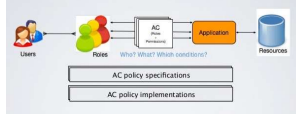





## Malware Countermeasure Approaches

- Ideal solution to the threat of malware is prevention:
  - Do not allow malware to get into the system in the first place
  - Block the ability of it to modify the system
- Four main elements of prevention (might overlap as categories):
  - Policy:** installation and maintenance of **Anti-Viruses** and **Anti-Malwares** to detect payload download or actions before they happen; Training of personnel on phishing, spam and opening emails or shadow I; **Access Control policies**
  - Awareness:** again, **training** of personnel, but also **collection of alarms/logs/events by IT security**, and control on installed applications from personnel
  - Vulnerability mitigation:** **patch system on time**, as soon as **vulnerabilities** are found
  - Threat mitigation:** **Virus Propagation containment**, **Virus execution containment**, **Checkpointing/Backup**, access controls on the applications and data stored on the system, to reduce the number of files that any user can access, and hence potentially infect or corrupt, as a result of them executing some malware code




Anti-virus Policy

Users → Rules → AC Policy Specifications → AC Policy Implementations → Application → Resources



7 Signs of Phishing



2 Dr. Valerio Formicola

The ideal solution to the threat of malware is prevention: Do not allow malware to get into the system in the first place, or block the ability of it to modify the system. This goal is, in general, nearly impossible to achieve, although taking suitable countermeasures to harden systems and users in preventing infection can significantly reduce the number of successful malware attacks. NIST SP 800-83 suggests there are four main elements of prevention: policy, awareness, vulnerability mitigation, and threat mitigation. Having a suitable policy to address malware prevention provides a basis for implementing appropriate preventative countermeasures.

One of the first countermeasures that should be employed is to ensure all systems are as current as possible, with all patches applied, in order to reduce the number of vulnerabilities that might be exploited on the system. The next is to set appropriate access controls on the applications and data stored on the system, to reduce the number of files that any user can access, and hence potentially infect or corrupt, as a result of them executing some malware code. These measures directly target the key propagation mechanisms used by worms, viruses, and some Trojans. We discuss them further in Chapter 12 when we discuss hardening operating systems and applications.

The third common propagation mechanism, which targets users in a social engineering attack, can be countered using appropriate user awareness and training. This aims to equip users to be more aware of these attacks, and less likely to take actions that result in their compromise. NIST SP 800-83 provides examples of suitable awareness issues. We will return to this topic in Chapter 17.

## In the case of infection from malware

- If prevention fails, technical mechanisms can be used to support the following threat mitigation options:
  - Detection:** Once the infection has occurred, determine that it has occurred (**detect**) and locate the malware (**locate**).
  - Identification:** Once detection has been achieved, identify the specific malware that has infected the system (**diagnosis/identification**).
  - Removal:** Once the specific malware has been identified, remove all traces of malware virus from all infected systems so that it cannot spread further (**removal**).

Detection might be successful but, identification not successful or removal impossible (non-reversible)



1. Removal of malware infected files.
2. Wipeout and restore safe backup in most extreme cases.
3. Re-manufacturing firmware in the case of UEFI/BIOS infections



3 Dr. Valerio Formicola



If prevention fails, then technical mechanisms can be used to support the following threat mitigation options:

- **Detection:** Once the infection has occurred, determine that it has occurred and locate the malware.
- **Identification:** Once detection has been achieved, identify the specific malware that has infected the system.
- **Removal:** Once the specific malware has been identified, remove all traces of malware virus from all infected systems so that it cannot spread further.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard any infected or malicious files and reload a clean backup version. In the case of some particularly nasty infections, this may require a complete wipe of all storage, and rebuild of the infected system from known clean media.

