How NTLM Relay Ruins Your Exchange Servers

Tianze Ding (@D1iv3)
Tianze Ding (@D1iv3)
Senior Security Researcher, Tencent Security Xuanwu Lab
Focusing on Active Directory Security / Cloud Security / Web Security
2022 MSRC Most Valuable Researchers
DEFCON / Black Hat Asia Speaker
Former CTF Player, 2018 HITB Singapore CTF Final 1st runner-up
Vulnerabilities in this talk

<table>
<thead>
<tr>
<th>Vulnerabilities</th>
<th>Affected Components</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2021-33768</td>
<td>Exchange Front End</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>CVE-2022-21980</td>
<td>Exchange Front End</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>CVE-2022-24516</td>
<td>Exchange Front End</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>CVE-2022-24477</td>
<td>Exchange Back End</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>CVE-2021-26414</td>
<td>Windows DCOM</td>
<td>Remote Code Execution</td>
</tr>
<tr>
<td>Won’t Fix</td>
<td>Exchange &amp; Active Directory</td>
<td>Privilege Escalation to Domain Admin</td>
</tr>
</tbody>
</table>
Some of vulnerabilities (CVE-2021-33768/CVE-2022-21979) in this attack surface was also found and reported to MSRC independently by Orange Tsai (@orange_8361), and named ProxyRelay.
NTLM Relay

NTLM Relay is a quite old MITM attack, but still very popular among Active Directory attacks.

Attacker needs to trick/coerce a victim user/computer to authenticate.
Coerced Authentication Methods

Many well-known MS-RPC APIs can coerce machine accounts authenticate to an arbitrary target with NTLM protocol (over SMB)

- MS-PRPNN (PrinterBug)
- MS-EFSR (Petitpotam)
- MS-PAR
- MS-DFSNM
- MS-FSRVP

```c
DWORD RpcRemoteFindFirstPrinterChangeNotificationEx(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD fdwFlags,
    [in] DWORD fdwOptions,
    [in, string, unique] wchar_t* pszLocalMachine,
    [in] DWORD dwPrinterLocal,
    [in, unique] RPC_V2_NOTIFY_OPTIONS* pOptions
);
```

```c
long EfsRpcOpenFileRaw(
    [in] handle_t binding_h,
    [out] PEXIMPORT_CONTEXT_HANDLE* hContext,
    [in, string] wchar_t* FileName,
    [in] long Flags
);
```

Set to a UNC path: `\192.168.1.100\path`

Some MS-RPC are enabled by default

Any low-privileged domain users / machine accounts can call these APIs remotely
NTLM Relay

- **Authentication**
  - Attacked target supports NTLM authentication
  - Relayed credentials need to be considered valid by target services
  - All domain users/computers can authenticate to all AD integration services and authentication will be accepted

- **Authorization**
  - Relayed credentials need to have special privileges to perform attacks, e.g., local administrator privileges
  - In most cases, low-privileged credentials (regular domain users / machine accounts) can’t do much harm
NTLM Relay Mitigations

● NTLM reflection protection
● SMB Signing
● LDAP Signing
● EPA (Extended Protection for Authentication)
  ○ Channel Binding
  ○ Service Binding
● Signing / Sealing restrictions implemented by individual services
  ○ Some MS-RPC Services
Why Exchange Server

- Exchange Server is the world's most famous enterprise mail solution
- Highly integrated with Active Directory
  - support AD authentication methods, NTLM/Kerberos
  - some Exchange users and groups have powerful privileges in Active Directory
  - ideal entry point for attackers to break Active Directory
- Complex implementation, software and network architecture
Architecture Overview

HTTP 80 / HTTPS 443

HTTP 81 / HTTPS 444
The Front End HttpProxy forward client access requests to The Back End
The Front End authenticates to Back End with machine account and SYSTEM account

Kerberos AP-REQ to Back-End with EXCHANGE$

Local NTLM authentication with SYSTEM account
The Front End and the Back End synchronize user identities through X-CommonAccessToken

GET /owa HTTP/1.1
X-FE-ClientIP: 192.168.2.1
X-Forwarded-For: 192.168.2.1
X-Forwarded-Port: 60388
X-MS-EdgeIP:
Authorization: Negotiate YIGbAYJKOZI...
msExchProxyUri: https://192.168.2.129/owa
X-IsFromCafe: 1
X-SourceCafeServer: EXCHANGE1.XLAB.SEC
X-CommonAccessToken: VgEAVAdXaw5kb3dzQwBBUJhc2ljTBjYTEF...
X-vDirObjectID: d2206b1e-fa8b-4b86-b24e-880597cbb33
Host: exchange1.xlab.sec:444
Cookie: PrivateComputer=true; ClientId=3BC60BD0BB8A452784D...
Connection: Keep-Alive

