Automated Malware Analysis

Why We Need Automated Malware Analysis (Sandboxes)

Automated malware analysis systems (or sandboxes) are one of the latest weapons in the arsenal of security vendors. Such systems execute an unknown malware program in an instrumented environment and monitor their execution. While such systems have been used as part of the manual analysis process for a while, they are increasingly used as the core of automated detection processes. The advantage of the approach is clear: It is possible to identify previously unseen (zero day) malware, as the observed activity in the sandbox is used as the basis for detection.

Goals of a dynamic analysis system (sandbox)

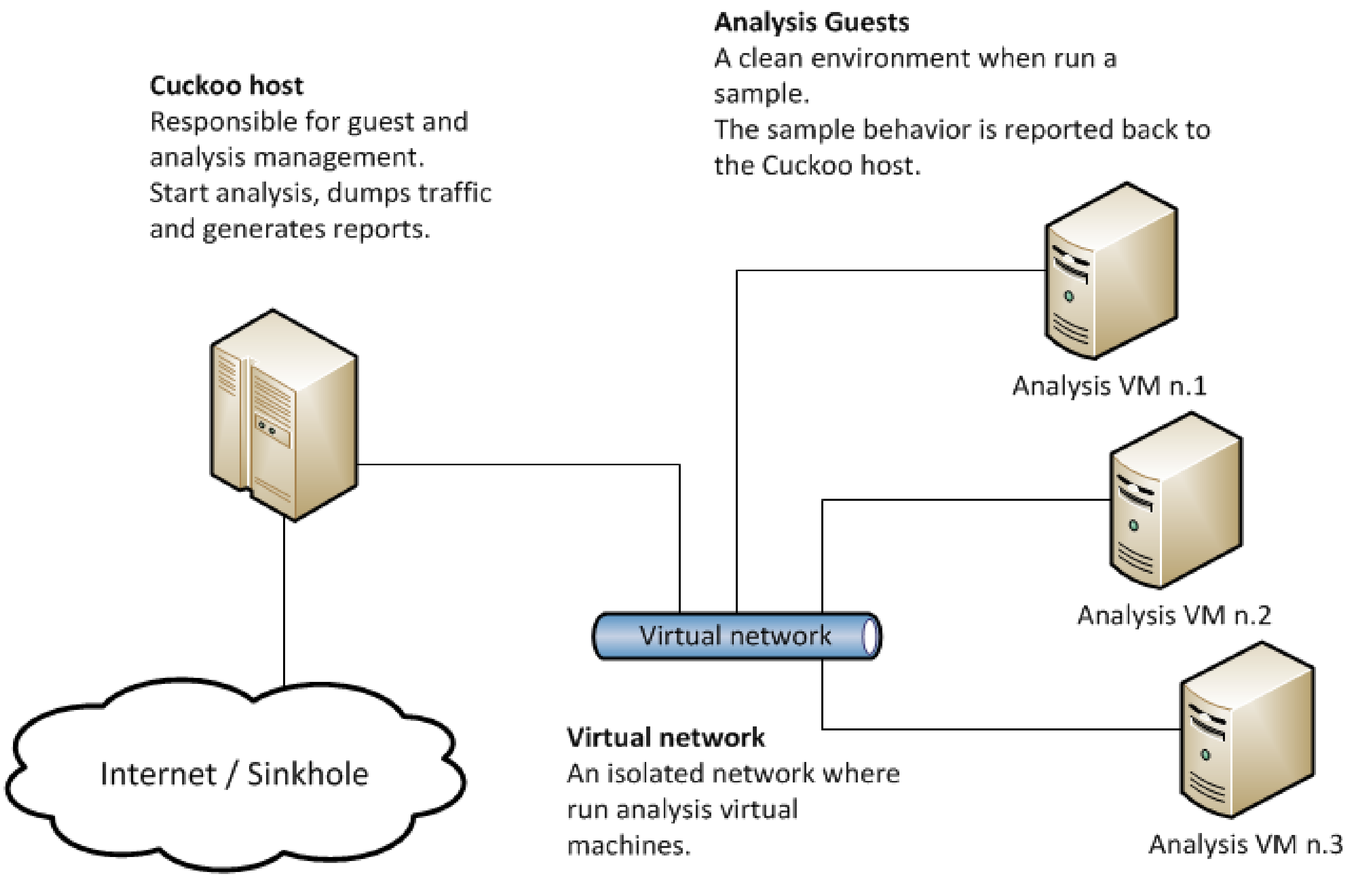
A good malware analysis sandbox has to achieve three goals: Visibility, resistance to detection, and scalability.

First, a sandbox has to see as much as possible of the execution of a program. Otherwise, it might miss relevant activity and cannot make solid deductions about the presence or absence of malicious behaviors.

Second, a sandbox has to perform monitoring in a fashion that makes it difficult to detect. Otherwise, it is easy for malware to identify the presence of the sandbox and, in response, alter its behavior to evade detection.

The third goal captures the desire to run many samples through a sandbox, in a way that the execution of one sample does not interfere with the execution of subsequent malware programs. Also, scalability means that it must be possible to analyze many samples in an automated fashion.