# Requirements and placement of malware detectors

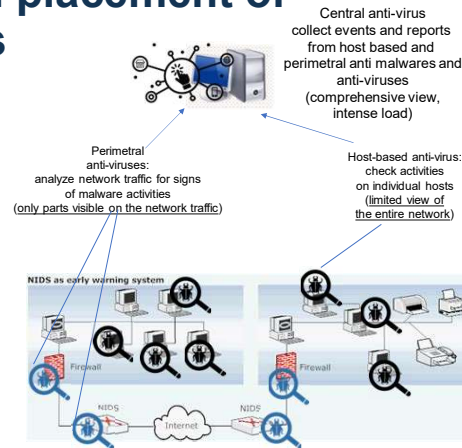
## Requirements of countermeasures for malware:

- **Generality:** The approach taken should be able to handle a wide variety of attacks.
- **Timeliness:** The approach should respond quickly so as to limit the number of infected programs or systems and the consequent activity.
- **Resiliency:** The approach should be resistant to evasion techniques employed by attackers to hide the presence of their malware.
- **Minimal denial-of-service costs:** The approach should result in minimal reduction in capacity or service due to the actions of the countermeasure software and should not significantly disrupt normal operation.
- **Transparency:** The countermeasure software and devices should not require modification to existing (legacy) OSs, application software, and hardware.
- **Global and local coverage:** The approach should be able to deal with attack sources both from outside and inside the enterprise network.

Achieving all these requirements often requires the use of multiple approaches.



4 Dr. Valerio Formicola



To begin, let us consider some requirements for effective malware countermeasures:

- **Generality:** The approach taken should be able to handle a wide variety of attacks.
- **Timeliness:** The approach should respond quickly so as to limit the number of infected programs or systems and the consequent activity.
- **Resiliency:** The approach should be resistant to evasion techniques employed by attackers to hide the presence of their malware.
- **Minimal denial-of-service costs:** The approach should result in minimal reduction in capacity or service due to the actions of the countermeasure software, and should not significantly disrupt normal operation.
- **Transparency:** The countermeasure software and devices should not require modification to existing (legacy) OSs, application software, and hardware.
- **Global and local coverage:** The approach should be able to deal with attack sources both from outside and inside the enterprise network.

Achieving all these requirements often requires the use of multiple approaches.

Detection of the presence of malware can occur in a number of locations. It may occur on the infected system, where some host-based “anti-virus” program is running, monitoring data imported into the system, and the execution and behavior of programs running on the system. Or, it may take place as part of the perimeter security mechanisms used in an organization’s firewall and intrusion detection systems (IDS). Lastly, detection may use distributed mechanisms that gather data from both host-based and perimeter sensors, potentially over a large number of networks and organizations, in order to obtain the largest scale view of the movement of malware. We now consider each of these approaches in more detail.

The first location where anti-virus software is used is on each end system. This gives the software the maximum access to information on not only the behavior of the malware as it interacts with the targeted system, but also the smallest overall view of malware activity. The use of anti-virus software on personal computers is now widespread, in part caused by the explosive growth in malware volume and activity. This software can be regarded as a form of host-based intrusion detection system, which we discuss more generally in Section 8.4. Advances in virus and other malware technology, and in antivirus technology and other countermeasures, go hand in hand. Early malware used relatively simple and easily detected code, and hence could be identified and purged with relatively simple anti-virus software packages. As the malware arms race has evolved, both the malware code and, necessarily, anti-virus software have grown more complex and sophisticated.

## Virus and malware detection

Virus identification is a balance to reduce these two imperatives as much as possible:

- **false negatives:** fail to detect an infection.
- **false positives:** detection of a virus where none exists.

# First generation anti-virus software

- **First generation: simple scanners**
  - Requires a malware signature to identify the malware
- **Limited to the detection of known malware**
  - high false negatives
- Scan compare the analyzed object with a database of signatures
- A **signature** is a virus fingerprint
  - E.g., a string with a sequence of instructions specific for each virus
  - Different from a digital signature
- A file is infected if there is a signature inside its code
  - Fast **pattern matching** techniques to search for signatures
- All the signatures together create the malware database that usually is proprietary