Base64 decode

| V \x01 \x00 | Version |
| T \x07 Windows | Token Type |
| C \x00 | Compressed Flag |
| A \x08 Kerberos | Authentication Type |
| L \x12 XLAB\Administrator | Logon Name |
| U \x2c S-1-5-21-2658105758-2410005936-383990995-500 User SID |
| G \x0d\x00\x00\x00 \x07\x00\x00\x00\x2c S-1-5-21-2658105758-2410005936-383990995-513 Group SIDs |
| \x00\x00\x00\x00 | |

The Back End use the X-CommonAccessToken to create a new user token
X-CommonAccessToken


```csharp
private bool TryGetCommonAccessToken(HttpContext httpContext,
                           Stopwatch stopwatch, out CommonAccessToken token)
{
    // ...
    string text = httpContext.Request.Headers["X-CommonAccessToken"];
    // ...
    bool flag;
    // ...
    flag = this.IsTokenSerializationAllowed(httpContext.User.Identity as WindowsIdentity);
    // ...
    if (!flag) {
        throw new BackendRehydrationException(SecurityStrings.
            SourceServerNoTokenSerializationPermission(safeName));
    }
    token = CommonAccessToken.Deserialize(text);
}
```

- Back End gets X-CommonAccessToken from HTTP header
- Check if the current user has TokenSerialization right
- Deserialize it to create a new user token
- Use the new user token to access Back End endpoints
Exchange Server Machine Account

- **TokenSerialization** ExtendedRight
  - ms-Exch-EPI-Token-Serialization (06386F89-BEFB-4e48-BAA1-559FD9221F78)
- Members of the **Exchange Servers group** have this right on Exchange Servers
- Exchange machine accounts will be added to Exchange Servers group when installing Exchange Servers

```
[PS] C:\Windows\system32> Get-ADPermission -Identity Exchange1 | where 
{$_._ExtendedRights -like "ms-Exch-EPI-Token-Serialization"} -and 
{-not $_._Deny} | ft -autosize Identity,User,ExtendedRights,Deny,IsInherited

<table>
<thead>
<tr>
<th>Identity</th>
<th>User</th>
<th>ExtendedRights</th>
<th>Deny</th>
<th>IsInherited</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCHANGE1</td>
<td>NT AUTHORITY\NETWORK SERVICE</td>
<td>{ms-Exch-EPI-Token-Serialization} False False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>EXCHANGE1</td>
<td>XLAB\Exchange Servers</td>
<td>{ms-Exch-EPI-Token-Serialization} False False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Mailbox Database 1810180856\EXCHANGE1</td>
<td>XLAB\Exchange Servers</td>
<td>{ms-Exch-EPI-Token-Serialization} False True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>EXCHANGE1\EXCHANGE1</td>
<td>NT AUTHORITY\NETWORK SERVICE</td>
<td>{ms-Exch-EPI-Token-Serialization} False True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>EXCHANGE1\EXCHANGE1</td>
<td>XLAB\Exchange Servers</td>
<td>{ms-Exch-EPI-Token-Serialization} False True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
```
Cluster architecture is widely used in enterprise environments for high availability.

The Front End and the Back End can be on different Exchange Servers.

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Pragma: no-cache
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Content-Encoding: gzip
X-FrontEnd-Begin: 2023-08-06T15:02:56.006
X-BacEnd-Begin: 2023-08-06T15:02:56.017
X-FrontEnd-Handler-Begin: 2023-08-06T15:02:56.007
X-BacEnd-End: 2023-08-06T15:02:56.037
X-BEServer: EXCHANGE-2
X-UA-Compatible: IE=EmulateIE7
Set-Cookie:
X-OWA-CANARY=
X-OWA-CHALLENGE=
Set-Cookie:
X-BacEndCookie=
X-OWA-Data=
X-FEServer: EXCHANGE-1
Date: Sun, 06 Aug 2023 07:02:55 GMT
Exchange Server - Cluster

- Exchange machine accounts will be used when Frontends authenticate to other Backends in different Exchange Servers
- All Exchange machine accounts are in the same group and have same privileges on all Exchange servers in the AD
- Exchange1 can access the backend of Exchange2 just like the backend of itself
NTLM Relay to Exchange Server

1. Coerce Exchange1$ to authenticate
2. Exchange1$'s NTLM messages
3. Exchange1$'s NTLM messages
4. Login success as Exchange1$ and get Exchange1$'s privilege
## Exchange Endpoints

<table>
<thead>
<tr>
<th>Endpoints</th>
<th>Description</th>
<th>Mail Access</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWA</td>
<td>Outlook Web App</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>EWS</td>
<td>Exchange Web Services, used by Outlook for macOS and Outlook add-ins</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>API</td>
<td>REST API, available in Exchange 2016 CU3 or newer</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>Microsoft-Server-ActiveSync</td>
<td>ActiveSync let you synchronize a mobile device with your Exchange mailbox</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>MAPI</td>
<td>MAPI over HTTP, used by modern Microsoft Outlook</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>RPC</td>
<td>Outlook Anywhere, used by Microsoft Outlook 2013, Outlook 2010, or Outlook 2007</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>Powershell</td>
<td>Used by Exchange PowerShell Cmdlets</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>ECP</td>
<td>Exchange Control Panel</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Autodiscover</td>
<td>Used by client application to configure itself</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OAB</td>
<td>Offline Address Book</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Front End endpoints NTLM support

