

# Antivirus

*Alessandro Pellegrini*  
*a.pellegrini@ing.uniroma2.it*

# What's an antivirus?

- An antivirus is a piece of software running in background of your computer
- It has multiple high-level goals:
  - prevent infections by detecting the presence of malicious software
  - remove malicious software from your computer
- It is meant to be a prevention
- Sometimes the malware is able to get in
- In this case the antivirus should be able to
  - disinfect infected programs
  - clean malware from the computer

# Main problems AVs face

- Malware tries to self-protect and hide
  - AV evasion techniques
- Malware can use undocumented features of the system
- The attack surface of modern systems is huge
  - applications
  - system services
  - operating systems
  - email
  - network
- A large number of new malicious applications are released on a daily basis
  - the goal has shifted from *intellectual satisfaction* to *money*
  - Also *spying on users* is an important goal (governments, shady organizations, ...)
- Bugs! (AVs are typically implemented in compiled languages.)

# Current State of AVs

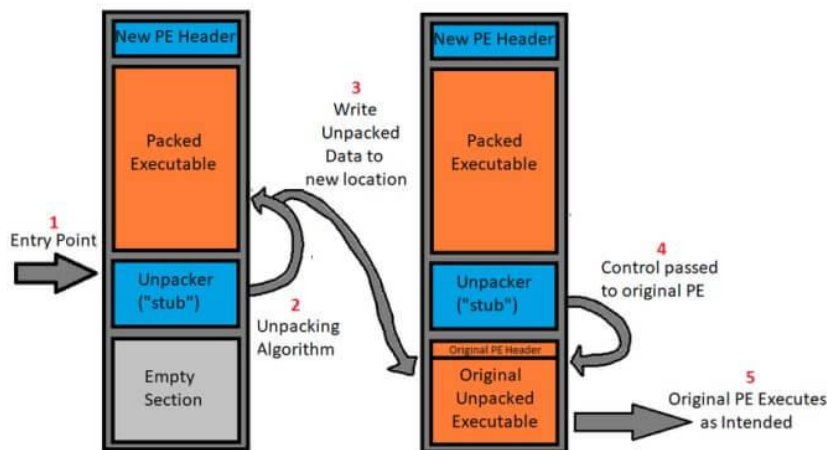
- In the early days, AVs were simple *scanners*
  - command line applications to identify malicious patterns in executable programs
- Modern AVs:
  - scan files in background
  - install firewalls
  - use browser add-ons
- An AV can receive thousands of unique malicious files every day
  - patterns are no longer a suitable way to detect malicious applications
  - *heuristics* have been introduced in most major products

# Main Components of an AV

- The kernel
- Command-line/GUI scanner
  - can be also *resident* or *real-time*, when they detect file creation/modification
- Daemons or system services
- File system filter drivers
- Network filter drivers
- Signature database
  - Signatures are the known patterns of malicious files
  - They are consumed by simple pattern-matching techniques
  - Can use CRC (checksums) or MD5 hashes
- Unpackers
- File format interpreters (PE, ELF, PNG, JPG, DOCX, PDF, [add your preferred extension here])
- Packet filters
- Self-protection mechanisms (avoid kill, ASLR, Data Execution Prevention, ...)
- Emulators

# Unpackers

- Malware is commonly packed with compressors and protectors
  - Sometimes they are just compressed files
  - They can also use encryption
- More advanced techniques transform the code into some bytecode
- This bytecode is coupled with a one or more (randomly-generated) virtual machines to run the original code of the malware



# Indications of a Packed Executable

- Lack of Imports in Import Address Table (IAT)
  - On Windows, often also lacking kernel32.dll and user32.dll
- Non-standard Section Names
- Sections with a small raw size but a large virtual size
- Low number of discernible strings
- Sections with RWX privileges
- jmp or call Instructions to registers/strange memory addresses
  - Many conventional packers store the address where to unpack in registers

