks.

Meta-virus and ROP gadgets

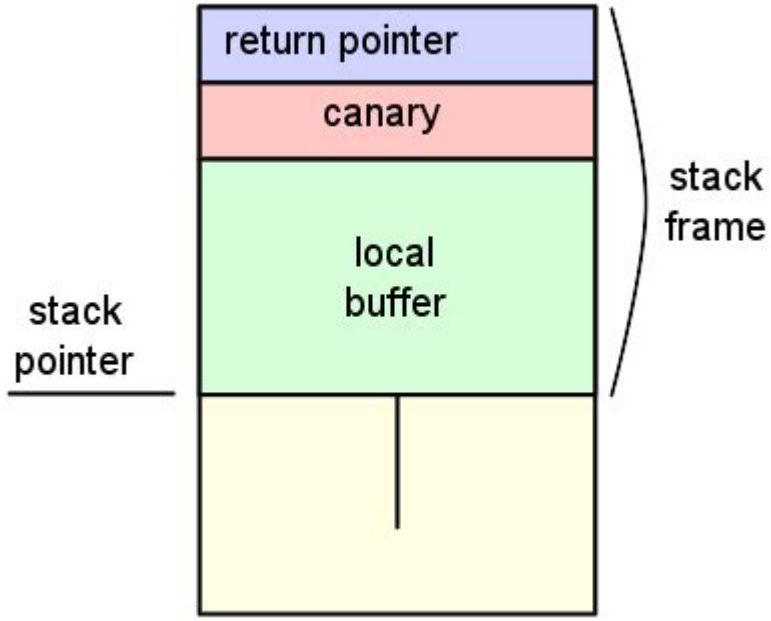
Ca' Foscari University of Venice



Stack Canaries

Ca' Foscari University of Venice

- Stack Canaries are a secret value placed on the stack which changes every time the program is started.
- Prior to a function return, the stack canary is checked and if it appears to be modified, the program exits immediately
- Inserted by modern compilers



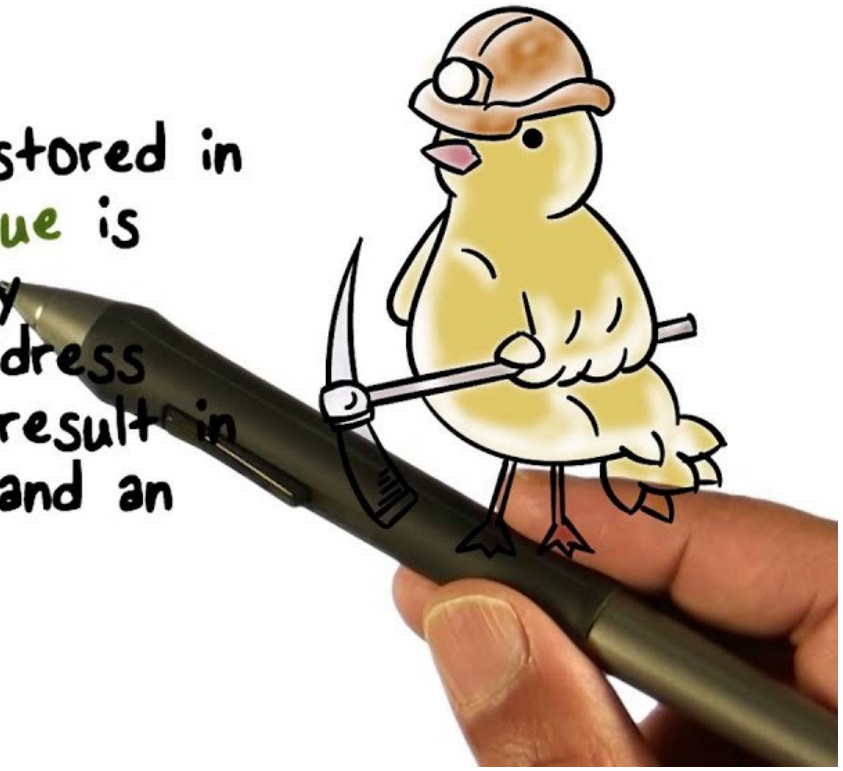
Stack Canaries

Ca' Foscari University of Venice

Thwarting Buffer Overflow Attacks

Stack canaries:

- When a return address is stored in a stack frame, a canary value is written just before it. Any attempt to rewrite the address using buffer overflow will result in the canary being rewritten and an overflow will be detected.