<table>
<thead>
<tr>
<th>Frontend Endpoints</th>
<th>Authentication Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>/EWS/</td>
<td>Kerberos, <strong>NTLM</strong></td>
</tr>
<tr>
<td>/mapi/emsmdb</td>
<td>Kerberos, <strong>NTLM</strong></td>
</tr>
<tr>
<td>/API/</td>
<td>Kerberos, <strong>NTLM</strong></td>
</tr>
<tr>
<td>/owa/Integrated/</td>
<td>Web Form, Kerberos, <strong>NTLM</strong></td>
</tr>
<tr>
<td>/Microsoft-Server-ActiveSync/Proxy</td>
<td>Basic, Kerberos, <strong>NTLM</strong></td>
</tr>
<tr>
<td>/rpc/rpcproxy.dll</td>
<td>Basic, Kerberos, <strong>NTLM</strong></td>
</tr>
<tr>
<td>/autodiscover/</td>
<td>Kerberos, <strong>NTLM</strong>, Basic</td>
</tr>
<tr>
<td>/oab/</td>
<td>Kerberos, <strong>NTLM</strong></td>
</tr>
<tr>
<td>/ecp/</td>
<td>Web Form</td>
</tr>
<tr>
<td>/Powershell/</td>
<td>Kerberos</td>
</tr>
</tbody>
</table>
**NTLM Relay to the Front End**

- The EPA is disabled on all Front-End endpoints by default

EPA is not compatible well with 7-layer load balancer by default

Enterprise IT administrators usually do not enable EPA on Exchange Servers
NTLM Relay to the Front End

Can we use X-CommonAccessToken to impersonate arbitrary user when we relay to the Front End?

Microsoft.Exchange.FrontEndHttpProxy.dll\HttpProxy\ProxyRequestHandler.cs

```csharp
protected virtual bool ShouldCopyHeaderToServerRequest(string headerName)
{
    return !string.Equals(headerName, "X-CommonAccessToken", StringComparison.OrdinalIgnoreCase) &&
            !string.Equals(headerName, Constants.XIsFromCache, StringComparison.OrdinalIgnoreCase) &&
            !string.Equals(headerName, Constants.XSourceCafeServer, StringComparison.OrdinalIgnoreCase) &&
            !string.Equals(headerName, Constants.MsExchProxyUri, StringComparison.OrdinalIgnoreCase) &&
            !string.Equals(headerName, "X-MSExchangeActivityCtx", StringComparison.OrdinalIgnoreCase) &&
            !string.Equals(headerName, "return-client-request-id", StringComparison.OrdinalIgnoreCase) &&
            !string.Equals(headerName, "X-Forwarded-For", StringComparison.OrdinalIgnoreCase) &&
            (!headerName.StartsWith(Constants.XBackendHeaderPrefix, StringComparison.OrdinalIgnoreCase)) &&
            !this.ClientRequest.GetHttpRequestBase().IsProbeRequest();
}
```

X-CommonAccessToken in the client request doesn’t be allowed to forward to the server request
Can we impersonate other users without X-CommonAccessToken?

The FrontEnd proxy will create a new X-CommonAccessToken with current user's identity.
Exchange EWS

Exchange Web Services API
- used by Outlook for macOS and Outlook add-ins by default
- provide full-ability SOAP API for accessing and manipulating emails, attachments, contacts, calendar events, etc.

Endpoint: /EWS/Exchange.asmx, with NTLM support

EWS API support token serialization natively

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsi="https://schemas.microsoft.com/exchange/services/2006/types">

<soap:Body>
  <GetFolder xmlns="https://schemas.microsoft.com/exchange/services/2006/messages"
xmlns:xsi="https://schemas.microsoft.com/exchange/services/2006/types">
    <FolderShape>
      <t:BaseShape>Default</t:BaseShape>
    </FolderShape>
    <FolderIds>
      <t:DistinguishedFolderId Id="inbox"/>
    </FolderIds>
  </GetFolder>
</soap:Body>
</soap:Envelope>
```
The **SerializedSecurityContext** element is used in the SOAP header for token serialization in server-to-server authentication.