YARA format signature

```

rule XProtect_Malware_Signature {
  meta:
    description: "XProtect_Malware_Signature"
  strings:
    $s1 = { 48 25 48 25 48 25 48 25 48 25 48 25 48 25 48 25 }
    $s2 = { 48 25 48 25 48 25 48 25 48 25 48 25 48 25 48 25 }
    $s3 = { 48 25 48 25 48 25 48 25 48 25 48 25 48 25 48 25 }
    $s4 = { 48 25 48 25 48 25 48 25 48 25 48 25 48 25 48 25 }
    $s5 = { 48 25 48 25 48 25 48 25 48 25 48 25 48 25 48 25 }
  condition:
    Meta and all of ($s1) and filesize > 20000
}

rule XProtect_TrojanDownloader_PayloadNames {
  strings:
    $str1 = "msi.dll.eng" wide fullword
    $str2 = "msutil.dll.url" wide ascii nocase
  condition:
    (uint16(0) == 0x504D) and
    (filesize > 1KB) and
    any of them
}

```

Input = ba aab ab  
 Pattern = \*\*\*\*\*ba\*\*\*\*\*ab  
 Output : true

Input = baaabab  
 Pattern = a \* ab  
 Output : false

Input = ba aab ab  
 Pattern = ba \* a?  
 Output : true

Virus Name	String Pattern (Signature)
Accom.128	89C3 B440 8A2E 2004 8A0E 2104 BA00
0	05CD 21E8 D500 BF50 04CD
Die.448	B440 B9E8 0133 D2CD 2172 1126 8955
	15B4 40B9 0500 BA5A 01CD
Xany.979	8B96 0906 B000 E85C FF8B D5B9 D303
	E864 FFC6 8602 0401 F8C3



CalPolyPomona

6 Dr. Valerio Formicola

A first-generation scanner requires a malware signature to identify the malware. The signature may contain “wildcards” but matches essentially the same structure and bit pattern in all copies of the malware. Such signature-specific scanners are limited to the detection of known malware. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length as a result of virus infection.

## Second Generation: heuristics

**Second generation: heuristic scanners**, so called because operate on the basis of experience (by comparing the suspicious file to the code and functions of known viruses).

- **Uses heuristic rules (rule-sets) to search for probable malware instances:** looks for fragments of malware code, e.g., checks for encryption loops at beginning of polymorphic virus
- **Another approach is integrity checking: checking the hash of files to determine an unusual change due to infection**
- **Heuristics are normally characterized as using a specific scoring algorithm that determines the likelihood of the scanned object being malicious**

### Detection limits:

- **Some false negatives:** It is likely to miss new viruses that contain previously unknown methods of operation not found in any known viruses.
- **high false positives:** legitimate programs with similar heuristics can trigger the detection

### Two categories of heuristics:

- **Static heuristic analysis:** involves decompiling/disassembling a suspected program that might contain the virus; then, examining the obtained source code of this program for comparing it to the source code of known viruses that have already been logged in a database. If enough of it matches what is in the database, the code gets flagged as a potential threat.
- **Dynamic heuristic analysis:** uses a virtual machine, which acts as a sandbox (later in these slides). For example, during a dynamic heuristic analysis, the program under observation may self-replicate, try to stay within resident memory after executing, overwrite files, or do other things that viruses are often programmed to do.



Examples of Heuristics (dynamic or static):

- Contacted (malicious) IP addresses
- Operations on files
- Propagation pattern
- Persistence in memory after task completion
- Areas of system that are changed
- Creation of registry and key records
- API calls and system calls (and/or their sequence)
- Control flow of a program (through CFGs)
- OpCodes
- Strings (or substrings) and other hybrid or novel features (e.g., file content).



Cal Poly Pomona

7 Dr. Valerio Formicola

A second-generation scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable malware instances. One class of such scanners looks for fragments of code that are often associated with malware. For example, a scanner may look for the beginning of an encryption loop used in a polymorphic virus and discover the encryption key. Once the key is discovered, the scanner can decrypt the malware to identify it, then remove the infection and return the program to service.

