Position COMpromised

Weaponizing Windows COM for Stealth Execution

(Part of Altered Security Hacker Summer 2026)

About me

- Security Researcher and Trainer
- LinkedIn - linkedin.com/in/manthan-chhabra/
- Developer of multiple tools used in the course
- Interested in on-prem red team and evasion
- Previous Trainings
  - DEF CON, BlackHat, AltSecCON, BruCON and more.

Altered Security

- Trained more than 50000 security professionals from more than 130 countries!
- Our Red Team Labs Platform enables labs to be:
  - Affordable
  - Easy to Access
  - Stable and provide great user experience
  - Fun to Solve
  - Big enough to feel enterprise-like

Red team labs	alteredsecurity.com/online-labs
Instructor-led bootcamps	alteredsecurity.com/bootcamps
GitHub	github.com/AlteredSecurity
Lab Platform	enterprisesecurity.io
Free Labs and Challenges	redlabs.enterprisesecurity.io

What is Hacker Summer

- Altered Security's month-long Red Teaming event.
  - Research blogs and webinars on Red Teaming for on-prem, Azure and AI.
  - Call for papers.
  - Discounts and giveaways.

Webinar Ground Rules

- You can ask questions on general channel (#general) on our Discord server (https://discord.com/invite/vcEwaRMwJe). We won't be able to monitor Zoom Chat.
- To ensure a smooth experience, participants are not allowed to use their microphone.
- Please maintain professional conduct and a respectful atmosphere throughout the webinar.

Agenda

- COM fundamentals
- Architecture
- Activation
- Enumeration
- Abuse techniques
- Detection
- Further studies

What is COM?

- Component Object Model (COM) is Microsoft's binary interoperability standard for reusable software components.

- Before COM, sharing compiled components across programming boundaries resulted in issues such as strict compiler dependencies and "DLL Hell."

Why COM Exists

- **Language Independence:** Connect C++, C#, VB, PowerShell, or script clients seamlessly.
- **Component Reusability:** Use or upgrade components without rebuilding dependent clients.
- **Transparent IPC:** Abstract the location of binaries (In-Proc DLLs or Out-of-Proc EXEs).

Key Terminology

- **Interface:** A defined group of related virtual functions that a COM class implements.
- **Object:** An instantiation of a COM class.
- **CLSID:** 128-bit GUID uniquely identifying a COM class.
- **IID:** Unique interface identifier.
- **ProgID:** String counterpart linked to a CLSID (e.g., WScript.Shell).
- **Apartment:** Logical execution container determining concurrency, thread safety, and security contexts.

Interfaces and VTables

- COM objects expose functionality through interfaces identified by IIDs.
- At a binary level, a COM interface is structured as a pointer to a table of function pointers, known as a Virtual Function Table (VTable).
- The client application traverses its local interface pointer, locates the target object's vtable address, and executes the exact function compiled inside the server binary to call a method.

IUnknown

- Every COM interface in Windows must inherit from IUnknown, permanently reserving the first three slots of every VTable for its core methods:
  - **QueryInterface:** Asks the target object dynamically at runtime if it supports a specific IID.
  - **AddRef:** Increments the reference counter when an interface is referenced.
  - **Release:** Decrements the reference counter. When the count hits 0, the object frees itself from memory.

Registry Structure

Registry Path (HKCR \ ...)	Associated Key Context	Core Operational Value
\CLSID\{GUID}	Class Identifier Map	The mapping database. Connects a CLSID to binary paths and programmatic rules.
\Interface\{GUID}	Interface Database	Maps interface signatures (IIDs) directly to their respective proxy/stub marshaling binaries.
\AppID\{GUID}	Application ID Configuration	Governs security configurations (launch permissions, access control, impersonation) for Out-of-Proc EXEs.
\TypeLib\{GUID}	Type Library Metadata	Hosts localized binary descriptions of classes and functions, allowing dynamic bindings.