```csharp
internal virtual AuthZClientInfo ProcessSerializedSecurityContextHeaders(Message request)
{
    // ...
    {
        if (!string.IsNullOrEmpty(text) && SmtpAddress.IsValidSmtpAddress(text))
        {
            serializedSecurityContextTypeForAS.PrimarySmtpAddress = text;
        }
        authZClientInfo = serializedSecurityContextTypeForAS.ToAuthZClientInfo();
    }
    // ...
```
EWS will create user token with the UserSid and GroupSids in the SerializedSecurityContext.

Users with token serialization right can impersonate any Exchange users on the EWS endpoint.
The attacker can perform NTLM relay to the Front-End EWS, impersonate arbitrary Exchange users to send emails, read emails, download attachments, do anything EWS supports.
NTLM Relay to the Font End - CVE-2021-33768

DEMO: https://youtu.be/I_HOLSztI4Q
Not allow machine account logins to the Front End anymore
CVE-2021-33768 - Patch Bypass

This branch is introduced in the same Security Update with the patch

```csharp
protected virtual void AddProtocolSpecificHeadersToServerRequest(WebHeaderValueCollection headers)
{
  ...
  if (this.AuthBehavior.AuthState != AuthState.BackEndFullAuth)
  {
    if (this.ClientRequest.IsAuthenticated)
    {
      string text = this.ClientRequest.Headers["X-CommonAccessToken"];  
      if (string.IsNullOrEmpty(text))
      {
        if (CommonAccessTokens.Deserialize(text).IsSystemOrMachineAccount())
        {
          throw new HttpException(400, "Bad context");
        }
        WindowsIdentity windowsIdentity = null;
        if (this.HttpContext != null && this.HttpContext.User != null)
        {
          windowsIdentity = (this.HttpContext.User.Identity as WindowsIdentity);
        }
        if (windowsIdentity != null || !windowsIdentity.IsSystemOrTrustedMachineAccount())
        {
          throw new HttpException(400, "Unauthorized to send context");
        }
      RequestDetailsLoggerBase.RequestDetailsLogger>.SafeAppendGenericInfo(this.Logger, "CT", "TMA");
      headers["X-CommonAccessToken"] = text;
    }
    else
    {
      // The patch for CVE-2021-33768
      // Not allow machine account logins to Front End
    }
  }
  
  // Add protocol specific headers to server request
}```

- Read X-CommonAccessToken from clientRequest directly.
- Deserialize X-CommonAccessToken, check if the identity is a machine account.
- If the user in the X-CommonAccessToken is not a machine account.
- X-CommonAccessToken from client request will be added to serverRequest.Headers.
CVE-2021-33768 – Patch Bypass

Microsoft.Exchange.FrontEndHttpProxy.dll\HttpProxy\ProxyRequestHandler.cs

```csharp
protected void PrepareServerRequest(HttpWebRequest serverRequest) {
    ...
    this.CopyHeadersToServerRequest(serverRequest);
    ...
    this.AddProtocolSpecificHeadersToServerRequest(serverRequest.Headers);
    ...
}
```

```csharp
protected virtual bool ShouldCopyHeaderToServerRequest(string headerName) {
    return !string.Equals(headerName, "X-CommonAccessToken", StringComparison.OrdinalIgnoreCase) &&
    !string.Equals(headerName, Constants.XIsFromCafe, StringComparison.OrdinalIgnoreCase) &&
    !string.Equals(headerName, Constants.XSourceCafeServer, StringComparison.OrdinalIgnoreCase) &&
    !string.Equals(headerName, Constants.XSExchProxyUri, StringComparison.OrdinalIgnoreCase) &&
    !string.Equals(headerName, "X-MSExchangeActivityCtx", StringComparison.OrdinalIgnoreCase) &&
    !string.Equals(headerName, "return-client-request-id", StringComparison.OrdinalIgnoreCase) &&
    !string.Equals(headerName, "X-Forwarded-For", StringComparison.OrdinalIgnoreCase) &&
    (!headerName.StartsWith(Constants.XBackendHeaderPrefix, StringComparison.OrdinalIgnoreCase) ||
    this.ClientRequest.GetHttpRequestBase().IsProbeRequest()));
}
```

PrepareServerRequest

|__ CopyHeadersToServerRequest
|__ ShouldCopyHeaderToServerRequest
|__ AddProtocolSpecificHeadersToServerRequest

ShouldCopyHeaderToServerRequest doesn’t allow X-CommonAccessToken in the client request to be forwarded to the Back End

AddProtocolSpecificHeadersToServerRequest is called after ShouldCopyHeaderToServerRequest

Attackers can forge any mailbox user’s X-CommonAccessToken, the Front End will forward it directly to the Back End
The Security Update for CVE-2021-33768 bring us a new and more powerful vulnerability

Attackers can perform NTLM relay to all Front End endpoints have NTLM support and impersonate arbitrary Exchange user
NTLM Relay to the Front End - CVE-2022-21980

All endpoints that support NTLM authentication and support email access are exploitable
● add X-CommonAccessToken to impersonate Exchange users
● implement the corresponding protocol to operate the mail

EWS
- /EWS/Exchange.asmx
- SOAP XML over HTTPS
  - FindFolder / FindItem / GetItem

ActiveSync
- /Microsoft-Server-ActiveSync/Proxy
- WBXML over HTTPS
  - FolderSync to dump collectionIds
  - Sync to dump email contents

OWA
- JSON with HTTPS
  - /owa/Integrated/service.svc?action=FindConversation
  - /owa/Integrated/service.svc?action=GetConversationItems

