

They See Me Roaming: Following APT29 by Taking a Deeper Look at Windows Credential Roaming | Mandiant

By Mandiant

Published: 2022-11-08 · Archived: 2026-04-05 12:40:18 UTC

Written by: Thibault Van Geluwe de Berlaere

In early 2022, Mandiant detected and responded to an incident where APT29 successfully phished a European diplomatic entity and ultimately abused the Windows Credential Roaming feature. The diplomatic-centric targeting is consistent with Russian strategic priorities as well as [historic APT29 targeting](#). Mandiant has been tracking APT29—a Russian espionage group that is [sponsored by the Foreign Intelligence Service \(SVR\)](#)—since at least 2014. Some APT29 activity is also publicly referred to as [Nobelium by Microsoft](#).

During the short timespan that APT29 was determined to be active inside the victim network, Mandiant observed numerous LDAP queries with atypical properties (Figure 1) performed against the Active Directory system.

```
4662 | Audit Success | An operation was performed on an object.  
Subject :  
Security ID: < redacted by Mandiant >  
Account Name: < redacted by Mandiant >  
Account Domain:  
Logon ID: 0x000000006d15eb96  
Object:  
Object Server: DS  
Object Type: %{bf967aba-0de6-11d0-a285-00aa003049e2}  
Object Name: < redacted by Mandiant >  
Handle ID: 0x0000000000000000  
Operation:  
Operation Type: Object Access  
Accesses: %%7688  
Access Mask: 0x00000100  
Properties: %%7688  
{771727b1-31b8-4cdf-ae62-4fe39fadf89e}  
{612cb747-c0e8-4f92-9221-fdd5f15b550d}  
{91e647de-d96f-4b70-9557-d63ff4f3ccd8}  
{b7ff5a38-0818-42b0-8110-d3d154c97f24}  
{bf967aba-0de6-11d0-a285-00aa003049e2}
```

Figure 1: Example of event log of LDAP query

The queried LDAP attributes relate to usual credential information gathering (e.g. `unixUserPassword`); however, one attribute in particular stood out: `{b7ff5a38-0818-42b0-8110-d3d154c97f24}` , or `msPKI-CredentialRoamingTokens` , which is [described by Microsoft](#) as ‘*storage of encrypted user credential token BLOBs for roaming*’. Upon further inspection, Mandiant identified that this attribute is part of a lesser-known feature of Active Directory: Credential Roaming.

A Deep Dive into Credential Roaming

Credential Roaming was introduced in Windows Server 2003 SP1 and is still supported on Windows 11 and Windows Server 2022. This feature was created to allow certificates (and other credentials) to ‘roam’ with the user.

For example: Consider a scenario where a corporation uses autoenrollment to automatically provision certificates for employees for the purpose of Secure/Multipurpose Internet Mail Extension (S/MIME) encryption. When user Alice logs on to device A, the autoenrollment process launches and she is enrolled into the corresponding certificate template. However, should Alice now log in to device B, she would receive a new certificate (because the certificate is device-local). Credential Roaming ensures that Alice’s first S/MIME certificate (from device A), including the private key, is saved to device B before the autoenrollment process kicks in. With Credential Roaming, Alice would only enroll one certificate, thereby removing the need for duplicate certificates and reducing the certificate management overhead.

