AV/EDR EVASION

WHO AM I

rasulbaharvandi

Red Teamer @ Golrang Industrial Group

Background

- Red Team
- Blue Team

Likes

- Sigint
- Reaserch in Windows

AGENDA

- What are Antivirus and EDR?
- History of Malware Detection
- How These Security Mechanisms Work
- How to Bypass These Mechanisms

WHAT ARE ANTIVIRUS & EDR

- **Antivirus (AV):** Software designed to detect, prevent, and remove malware from computers. It typically uses signature-based detection.

- **Endpoint Detection and Response (EDR):** Advanced security solutions that monitor endpoints (devices like laptops, servers, or desktops) to detect suspicious activities, investigate incidents, and respond to threats in real time.

EVOLUTION

- The Beginnings: Signature-Based Detection (1980s–1990s)
- Heuristic Analysis (1990s–2000s)
- Behavior-Based Detection (2000s–2010s)
- Cloud-Based Detection (2010s)
- Machine Learning & AI-Powered Detection (2015–Present)
- Endpoint Detection and Response (EDR) (2015–Present)
- Hybrid Detection (2020–Present)

CORE COMPONENTS

- AV systems rely on multiple layers of integration with the operating system:
- **User-Mode Components**:
  - Scanning engines, user interfaces, and heuristics.
  - Limited access to system internals (requires elevated privileges for deeper inspection).
- **Kernel-Mode Components**:
  - Drivers and kernel hooks.
  - Operate with high privileges, enabling AV software to monitor and control system processes at a low level.

HOOKING

- User-Mode Hooking
- Kernel-Mode Hooking

USER-MODE HOOKING

- API Hooking :
  - The AV replaces the address of key APIs (e.g., CreateFile, WriteProcessMemory) in the Import Address Table (IAT) of processes.
  - When malware calls these APIs, the AV's code is executed first.

- Inline Hooking :
  - The AV modifies the prologue of a function to redirect execution to its own monitoring code.
  - When malware calls these APIs, the AV's code is executed first.

KERNEL-MODE HOOKING

- **System Service Descriptor Table (SSDT) Hooking**
  - SSDT contains addresses of kernel functions. The AV modifies these entries to point to its driver functions.
  - Example: Redirecting NtCreateFile to inspect file creation events.
- **Interrupt Descriptor Table (IDT) Hooking**
  - Used to monitor interrupts (e.g., system calls via int 0x2E on older Windows systems).
- **Kernel Callbacks**
  - Modern AVs register callbacks (e.g., PsSetCreateProcessNotifyRoutine) instead of modifying critical structures.

KERNEL CALLBACK TABLES OVERVIEW

- Windows provides mechanisms for drivers to register callbacks for certain kernel-level events. These callbacks allow monitoring tools (like EDRs) to track key system activities.
- Examples of Kernel Callbacks:

- Process Creation Callbacks (PsSetCreateProcessNotifyRoutine):
- Invoked when a new process is created or terminated.
- Thread Creation Callbacks (PsSetCreateThreadNotifyRoutine):
- Triggered when a thread is created or terminated.
- Image Load Callbacks (PsSetLoadImageNotifyRoutine):
- Called when a module (e.g., DLL or EXE) is loaded into memory.
- Registry Callbacks (CmRegisterCallback):
- Used to monitor registry operations.

- EDRs leverage these callbacks to collect telemetry on system activities and detect malicious behavior.

DRIVER ARCHITECTURE

- **File System Filtering**
- **Process Monitoring**
- **Memory Scanning**
- **Registry Monitoring**
- **Networking Hooks**

REAL-TIME MONITORING MECHANISMS

- **Kernel Callbacks**
  - AVs use callback routines to monitor events like process creation (PsSetCreateProcessNotifyRoutine), thread creation, or file operations.
- **Inline Patch Guard Bypasses**
  - Some AVs bypass PatchGuard restrictions by dynamically generating and injecting code into kernel-mode threads.
- **Sandboxing**
  - Drivers isolate unknown programs in a controlled environment to observe behavior without impacting the host system.

