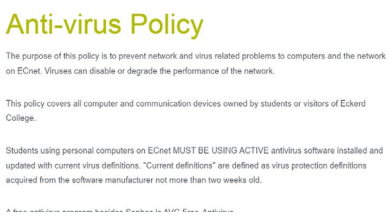
Countermeasures against malware

Malware Countermeasure Approaches

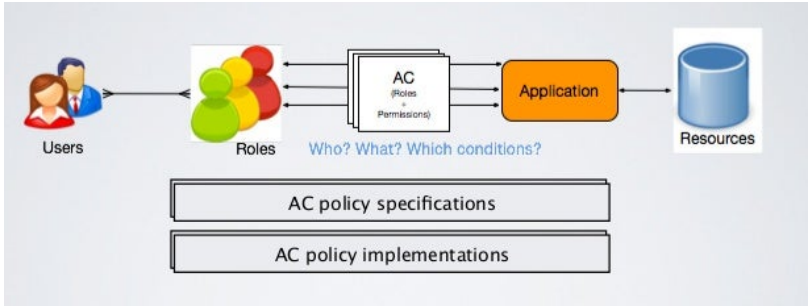
- Ideal solution to the threat of malware is prevention:
  1. Do not allow malware to get into the system in the first place
  2. Block the ability of it to modify the system
- Four main elements of prevention (might overlap as a categories):
  - Policy: installation and maintenance of Anti-Viruses and 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

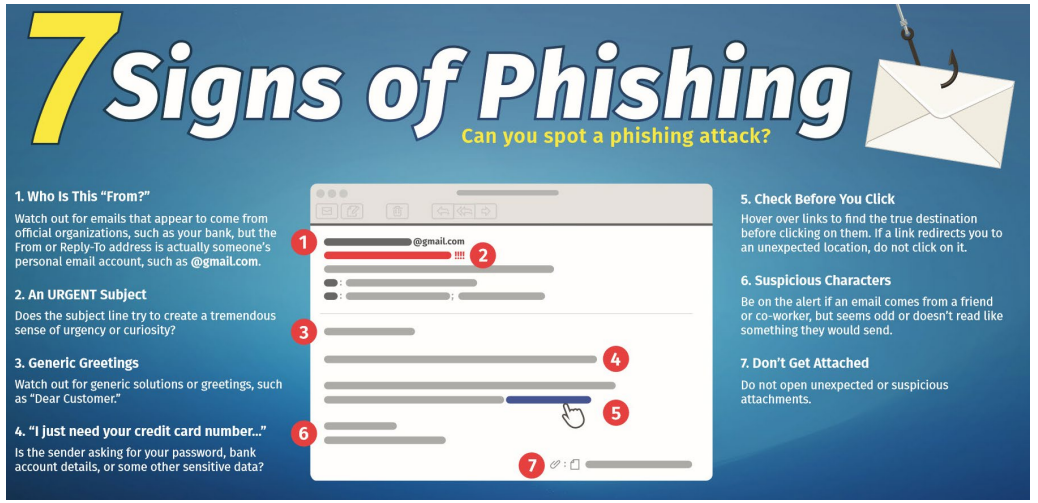

Anti-virus Policy

The purpose of this policy is to prevent network and virus related problems to computers and the network on ECnet. Viruses can disable or degrade the performance of the network.

This policy covers all computer and communication devices owned by students or visitors of Eckerd College.

Students using personal computers on ECnet MUST BE USING ACTIVE antivirus software installed and updated with current virus definitions. "Current definitions" are defined as virus protection definitions acquired from the software manufacturer not more than two weeks old.



