Secure Downloads

Background

Relevant Technologies

Patent 5,974,549 – In this invention a security monitor is put into the address space of an application such as a web browser so that it can monitor any downloaded components (such as Active X components) to search for any security issues (viruses, worms, spyware, etc.).

Patent 5,305,456 - In this invention uses an interface so that the operating systems security parameters are extended to include a specific application.

Patent 1,1409,276 – In this invention software/components are cryptographically signed for safe distribution over a network.

Patent 7,107,618 – In this invention verifies that data that is sent is free from viruses as determined by comparison to a known virus data file. A digital certificate is sent with the data to confirm its status as virus free.

The New Process

The problem with the processes in all above patents is that they obviously do not truly guarantee that the software, data, attachments are truly safe. They can help to ameliorate the problem of unsafe downloads, but they do not eliminate it. For example attempting to isolate a downloaded component into a separate memory space, does not prevent that component (if it is malicious) from accessing the System Registry, files, etc.

In this new process the host operating system also has a default virtual machine. Unlike most virtual machines this one would:

a. Be relatively small. It would hold the operating system and a small area for software testing. Hard drive usage of approximately 5% of the total hard drive space would be more than adequate.
b. It would have an operating system identical to the host operating system. In other words if the host is Windows 7, then the virtual machine would also be Windows 7. Most often virtual machines are utilized to host alternative operating systems, such as Linux on a Windows machine. However the purpose of this VM is to allow testing of downloaded components and data prior to transferring them to the host. Therefore the VM would need to be the exact same operating system (including patches and updates) as the host machine.
c. It would mirror the host operating system configuration including ports, user accounts etc. This would be necessary in order to fully test downloaded data. Any difference in configuration between the host OS and the VM could lead to different conditions in existing in each, that might affect any testing done on the VM and make such testing inapplicable to the host OS>

d. There would have no access from the virtual machine to the host operating system. In many virtual machine implementations, the VM has a drive mapped to the host operating system and can thus copy files to and from the host operating system. In this process that mapped drive would not exist.

When software is downloaded it is actually held in memory, not saved to the hard drive, and is sent to the virtual machine. In one embodiment of this process, that would be accomplished via SCP (Secure Copy) with SSH (Secure Shell). In another embodiment of the process it could be transferred using telnet, sftp (secure FTP) or any other remote access process. A general overview of this process is shown in figure 1.1.