CLSID Registration Keys

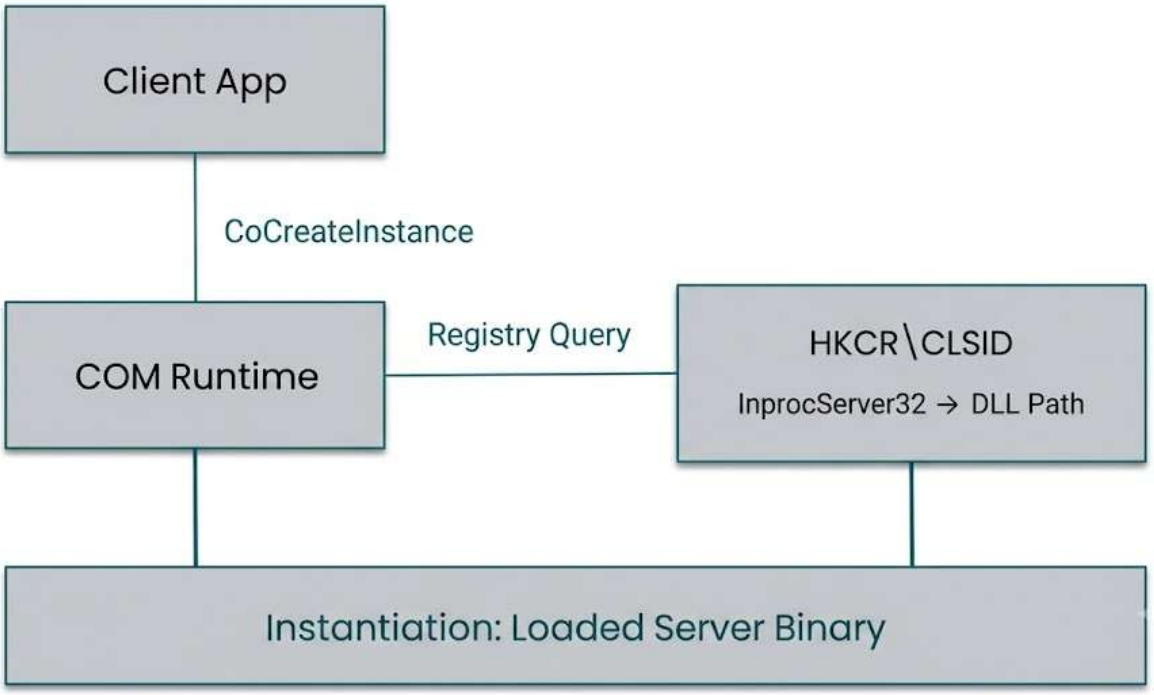
- **InprocServer32:** Identifies an in-process dynamic-link library (DLL). Maps directly to the local absolute file path.
- **LocalServer32:** Identifies an out-of-process executable file (EXE). Spawns separate processes when called.
- **TreatAs:** Contains another target CLSID GUID. Forces COM runtime to emulate another class, useful for backward compatibility and attacker redirection.
- **ProgID/VersionIndependentProgID**: Contains the human-readable string that maps backward to the raw CLSID GUID.

COM Activation Flow

- When a client requests a COM object, it never instantiates the component binary directly. It talks strictly to the COM Runtime:

1. The client executes initialization and calls CoCreateInstance(CLSID).
2. COM Runtime maps the CLSID back to a registry configuration hive.
3. The system identifies the target binary implementation (DLL or EXE).
4. The server binary is loaded into the appropriate apartment space.

COM Activation Flow



In-Process vs. Out-of-Process

- **In-process COM:** The COM server is implemented as a DLL and loaded directly into the client application's memory space. Fast but lacks application sandboxing isolation.

- **Out-of-process COM:** The COM server runs as a separate EXE. To access this object, requests cross process boundaries via RPC. This is more resilient but slower due to Marshaling.

COM Apartments

- Logical boundaries containing COM objects.
- Three types of apartments:
  - **Single-Threaded Apartment (STA):** One dedicated thread executes object calls. Inbound calls are serialized via Windows message queues.
  - **Multi-Threaded Apartment (MTA):** Multiple threads can access objects concurrently. The developer is fully responsible for concurrency locks.
  - **Neutral Apartment (NA):** Execution occurs on the caller's thread, eliminating thread context-switching overhead.