MAPI
- /mapi/emsmdb/?MailboxId=victimmailboxid
- MS-OXPROPS over HTTPS
  - RopGetPropertiesListRequest
  - RopGetPropertiesSpecificRequest

API
- JSON with HTTPS
  - /api/v2.0/users/victim@xlab.sec/mailFolders/inbox/messages

RPC
- /rpc/rpcproxy.dll
- MS-OXPROPS over MS-RPC over HTTP(S)
### RPC (Outlook Anywhere)

Endpoint: /rpc/rpcproxy.dll, works as an RPC proxy

- Requires two connections RPC_IN_DATA and RPC_OUT_DATA
- MAPI (MS-OXPROPS) over MS-RPC (MS-OXCRPC) over HTTP(S)

![Diagram showing the RPC connection flow](image)
RPC (Outlook Anywhere)

RPC authentication
- The RPC proxy allows client to skip authentication on the RPC level to get a faster connection
- RPC auth type RPC_C_AUTHN_NONE
- RPC auth level RPC_C_AUTHN_LEVEL_NONE
- No NTLM relay protection on the RPC level

NTLM relay to the Front-End /rpc/rpcproxy.dll
- Trigger NTLM relay twice, login to RPC_IN_DATA and RPC_OUT_DATA as Exchange machine account
- Add X-CommonAccessToken HTTP Header to impersonate arbitrary Exchange user
- Skip RPC authentication to prevent signing / sealing on the RPC level
- Use MAPI(MS-OXPROPS) protocol to access arbitrary emails, attachments, etc.
## NTLM Relay to the Front End - CVE-2022-21980

<table>
<thead>
<tr>
<th>Frontend Endpoints</th>
<th>Authentication Methods</th>
<th>Affected</th>
<th>Exploitable</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>/EWS/</td>
<td>Kerberos, NTLM</td>
<td>✅</td>
<td>✅</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/mapi/emsmdb</td>
<td>Kerberos, NTLM</td>
<td>✅</td>
<td>✅</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/API/</td>
<td>Kerberos, NTLM</td>
<td>✅</td>
<td>✅</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/owa/Integrated/</td>
<td>Web Form, Kerberos, NTLM</td>
<td>✅</td>
<td>✅</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/Microsoft-Server-ActiveSync/Proxy</td>
<td>Basic, Kerberos, NTLM</td>
<td>✅</td>
<td>✅</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/rpc/rpcproxy.dll</td>
<td>Basic, Kerberos, NTLM</td>
<td>✅</td>
<td>✅</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/autodiscover/</td>
<td>Kerberos, NTLM, Basic</td>
<td>✅</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/oab/</td>
<td>Kerberos, NTLM</td>
<td>✅</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/ecp/</td>
<td>Web Form</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/Powershell/</td>
<td>Kerberos</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
NTLM Relay to the Front End - CVE-2022-21980

DEMOS: https://www.youtube.com/playlist?list=PLtZO9vwOND910VlyxVOEPoTJNZMWBtv9y
NTLM Relay to the Back End

- Everyone can access the Back End port 444 (no IP/Host whitelist by default)
- All Back End endpoints support NTLM authentication, and the EPA is disabled by default
- The EPA on the Backend needs to be disabled by design, if enabled it will break the communication between the Front-End and the Back-End

If EPA is set to accept or required, the frontend will fail (401) to authenticate to the backend.
EPA on Back End Endpoints

```csharp
internal static string GenerateKerberosAuthHeader(string host, int traceContext, ref AuthenticationContext authenticationContext, ref string kerberosChallenge)
{
    // ...
    authenticationContext = new AuthenticationContext();
    string text = Constants.SpnPrefixForHttp + host;
    // ...
    authenticationContext.InitializeForOutboundNegotiate(
        AuthenticationMechanism.Kerberos, text, null, null
    );
    // ...
    SecurityStatus securityStatus = authenticationContext.
    | NegotiateSecurityContext(inputBuffer, out bytes);
    // ...

    public SecurityStatus InitializeForOutboundNegotiate(AuthenticationMechanism mechanism,
        string spn, string username, string domain, SecureString password)
    {
        // ...
        this.ssiContext = this.CreateSspiContext();
        bool flag = this.packageName.Equals("Kerberos", StringComparison.OrdinalIgnoreCase);
        return this.ssiContext.InitializeForOutboundAuthentication(
            this.packageName, spn, @default, true,
            flag ? null : this.channelBindingToken
        );
    // ...
}
```

No channel binding token when generating Kerberos AP-REQ

EPA is not supported in the code level by design
EPA on Back End Endpoints

● No Channel Binding Token (CBT) when the Front End authenticates to the Back End

Kerberos AP-REQ authenticator generated by the Front End, with CBT set to all zeros
NTLM Relay to the Back End - CVE-2022-24477