Brute-forcing a Stack Canary

Ca' Foscari University of Venice

- The canary is determined when the program starts up for the first time which means that if the program forks, it keeps the same stack configuration in the child process.
- This means that if the input that can overwrite the canary is sent to the child, we can use whether it crashes as an oracle and brute-force 1 byte at a time!

Rootkit

What is a rootkit

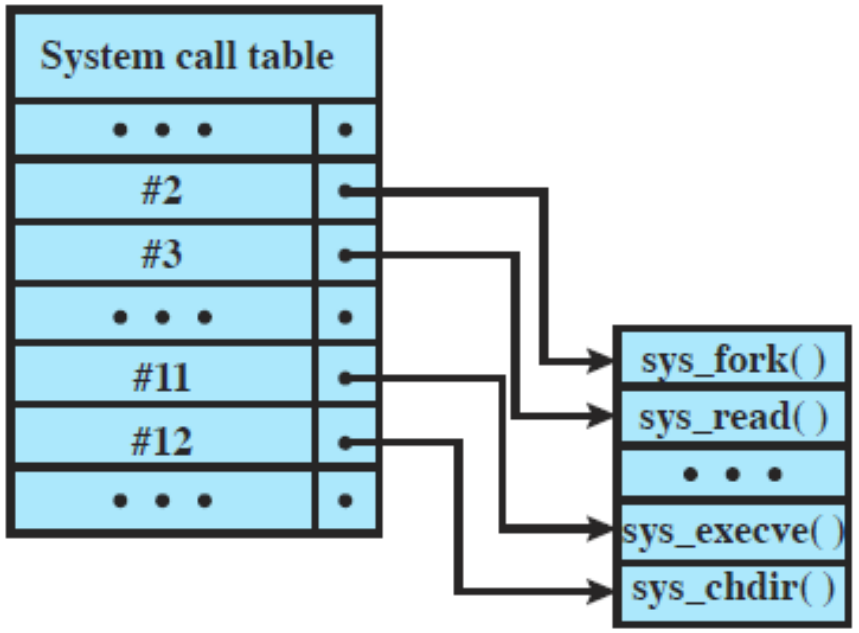
- A rootkit is a set of software tools that, when installed on a computer, provides remote access to resources, files and system information without the owner's knowledge.
- Law enforcement utilizes various types of rootkits to secretly monitor activity on computers for surveillance purposes
- Hackers can also install rootkits on the computers of unsuspecting victims.
- A rootkit is a component that uses stealth to maintain a persistent and undetectable presence on the machine