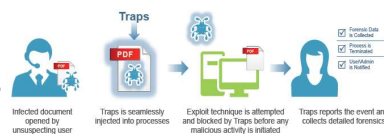
Another second-generation approach is integrity checking. A checksum can be appended to each program. If malware alters or replaces some program without changing the checksum, then an integrity check will catch this change. To counter malware that is sophisticated enough to change the checksum when it alters a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the malware cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the malware is prevented from adjusting the program to produce the same hash code as before. If a protected list of programs in trusted locations is kept, this approach can also detect attempts to replace or install rogue code or programs in these locations.



## 3rd Generations of Anti-Virus Software

### • Third generation: activity traps

- Memory-resident programs that identify malware by its actions rather than its structure in an infected program
  - E.g., reading a password file and sending it over the Internet
- Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of malware. Rather, it is necessary only to identify the small set of actions that indicate malicious activity is being attempted and then to intervene.
  - For example, an activity trap might provide API monitors to check the malware actions or a set of fake email addresses to catch the spread
- Very important is also the activity after a trap effectively stopped a malware (malware intelligence):  
**Forensic analysis** of memory-resident malware can be achieved (e.g., with a tool such as AccessData FTK Imager), to capture a copy of an infected device's memory contents for analysis.
- Activity traps act like **honeypot for malwares**
  - Note: a full honeypot is a more complex host than a malware honeypot and is not only focused on malwares, rather catching all attacker behavior within the "trap".



```

root@kali:~# volatility --profile=win7SP1x86 -f memdump.mem netscan | grep ESTABLISHED
tcpv4 10.24.130.4:49101 10.24.130.3:31337 ESTABLISHED -1
    
```



8 Dr. Valerio Formicola

```

root@kali:~# volatility --profile=win7SP1x86 -f memdump.mem netscan | grep ESTABLISHED
tcpv4 10.24.130.4:49101 10.24.130.3:31337 ESTABLISHED -1
    
```

Third-generation programs are memory-resident programs that identify malware by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of malware. Rather, it is necessary only to identify the small set of actions that indicate malicious activity is being attempted and then to intervene.

Example of trap operation:

The Traps agent injects itself into each process as it is started. If the process attempts to execute any of the core attack techniques, the exploit attempt will fail because Traps had made the process impervious to those techniques. Traps will immediately block that technique, terminate the process, and notify both the user and the admin that an attack was prevented and report all of the details to the an endpoint security manager.

Forensic information available after an attack has been prevented is unavoidably less than the information available about an attack that has succeeded and done damage. Despite that, there is still a great amount of intelligence that can be gathered. By capturing all the forensics of the attempted

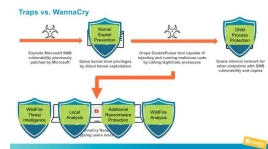
attack, organizations can apply proactive defenses to other endpoints that may not be protected. |

Example of traps: <https://www.paloguard.com/Endpoint-Protection.asp>

## 4th Generations of Anti-Virus Software

- **Fourth generation: full-featured protection**

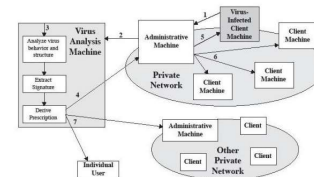
- Packages consisting of a variety of anti-virus techniques used in conjunction
- These includes scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.



<https://www.paloaltonetworks.com/blog/2019/01/preventing-malware-ransomware-traps/>



9 Dr. Valerio Formicola



Fourth-generation products are packages consisting of a variety of anti-virus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of malware to penetrate a system and then limits the ability of a malware to update files in order to propagate.

The arms race continues. With fourth-generation packages, a more comprehensive defense strategy is employed, broadening the scope of defense to more general-purpose computer security measures. These include more sophisticated anti-virus approaches.

