# Bypassing Antivirus Detection

## On-Disk Evasion :

* Various techniques to obfuscate files stored on a physical disk are

### Packers :

* Like UPX, Doesn’t work with modern antivirus.

### Obfuscators :

* reorganize and mutate code in a way that makes it more difficult to reverse-engineer.
* This includes replacing instructions with semantically equivalent ones, inserting irrelevant instructions or “dead code”, splitting or reordering functions, and so on.
* this technique is also marginally effective against signature-based AV detection.

### Crypters :

* “Crypter” software cryptographically alters executable code, adding a decrypting stub that restores the original code upon execution.
* This decryption happens in-memory, leaving only the encrypted code on-disk.

### Software Protectors :

* Highly effective antivirus evasion requires a combination of all of the previous techniques in addition to other advanced ones, including anti-reversing, anti-debugging, virtual machine emulation detection, and so on.
* In most cases, software protectors were designed for legitimate purposes but can also be used to bypass AV detection.
* Among commercially available tools, **The Enigma Protector** in particular can successfully be used to bypass antivirus products

## In-Memory Evasion:

### Remote Process Memory Injection:

* This technique attempts to inject the payload into another valid PE that is not malicious
* The most common method of doing this is by leveraging a set of Windows APIs
* First, we would use the OpenProcess function to obtain a valid HANDLE to a target process that we have permissions to access.
* After obtaining the HANDLE, we would allocate memory in the context of that process by calling a Windows API such as VirtualAllocEx.
* Once the memory has been allocated in the remote process, we would copy the malicious payload to the newly allocated memory using WriteProcessMemory.
* After the payload has been successfully copied, it is usually executed in memory in a separate thread using the CreateRemoteThread API.

### Reflective DLL Injection :

* Unlike regular DLL injection, which implies loading a malicious DLL from disk using the LoadLibrary API, this technique attempts to load a DLL stored by the attacker in the process memory.

### Process Hollowing :

* Dss

### Inline hooking :