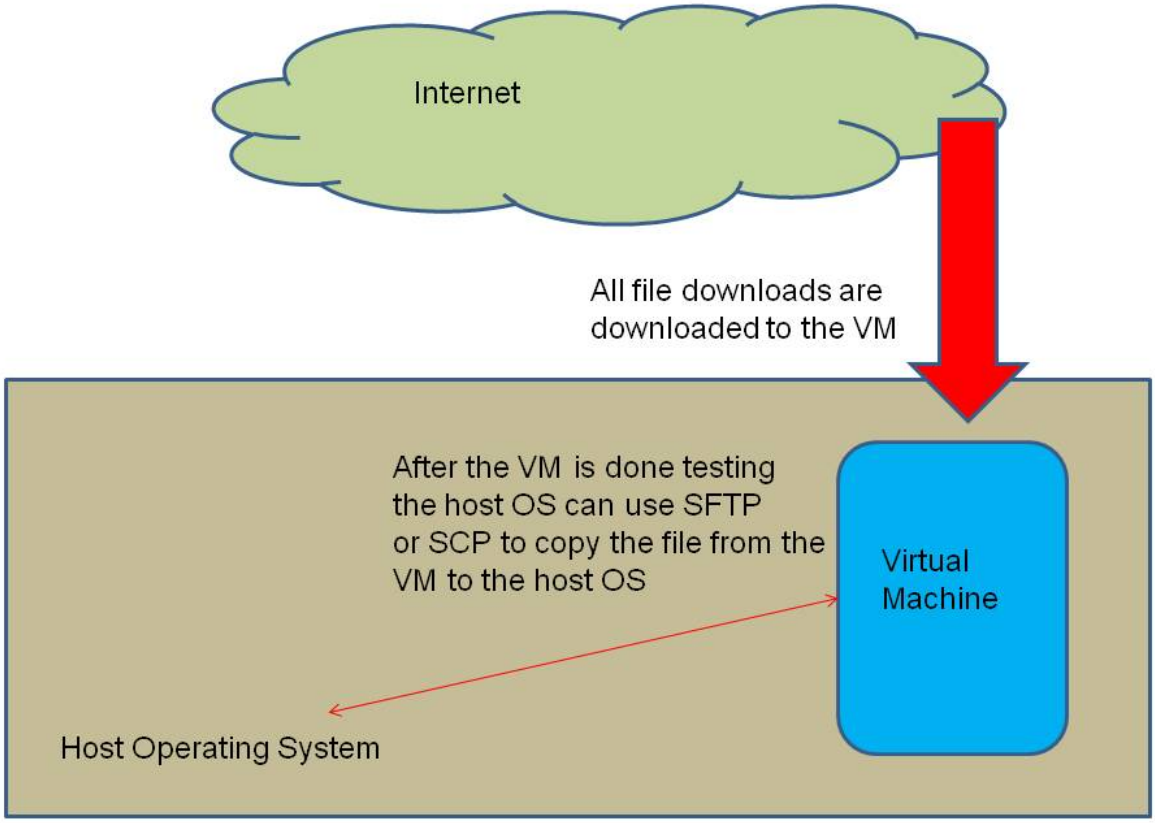

Figure 1.1 Overview

In the preferred embodiment of this operating system the host operating system is modified so that any file download such as via FTP, HTTP, or other download mechanism would be re-routed to the VM ip address. Any request (such as clicking on a download link in a browser) would trigger an SSH message sent to the virtual machine. That message would contain the IP or URL of the download and would direct the VM to execute a download. This is shown in figure 1.2.