KERNEL PATCH PROTECTION (PATCHGUARD)

- PatchGuard periodically checks the integrity of critical kernel structures, including:

- System Service Descriptor Table (SSDT).
- Interrupt Descriptor Table (IDT).
- Global Descriptor Table (GDT).
- Kernel code sections.

- If a violation is detected, PatchGuard triggers a system crash (BSOD) to prevent further exploitation.

EVADING

- Obfuscation Techniques
- Packing and Encryption
- Living Off the Land (LOTL) Techniques
- Process Injection
- Anti-Analysis Techniques
- Code Injection and Reflection
- API Hooking Evasion
- Network Evasion
- Memory-Based Techniques

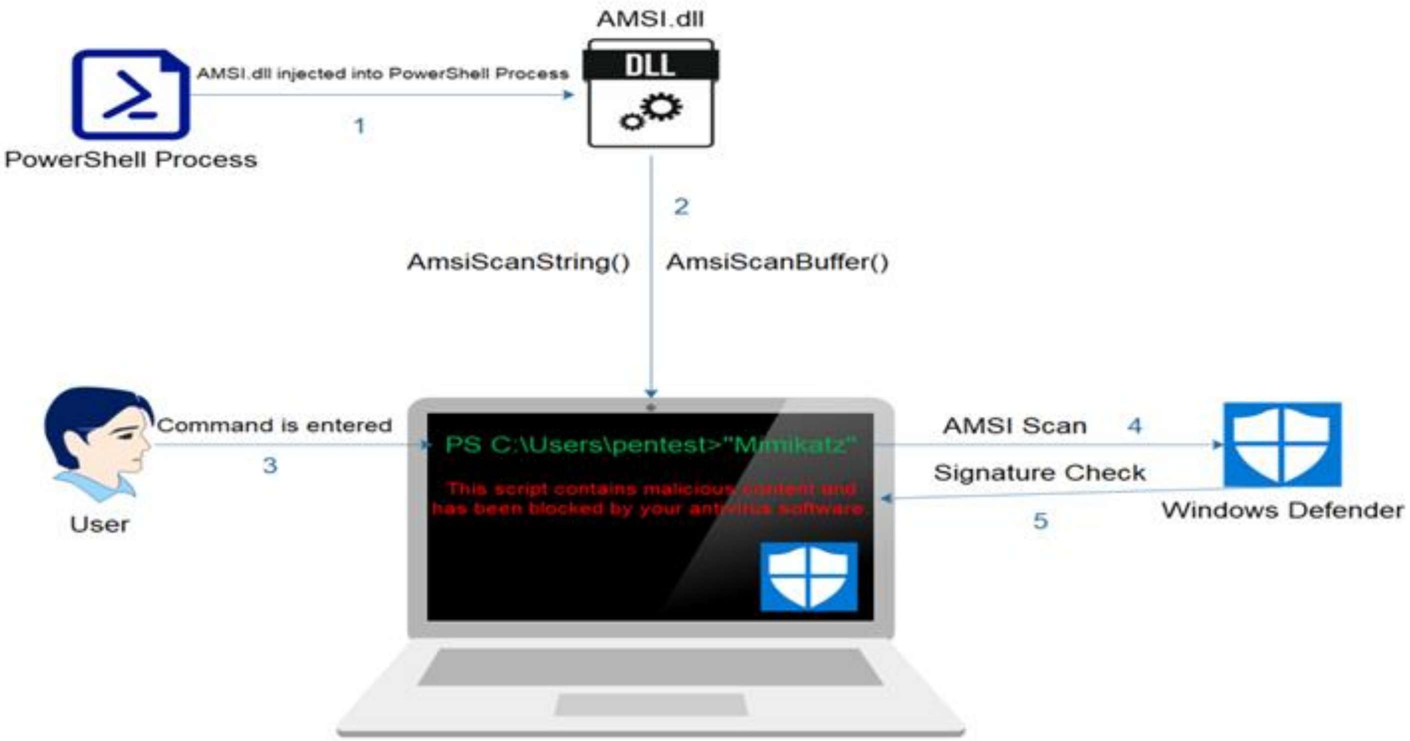
OBFUSCATION TECHNIQUES

- String Obfuscation
  - Rotr32
  - Stack Strings
- AES
- IAT Hiding

LIVING OFF THE LAND

- **Substring Execution**
- Ordinal Number
- Token Execution

AMSI (ANTIMALWARE SCAN INTERFACE)



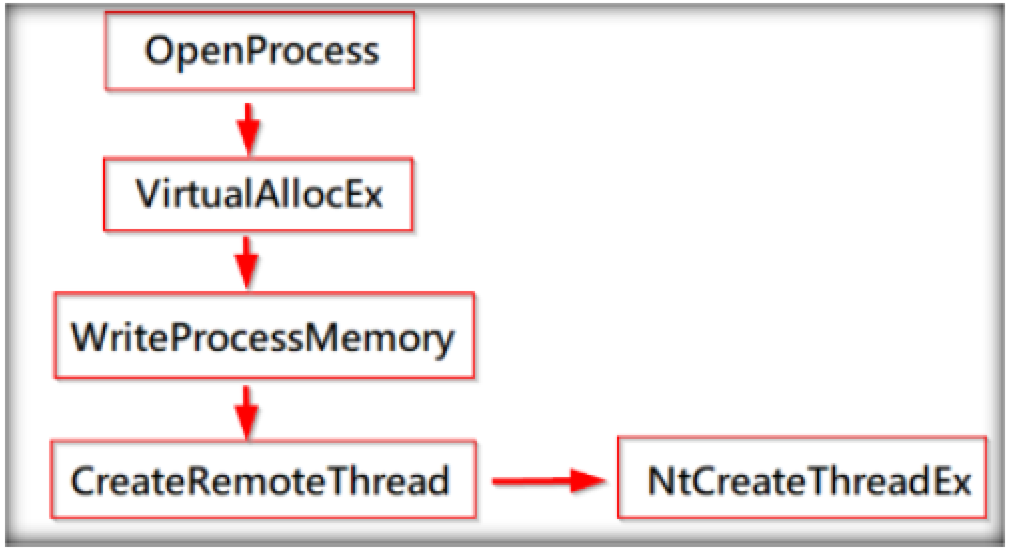
```
PS C:\Users\pentest>"Mimikatz"
This script contains malicious content and
has been blocked by your antivirus software.
```

PROTECTED PROCESS LIGHT (PPL) SIGNER LEVELS

- PPL uses a hierarchy of trust levels, with each level granting specific privileges:
  - WinTcb (Windows Trusted Computing Base):
    - The highest level of trust.
    - Reserved for the core operating system and highly sensitive processes.
    - Example: lsass.exe (Local Security Authority Subsystem Service).
  - Windows:
    - Processes that are signed by Microsoft and are part of the OS.
    - Example: winlogon.exe.
  - Windows Antimalware:
    - Designed for antivirus and EDR processes.
    - Example: Processes related to Microsoft Defender and third-party AV/EDRs.
  - App:
    - Protects applications with specific signing requirements.
    - Example: Media-related processes with DRM.