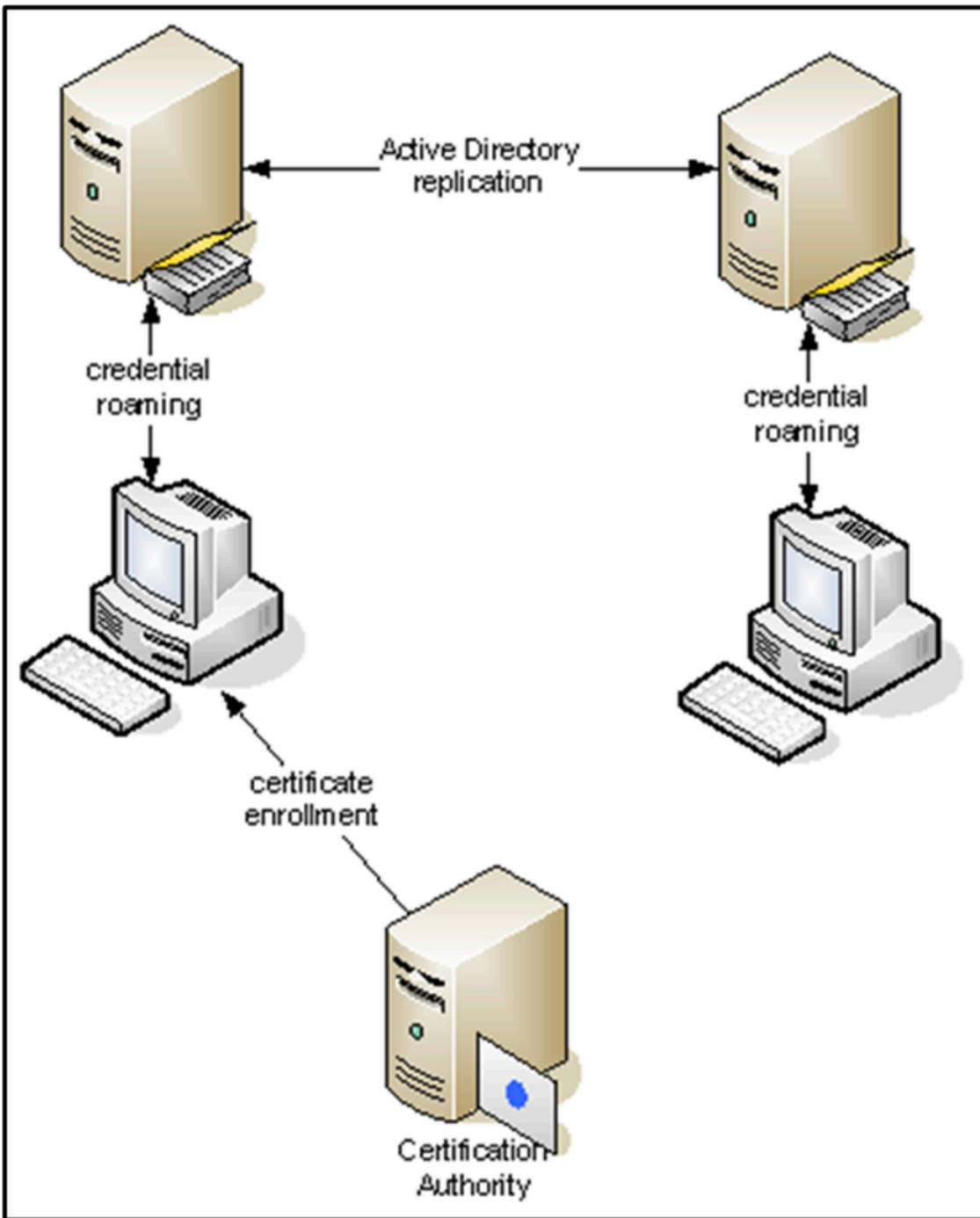


Figure 2: Credential Roaming diagram ([source](#))

Any kind of certificates, including certificates from external sources (such as public PKI vendors), are supported (with the exclusion of certificates where the private key is stored in hardware [e.g. TPM]). More examples and details are available in a [Microsoft whitepaper on Credential Roaming, published in 2012](#).

Windows Vista [extended the credential roaming functionality](#) so that usernames and passwords stored in the Windows Credential Manager can also be roamed between computers. This functionality [was removed in Windows 7](#), presumably due to security precautions.

Credential Roaming was touched upon briefly by Michael Grafnetter in his blog post [Extracting Roamed Private Keys from Active Directory](#), where Mr. Grafnetter explains how his DSInternals toolkit can be used to extract the roamed credentials from Active Directory and how the popular Mimikatz tool can be used to decrypt the DPAPI secrets with the DPAPI Domain Backup Key.

Credential Roaming synchronizes certificates and credentials (called ‘Roaming Tokens’) by using the user’s Active Directory account as a datastore. The 2012 Microsoft whitepaper identifies the following LDAP properties are used in Credential Roaming:

- msPKI-CredentialRoamingTokens
- msPKIRoamingTimeStamp
- msPKIDPAPIMasterKeys
- msPKIAccountCredentials

These attributes form the [Private-Information](#) property set. The last attribute, msPKIAccountCredentials, is where the Roaming Tokens are stored. The msPKIRoamingTimeStamp attribute contains the last update time of msPKIAccountCredentials, and msPKIDPAPIMasterKeys contains the user’s Data Protection API (DPAPI) Master Keys.