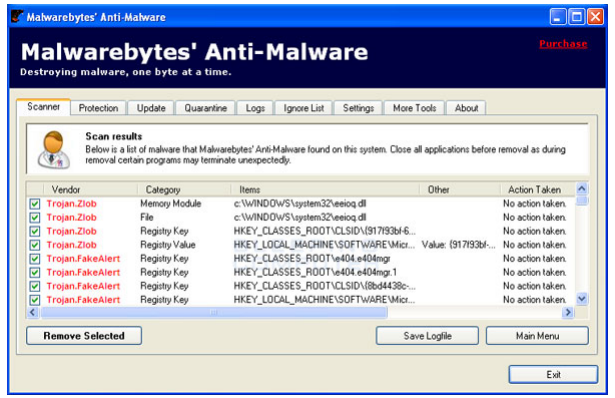


In the case of infection from malware

- If prevention fails, technical mechanisms can be used to support the following threat mitigation options:
  1. **Detection**: Once the infection has occurred, determine that it has occurred (detect) and locate the malware (locate).
  2. **Identification:** Once detection has been achieved, identify the specific malware that has infected the system (diagnosis/identification).
  3. **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



Requirements and placement of malware detectors

Requirements of countermeasures for malwares:

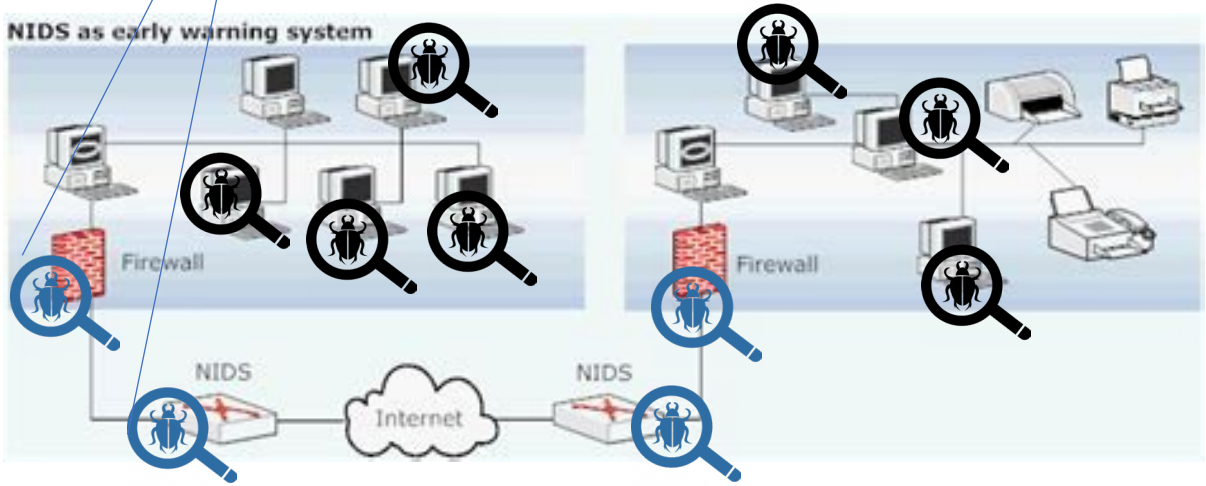
- **Generality:** The approach taken should be able to handle a wide variety of attacks.
- **Timeliness:** The approach should respond quickly so as to limit the number of infected programs or systems and the consequent activity.
- **Resiliency:** The approach should be resistant to evasion techniques 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.

Central anti-virus collect events and reports from host based and perimetral anti malwares and anti-viruses (comprehensive view, intense load)

Perimetral anti-viruses: analyze network traffic for signs of malware activities (only parts visible on the network traffic)

Host-based anti-virus: check activities on individual hosts (limited view of the entire network)



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_MACOS_6175e25
{
    meta:
        description = "MACOS.6175e25"
    strings:
        $a1 = { 00 25 40 25 40 25 40 25 40 00 25 63 00 }
        $a2 = { 64 65 6c 65 74 65 41 70 70 42 79 53 65 6c 66 }
        $a3 = { 65 6e 63 72 79 70 74 44 65 63 72 79 70 74 4f 70 65 72 61 74 69 6f 6e }
        $a4 = { 45 6e 63 6f 64 65 44 65 63 6f 64 65 4f 70 73 }
        $a5 = { 63 72 65 61 74 46 69 6c 65 4f 6e 54 65 6d 70 3a 73 63 72 70 4e 61 6d 65 3a }
    condition:
        Macho and all of ($a*) and filesize < 200KB
}
```