## Host-Based Dynamic/Behavior Malware Analysis

- Integrates with the operating system of a host computer and monitors program behavior in real time for malicious action
  - Blocks potentially malicious actions before they have a chance to affect the system
  - Blocks software in real time so it has an advantage over anti-virus detection techniques such as fingerprinting or heuristics
- Advantages
  - If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software. This gives it a fundamental advantage over such established antivirus detection techniques as fingerprinting (first generation) or heuristics.
- Limitations
  - Because malicious code must run on the target machine before all its behaviors can be identified, it can cause harm before it has been detected and blocked

Monitored behaviors can include the following:

- Attempts to open, view, delete, and/or modify files;
- Attempts to format disk drives and other unrecoverable disk operations;
- Modifications to the logic of executable files or macros;
- Modification of critical system settings, such as start-up settings;
- Scripting of e-mail and instant messaging clients to send executable content; and
- Initiation of network communications.

```
"Time of Day", "Process Name", "PID", "Operation", "Path"
"3:11:17.8988718 PM", "Explorer.EXE", "2368", "RegQueryV
"3:11:17.8988843 PM", "Explorer.EXE", "2368", "RegSetVal
"3:11:17.8988899 PM", "Explorer.EXE", "2368", "RegSetVal
"3:11:17.9011089 PM", "SearchIndexer.exe", "2700", "File
"3:11:17.9011173 PM", "SearchIndexer.exe", "2700", "File
"3:11:17.9011237 PM", "SearchIndexer.exe", "2700", "File
"3:11:18.0114931 PM", "WShareTray.exe", "2520", "CreateF
```

Logging system actions



CalPolyPomona

10 Dr. Valerio Formicola

Unlike heuristics or fingerprint-based scanners, dynamic malware analysis or behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real time for malicious actions [CONR02, EGEL12]. It is a type of host-based intrusion prevention system, which we will discuss further in Section 9.6. This software monitors the behavior of possibly malicious code, looking for potentially malicious actions, similar to the sandbox systems we discussed in the previous section. However, it then has the capability to block malicious actions before they can affect the target system. Monitored behaviors can include the following:

- Attempts to open, view, delete, and/or modify files;
- Attempts to format disk drives and other unrecoverable disk operations;
- Modifications to the logic of executable files or macros;
- Modification of critical system settings, such as start-up settings;

- Scripting of e-mail and instant messaging clients to send executable content; and
- Initiation of network communications.

Because dynamic analysis software can block suspicious software in real time, it has an advantage over such established anti-virus detection techniques as fingerprinting or heuristics. There are literally trillions of different ways to obfuscate and rearrange the instructions of a virus or worm, many of which will evade detection by a fingerprint scanner or heuristic. But eventually, malicious code must make a well-defined request to the operating system. Given that the behavior blocker can intercept all such requests, it can identify and block malicious actions regardless of how obfuscated the program logic appears to be.

Dynamic analysis alone has limitations. Because the malicious code must run on the target machine before all its behaviors can be identified, it can cause harm before it has been detected and blocked. For example, a new item of malware might shuffle a number of seemingly unimportant files around the hard drive before modifying a single file and being blocked. Even though the actual modification was blocked, the user may be unable to locate his or her files, causing a loss to productivity or possibly worse.

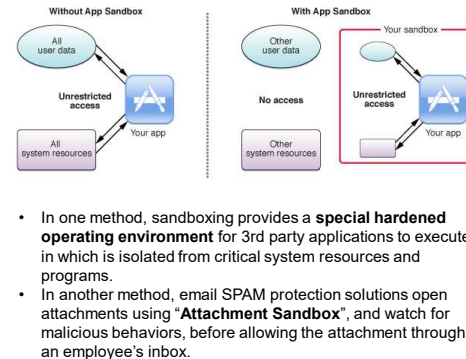
## Sandbox Analysis (1/2)

- A **sandbox** is a safe, isolated environment.
  - Can be based on emulation (CPU, I/O, Networks, memories, file systems) or virtual machines
- **Sandbox Analysis:** Running potentially malicious code in an **emulated sandbox or on a virtual machine**
  - Allows the code to execute in a controlled environment where its behavior can be closely monitored without threatening the security of a real system
  - Running potentially malicious software in such environments enables the detection of complex encrypted, polymorphic, or metamorphic malware
- The most difficult design issue with sandbox analysis is to determine **how long to run each interpretation**
  - Some malwares might have a logic bomb that activates them after some time to not be caught in a sandbox



CalPolyPomona

11 Dr. Valerio Formicola



- In one method, sandboxing provides a **special hardened operating environment** for 3rd party applications to execute in which is isolated from critical system resources and programs.
- In another method, email SPAM protection solutions open attachments using "**Attachment Sandbox**", and watch for malicious behaviors, before allowing the attachment through to an employee's inbox.