Credential Roaming is implemented using a Scheduled Task (Figure 3) at \Microsoft\Windows\CertificateServicesClient\UserTask-Roam. This scheduled task launches a Component Object Model (COM) object with CLSID {58FB76B9-AC85-4E55-AC04-427593B1D060}, corresponding with the dimsjob.dll DLL (the Credential Roaming service was [formerly named the Digital Identity Management Service \[DIMS\]](#)). The string “KEYROAMING” is passed as an argument.

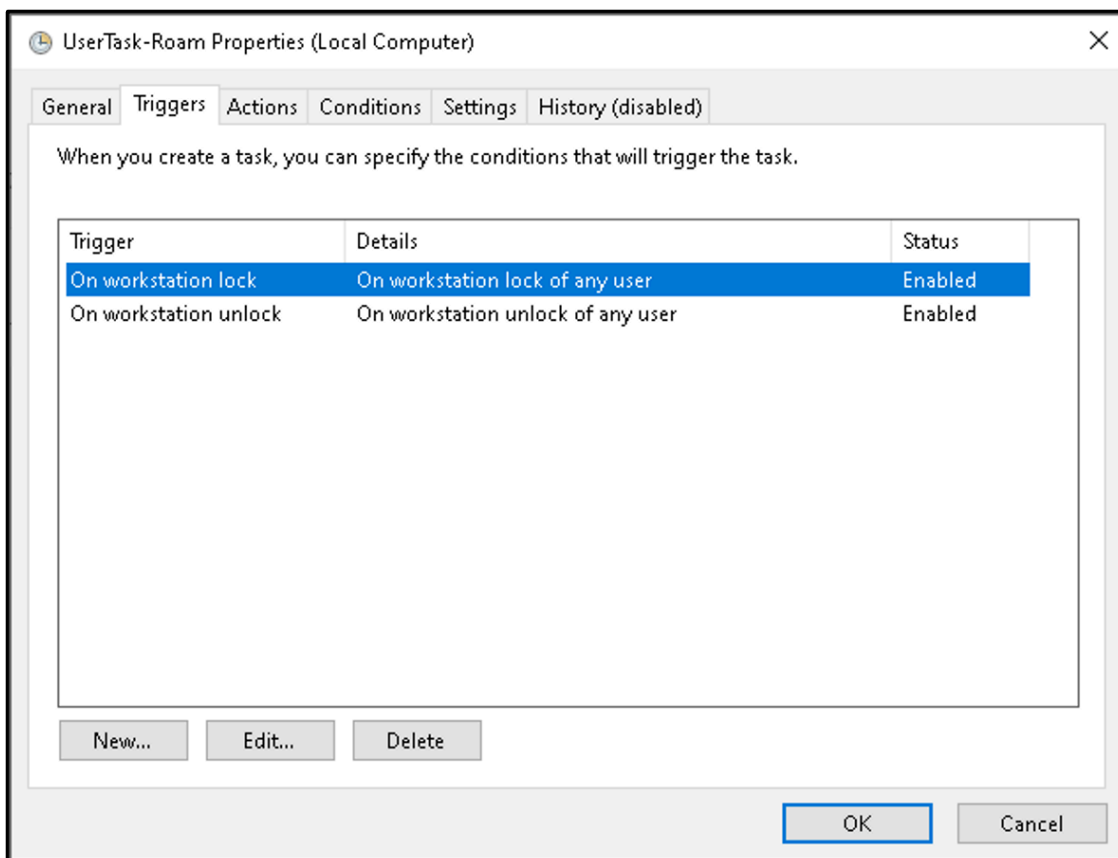


Figure 3: Scheduled Task that launches Credential Roaming

By examining the `dimsjob.dll` DLL entry point, Mandiant observed that `dimsjob.dll` loads another DLL, `dimsroam.dll`, to perform the Credential Roaming functionality (for the purposes of simplicity, the DLLs examined in this article are from Windows Server 2008 R2. Recent versions of Windows use various COM objects to handle Credential Roaming, but the same principles apply).

```

if (local_48 != '\0') {
    if ((this->cProvider).lpProcAddress == (FARPROC)NULL) {
        CProvider::Load(&this->cProvider, (unsigned_short *)L"dimsroam.dll", "DimsRoamEntry");
    }
    if (((undefined **)WPP_GLOBAL_Control != &WPP_GLOBAL_Control) &&
        ((WPP_GLOBAL_Control[0x1c] & 0x10) != 0)) {
        WPP_SF_D(*(undefined8 *) (WPP_GLOBAL_Control + 0x10), 0xc, &DAT_7ff707e13c0, uVar6);
    }
    CProvider::operator()(&this->cProvider, uVar6, (void *)0x0, 0);
}
    
```

Figure 4: Snippet of code from `dimsjob.dll!CDims::Notify` where `dimsroam.dll!DimsRoamEntry` is called

Mandiant then identified the binary structure of the entries in the `msPKIAccountCredentials` LDAP attribute (Figure 5).

ffset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	25	35	5C	31	41	42	44	30	42	33	38	45	41	36	37	32	§5\IABD0B38EA672
00000010	35	37	32	41	33	42	32	39	34	42	42	31	38	41	35	38	§72A3B294BB18A58