```
rule Korplug_TrojanLoader_PayloadNames {
    strings:
        $str1 = "msi.dll.eng" wide fullword
        $str2 = "McUtil.dll.url" wide ascii nocase
condition:
    (uint16(0) == 0x5A4D) and
    (filesize < 11KB) and
    any of them
}
```

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

```
                 No matching text
Input =          baaabab
Pattern =        a  *    ab
Output : false
```

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

Virus Name	*String Pattern (Signature)*
Accom.1280	89C3 B440 8A2E 2004 8A0E 2104 BA00 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

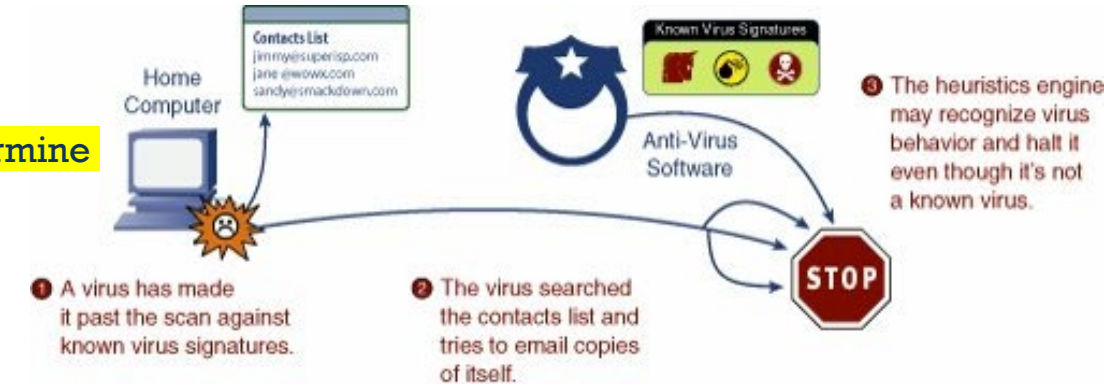
Second Generation: heuristics

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



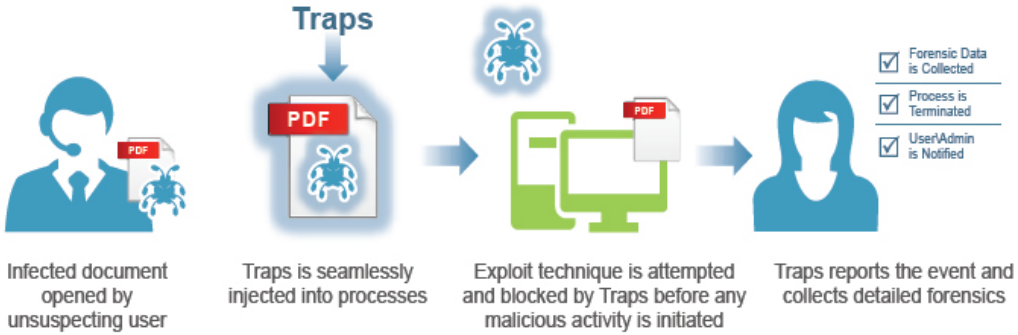
- Two categories of heuristics:
  - **Static heuristic analysis**: involves decompiling/disassembling a suspected program that might contain the virus; then, examining the obtained source code of the program for comparing it to the source code of known viruses that have already been logged in a database. 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).

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".

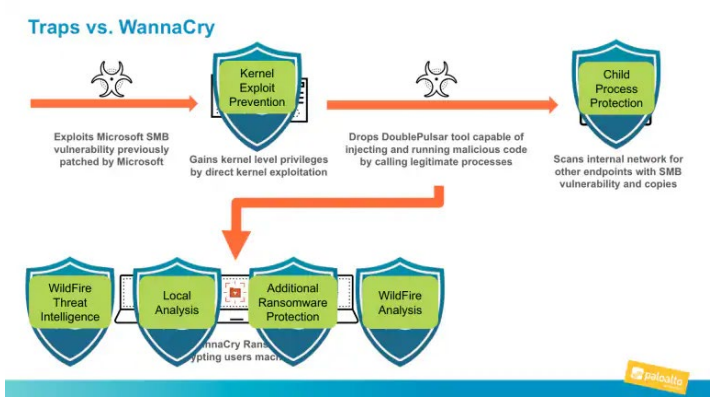


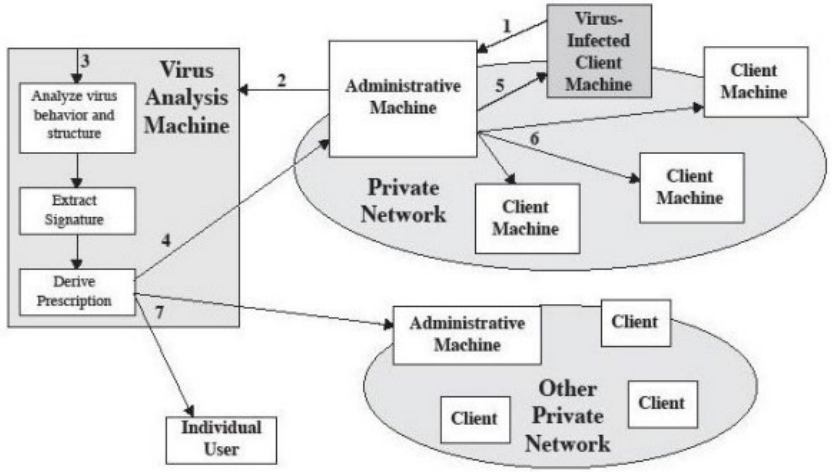

