Vulnerabilities and exploits

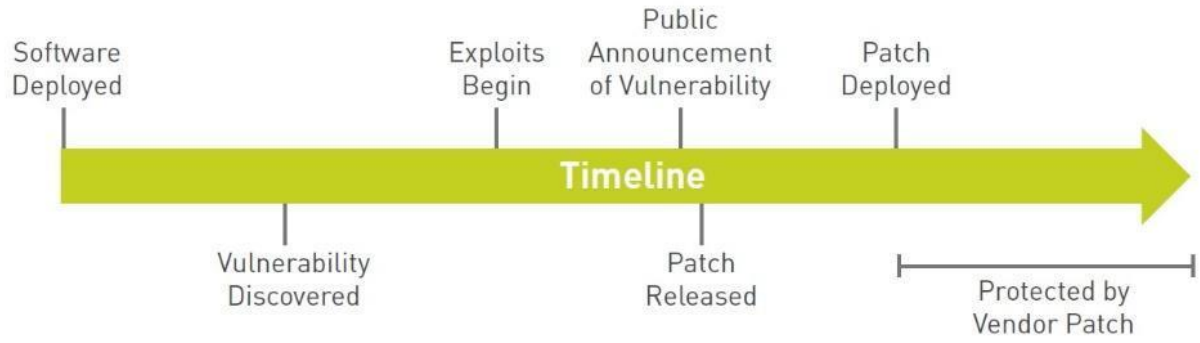
An exploit is a type of malware that takes advantage of a vulnerability in installed endpoint or server software such as a web browser, Adobe Flash, Java, or Microsoft Office. An attacker crafts an exploit that targets a software vulnerability, causing the software to perform functions or execute code on behalf of the attacker.

Vulnerabilities are routinely discovered in software at an alarming rate. Vulnerabilities may exist in software when the software is initially developed and released, or vulnerabilities may be inadvertently created, or even reintroduced, when subsequent version updates or security patches are installed. According to research by Palo Alto Networks, 78 percent of exploits take advantage of vulnerabilities that are less than two years old.

Security patches are developed by software vendors as quickly as possible after a vulnerability has been discovered in their software. However, an attacker may learn of a vulnerability and begin exploiting it before the software vendor is aware of the vulnerability or has an opportunity to develop a patch. This delay between the discovery of a vulnerability and development and release of a patch is known as a zero-day threat (or exploit). It may be months or years before a vulnerability is announced publicly. After a security patch becomes available, time inevitably is required for organizations to properly test and deploy the patch on all affected systems. During this time, a system running the vulnerable software is at risk of being exploited by an attacker (see Figure 1-2).

Figure 1-2

Vulnerabilities can be exploited from the time software is deployed until it is patched.



Exploits can be embedded in seemingly innocuous data files (such as Microsoft Word documents, PDF files, and webpages), or they can target vulnerable network services. Exploits are particularly dangerous because they are often packaged in legitimate files that do not trigger anti-malware (or antivirus) software and are therefore not easily detected.

Creation of an exploit data file is a two-step process. The first step is to embed a small piece of malicious code within the data file. However, the attacker still must trick the application into running the malicious code. Thus, the second part of the exploit typically involves memory corruption techniques that allow the attacker's code to be inserted into the execution flow of the vulnerable software. After that happens, a legitimate application, such as a document viewer or web browser, will perform actions on behalf of the attacker, such as establishing communication and providing the ability to upload additional malware to the target endpoint. Because the application being exploited is a legitimate application, traditional signature-based antivirus and whitelisting software have virtually no effectiveness against these attacks.

Although there are many thousands of exploits, they all rely on a small set of core techniques that change infrequently. For example, a *heap spray* is an attempt to insert the attacker's code into multiple locations within the memory heap, hoping that one of those locations will be called by the process and executed. Some attacks may involve more steps, some may involve fewer, but typically three to five core techniques must be used to exploit an application. Regardless of the attack or its complexity, for the attack to be successful the attacker must execute a series of these core exploit techniques in sequence, like navigating a maze to reach its objective (see Figure 1-3).

Key Terms

Heap spray is a technique used to facilitate arbitrary code execution by injecting a certain sequence of bytes into the memory of a target process.

Figure 1-3

Exploits rely on a series of core attack techniques to succeed.