Rootkits

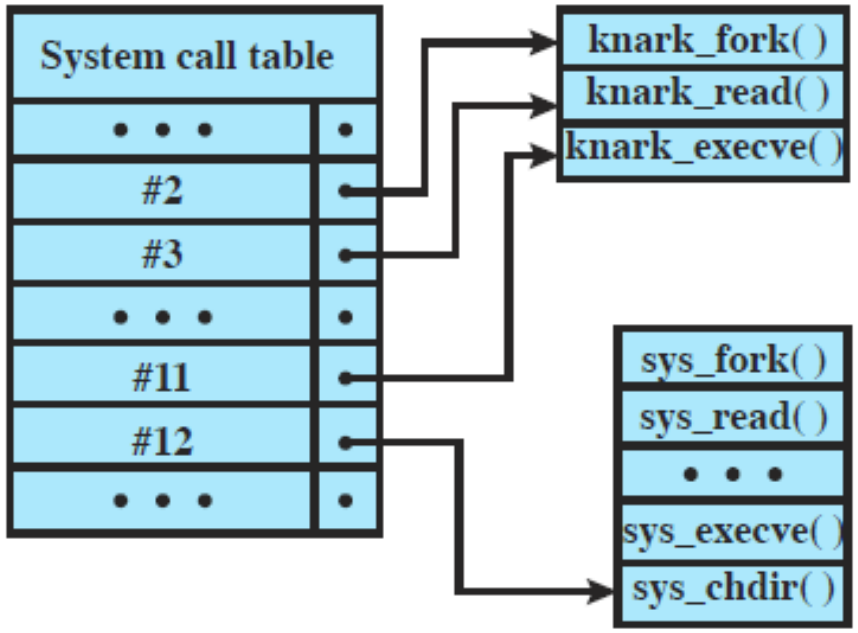
- Replace function table entries.
  - New version performs extra checks to hide information before performing original call.
  - Can replace Windows API pointers (user mode)
  - Can also replace syscall table pointers
- Both require privilege, but most Windows installs require privilege anyway
- Techniques apply equally well to Linux and Mac

Rootkit Infiltration

Ca' Foscari University of Venice



(a) Normal kernel memory layout



(b) After nkark install

Rootkit Countermeasures

Ca' Foscari University of Venice

- Hard to defend/detect
- User mode - Look for discrepancies
  - Between results of different APIs
  - Between API results and direct access to storage
  - Rootkit revealers

Subverting the Kernel

Ca' Foscari University of Venice

- Kernel task
- Process management
- File access
- Memory management
- Network management