Attackers can perform NTLM relay to all Back-End endpoints, and impersonate arbitrary Exchange user with X-CommonAccessToken to read emails / RCE
# NTLM Relay to the Back End - CVE-2022-24477

<table>
<thead>
<tr>
<th>Backend Endpoints</th>
<th>Authentication Methods</th>
<th>Affected</th>
<th>Exploitable</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Powershell/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Remote Code Execution</td>
</tr>
<tr>
<td>/ecp/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Remote Code Execution</td>
</tr>
<tr>
<td>/EWS/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/mapi/emsmdb</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/API/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/owa/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/Microsoft-Server-ActiveSync/Proxy</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/rpc/rpcproxy.dll</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/RpcWithCert/rpcproxy.dll</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>✓</td>
<td>Arbitrary Mailbox Takeover</td>
</tr>
<tr>
<td>/autodiscover/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/oab/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/PushNotifications/</td>
<td>Kerberos, NTLM</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
NTLM Relay to the Back End – RCE

The /Powershell and /ECP are mainly for Exchange management, support NTLM authentication on the Back End

> GET /ecp/ HTTP/1.1
> Host: exchange1.xlab.se:444
> User-Agent: curl/7.85.0
> Accept: */*
> X-IsFromCafe: 1
>
> < HTTP/1.1 401 Unauthorized
> Cache-Control: private
> Server: Microsoft-IIS/10.0
> X-Content-Type-Options: nosniff
> X-Frame-Options: SameOrigin
> X-AspNet-Version: 4.0.30319
> WWW-Authenticate: Negotiate
> WWW-Authenticate: NTLM
> X-Powered-By: ASP.NET
> X-UA-Compatible: IE=10
> Date: Mon, 07 Aug 2023 15:01:39 GMT
> Content-Length: 1181

> GET /powershell HTTP/1.1
> Host: exchange1.xlab.se:444
> User-Agent: curl/7.85.0
> Accept: */*
>
> < HTTP/1.1 401 Unauthorized
> Content-Type: text/html
> Server: Microsoft-IIS/10.0
> request-id: ead1487f-96f9-4bdc-984f-c94d7161341f
> WWW-Authenticate: Negotiate
> WWW-Authenticate: NTLM
> X-Powered-By: ASP.NET
> Date: Mon, 07 Aug 2023 15:16:05 GMT
> Content-Length: 1181
NTLM Relay to the Back End – Powershell

- Attackers can impersonate administrator on the Back-End /Powershell with X-CommonAccessToken, and execute arbitrary Exchange Cmdlet
- Exchange PowerShell doesn’t support to execute Windows commands like the native PowerShell of Windows, it only support cmdlets implemented by Exchange
- How to RCE? Maybe we need found a new Post-Auth RCE on Exchange Cmdlets?
### NTLM Relay to the Back End – Powershell

- Role-based access control management Cmdlets

#### role-based-access-control

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add-ManagementRoleEntry</td>
<td>This cmdlet is available in on-premises Exchange and in the cloud-based service. Some parameters and settings may be exclusive to one environment or the other. Use the Add-ManagementRoleEntry cmdlet to add management role entries to an existing management role. For information about the parameter sets in the Syntax section below, see Exchange cmdlet syntax.</td>
<td>Add-RoleGroupMember [-Identity] &lt;RoleGroupIdParameter&gt; -Member &lt;SecurityPrincipalIdParameter&gt; [-BypassSecurityGroupManagerCheck] [-Confirm] [-DomainController &lt;Fqdn&gt;] [-WhatIf] [&lt;CommonParameters&gt;]</td>
</tr>
<tr>
<td>Add-RoleGroupMember</td>
<td>This cmdlet is available in on-premises Exchange and in the cloud-based service. Some parameters and settings may be exclusive to one environment or the other. Use the Add-RoleGroupMember cmdlet to add members to a management role group. For information about the parameter sets in the Syntax section below, see Exchange cmdlet syntax.</td>
<td>Add-RoleGroupMember [-Identity] &lt;RoleGroupIdParameter&gt; -Member &lt;SecurityPrincipalIdParameter&gt; [-BypassSecurityGroupManagerCheck] [-Confirm] [-DomainController &lt;Fqdn&gt;] [-WhatIf] [&lt;CommonParameters&gt;]</td>
</tr>
<tr>
<td>Get-ManagementRole</td>
<td>This cmdlet is available in on-premises Exchange and in the cloud-based service. Some parameters and settings may be exclusive to one environment or the other. Use the Get-ManagementRole cmdlet to view management roles that have been created in your organization. For information about the parameter sets in the Syntax section below, see Exchange cmdlet syntax.</td>
<td>Get-ManagementRole</td>
</tr>
</tbody>
</table>
NTLM Relay to the Back End – Powershell

“Organization Management” is one of these built-in role groups, which is also a member of local administrators of Exchange Servers