```
Process: TPS Report.exe Pid: 2504 Address: 0x20000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00020000  db cf d9 74 24 f4 bf a6 70 1c 12 58 29 c9 b1 54   ...t$...p..X)..T
0x00020010  31 78 18 03 78 18 83 e8 fc e2 f5 fc e8 82 00 00   1x..x...........
0x00020020  00 60 89 e5 31 c0 64 8b 50 30 8b 52 0c 8b 52 14   .`..1.d.P0.R..R.
0x00020030  8b 72 28 0f b7 4a 26 31 ff ac 3c 61 7c 02 2c 20   .r(..J&1..<a|.,.

0x00020000 dbcf            FCMOVNE ST0, ST7
0x00020002 d97424f4        FNSTENV [ESP-0xc]
0x00020006 bfa6701c12      MOV EDI, 0x121c70a6
0x0002000b 58              POP EAX
0x0002000c 29c9            SUB ECX, ECX
0x0002000e b154            MOV CL, 0x54
0x00020010 317818          XOR [EAX+0x18], EDI
0x00020013 037818          ADD EDI, [EAX+0x18]
0x00020016 83e8fc          SUB EAX, -0x4
0x00020019 e2f5            LOOP 0x20010
0x0002001b fc              CLD
0x0002001c e882000000      CALL 0x200a3
0x00020021 60              PUSHA
0x00020022 89e5            MOV EBP, ESP
0x00020024 31c0            XOR EAX, EAX
0x00020026 648b5030        MOV EDX, [FS:EAX+0x30]
0x0002002a 8b520c          MOV EDX, [EDX+0xc]
0x0002002d 8b5214          MOV EDX, [EDX+0x14]
0x00020030 8b7228          MOV ESI, [EDX+0x28]
0x00020033 0fb74a26        MOVZX ECX, WORD [EDX+0x26]
```