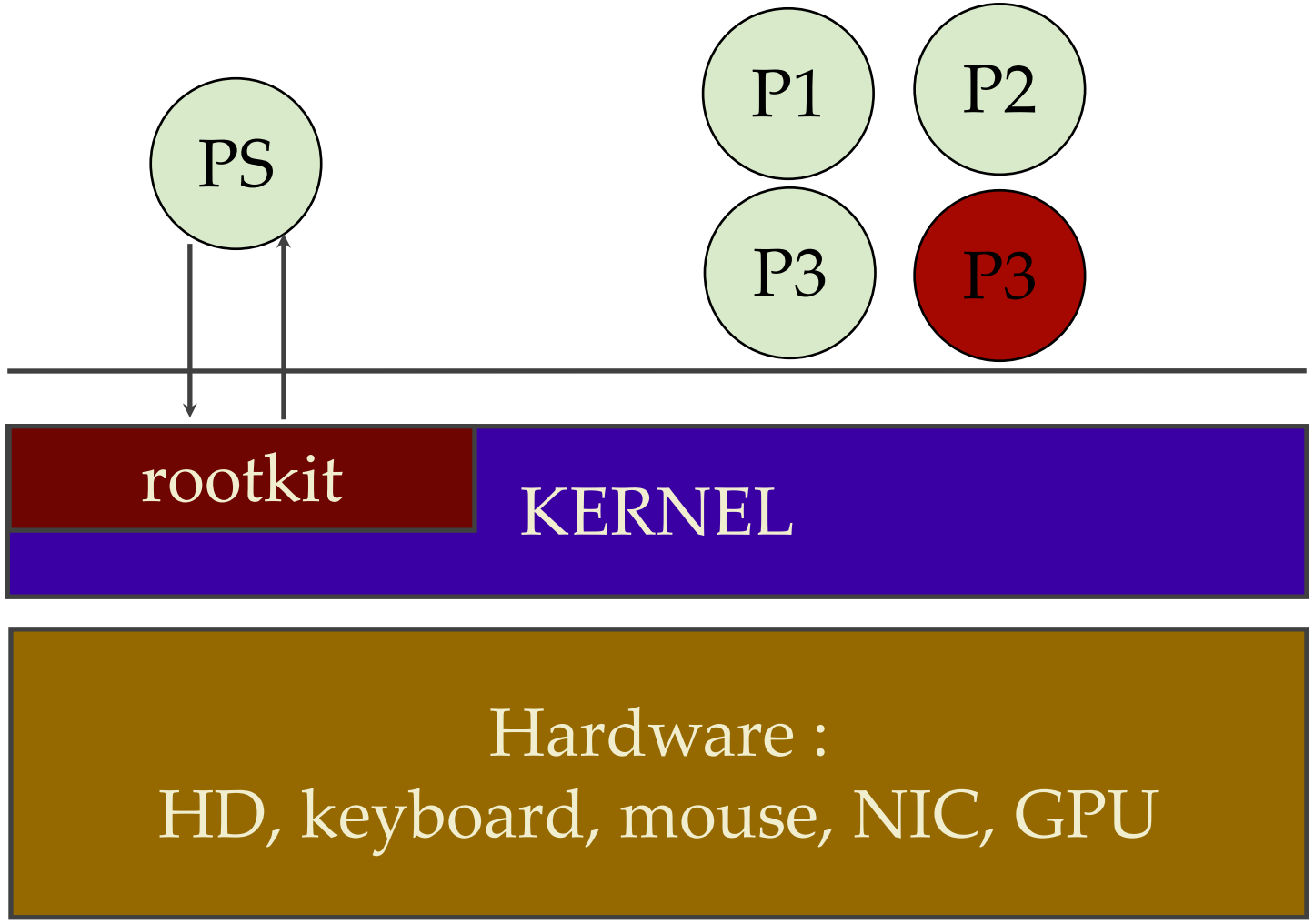
What to hide

→Process

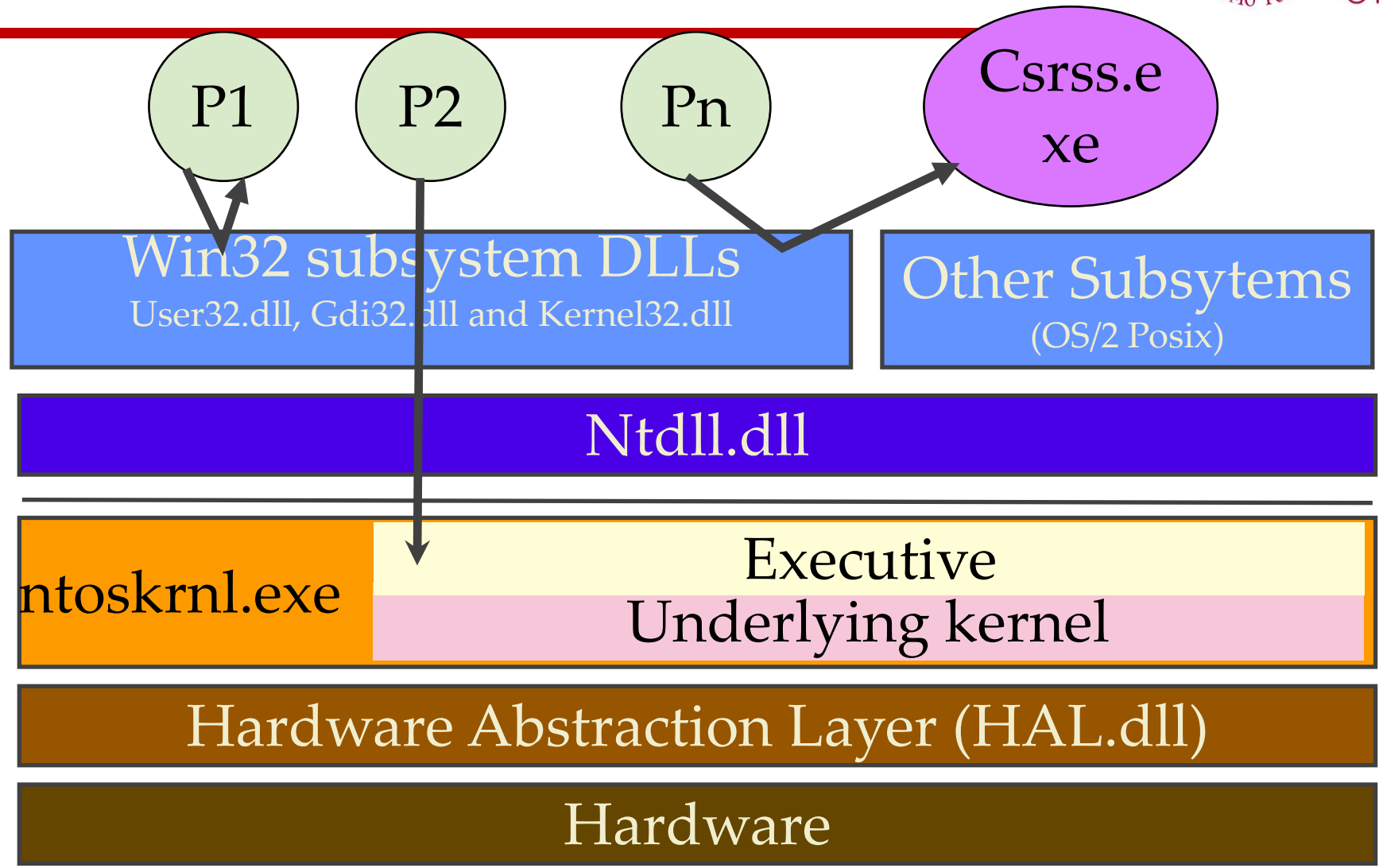
→Files