00000020	46	32	42	00	00	00	00	00	00	00	00	00	00	00	00	00	F2B.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	4C	4E	7A	85	3C	19	D8	01	00	00	00	00	DB	A5	67	E1	LNz<.0.1.1.Uygá
00000070	07	7F	2F	1E	23	A1	E9	69	A8	EC	95	0A	FF	69	C0	DA	...#;éi'i+.yiãÜ
00000080	76	01	00	00	01	00	00	00	6A	01	00	00	00	00	00	00	v...j.....
00000090	01	00	00	00	D0	8C	9D	DF	01	15	D1	11	8C	7A	00	C0	...B.E.B..N.Gz.À
000000A0	4F	C2	97	EB	01	00	00	00	23	B7	23	CC	32	77	F3	44	OÄ-é...#.#i2wöD
000000B0	94	69	46	58	74	8F	CE	6E	00	00	00	20	3A	00	00	00	"iFXt.in...:...
000000C0	45	00	6E	00	74	00	65	00	72	00	70	00	72	00	69	00	E.n.t.e.r.p.r.i.
000000D0	73	00	65	00	20	00	43	00	72	00	65	00	64	00	65	00	s.e..C.r.e.d.e.
000000E0	6E	00	74	00	69	00	61	00	6C	00	20	00	44	00	61	00	n.t.i.a.l..D.a.
000000F0	74	00	61	00	0D	00	0A	00	00	00	03	66	00	00	A8	00	t.a.....f...
00000100	00	00	10	00	00	00	AF	2C	27	80	34	2F	C2	31	63	6A, 'e4/Älcj
00000110	67	14	48	54	64	C4	00	00	00	00	04	80	00	00	A0	00	g.HTdÄ.....é...
00000120	00	00	10	00	00	00	C4	6C	7E	6C	70	6C	93	51	FB	16Äl~lpl"OÄ.
00000130	8E	AA	74	B8	95	65	A8	00	00	00	EE	B5	40	0A	6B	02	Ž*t,*e...iþ@.k.
00000140	5A	8C	2F	FE	83	AA	56	2D	75	F3	EE	13	61	23	05	78	ZÉ/pf*V-uó1.a#x
00000150	7D	81	07	71	5E	6C	23	9F	9A	33	97	1C	D1	F9	C9	8A	}..q^1#ÿs3-.ñuëŠ
00000160	61	24	AE	64	88	11	FC	53	16	35	F5	2D	10	6D	CC	E6	a§ød*.üs.5ö-.mIæ
00000170	49	82	00	E8	9C	9E	2F	66	DA	D3	0C	8D	C1	78	FA	CF	I,.ëez/fÜÖ..ÄxüI
00000180	52	30	CE	36	92	E6	17	72	7C	A1	61	99	98	17	CD	A0	ROİé'e.r ;a™.í
00000190	79	F9	F9	A7	B0	0F	D9	09	E1	17	08	E6	43	31	96	8A	ÿüü\$°.Ü.á..æCl-Š
000001A0	DD	90	AA	0A	E1	D6	0A	83	6F	12	0D	75	54	FA	65	48	Ý.*.áÖ.fo..uTúeH
000001B0	12	EC	95	6A	44	D3	4E	81	F2	19	38	F0	E4	E1	51	79	.i+jDÓN.ö.8öäáQy
000001C0	F3	2E	5A	A1	97	0C	8E	14	95	C4	AE	2A	96	A5	AA	C8	ó.Z -..Ž.*Äø*-#*E
000001D0	21	DB	91	33	C5	1E	95	8D	B9	7C	46	6A	FD	03	08	6D	!Ü'3Ä..' Fjý..m
000001E0	89	92	14	00	00	00	A1	F7	4B	E5	DE	10	B8	DC	2C	44	‰'...; =KÄF.Ü,D
000001F0	31	8C	93	C5	CD	6A	62	7B	6F	AD							læ"Äíjb{ø.