```
root@kali:~# volatility --profile=Win7SP1x86 -f memdump.mem netscan | grep ESTABLISHED
Volatility Foundation Volatility Framework 2.6
0x2eedddf8         TCPv4    10.24.130.4:49161              10.24.130.3:31337    ESTABLISHED      -1
```

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/

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:

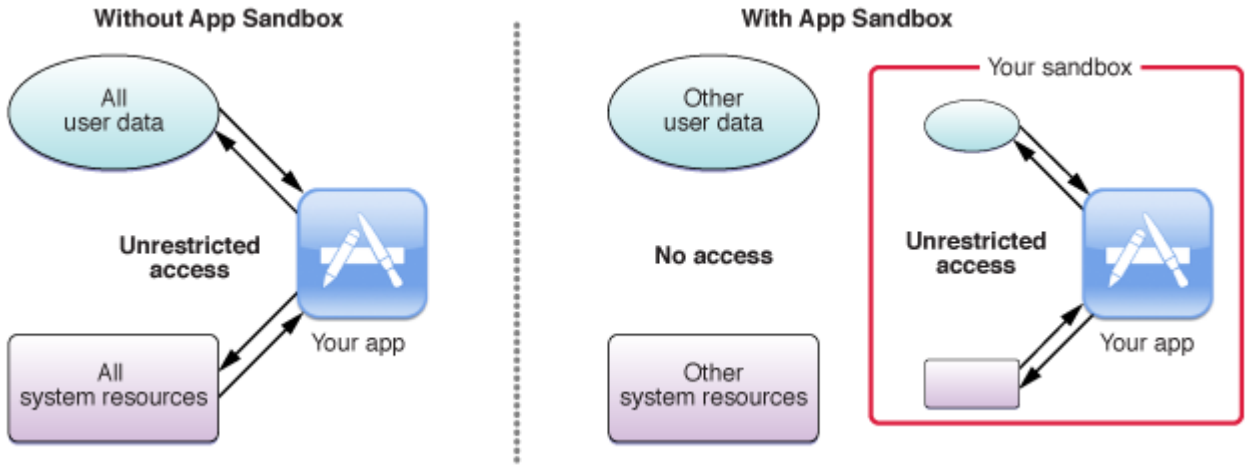
- Attempts to open, view, delete, and/or modify files;
- Attempts to format disk drives and other unrecoverable disk operations;
- Modifications to the logic of executable files or macros;
- Modification of critical system settings, such as start-up settings;
- Scripting of e-mail and instant messaging 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","VMwareTray.exe","2520","CreateF
```

Logging system actions

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



- 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.

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.