One method of detecting and analyzing malware involves running potentially malicious code in an emulated sandbox or on a virtual machine. These allow the code to execute in a controlled environment, where its behavior can be closely monitored without threatening the security of a real system. These environments range from sandbox emulators that simulate memory and CPU of a target system, up to full virtual machines, of the type we will discuss in Section 12.8, that replicate the full functionality of target systems, but which can easily be restored to a known state. Running potentially malicious software in such environments enables the detection of complex encrypted, polymorphic, or metamorphic malware. The code must transform itself into the required machine instructions, which it then executes to perform the intended malicious actions. The resulting unpacked, transformed, or decrypted code can then be scanned for known malware signatures, or its behavior monitored as execution continues for possibly malicious activity [EGEL12, KERA16]. This extended analysis can be used to develop anti-virus signatures for new, unknown malware.

The most difficult design issue with sandbox analysis is to determine how long to run each interpretation.

Typically, malware elements are activated soon after a program begins executing, but recent malware increasingly uses evasion approaches such as extended sleep to evade detection in the analysis time used by sandbox systems [KERA16]. The longer the scanner emulates a particular program, the more likely it is to catch any hidden malware. However, the sandbox analysis has only a limited amount of time and resources available, given the need to analyze large amounts of potential malware.

As analysis techniques improve, an arms race has developed between malware authors and defenders. Some malware checks to see if it is running in a sandbox or virtualized environment, and suppresses malicious behavior if so. Other malware includes extended sleep periods before engaging in malicious activity, in an attempt to evade detection before the analysis terminates. Or the malware may include a logic bomb looking for a specific date, or specific system type or network location before engaging in malicious activity, which the sandbox environment does not match. In response, analysts adapt their sandbox environments to attempt to evade these tests. This race continues.

## Sandbox Analysis (2/2)

### What a sandbox analysis produces

- Provides file system, registry keys, and network traffic monitoring in controlled environment and produces a well formed report
- Using a sandbox is more efficient and sometimes more effective
- Configure your own sandbox such as Joebox, GFI Sandbox, and Cuckoo Sandbox.
- Use public sandbox such as ThreatExpert, GFI ThreatTrack, and Anubis
  - Do not submit malware to a public sandbox if it reveals sensitive information about your organization and/or customer.

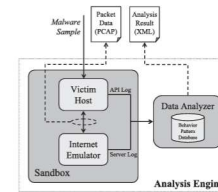
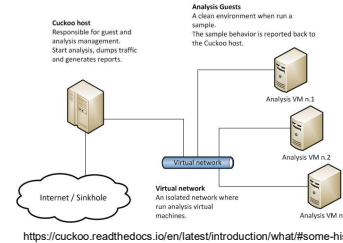


Fig. 3 Sandbox and data analyzer.

### Cuckoo Sandbox project



[https://www.jstage.jst.go.jp/article/transinf/E92.D/5/E92.D\\_5\\_945/\\_pdf/-char/en](https://www.jstage.jst.go.jp/article/transinf/E92.D/5/E92.D_5_945/_pdf/-char/en)

API hooking is a technique very common in SandBox analysis:

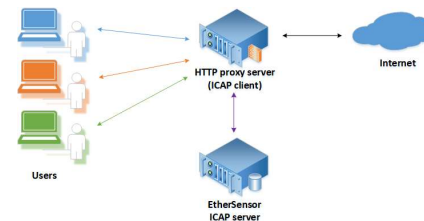
<https://www.cynet.com/attack-techniques-hands-on/api-hooking/>

In [computer programming](#), the term **hooking** covers a range of techniques used to alter or augment the behaviour of an [operating system](#), of [applications](#), or of other software components by intercepting [function calls](#) or [messages](#) or [events](#) passed between [software components](#). Code that handles such intercepted function calls, events or messages is called a **hook**.