(3 bytes) Roaming Token Type
 (92 bytes) Identifier
 (8 bytes) Update Timestamp
 (2 bytes) Flags
 (2 bytes) Always 00 00
 (20 bytes) SHA1 hash of data
 (4 bytes) Size of data
 (variable size) Data

Figure 5: Binary structure of Roaming Tokens

The binary structure starts off with an indication of which type of Roaming Token this entry represents. Mandiant identified the following types:

- %0: DPAPI Master Key
- %1: CAPI Private Key (RSA)
- %2: CAPI Private Key (DSA)
- %3: CAPI Certificate

- %4: CAPI Certificate Request
- %5: Username/Password (Enterprise Credential Data)
- %6: (unknown – presumably unused)
- %7: CNG Certificate
- %8: CNG Certificate Request
- %9: CNG Private Key

Next follows the identifier of the Roaming Token. This is the filename of the corresponding file on disk. The structure continues with the last update timestamp of the Roaming Token, some NULL bytes (padding) and the SHA1 hash of the Roaming Token data. Finally, the size of the Roaming Token data is included (4 bytes integer) and the raw data of the Roaming Token data.

When `dimsroam.dll` launches, it retrieves these structures from the `msPKIAccountCredentials` LDAP attribute of the current user. For every entry, it determines if there already exists a local file that corresponds to the Roaming Token. If such a file is found, `dimsroam.dll` will compare the last file write time and the SHA1 hash and update the file if necessary. If such a local file is not found, `dimsroam.dll` identifies the correct save location for the binary data based on the type of the Roaming Token (Figure 6).

```

2 LPWSTR * GetSaveFolderLocationForFileType
3         (uint RoamingTokenType, LPWSTR pszStringSid, int *param_3, LPWSTR *OutBuffer)
4
5 {
6     short *psVar1;
7     LPWSTR *NewEndOfBuffer;
8
9     if (RoamingTokenType == 7) {
10        RoamingTokenType = 3;
11    }
12    else {
13        if (RoamingTokenType == 8) {
14            RoamingTokenType = 4;
15        }
16        if (5 < RoamingTokenType) {
17            return (LPWSTR *)0x0;
18        }
19    }
20
21        /* Determine folder based on file type
22         0 = "Microsoft\Protect\"
23         1 = "Microsoft\Crypto\RSA\"
24         2 = "Microsoft\Crypto\DSS\"
25         3 = "Microsoft\SystemCertificates\My\"
26         4 = "Microsoft\SystemCertificates\Req\"
27         5 = "Microsoft\Credentials" */
27 psVar1 = (short *)(&PTR_u_Microsoft\Protect\_7ff33be16c0)[RoamingTokenType];

```

Figure 6: Snippet from `dimsroam.dll` where the save location is determined based on the Roaming Token type (the path is prepended with the user's

To determine the final save location of the roaming token, the identifier (byte 0x03 to 0x0F) is appended to the folder path string (Figure 7).

```

25     ppWVar3 = GetSaveFolderLocationForFileType
26             (pBuffer[1] - 0x30,pszStringSid,(int *) (FilePathBuffer + 260
27             (LPWSTR *) (FilePathBuffer + (~FilePathLength - 1)));
28     if (ppWVar3 != (LPWSTR *) NULL) {
29         uVar4 = (ulonglong)((int)uVar6 + 94);
30         FilePathLength = (longlong)FilePathBuffer + (518 - (longlong)ppWVar3) >> 1;
31         if ((longlong)uVar4 < (longlong)FilePathLength) {
32             FilePathLength = uVar4;
33         }
34         uVar4 = 0;
35         if (FilePathLength != 0) {
36             do {
37                 if (pBuffer[uVar4 + 2] == 0) break;
38                 *(ushort *) ((longlong)ppWVar3 + uVar4 * 2) = (ushort)pBuffer[uVar4 + 2];
39                 uVar4 = uVar4 + 1;
40             } while (uVar4 < FilePathLength);
41         }