<table>
<thead>
<tr>
<th>Role group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization Management</td>
<td>Administrators who are members of the Organization Management role group have administrative access to the entire Exchange 2013 organization and can perform almost any task against any Exchange 2013 object, with some exceptions. By default, members of this role group can't perform mailbox searches and management of unscoped top-level management roles.</td>
</tr>
</tbody>
</table>

Add-RoleGroupMember ’Organization Management’ -Member attacker
NTLM Relay to the Back End Powershell - RCE

RCE with PsExec, WmiExec, WinRM ...
NTLM Relay to the Back End Powershell - RCE

DEMO: https://youtu.be/Y7uVtfZ3jcU
NTLM Relay to the Back End – ECP

- Two methods to impersonate other users on the Back End ECP
  - X-CommonAccessToken HTTP Header
  - `/ecp/ProxyLogon.ecp` create new user token from the XML in POST body
    - Also used by the well-known ProxyLogon vulnerability

GET /ecp/ HTTP/1.1
X-FE-ClientIP: 192.168.2.1
X-Forwarded-For: 192.168.2.1
X-Forwarded-Port: 57974
Authorization: Negotiate YIIGbQYJKoZIhv...
X-ExCompId: ClientAccessFrontEnd
X-MSExchangeActivityCtx:
V=1.0.0.0;Id=8aeb3e99-7e8b-44f8-9fb9-bf5350644662;C=;P=
msExchClientPath: %2Fecp%2F
msExchProxyUri: https://192.168.2.10/ecp/
X-IsFromCafe: 1
X-SourceCafeServer: EXCHANGE1.XLAB.SEC
X-CommonAccessToken: VgEAVAdXaW5kb...
X-vDirObjectId: 6627fd03-e475-454b-90b5-166b1adc5e66
Host: exchange1.xlab.sec:444

A valid X-vDirObjectId is required in the request header

Front-End ECP Virtual Directory GUID

View-Only Configuration role can read it with Get-EcpVirtualDirectory cmdlet
NTLM Relay to the Back End ECP - RCE

ECP also support add users to Organization Management group

Attackers with View-Only Configuration role can impersonate administrator on the Back-End ECP with NTLM relay and add himself to Organization Management, and achieve RCE on Exchange Servers with PsExec, WmiExec, WinRM ...
Relay to the Back End - CVE-2022-24477

DEMOS: https://www.youtube.com/playlist?list=PLtZO9vwOND92_EcfyXo90lHtLng8aIEQT
Patch - Extended Protection

- Extended Protection is supported on Exchange Server 2013, 2016 and 2019 starting with the August 2022 Exchange Server Security Update (SU) releases.
- This protection is accomplished by Channel Binding Token (CBT) and mainly for SSL connections.
- Customers need to enable the Extended Protection manually.
- All exploitable Front End endpoints and Back End endpoints are recommended to enabled Extended Protection.

Patch Script: ExchangeExtendedProtectionManagement.ps1
NTLM Relay to the Front End /RPC over HTTP(80)

Extended Protection only protects HTTPS connections
SSLOffLoading is enabled by default in the Front End /RPC, which means /RPC endpoint also supports HTTP 80

![Get-OutlookAnywhere output]

CVE-2022-24516
NTLM relay to the Front End /RPC on HTTP 80 can also lead to arbitrary mailbox takeover

The patch script ExtendedProtection ExchangeExtendedProtectionManagement.ps1 will turn off the SSLOffLoading for the Front End /RPC when enable the Extended Protection
Patch Bypass?

The Extended Protection is still not enabled on the frontend and backend AutoDiscover

<table>
<thead>
<tr>
<th>IIS Website</th>
<th>Virtual Directory</th>
<th>Recommended Extended Protection</th>
<th>Recommended sslFlags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Website</td>
<td>API</td>
<td>Required</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>AutoDiscover</td>
<td>Off</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>ECP</td>
<td>Required</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>EWS</td>
<td>Accept (UI) /Allow (Script)</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>MAPI</td>
<td>Required</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>Microsoft-Server-ActiveSync</td>
<td>Accept (UI) /Allow (Script)</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>OAB</td>
<td>Required</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>OWA</td>
<td>Required</td>
<td>Ssl,Ssl128</td>
</tr>
<tr>
<td>Default Website</td>
<td>PowerShell</td>
<td>Required</td>
<td>SslNegotiateCert</td>
</tr>
<tr>
<td>Default Website</td>
<td>RPC</td>
<td>Required</td>
<td>Ssl,Ssl128</td>
</tr>
</tbody>
</table>

If you found an SSRF on AutoDiscover endpoint (like ProxyNotShell), this attack will still work.
Exchange Server Machine Account

- All members of Exchange Trusted Subsystem have **local administrator privileges** on all Exchange Servers.
- **All Exchange machine accounts** will be added to this group during Exchange Server installation.

```bash
PS C:\> hostname
exchange1
PS C:\> net localgroup administrators
Alias name   administrators
Comment       Administrators have complete and unrestricted access to the computer/domain
Members

PS C:\> net group "Exchange Trusted Subsystem" /domain
The request will be processed at a domain controller for domain xlab.sec.

Administrator
XLAB\Domain Admins
XLAB\Exchange Trusted Subsystem
XLAB\Organization Management
The command completed successfully.
```
NTLM Relay to Windows Services

