Introduction To Malware Development

What is Malware?

Malware is a type of software specifically designed to perform malicious actions such as gaining unauthorized access to a machine or stealing sensitive data from a machine. The term "malware" is often associated with illegal or criminal conduct but it can also be used by ethical hackers such as penetration testers and red teamers for an authorized security assessment of an organization.

MalDev Academy assumes that users enrolled in this course will use the knowledge learned for ethical and legal purposes only. Any other uses can result in criminal charges and MalDev Academy will not be responsible for this.

Why Learn Malware Development?

There are several reasons why someone would want to learn malware development. From an offensive security perspective, testers will often need to perform certain malicious tasks against a client's environment. Testers generally have three main options when it comes to the types of tools used in an engagement:

1. Open-Source Tools (OSTs) - These tools are generally signatured by security vendors and detected in any decently protected or mature organization. They are not always reliable when engaging in an offensive security assessment.

2. Purchasing Tools - Teams with larger budgets will often opt to purchase tools in order to save valuable time during engagements. Similar to custom tools, these are generally closed-source and have a better chance of evading security solutions.

3. Developing Custom Tools - Because these tools are custom-built, they have not been analyzed or signatured by security vendors which gives the attacking team an advantage when it comes to detection. This is where malware development knowledge becomes paramount for a more successful offensive security assessment.

What Programming Language Should Be Used?

Technically speaking any programming language can be used to build malware such as Python, PowerShell, C#, C, C++ and Go. With that being said, there are a few reasons that some programming languages prevail over others when it comes to malware development and it usually boils down to the following points:

- Certain programming languages are more difficult to reverse engineer. It should always be a part of the attacker's goal to ensure defenders have limited understanding as to how the malware behaves

- Some programming languages require prerequisites on the target system. For example, executing a Python script requires an interpreter present on the target machine. Without the Python interpreter present on the machine, it is impossible to execute Python-based malware.
- Depending on the programming language the generated file size will differ.

High-level vs Low-level Programming Languages

Programming languages can be classified into two different groups, high-level and low-level.

- High-level - Generally more abstracted from the operating system, less efficient with memory and provides the developer with less overall control due to the abstraction of several complex functions. An example of a high-level programming language is Python.
- Low-Level - Provides a way to interact with the operating system at an intimate level and provides the developer more freedom when interacting with the system. An example of a low-level programming language is C.

Given the previous explanations, it should become clear why low-level programming languages have been the preferred choice in malware development, especially when targeting Windows machines.

Windows Malware Development

The Windows malware development scene has shifted within the past few years and is now highly focused on evading host-based security solutions such as Antivirus (AV) and Endpoint Detection and Response (EDR). With the advancement in technology, it is no longer sufficient to build malware that executes suspicious commands or performs "malware-like" actions.

MalDev Academy will teach you to build evasive malware that can be used in real engagements. The modules will also call out non-opsec actions or actions that will likely have your malware detected by security solutions or blue teams.

Malware Development Life Cycle

Fundamentally, malware is a piece of software designed to perform certain actions. Successful software implementations require a process that's known as the Software Development Life Cycle (SDLC). Similarly, a well-built and complex malware will require a tailored version of the SDLC referred to as the Malware Development Life Cycle (MDLC).

Although the MDLC is not necessarily a formalized process, it is used in MalDev Academy to give the readers an easy way to understand the development process. The MDLC consists of 5 main stages:

1. Development - Begin the development or refinement of functionality within the malware.
2. Testing - Perform tests to uncover hidden bugs within the so-far developed code.
3. Offline AV/EDR Testing - Run the developed malware against as many security products as possible. It's important that the testing is conducted offline to ensure no samples are sent to the

security vendors. Using Microsoft Defender, this is achieved by disabling the automated sample submissions & cloud-delivered protection option.

4. Online AV/EDR Testing - Run the developed malware against the security products with internet connectivity. Cloud engines are often key components in AVs/EDRs and therefore testing your malware against these components is crucial to gain more accurate results. Be cautious as this step may result in samples being sent to the security solution's cloud engine.

5. IoC (Indicators of Compromise) Analysis - In this stage, you become the threat hunter or malware analyst. Analyze the malware and pull out IoCs that can potentially be used to detect or signature the malware.

6. Return to step 1.