```

Figure 7: Identifier string is appended to folder path string

This file path is then directly passed to a `kernel32!CreateFileW` API call (Figure 8), where the Roaming Token data will be written.

```

65     uVar5 = GetSaveLocationForRoamingToken(pBuffer,FilePath);
66     DVar2 = (DWORD)uVar5;
67     pBVar10 = lpBuffer;
68     if (DVar2 == 0) {
69         /* If FileType = 0 (Masterkey), attributes becomes 0x8000006 (GENERIC_READ |
70         FILE_ATTRIBUTE_SYSTEM | FILE_ATTRIBUTE_HIDDEN), otherwise attributes becomes
71         0x108000004 (0x100000000? | GENERIC_READ | FILE_ATTRIBUTE_SYSTEM) */
72         dwFlagsAndAttributes = (-(uint)(FileType != '0') & 0xffffffff) + 0x8000006;
73         hFile = CreateFileW(FilePath,3,0,(LPSECURITY_ATTRIBUTES)0x0,OPEN_ALWAYS,dwFlagsAndAttributes,
74             (HANDLE)0x0);
75         /* If file couldn't be created because of path not found; create the path */
76         if (hFile == (HANDLE)INVALID_HANDLE_VALUE) {
77             DVar2 = GetLastError();
78             if (DVar2 == ERROR_PATH_NOT_FOUND) {
79                 uVar5 = INVALID_HANDLE_VALUE;
80                 pwVar8 = FilePath;
81                 do {
82                     if (uVar5 == 0) break;
83                     uVar5 = uVar5 - 1;
84                     wVar1 = *pwVar8;
85                     pwVar8 = pwVar8 + 1;
86                 } while (wVar1 != L'\0');
87                 DVar2 = DRR_CreateDirectory(FilePath,~uVar5 - 1);
88                 if (DVar2 == ERROR_SUCCESS) {
89                     hFile = CreateFileW(FilePath,3,0,(LPSECURITY_ATTRIBUTES)0x0,OPEN_ALWAYS,
90                         dwFlagsAndAttributes,(HANDLE)0x0);

```

Figure 8: The modified file path is passed to kernel32!CreateFileW

CVE-2022-30170: Arbitrary File Write turns Remote Code Execution

The aforementioned behavior introduces an Arbitrary File Write vulnerability: the file path is not properly sanitized and may contain directory traversal (“..”) characters. If an attacker can control the

`msPKIAccountCredentials` LDAP attribute, they may add a malicious Roaming Token entry where the identifier string contains directory traversal characters and thereby write an arbitrary number of bytes to any file on the file

This vulnerability was reported to MSRC in April 2022 and was classified as an ‘Elevation of Privilege’ vulnerability. Microsoft assigned [CVE-2022-30170](#) and published [KB5017365](#) and [KB5017367](#) on September 13 2022 to address the issue. Mandiant published this vulnerability under [MNDT-2022-0038](#).

An Attacker's Perspective

The use of Credential Roaming in an organization allows attackers (and Red Teams) to abuse the saved credentials for the purposes of privilege escalation. The author identifies the following situations that could allow an attacker to abuse Credential Roaming:

1. An organization has not applied the September 2022 patch to each system where Credential Roaming is used.

The affected systems are vulnerable to CVE-2022-30170 – an attacker can abuse this vulnerability to write arbitrary files to the affected systems in the context of any users they can control, possibly allowing for lateral movement. Using this technique, an attacker can spread to any affected system accessed by the victim user, including systems that are possibly unknown to the attacker at the time of compromise, in a fully automatic fashion.

Note that the attacker requires write access to the victim user’s Active Directory account, either by having access to the account itself or through another AD account with sufficient privileges over the victim account. Credential Roaming must be configured and in use on the victim user and system for the vulnerability to be exploitable.