# Signatures

- Signatures are patterns (or pieces of code) that try to match some file against some known malware/exploit
  - In some cases, they are used to check the presence of strings or sequences of bytes in a file
- In many cases, they are actual programs that are run to parse one specific file
- These programs try to be small and quick, and focus on the results of security engineers
- They try to be as general as possible, but the complexity of the file formats can make it quite simple to evade signature-based scanning
  - A typical example is that of PDF files



# Evasion Techniques: Ineffective Code Sequences

- A form of *polymorphic code*

```
00401005 8BF0      MOV ESI,EAX
00401007 3E:8A00   MOV AL,BYTE PTR DS:[EAX]
0040100A 84C0      TEST AL,AL
0040100C v 74 46    JE SHORT Test.00401054
0040100E 53        PUSH EBX
0040100F 3E:8F05 74F940 POP DWORD PTR DS:[40F974]
00401016 D3DB      RCR EBX,CL
00401018 0FCB      BSWAP EBX
0040101A 68 56104000 PUSH Test.00401056
0040101F 5B        POP EBX
00401020 3E:8903   MOV DWORD PTR DS:[EBX],EAX
00401023 43        INC EBX
00401024 0FBDC2   BSR EAX,EDX
00401027 A9 46A978DC TEST EAX,DC78A946
0040102C 8BC2     MOV EAX,EDX
0040102E 52        PUSH EDX
0040102F B6 86     MOV DH,86
00401031 B3 27     MOV BL,27
00401033 B8 7CFAA17F MOV EAX,7FA1FA7C
00401038 v EB 01    JMP SHORT Test.0040103B
0040103A 90        NOP
0040103B 0FBCC2   BSF EAX,EDX
0040103E 3E:C705 FC8841 MOV DWORD PTR DS:[4188FC],0
00401049 2D 210DE8B9 SUB EAX,B9E80D21
0040104E 69DA E577D49D IMUL EBX,EDX,9DD477E5
```



```
00401005 8BF0      MOV ESI,EAX
00401007 3E:8A00   MOV AL,BYTE PTR DS:[EAX]
0040100A 84C0      TEST AL,AL
0040100C v 74 4D    JE SHORT Test.0040105B
0040100E 53        PUSH EBX
0040100F 3E:8F05 74F940 POP DWORD PTR DS:[40F974]
00401016 D3DB      RCR EBX,CL
00401018 0FCB      BSWAP EBX
0040101A 68 5D104000 PUSH Test.0040105D
0040101F 5B        POP EBX
00401020 3E:8903   MOV DWORD PTR DS:[EBX],EAX
00401023 43        INC EBX
00401024 0FBDC2   BSR EAX,EDX
00401027 A9 46A978DC TEST EAX,DC78A946
0040102C 8BC2     MOV EAX,EDX
0040102E 90        NOP
0040102F 90        NOP
00401030 42        INC EDX
00401031 52        PUSH EDX
00401032 FE0C24   DEC BYTE PTR SS:[ESP]
00401035 40        DEC EDX
00401036 B6 86     MOV DH,86
00401038 B3 27     MOV BL,27
0040103A B8 7CFAA17F MOV EAX,7FA1FA7C
0040103F v EB 01    JMP SHORT Test.00401042
00401041 90        NOP
00401042 0FBCC2   BSF EAX,EDX
00401045 3E:C705 FC8841 MOV DWORD PTR DS:[4188FC],0
00401050 2D 210DE8B9 SUB EAX,B9E80D21
00401055 69DA E577D49D IMUL EBX,EDX,9DD477E5
```

# Evasion Techniques: Code Transposition

- Make your code some spaghetti code
- The signature becomes effectively altered



# Polymorphic viruses

- Polymorphic viruses change the decryptor body in the successive generation, during the infection
  - This is similar in spirit to polymorphic code
- Simplest implementations use a large number of decryptors, on the order of hundreds (*olygomorphic* viruses).
- Multiple decryptors can be used at once
- The pieces used to build the decryptors are usually common code
  - signature based detection doesn't behave well in this context