## Perimeter Scanning Approaches (1 of 2)

- Anti-virus software typically included in e-mail and Web proxy services running on an organization's firewall and IDS
- May also be included in the traffic analysis component of an IDS
- May include intrusion prevention measures, blocking the flow of any suspicious traffic
- Approach is limited to scanning malwares



Cal Poly Pomona

13 Dr. Valerio Formicola

The next location where anti-virus software is used is on an organization's firewall and IDS. It is typically included in e-mail and Web proxy services running on these systems. It may also be included in the traffic analysis component of an IDS. This gives the anti-virus software access to malware in transit over a network connection to any of the organization's systems, providing a larger scale view of malware activity. This software may also include intrusion prevention measures, blocking the flow of any suspicious traffic, thus preventing it reaching and compromising some target system, either inside or outside the organization.

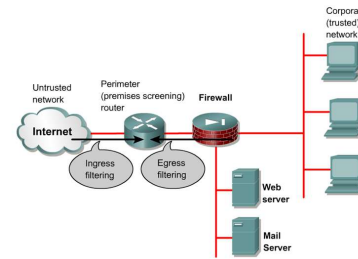
There is also a protocol to perform perimetral operations like the **Internet Content Adaptation Protocol (ICAP)**. It was introduced in 1999 by the ICAP forum. It is a lightweight HTTP-based RPC protocol designed to offload processing of Internet-based content to dedicated servers. Specifically, the goal of ICAP is to offload tasks like antivirus scanning onto specialized servers to increase network throughput. As an open protocol, the support of ICAP AV scanning for large storage arrays or secure web gateways is ubiquitous. Most of these devices can be configured to forward files when they are open, written, or transferred to an ICAP AV scanner.

[https://en.wikipedia.org/wiki/Internet\\_Content\\_Adaptation\\_Protocol](https://en.wikipedia.org/wiki/Internet_Content_Adaptation_Protocol)

## Perimeter Scanning Approaches (2 of 2)

Two types of monitoring software

- **Ingress monitors**
  - Located at the border between the enterprise network and the Internet
  - One technique is to look for incoming traffic to unused local IP addresses
- **Egress monitors**
  - Located at the egress point of individual LANs as well as at the border between the enterprise network and the Internet
  - Monitors outgoing traffic for signs of scanning or other suspicious behavior



14 Dr. Valerio Formicola

However, this approach is limited to scanning the malware content, as it does not have access to any behavior observed when it runs on an infected system. Two types of monitoring software may be used:

- **Ingress monitors:** These are located at the border between the enterprise network and the Internet. They can be part of the ingress filtering software of a border router or external firewall or a separate passive monitor. A honeypot can also capture incoming malware traffic. An example of a detection technique for an ingress monitor is to look for incoming traffic to unused local IP addresses.
- **Egress monitors:** These can be located at the egress point of individual LANs on the enterprise network as well as at the border between the enterprise network and the Internet. In the former case, the egress monitor can be part of the egress filtering software of a LAN router or switch. As with ingress monitors, the external firewall or a honeypot can house the monitoring software. Indeed, the two types of monitors can be collocated. The egress monitor is designed to catch the source of a malware attack by monitoring outgoing traffic for signs of scanning or other suspicious behavior.

Perimeter monitoring can also assist in detecting and responding to botnet activity by detecting abnormal traffic patterns associated with this activity. Once bots are activated and an attack is underway, such monitoring can be used to detect the attack. However, the primary objective is to try to detect and disable the botnet during its construction phase, using the various scanning techniques we have just discussed, identifying and blocking the malware that is used to propagate this type of payload.