2. An attacker gained Domain Administrator privileges in an organization where Credential Roaming is in use or was used in the past without proper clean-up.

In this scenario, the attacker can retrieve the DPAPI Domain Backup Key and decrypt all credentials stored in the Active Directory attributes for Credential Roaming. In his blog post ‘[Extracting Roamed Private Keys from Active Directory](#)’, Mr. Grafnetter explains how his `DSInternals` toolkit can be used to extract the roamed credentials from Active Directory and how Mimikatz tool can be used to decrypt the DPAPI secrets with the DPAPI Domain Backup Key.

Note that even if your organization does not currently use Credential Roaming, but used Credential Roaming in the past, credentials may still be stored in the Active Directory! In their 2012 whitepaper, [Microsoft explains how system administrators should decommission Certificate Roaming](#). The decommissioning process includes manual deletion of the Roaming Credentials from Active Directory (clearing the `msPKIAccountCredentials`, `msPKIRoamingTimeStamp` and `msPKIDPAPIMasterKeys` LDAP attributes). Organizations that failed to perform this clean-up process may still have sensitive secrets stored in their Active Directory environment.

Additionally, if the organization uses (or used) Credential Roaming with Windows Vista-era machines, not only certificates/private keys but also usernames and passwords may be stored in the Active Directory ([Windows 7 removed the ability to roam usernames and passwords](#), presumably due to security concerns).

3. An attacker has access to the cleartext password of a user where Credential Roaming is in use or was in use in the past.

As in scenario (2), the attacker can authenticate as the victim user and retrieve the Credential Roaming attributes from Active Directory. With the user's cleartext password, the attacker can decrypt the DPAPI master key and in turn obtain the credentials stored in the Credential Roaming attributes.

4. An attacker has read access to the `msPKIDPAPIMasterKeys` attribute on a victim account, but does not have the cleartext password of the victim user.

By reading the `msPKIDPAPIMasterKeys` attribute, an attacker can extract the DPAPI Master Key for a user and use the [DPAPImk2john.py](#) Python script from the popular John the Ripper password cracking software to extract the user's password hash. This hash can then be cracked offline using either [John The Ripper \(john\)](#) or [hashcat](#).

Recommendations

Mandiant recommends organizations to check whether Credential Roaming is in use in their environment; and if so, apply the September 2022 patch urgently to remediate CVE-2022-30170. Additionally, organizations that have used Credential Roaming in the past should investigate if the proper clean-up process ([as described by Microsoft](#)) was followed.

Future Work

While this research certainly deepens our understanding of Credential Roaming and offers insight into why APT29 is actively querying the related LDAP attributes in Active Directory, the attribute that Mandiant IR consultants observed (`msPKI-CredentialRoamingTokens {b7ff5a38-0818-42b0-8110-d3d154c97f24}`) is not featured in the inner workings of Credential Roaming. Mandiant was—as of yet—unable to determine how (or if) this attribute is used in Credential Roaming.

References

1. [Certs On Wheels: Understanding Credential Roaming](#) – Microsoft
2. [Extracting Roamed Private Keys from Active Directory](#) – Michael Grafnetter
3. [Windows: Credential Roaming \(Whitepaper\)](#) – Microsoft
4. [CVE-2022-30170: Windows Credential Roaming Service Elevation of Privilege Vulnerability](#) – Microsoft
5. [MNDT-2022-0038: Windows Credential Roaming Service Elevation of Privilege Vulnerability](#)

Appendix: Disclosure Timeline for CVE-2022-30170

- 20 April 2022 - Issue submitted to Microsoft
- 26 April 2022 - Case opened
- 18 May 2022 - Microsoft confirms issue
- 01 June 2022 - Microsoft classifies issue as a 'Defense in Depth' vulnerability
- 07 June 2022 - Re-explain scope and impact to MSRC

- 09 June 2022 - MSRC re-evaluates severity of the issue
- 17 June 2022 - MSRC assigns CVE-2022-30170
- 13 September 2022 - Patch released

Posted in

- [Threat Intelligence](#)
- [Security & Identity](#)

Source: <https://www.mandiant.com/resources/blog/apt29-windows-credential-roaming>