# Metamorphic virus

- Metamorphic viruses output a logically-equivalent version of its own code under some interpretation.
- Metamorphic viruses often translate their own binary code into a temporary representation which is altered
- The altered temporary representation is then translated back to machine code again
- The main difference from polymorphic viruses is that also the encryption/decryption engine is modified

# Behavioral Signatures

- Behavior-based malware detection techniques observe the behavior of a program to conclude whether it is malicious or not
- A behavior based detector concludes whether a program is malicious by inspecting what it does rather than what it says.
- Histogram-based malicious code detection (Symantec)
  - Keep track of occurrence of instructions
  - Keep track of occurrence of *pairs* of instructions
  - Use the histogram (binning) to match against signatures
- Such an approach is typically used in emulated executions

# Heuristic Engines

- Typically based on ML techniques, especially classifiers
- API call sequences as a feature of a malware
  - The interaction of a malware with the OS can be an indication of the actions that the application is taking
  - Typically based on classification: millions of malware and benignware used as training sets
- OpCode Sequences
  - The sequence (or binning) of machine instructions can be an indication of the operations that the applications is carrying out
  - Applications are analyzed to determine the frequency of opcodes in the binaries
  - Benignware and malware datasets are filtered using the Mutual Information filtering method
  - Weighted Term Frequency is the used to make a suitable feature vector extracted from executables

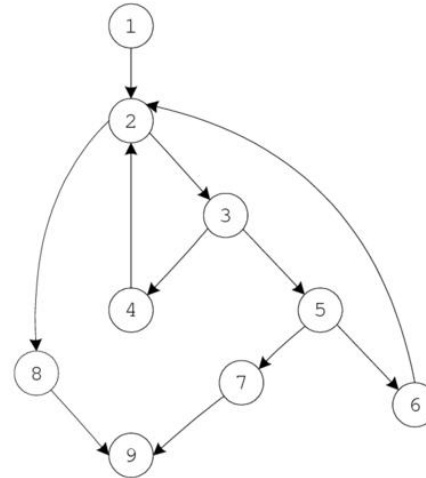
# Heuristic Engines

- Call Function Graph is a technique to extract *basic blocks* from a binary
- The links between basic blocks can be an indication of the *semantic* of an application, independently of possible mutations

Source Program:

```
int binsearch(int x, int v[], int n)
{
  1 | int low, high, mid;
    | low = 0;
    | high = n - 1;
    | while (low <= high) | 2
    | {
    |   3 | mid = (low + high)/2;
    |     | if (x < v[mid])
    |       | high = mid - 1; | 4
    |   5 | else if (x > v[mid])
    |       | low = mid + 1; | 6
    |   7 | else return mid;
    | }
    | return -1; | 8
  } | 9
```

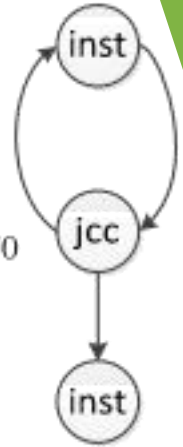
CFG:



# Heuristic Engines

- Four types of instruction are considered:
  - non-conditional jumps (jmp)
  - conditional jumps (jcc)
  - function calls (call)
  - function returns (ret).
- They abstract any contiguous sequence of instructions in a node named “inst”,
- The end of the program comes in a node named “end”.
- These nodes are then reduced:
  - any node of kind inst or jmp, is removed from the graph
  - all its predecessors are linked to its unique successor.
- This graph is used as a signature

```
0x1288 push ebp
0x128b mov ebp, esp
0x1291 lea edi, [0x405814]
0x1293 mov eax, [ebp + 0x8]
0x1299 cmp dword [0x4056c5], 0x270
0x12a3 jnz 0x1288
0x12a5 pop edi
```