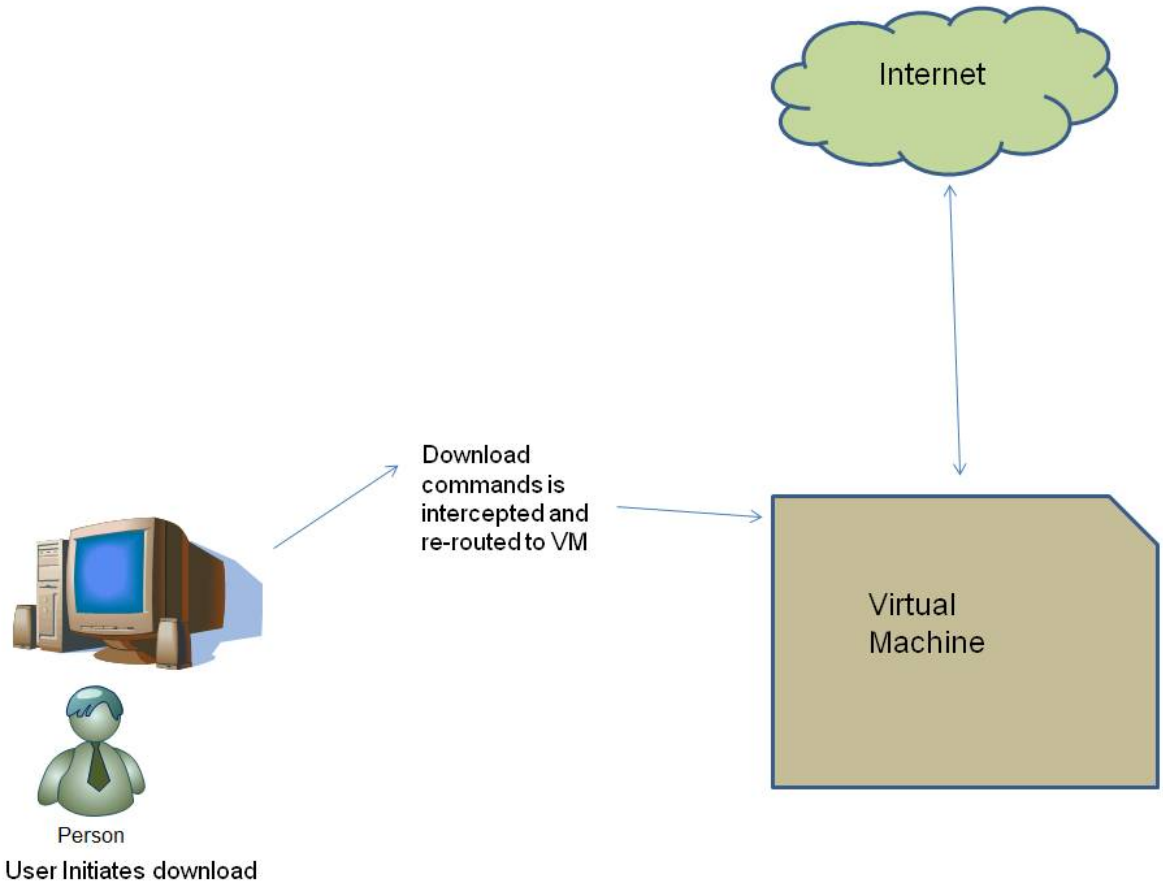

Figure 1.2 Download Interception

Once the download is on the virtual machine, should it be malware (virus, spyware, Trojan horse, adware, worm, etc.) it would have zero effect on the host machine.

Then the virtual machine would, at a minimum run a full virus scan. For added security the virtual machine would not be able to send any communication to the host machine, even to send notification to the host when the virus scan is done. Rather the host machine would periodically query the virtual machine (for example using SSH) to see of the virus scan was done and if the result was negative (i.e. no malware). Then the host machine would initiate a SCP (secure copy using SSH) or sftp (Secure FTP) connection to the virtual machine and copy the now verified software onto the host machine. It would then be deleted from the virtual machine, thus leaving space for the next software to be tested.

A detailed diagram of communication between host and VM is shown in image 1.3.