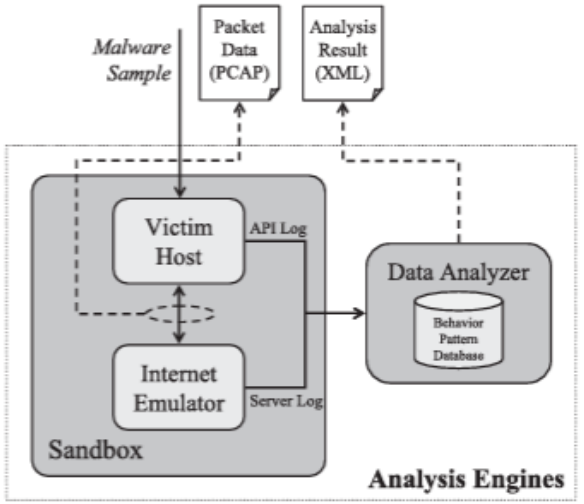
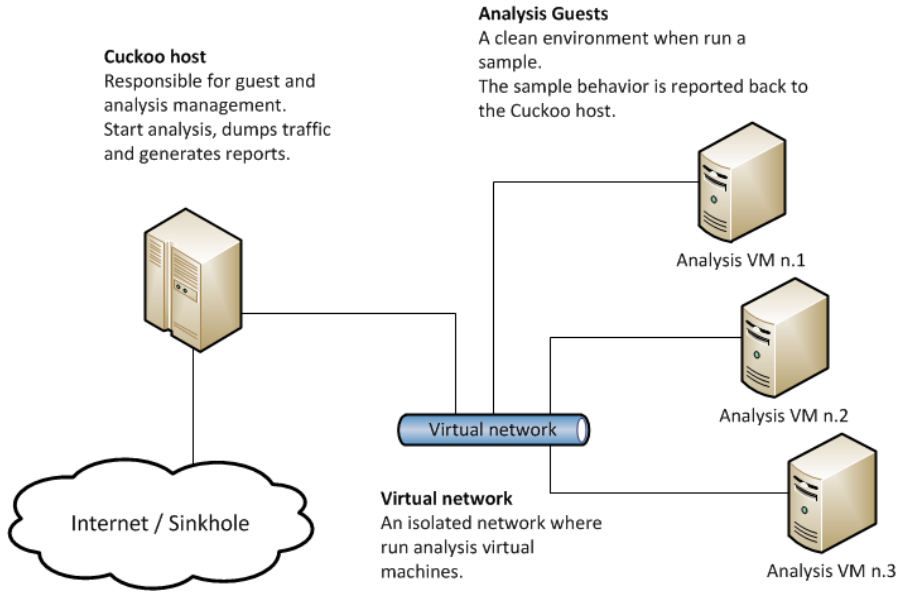


Fig. 3 Sandbox and data analyzer.

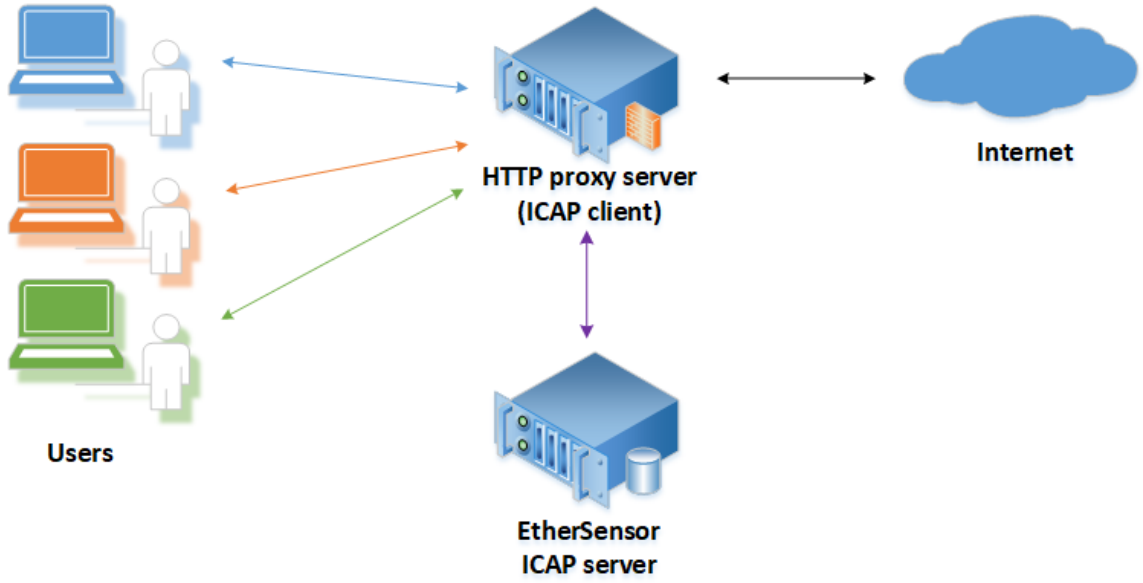
Cuckoo Sandbox project



https://cuckoo.readthedocs.io/en/latest/introduction/what/#some-history

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



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