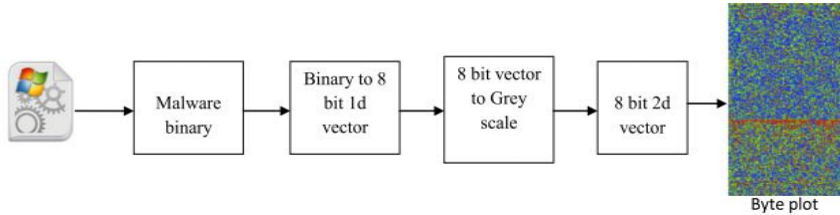


# n-grams

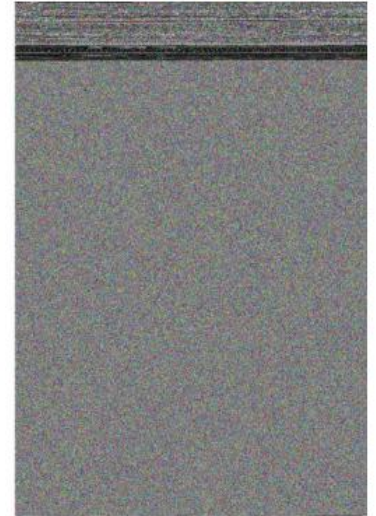
- In linguistics, an *n-gram* is a contiguous sequence of  $n$  items from a given sample of text or speech
  - co-occurring words within a given window
  - when computing the n-grams you typically move one word forward
- Example: “*The cow jumps over the moon*”, with  $n=2$ 
  - the cow
  - cow jumps
  - jumps over
  - over the
  - the moon
- n-grams over bytes are used with ML methods to detect malicious executables.
- Typical classifiers: Naive-Bayes, Support Vector Machine, Decision Tree
- Boosted decision tree typically give the best classification results

# Byteplots

- Transform binaries into images
- Compare the images against known malware



Benign executable



Malware

# Emulators

- Signature-based detection or heuristics can be easily circumvented
- Some AVs carry out emulators that allow running applications in sandboxes
  - They implement CPU ISAs
  - They implement APIs offered by operating systems
- Running an application in a sandbox can allow to bypass malware obfuscation techniques
- The application is analyzed within the sandbox

# Evasion Techniques: Fingerprinting

- Emulators cannot fully emulate all the services from the OS
- Many functions are simple stubs that return hardcoded values
- An example: `OpenMutexW` in Comodo:
  - always returns the magic value `0xBBBB`
  - Chances that an actual call to `OpenMutexW` returns this value are low
- A malware can easily check against hardcoded values returned from stubs
- If a stub is detected, the malware will run some legitimate code

# Encrypted Traffic Analysis

- Some antivirus products advertise that they can inspect HTTPS
- They use the same actions that malware does to inspect network traffic and protect customers
  - They launch a MITM attack and install a certificate signed with a trusted certificate authority for the specific domain to be inspected
  - Or, they create new certificates for each new site that its users visit, signing them with a valid CA
  - This is done in Windows by installing a new root certificate
- This approach might lower security
  - It breaks HTTP Public Key Pinning (HPKP)
  - If the implementation is not right, the traffic is vulnerable to TLS attacks
  - Some implementations accept 8-bit Diffie-Hellman key exchanges!
  - A weaker TLS implementation might make vulnerable also antivirus updates

# Should you trust your AV?

- AVs are complex pieces of software
  - They play an invaluable role at reducing the dissemination of malware
- They are pieces of software anyhow
  - Detection techniques can be jeopardized
  - Some implementations might be buggy
  - Some techniques might lower the security of your system
- Having an antivirus installed is often referred to as the ultimate solution to computer security
- Remember: a secure computer does not exist!
- Feeling safe just because you installed a software might lower the attention of the user