PROCESS INJECTION

Classic Process Injection
Create Remote Thread
Fiber Injection
APC Injection

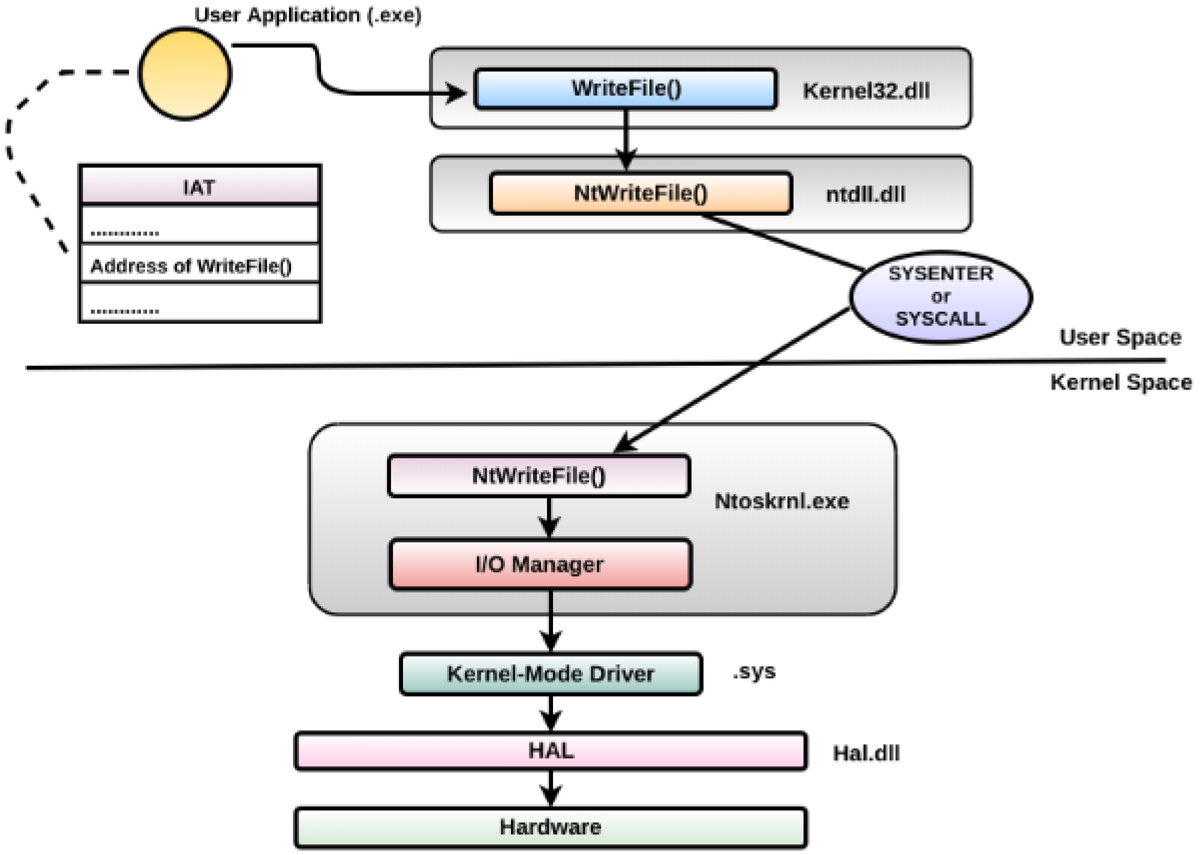

OpenProcess
VirtualAllocEx
WriteProcessMemory
CreateRemoteThread
NtCreateThreadEx

ANTI-ANALYSIS TECHNIQUES

- **Detecting Debuggers**
- **Detecting Debugger Via NtQueryInformationProcess**
- **BlackListed**
- **Breakpoint Detection Via GetTickCount64**
- **Detecting Delays**
- **Previously Mounted USBs Check**

API HOOKING EVASION

- Direct syscall
- In-direct syscall


User Application (.exe)
WriteFile()
Kernel32.dll
IAT
....
Address of WriteFile()
....
NtWriteFile()
ntdll.dll
SYSENTER
or
SYSCALL
User Space
Kernel Space
NtWriteFile()
Ntoskrnl.exe
I/O Manager
Kernel-Mode Driver
.sys
HAL
Hal.dll
Hardware