38. Malware

38. Malware

38.1. Overview

Malware, or malicious software (also known as malcode), is any type of attacker code that runs on victim computers. One of the primary ways that malware is able to propagate is through self-replicating code, which is a code snippet that outputs a copy of itself (usually to send to other people). For example, suppose a piece of malware runs on your computer; in addition to, say deleting all your files or turning on your webcam, the malware also outputs a copy of itself and sends it to other computers, thereby infecting other devices.

Viruses and worms are two categories of self-propagating malware wherein the malicious code sends copies of itself to other users. A virus is a piece of malware or malcode that requires some user action to propagate, meaning that the user has to take some action in order for the virus to spread. Usually, once the computer gets infected, a piece of code is stored somewhere on the infected computer. Then, when the user runs the code, the virus gets spread to other users.

On the other hand, a worm is a piece of malware or malcode that does not require user action to propagate. Usually, rather than the infection happening on code that is stored on the computer that gets run later, it instead infects a computer by altering some already-running code. As such, no user interaction is required for the worm to spread to other users.

One possible application of malware is to construct a botnet. A botnet is a set of compromised machines, or bots, that are under centralized control, allowing the owner of the botnet to own a huge amount of resources that could be used for other attacks (like DoS). An attacker could use a virus or a worm to infect a large number of machines, causing every infected machine to now be under the control of the attacker.

38.2. Viruses

Recall that viruses are forms of malware that require user action to propagate, meaning that it usually infects a computer by altering some stored code and when the user runs the code, the malicious code spreads to other users. For example, an attacker could infect the start-up code of an application, meaning that when the user tries to open the application, the malcode will run and look for opportunities to infect more systems (i.e. forward itself to other users, copy itself onto a USB drive, etc.).

One common approach to detecting viruses is through signature-based detection. Since viruses are self-propagating, they often use copies of the same code. Since signature-based detection uses patterns of known attacks, a signature can be created on the virus (since the virus has been infecting several computers in the same manner using the same code snippet). So, the signature-based detection system will capture a virus on one system (usually through a sacrificial computer which opens a bunch of malicious files on purpose) and look for bytes corresponding to the malcode on other systems. Antivirus software performs these checks for you by usually including a checklist of common viruses. Most antiviruses will incorporate some form of signature-based detection and will use the signatures of these viruses to ensure that your computer is not infected. Stronger antivirus softwares will likely have a greater number of virus signatures than weaker ones, ensuring that your computer is protected from a wider range of attacks.

Viruses have existed for several decades, and there is a constant race that exists between attackers writing viruses and antivirus companies detecting viruses. As this arms race continues, propagation strategies of modern malware have evolved. Attackers tend to look for evasion strategies as they don't want to be detected by the antivirus software. As such, they could change the appearance of the virus so that each copy looks different, thus making signature-based detection much harder. Rather than changing the virus's appearance manually, certain evasion strategies attempt to automate this process through polymorphic code. In this arms race, since the attacker can see what detection strategies the antivirus software is using, but the antivirus cannot see what attacks the attacker is planning, the attackers often have a slight advantage. In other words, the attackers can see the defense strategies employed by the antivirus companies and therefore write evasion strategies to get around them (namely, the attacker knows the system). Therefore, since the detectors have to usually publish their code first, they are at a bit of a disadvantage.

38.3. Polymorphic and Metamorphic Code

In an attempt to continuously change the virus's appearance to avoid signature-based detection, attacks employ polymorphic code wherein each time the virus propagates, it inserts an encrypted copy of the malcode. This code also includes the key and the decryptor, so when the code runs, it uses the key and decryptor to obtain the original, plaintext malcode. Since encryption schemes produce different looking outputs on repeated encryptions (with IND-CPA secure schemes), the attacker is able to change the appearance of the virus to help avoid signature-based detections. However, note that encryption is being used for obfuscation and not for confidentiality. Namely, the attacker is not trying to hide the contents of the virus (rather, the malcode is going to get run eventually and the decoder and the key are sent in plaintext), but simply avoid detection by making every copy of the virus look different. As such, this also means that weaker encryption algorithms, like ECB, can be used (since our goal is not confidentiality) and the decryption keys can be sent in plaintext.

