Understanding EDR and Antivirus: How Modern Endpoint Security Works

By Reda Ouzidane

Introduction

This document provides a concise, yet technical overview of how traditional Antivirus (AV) and modern Endpoint Detection and Response (EDR) systems operate, particularly from a red team or cybersecurity researcher's perspective.

1 Antivirus (AV)

Antivirus is a traditional tool designed to detect and stop known types of malware such as viruses, trojans, and worms. It uses signature-based detection, heuristic scanning, and sometimes basic behavioral analysis.

How It Works

- **Signature Scan:** Compares file hashes or byte patterns to a database of known malware.
- **Heuristics:** Detects suspicious patterns in code, like encryption routines or file manipulation loops.
- **Real-time Monitoring:** Observes system activity for known malicious patterns.
- **Sandboxing:** Executes unknown files in isolated environments to analyze their behavior.

2 Endpoint Detection and Response (EDR)

EDR systems go beyond Antivirus. They provide detection, telemetry, and incident response capabilities to defend against modern threats, including fileless malware and advanced persistent threats (APTs).

Key Capabilities

- Visibility into process creation, file changes, registry edits, network activity, and memory injections.
- Behavioral detection using pattern recognition and machine learning.
- Response capabilities like process termination, endpoint isolation, or memory dump collection.

How EDR Works

- **Sensor/Agent:** Runs on each endpoint to capture telemetry.
- **API Hooking:** Hooks Windows API calls like `CreateRemoteThread`, `WriteProcessMemory`, etc.
- **Kernel Monitoring:** Some EDRs use kernel drivers to monitor syscalls and detect rootkits or stealthy malware.
- **Cloud Analytics:** Sends suspicious activity to a cloud backend for deeper analysis using AI/ML.
- **Security Console:** Allows analysts to investigate, correlate, and respond to threats.

3 Real-World Example

Malware Tries to Inject Into lsass.exe

1. EDR hooks `OpenProcess()` and flags the attempt to access `lsass.exe`.
2. Behavioral analysis determines it's suspicious or malicious.
3. An alert is generated and optionally the action is blocked.
4. Incident responders investigate and isolate the affected machine.

4 Comparison: AV vs. EDR

Key Differences

Feature	Antivirus (AV)	EDR
Detection Method	Signature & Heuristics	Behavioral + AI + Syscalls
Response	Quarantine, Block	Investigate, Isolate, Remediate
Visibility	Low to Medium	High (processes, memory, network)
Prevention	Basic	Advanced (fileless attacks, LOLBins)
Logging	Minimal	Full forensic logging

5 EDR Evasion Techniques

Red teamers and malware developers use several methods to bypass AV and EDR systems:

- **Obfuscation:** Hide recognizable patterns to evade signature detection.
- **Packing/Encryption:** Encode payloads to avoid static analysis.
- **Living-Off-The-Land (LOLBins):** Use trusted Windows binaries like `powershell.exe`, `regsvr32.exe`.
- **DLL Unhooking:** Remove EDR hooks from userland DLLs like `ntdll.dll`.
- **Direct Syscalls:** Avoid API hooks by making system calls directly.
- **Manual Mapping:** Load DLLs without using `LoadLibrary`.

6 Testing EDR Defenses

- Use open-source tools like `Sysmon` + `ELK stack`.
- Simulate attacks with Atomic Red Team or `Invoke-AtomicRedTeam`.
- Experiment in labs using tools like `Velociraptor` or `Elastic EDR`.

Native API and API32

What is Native API?

The Native API refers to the underlying set of system calls that allow applications to interact directly with the Windows kernel. These system calls are low-level operations that are used to perform tasks like memory management, process control, and file I/O operations. Native APIs are typically used by system-level components and can be accessed through the `ntdll.dll` library, which acts as a bridge between user-mode applications and the kernel.

What is API32?

API32 is not a formal term used in Windows programming, but it could be a reference to the set of 32-bit application programming interfaces (APIs) available on Windows operating systems. These APIs allow programs to interface with system-level components like the file system, network, and hardware. It's important to distinguish between the general Windows API (which includes functions like those in `kernel32.dll`) and the Native API (which operates at a lower level in the kernel).

7 What Are Hooks?

Hooks are a method used by EDR solutions, malware, and other software to monitor or manipulate specific functions or system behavior at the API level. Hooks are typically implemented by modifying the function pointers or replacing the addresses of certain APIs with the address of custom code. This allows the hooked function to execute additional behavior, such as logging, modifying data, or blocking malicious actions.

How Hooks Are Used in AV and EDR

- **AV and EDR Solutions**: AV and EDR products often use hooks to intercept system calls related to file access, process creation, and network activity. By hooking these functions, the EDR can inspect the behavior of processes and determine whether they are suspicious or malicious. For example, the EDR might hook `CreateProcess()` to prevent the launch of malicious executables. - **Malware**: Malware can also use hooks to hide its activities from security solutions. By hooking system APIs like `CreateRemoteThread()`, malware can prevent detection while performing malicious actions like injecting code into another process.

Conclusion

Understanding how AV and EDR solutions work is critical for both defenders and attackers. By studying detection methods and telemetry, security professionals can enhance both offensive tactics and defensive monitoring strategies.

Written by Reda Ouzidane