Case Study :CuCkoo Sandbox


Cuckoo host
Responsible for guest and analysis management.
Start analysis, dumps traffic and generates reports.
Analysis Guests
A clean environment when run a sample.
The sample behavior is reported back to the Cuckoo host.
Analysis VM n.1
Analysis VM n.2
Analysis VM n.3
Virtual network
Virtual network
An isolated network where run analysis virtual machines.
Internet / Sinkhole

Emulation versus virtualization

How we can build a sandbox that can collect this data in a way that makes it difficult for malware to detect. The two main options are virtualization and emulation.

An emulator is a software program that simulates the functionality of another program or a piece of hardware. Since an emulator implements functionality in software, it provides great flexibility. For example, consider an emulator that simulates the system hardware (such as the CPU and physical memory). When you run a guest program P on top of this emulated hardware, the system can collect very detailed information about the execution of P. The guest program might even be written for a different CPU architecture than the actual CPU that the emulator runs on. This mechanism allows, for example, to run an Android program, written for ARM, on top of an emulator that runs on an x86 host. The drawback of emulation is that the software layer incurs a performance penalty. The potential performance impact has to be carefully addressed to make the analysis system scalable.

With virtualization, the guest program P actually runs on the underlying hardware. The virtualization software (the hypervisor) only controls and mediates the accesses of different programs (or different virtual machines) to the underlying hardware. In this fashion, the different virtual machines are independent and

isolated from each other. However, when a program in a virtual machine is executing, it is occupying the actual physical resources, and as a result, the hypervisor (and the malware analysis system) cannot run simultaneously. This makes detailed data collection challenging. Moreover, it is hard to entirely hide the hypervisor from the prying eyes of malware programs.The advantage is that programs in virtual machines can run at essentially native speed.

Why Sandbox FAILS?

Malware samples can use the following techniques to detect whether they are being executed in an automated malware analysis environment:

❿ Detecting a sandbox:a sandbox provides a virtual environment where a malware sample can be executed to determine whether the sample is malicious or not.
❿ Detecting a debugger:when malware is analyzed in a debugger, it can use different functions and techniques to detect if it’s being analyzed. The debugger is usually used when analyzing malware samples manually and not by automated cloud malware analysis services, but can still provide different barriers a malware analyst must overcome to be able to analyze malicious samples.
❿ Detecting a virtual environment:almost all of the cloud automated malware services are analyzing malware samples in a virtualized environment. This is because they provide many advantages that are quite useful when doing malware analysis. An especially useful feature is snapshots, which can be used to revert the virtual machine prior to malware infection. So we can setup a virtual machine, usually running Windows operating system, install all the required tools that we need for malware analysis, and create a snapshot of that virtual machine. Then we can run the malware inside that virtual machine, obtaining all the interesting pieces of information we can get in order to determine whether the sample is malicious and what it does. After the analysis is complete, we can revert back to the snapshot we had created earlier and start with a clean system ready to analyze another malware sample.

Solution :Adding Machine Learning Module to Sandboxes

Right now machine learning (ML) is making progress versus standard sandbox or virus scanning. To explain at a fairly technical level, a sandbox is created to run an

unknown executable, and the changes it makes in the sandbox environment is used heuristically to determine if the executable is malware. Virus scanning is similar, but it uses files at rest. There are problems with both of these cyber security processes.

Many executables don't exist on their own, but instead wait for user input on when to stop execution (File->Close). Sandboxing must run applications for a given period of time, and then kill the application to examine the sandbox. This can be defeated by an application performing tasks that appear safe for a given period of time before executing the actual payload on the real host since the sandbox shuts down too early. The sandbox can also be detected by an executable examining the environment in a subtle way, and once it determines it is in a sandbox, act differently to appear benign. Since each version of the product has a given set of sandboxes, hackers write libraries of code that detect such sandboxes.
ML is now being used to replace heuristic analysis with a learned profile. Since ML can detect the features of malware, it no longer has to be a scoring system. Instead, security specialists can rely on the ability of ML to detect and provide a degree of certainty to the detection. Using this system, more positive detections can be found by features instead of exact matches or scored matches. This will help defeat polymorphic viral code at rest, instead of introducing the limitations of a sandbox. When most companies speak of ML, they are referring to this type of learning which solves a classification problem.

How is machine learning improving cyber security?
ML improves cyber security by providing detection based on hidden variables that are difficult for humans to analyze. In all cases, given enough time, a team of data scientists can find such relationships in data, but this is often a slow process. By using ML, we can allow a system to do this for us. That means different attacks can be detected based on previous data, even if the attacks use different vectors. In other words, it moves from "heuristics" that require human learning to teach the system how to grade behaviors, to using features in the data that are often deeper and harder to find. This can be performed much faster since no human analysis is required.

Are attacks more commonly manual or automated?
Most attacks are automated, as many hackers have libraries of attack tools they use. These tools have been developed by manually attacking systems that are local to the developer. I can attack my own system using a given version of the operating system or application, determine how this works, and create an automated tool to perform the action. This tool might be manually targeted, but once initiated it will automatically perform all actions. That is not to say that these tools are not used in sequence with the output of one tool determining the next one to use, but whenever possible even that can be automated.