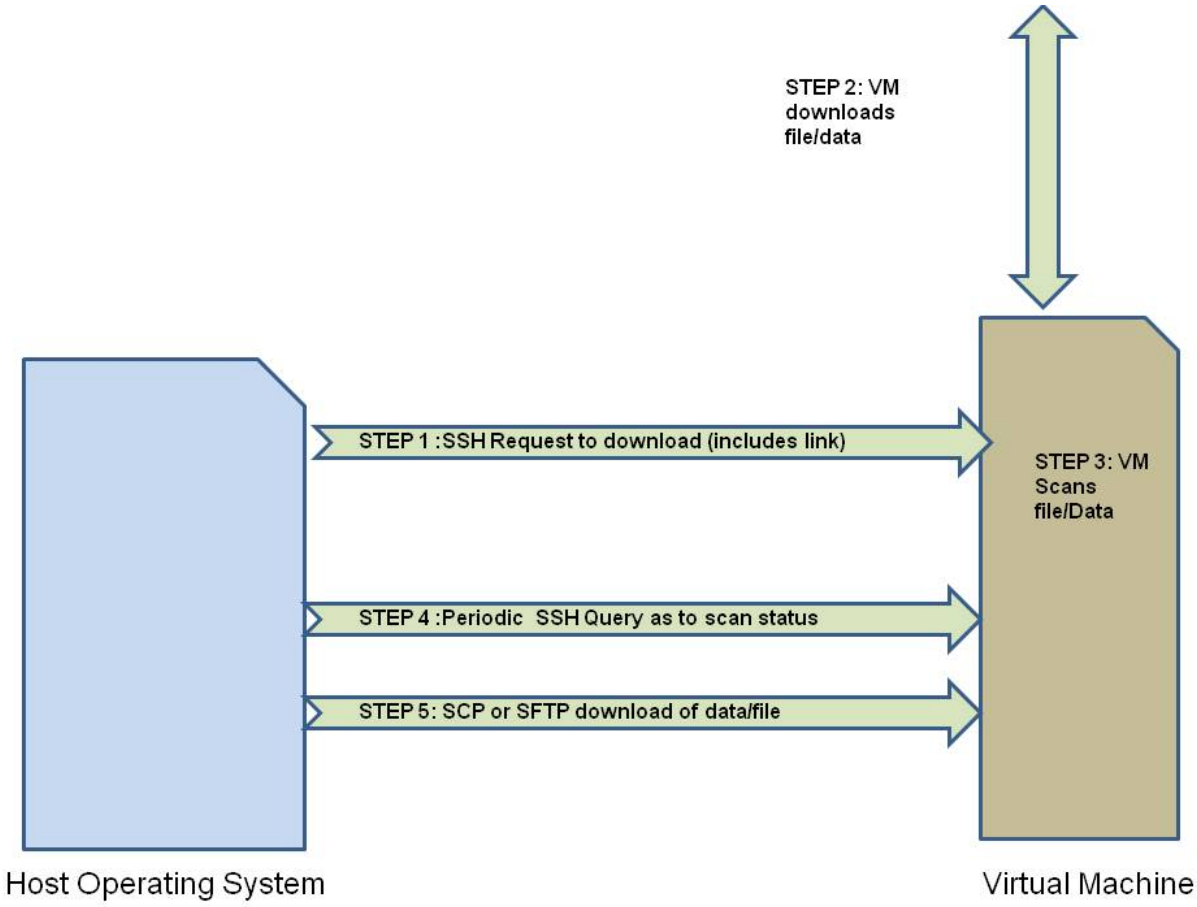

Figure 1.3 Detailed diagram of VM-Host Communication

As you can readily see this is a 5 step process

1. The Host Operating System uses SSH to send a request to the Virtual Machine asking it to download a given file/data/component. That request will include the URL or IP address from which to download.
2. The VM executes that download.
3. The VM begins its scanning. That could be simply a virus scan, or something much more detailed and extensive.
4. The Host Operating System periodically queries the VM (via SSH) as to the status of the scan.
5. When a query returns that the scan is complete, the Host Operating System then initiates a SCP or SFTP connection to the VM And downloads the data/file/component that has now been deemed secure.

The purpose of having the virtual machine only accessible via secure communication (SSH, STP, etc.) is to make very certain that it is truly completely isolated from the host. For the same reason communication with the virtual machine would only be one way (i.e. from the host to the VM). The VM would be unable to initiate any communication with the host.

In yet another embodiment of this process, additional testing could be done on the virtual machine. For example some rather infamous viruses have done their damage on a specific date. One test that could be done on the virtual machine would be to increase the system time by one year and examine the results. In yet other embodiments additional tests could be done including monitoring to see if the software downloaded attempted to alter the Windows Registry (for Windows systems), affect system logs, open any ports, scan any documents, or conduct any other undesirable activities.

This process would introduce a delay on downloads of anywhere from a few minutes to a few hours (depending on the scans and tests done). However it would make for truly safe downloads from the internet. Something that has been unattainable by any previous means.

As to how this process could be implemented, there are primarily three mechanisms:

1. In the first, and preferred embodiment of this process, is to have it integrated into the operating system itself by an operating system vendor. In that way any request for downloads would be routed through the VM security system. For example in the Windows Operating System's all network traffic is routed though a few dlls (Dynamic Linked Library). An additional dll could be added to the operating system in order to facilitate this process being integrated into the operating system. That process is shown in figure 1.4