→Network traffic

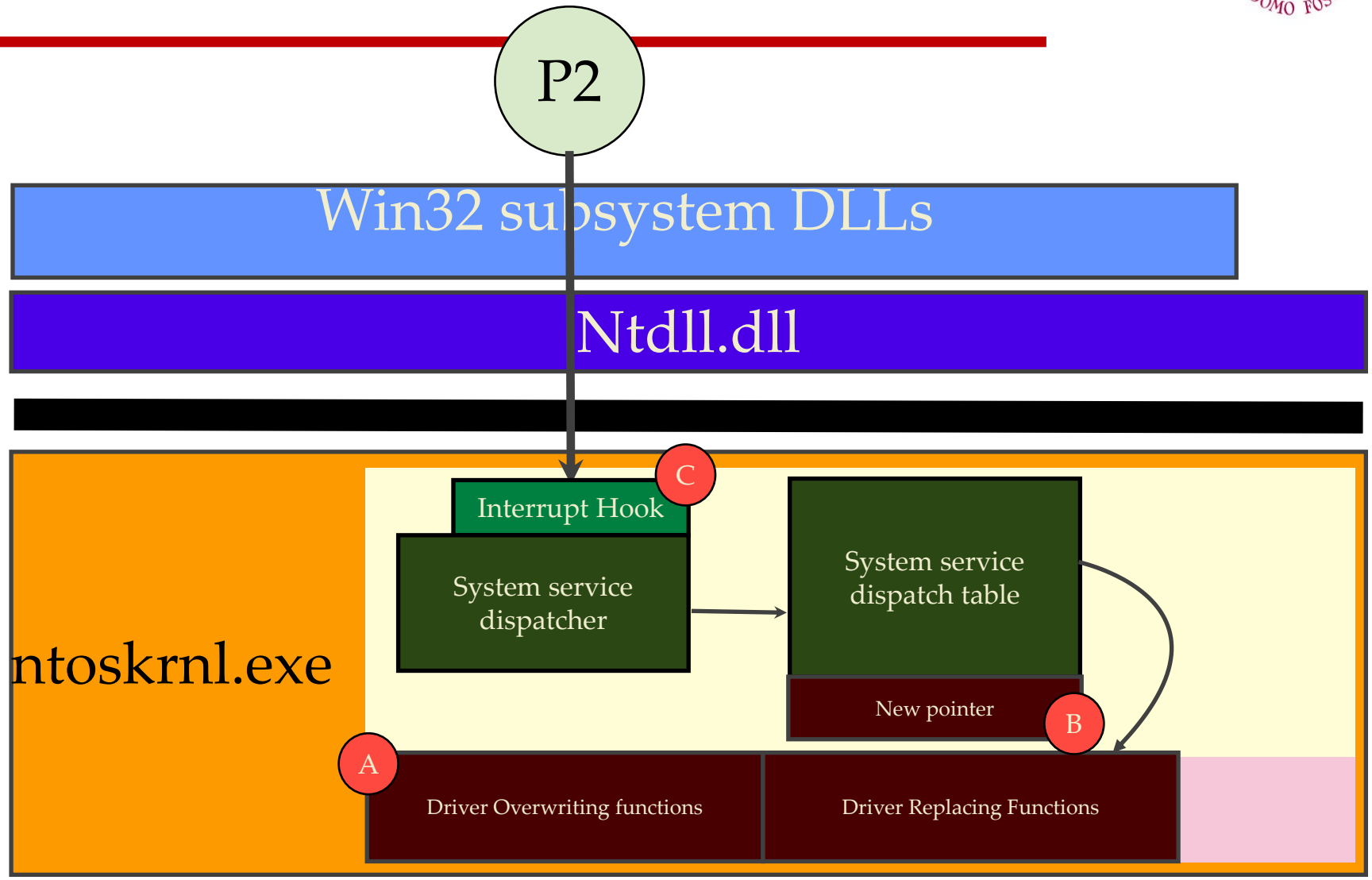
Kernel rootkit



Windows Kernel



Kernel Device driver