Why Apartments Matter

- **Thread Safety:** Apartments isolate objects, preventing multi-threaded apps from causing memory corruption in single-threaded components.
- **Synchronization:** The OS manages concurrency automatically based on apartment type for sequential access or enabling parallel execution.
- **Marshaling:** COM blocks direct cross-apartment memory pointers, forcing data serialization and transport through Proxies and Stubs.

COM Security

- **Authentication:** Integrated with the Windows Security Support Provider Interface (SSPI) to validate clients before connections are established.
- **Authorization:** Enforced access checks against explicit ACLs defined in the registry.
- **Impersonation:** Allows servers to impersonate the client's token identity and safely execute methods using the client’s security context.

Impersonation Levels

- **Anonymous:** The client is invisible; the server cannot identify the caller and uses its own default or guest context.
- **Identify:** The server can inspect the client's identity for access checks but cannot run code or access resources using their token.
- **Impersonate:** The server can execute tasks and access local resources using the client's token but cannot pass credentials over the network.
- **Delegate:** The server can clone and pass the client's credentials across network boundaries to access remote secondary servers.

Enumerating COM Objects

- **Registry:** Directly queries keys like HKCR\CLSID
- **WMI:** Query the "Win32_COMClass" objects.
- **OleView.NET**


IT'S
DEMO TIME!

Why Attackers abuse COM

- Security agents (EDRs, AV) trust operating system binaries. COM triggers command execution through trusted proxy processes.
- Out-of-Proc instantiation forces svchost.exe, rundll32.exe or dllhost.exe to spawn the target application.
- Passing execution arguments via in-memory COM interface methods leaves process-creation command logs clean.

Execution via COM

- WScript.Shell
- Shell.Application
- Office automation
- MMC20.Application
- ShellWindows
- And many, many more..

..again

Auto-Elevated COM Objects

- Windows grants full administrative access to pre-approved COM objects without prompting the user.
- Attackers use the elevation moniker to instantiate trusted COM classes straight from a low or medium-integrity context.

Elevation:Administrator!new:{CLSID}

ICMLuaUtil UAC Bypass

- Historical COM-based UAC bypass primitive.
- Targets the CMSTPLUA COM service with the GUID: {3E5FC7F9-9A51-4367-9063-A120244FBEC7}
- Elevation registry key has the value 'Enabled' set to '1'.
- Malicious binary modifies its PEB to impersonate a trusted binary.
- Invokes an undocumented interface method called ShellExec inside cmlua.dll.

COM for Persistence

- **Registry Search Order:** Placing a duplicate CLSID entry in the HKCU hive which takes precedence over HKLM.
- **InprocServer32:** Modifying the InprocServer32 key to point to a malicious DLL.
- **TreatAs:** Rerouting benign COM requests to an attacker-controlled CLSID.

Monitoring and Recommendations

- Monitor Sysmon Events:
  - Event 1: Process creation of dllhost.exe with unusual arguments.
  - Event 7: DLL load operations using dllhost.exe from writable paths.
  - Event 13: Audit registry creation and modifications targeted inside HKCU\Software\Classes\CLSID*.
- Utilize Event Tracing for Windows (ETW) by subscribing to the Microsoft-Windows-COM-Runtime provider.
- Maintain a known-good inventory of active COM registrations and alert on anomalies.

But wait, there's more..

- **DCOM:**
  - Extends COM architecture across network using RPC.
  - Abused to execute commands filelessly on remote servers.
- **Cross-Session Activation:**
  - Session-to-Session Activation with a Session Moniker.
  - Creates a COM object in a different logon session than the caller.
  - "RunAs Interactive User" object activation mode configured.
  - "SpeechRuntimeMove" by S3cur3Th1sSh1t targets the Speech Named Pipe COM and spawns SpeechRuntime.exe.

Thank you

- Please provide feedback.
- Discord (#general channel for this webinar) - https://discord.com/invite/vcEwaRMwJe
- Red Team labs: https://www.alteredsecurity.com/online-labs
- Bootcamps: https://www.alteredsecurity.com/bootcamps
- Free Azure Red Team Labs: https://redlabs.enterprisesecurity.io/