One possible defense against polymorphic code is to simply add a signature for detecting the decryptor code. For example, a possible signature could be a key being used to decrypt a certain piece of code. However, this raises a lot of false positives since there are a lot of pieces of code that are not malware, which use a key to decrypt other pieces of code. Furthermore, another issue arises if the decryptor code is scattered across different parts of memory as matching several small instructions is a lot harder than matching one big block of code. Another possible defense is to run the potentially dangerous code in a sandbox, or an isolated environment, where if something goes terribly wrong, nothing outside of the sandbox is affected. For example, if a piece of code performs a decryption mechanism, the machine could execute the code in a sandbox (like a VM), thus allowing us to analyze the code structure without actually executing the code in a dangerous environment.

In addition to polymorphic code, metamorphic code is another way to try to avoid signature-based detection. Here, each time the virus propagates, it generates a semantically different version of the code. In other words, the code performs the same high-level action, but with minor differences in execution, like changing variable names or changing the order of certain operations or using a for loop instead of a while loop. Usually included in metamorphic code is a code rewriter which changes the code randomly each time. Note that the rewriter can also change the rewriter code in addition to the virus code before propagating the virus to ensure that the entire malcode looks different.

Because the code is now changing, there is now no easy pattern to find the malcode, meaning that signature-based detection is extremely difficult. However, it does let us use behavioral-detection instead, wherein we analyze the behavior of the code instead of the syntax (since the syntax is continuously changing). As such, we now look at the effect of the instructions rather than the appearance of the instructions. However, viruses can subvert behavioral detection; for example, the virus could delay analysis by waiting a long time before executing the malicious code or it could detect that the code is being analyzed (run in a debugger or a sandbox) and could choose different, "normal" behavior.

Theoretically however, it is pretty much impossible to write a perfect algorithm to separate malicious code from safe code (though if you do manage to write something that accomplishes this task, you would have solved the halting problem!). Rather, antivirus softwares usually try to simply look for new and unfamiliar code. The software company keeps a central repository of previously-seen code and if some code has never been seen before, it treats that piece of code as malicious. Flagging unfamiliar code is a powerful defense as it employs a signature-based detection system to detect malicious behavior as well as a strategy for people avoiding the first detector. In other words, if the attacker does not modify the code for each propagation, it will have a detectable signature and if the attacker modifies the code each time, it always appears as new, and therefore suspicious.

38.4. Worms

Worms are pieces of code that, unlike viruses, do not require user action to propagate; instead, they usually infect a computer by altering some already-running code. Since worms want to run immediately, they usually randomly choose machines by randomly generating 32-bit IP addresses and try connecting to them in an attempt to propagate. Essentially, worms want to directly inject malcode into a lot of different computers very quickly. To find the different computers to inject, the worm will either try to connect to random machines or will use a pre-generated "hit-list".

Worms can potentially spread extremely quickly since they parallelize the process of propagation and replication. As more computers are infected, more computers are available to spread the worm further. While viruses have the same property, they usually spread more slowly since user action is needed to activate the virus. As such, worm propagation can be modeled as an infectious epidemic and computer scientists often use the same models that biologists use to model their spread of infectious diseases. Similar to epidemics, the spread of the worm depends on the size of the population, the proportion of the population that is vulnerable to the infection, the number of infected hosts, and the contact rate, or how often the infected host communicates with other hosts. The number of infected hosts grows logistically, meaning that the initial growth is exponential, since as more hosts are infected, there are more opportunities to infect, but later growth slows down as it becomes harder to find new non-infected hosts to infect.