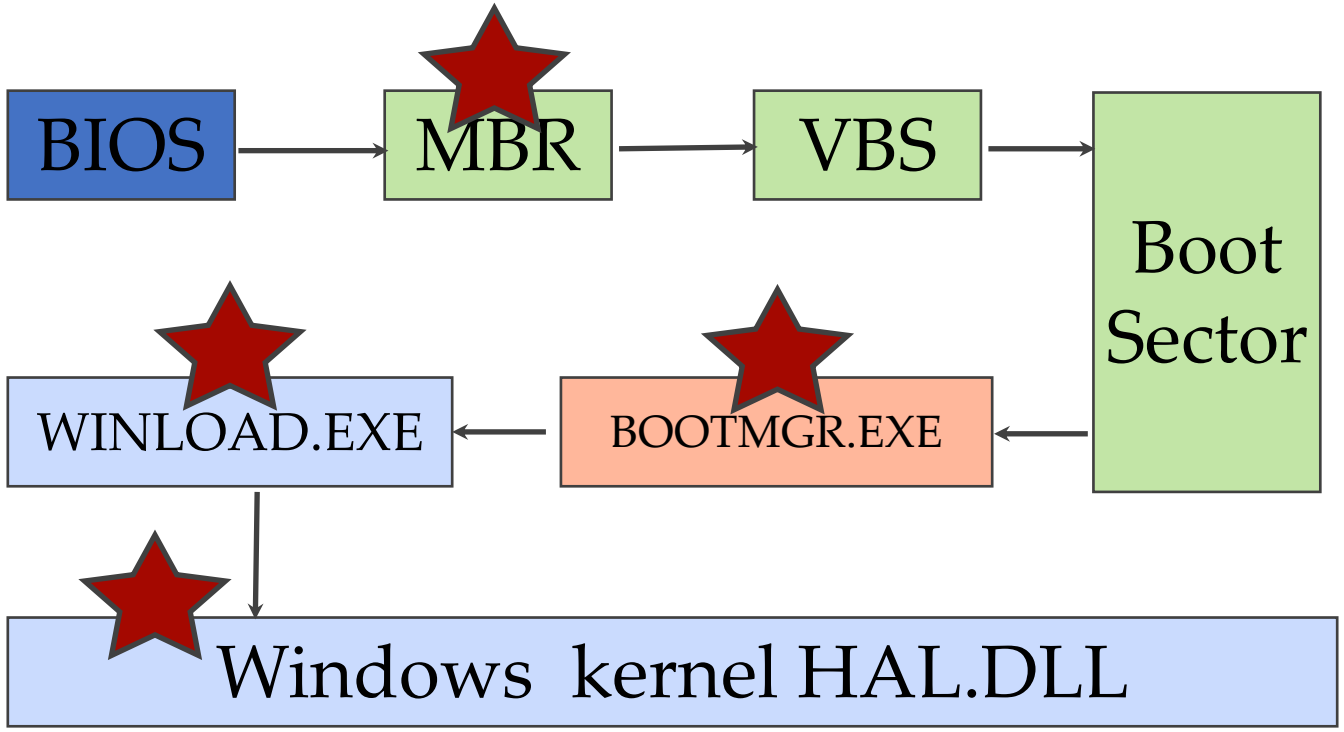
MBR/Bootkit

Ca' Foscari University of Venice

- Bootkits can be used to avoid all protections of an OS, because OS consider that the system was in trusted stated at the moment the OS boot loader took control.