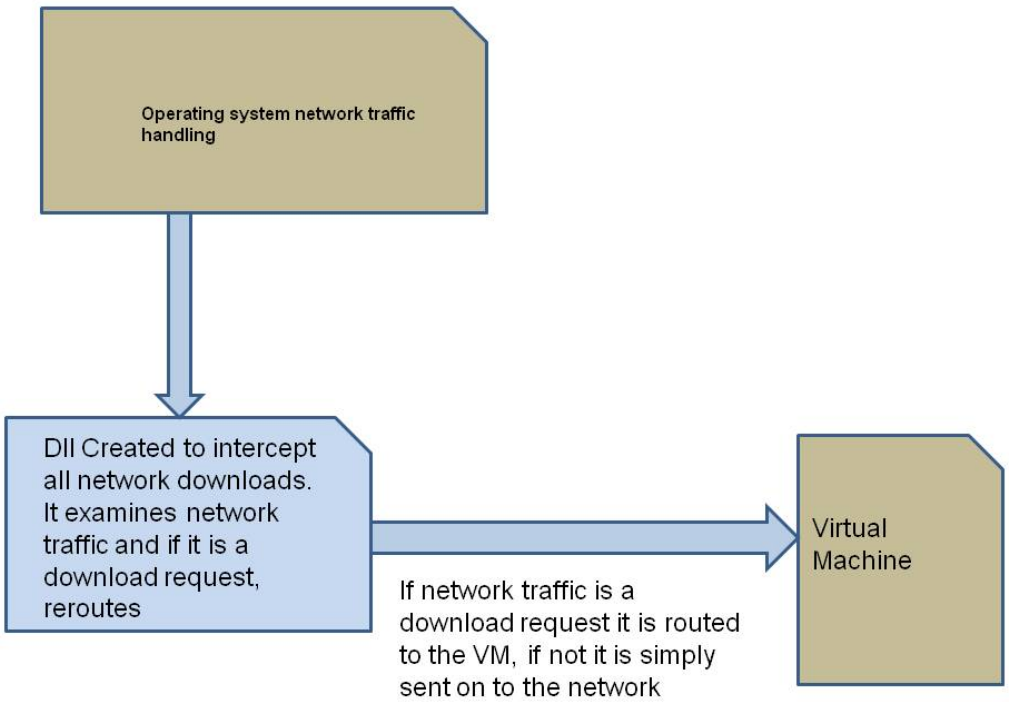

Figure 1.4 Integrating this process with the Operating System

A similar process could be adapted for other operating systems such as Unix, Linux, Free BSD, etc.

2. In another embodiment of the process, it would be integrated into a browser. In this way any download requests made via that browser would be routed through the VM security system. This would effectively make that browser a more secure alternative to other browsers. However downloads via email links, other browsers, or other sources would not be protected. This is shown in figure 1.5.

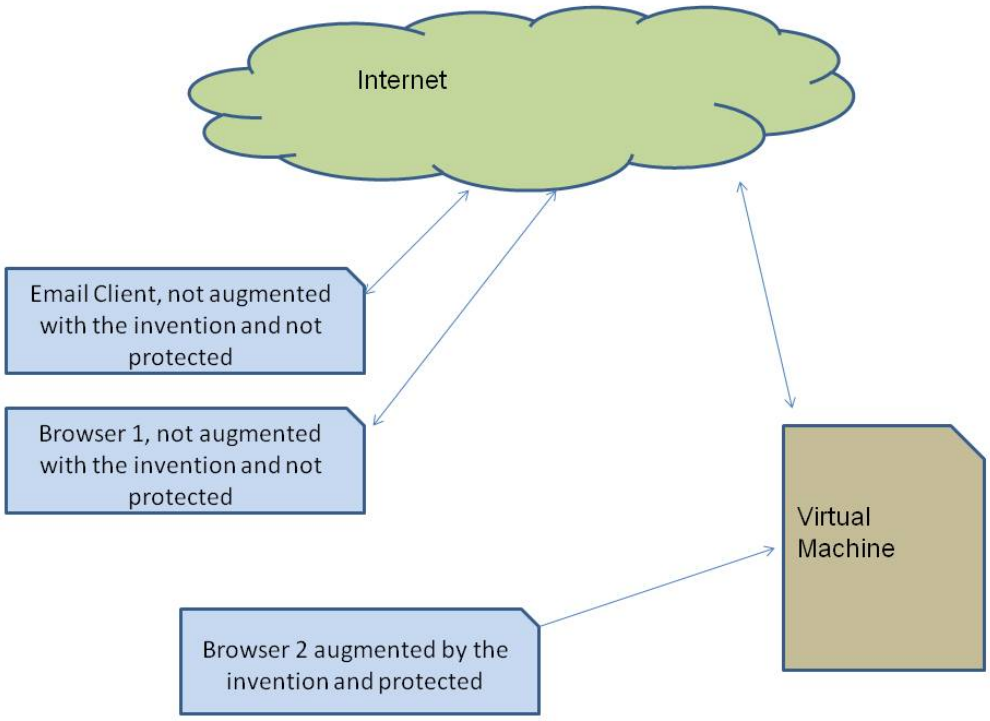

Figure 1.5 Secure Browsing

3. In yet another embodiment of the process, it would be a separate product installed on a given system. The host operating system would have to be modified (such as changing the registry in Windows) so that download requests would be routed through the VM security system.