Boot Sequence

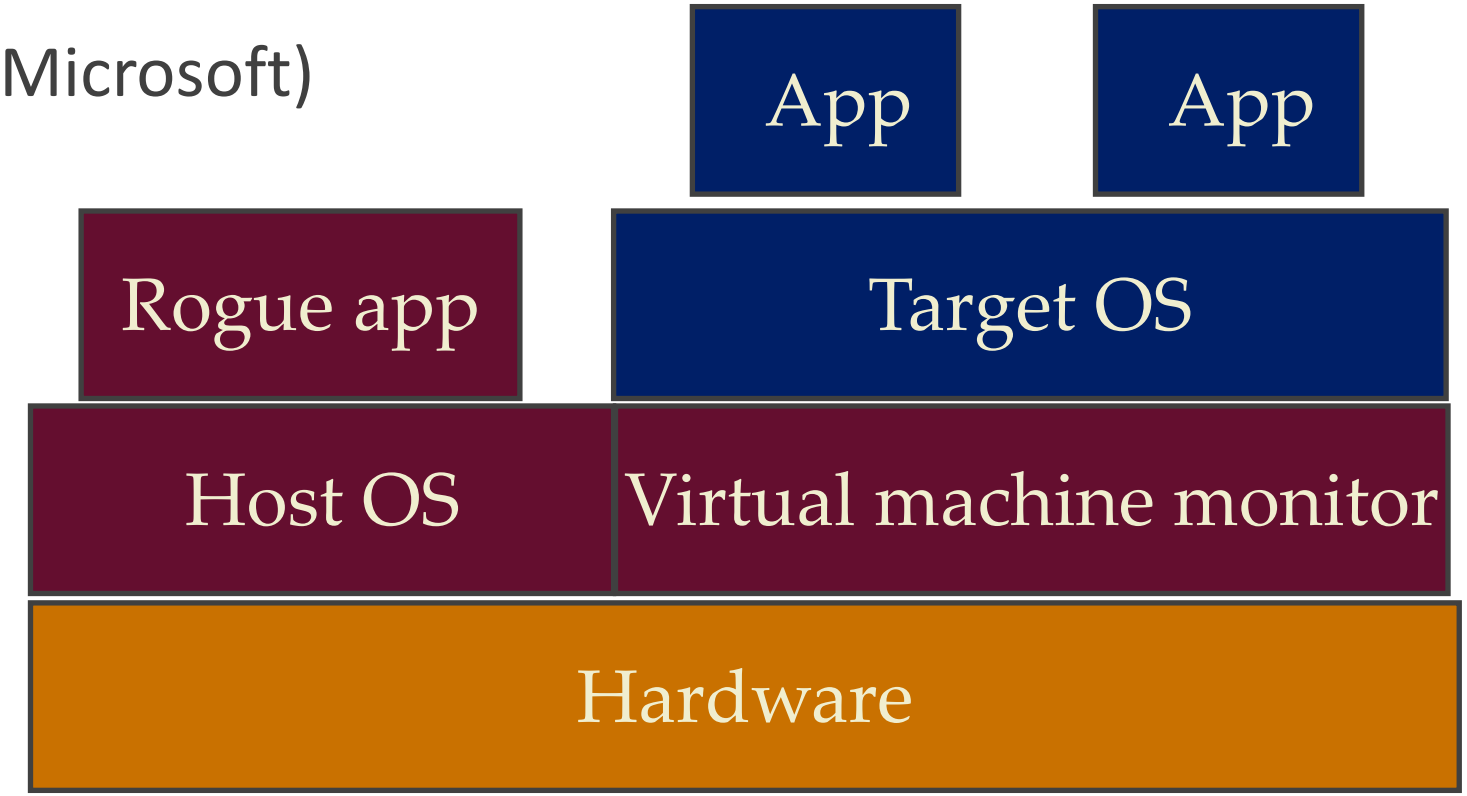
Ca' Foscari University of Venice



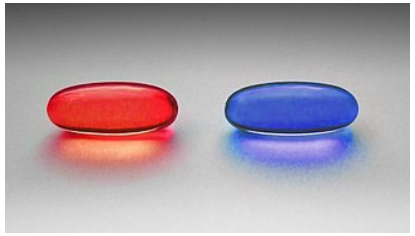
Hypervisor rootkit

Ca' Foscari University of Venice

- Blue pill
- SubVirt (Microsoft)



Blue Pill and Red Pill



Ca' Foscari University of Venice

- Blue Pill is the codename for a rootkit based on x86 virtualization.
- Blue Pill traps a running instance of the operating system by starting a thin hypervisor and virtualizing the rest of the machine under it.
  - The previous operating system would still maintain its existing references to all devices and files, but nearly anything, including hardware interrupts, requests for data and even the system time could be intercepted (and a fake response sent) by the hypervisor.
  - It was designed by Joanna Rutkowska and originally demonstrated at the Black Hat Briefings on 3/8/2006, with a reference implementation for the Microsoft Windows Vista kernel.
- Red Pill is a technique to detect the presence of a virtual machine
  - also developed by Joanna Rutkowska.

References

Ca' Foscari University of Venice

IMPORTANT: These are 3rd party websites so please review what you download for malware/suspicious content before use.

1. Cuckoo Sandbox: http://www.cuckoosandbox.org/

2. Zero Wine Malware Analysis Tool: http://zerowine.sourceforge.net/

3. Malheur: http://www.mlsec.org/malheur/

4. Radare: http://www.radare.org/

5. Interactive Disassembler Pro: http://www.hex-rays.com/

6. REGoogle: http://regoogle.carnivore.it/

7. ClamAV: http://www.clamav.net/

8. SysAnalyzer: https://github.com/dzzie/SysAnalyzer

9. Example technique from - Detecting exploits in electronic objects,

Alexander Shipp: http://www.google.com/patents/US20080134333

References

Ca' Foscari University of Venice

IMPORTANT: These are 3rd party websites so please review what you download for malware/suspicious content before use.

10. Malcode Analyst Pack: https://github.com/dzzie/MAP

11. VirusTotal: http://www.virustotal.com/

12. The Malware Hash Registry: http://www.team-cymru.org/Services/MHR/

13. ThreatExpert: http://www.threatexpert.com

14. McAfee SiteAdvisor: http://www.siteadvisor.com/

15. McAfee Free Tools: http://www.mcafee.com/us/downloads/free-tools/

16. Collaborative RCE Tool Library:

http://www.woodmann.com/collaborative/tools/index.php/Category:RCE_Tools

17. Ghidra: https://ghidra-sre.org/