- MS-RPC over SMB (ncacn_np)

```powershell
PS C:\> Get-SmbServerConfiguration | select RequireSecuritySignature

RequireSecuritySignature
------------------------
True
```

SMB signing is enabled by default On Exchange Servers

- WinRM (Powershell Remoting)

```powershell
PS C:\> winrm get winrm/config/service/auth

Auth
-----
Basic = false
Kerberos = true
Negotiate = true
Certificate = false
CredSSP = false
CbtHardeningLevel = Relaxed
```

HTTP: Signing and Sealing are required
HTTPS: EPA Channel Binding is enabled

- MS-RPC over TCP (ncacn_ip_tcp)
  - Many RPC interfaces support ncacn_ip_tcp transport, MS-PAR, MS-TSCH, DCOM, WMI...
  - Each RPC interface implement its own security policies
NTLM Relay to MS-RPC (ncacn_ip_tcp)

- RPC clients can set the auth type to RPC_C_AUTHN_WINNT to use NTLMSSP
- RPC authentication level and RPC signing / sealing
  - The RPC_C_AUTHN_LEVEL_CONNECT authentication level indicates that the RPC connection does not need to be sealed and signed

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPC_C_AUTHN_LEVEL_DEFAULT</td>
<td>0x00</td>
<td>Same as RPC_C_AUTHN_LEVEL_CONNECT</td>
</tr>
<tr>
<td>RPC_C_AUTHN_LEVEL_NONE</td>
<td>0x01</td>
<td>No authentication.</td>
</tr>
<tr>
<td>RPC_C_AUTHN_LEVEL_CONNECT</td>
<td>0x02</td>
<td>Authenticated the credentials of the client and server.</td>
</tr>
<tr>
<td>RPC_C_AUTHN_LEVEL_CALL</td>
<td>0x03</td>
<td>Same as RPC_C_AUTHN_LEVEL_PKT.</td>
</tr>
<tr>
<td>RPC_C_AUTHN_LEVEL_PKT</td>
<td>0x04</td>
<td>Same as RPC_C_AUTHN_LEVEL_CONNECT but also prevents replay attacks.</td>
</tr>
<tr>
<td>RPC_C_AUTHN_LEVEL_PKT_INTEGRITY</td>
<td>0x05</td>
<td>Same as RPC_C_AUTHN_LEVEL_PKT but also verifies that none of the data transferred between the client and server has been modified.</td>
</tr>
<tr>
<td>RPC_C_AUTHN_LEVEL_PKT_PRIVACY</td>
<td>0x06</td>
<td>Same as RPC_C_AUTHN_LEVEL_PKT_INTEGRITY but also ensures that the data transferred can only be seen unencrypted by the client and the server.</td>
</tr>
</tbody>
</table>
NTLM Relay to DCOM

- DCOM utilizes MS-RPC (ncacn_ip_tcp) as its underlying communication protocol to enable remote COM object communication.
- Signing and sealing are not force enabled on DCOM servers.
- DCOM clients can set the RPC authentication level to RPC_C_AUTHN_LEVEL_CONNECT to avoid signing and sealing, which can disable the protection for NTLM relay.
- DCOM clients communicate with RPC servers using port 135 and a dynamic port assigned by EPM (endpoint mapper), both connections require NTLM authentication.
NTLM Relay to DCOM

- MMC20.Application COM object
  - CLSID 49B2791A-B1AE-4C90-9B8E-E860BA07F889
  - has the Document.ActiveView.ExecuteShellCommand method which support to execute commands remotely
NTLM Relay to DCOM – Exchange Server RCE

Exchange-1

Exchange-1$

Member of

Exchange Trusted Subsystem
local administrator → all Exchange Servers

Attacker

Exchange-A$ over DCOM

MMC20.Application
Document.ActiveView.
ExecuteShellCommand(calc)

Member of

Exchange-2

DCOM

Windows
NTLM relay to DCOM – Exchange Server RCE

DEMO: https://youtu.be/ABylzLx7RiQ
Patch – CVE-2021-26414

● The patch for this vulnerability was released on Patch Tuesday in June 2021.
  ○ The minimum authentication level required by DCOM is set to \RPC_C_AUTHN_LEVEL_PKT_INTEGRITY
  ○ But this patch is not enabled by default, customers need to manually set “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole\AppCompat
    RequireIntegrityActivationAuthenticationLevel = 1” to active the patch
● June 2022, Microsoft released a security update to enable the patch by default, but still with the ability to disable it using the registry key.
● March 2023, the patch is enabled by default with no ability to disable it.
Privilege Escalation to Domain Admin

PrivExchange (fixed in 2019)

- Exchange EWS has a feature which can make it authenticate to an attacker with the Exchange machine account
- The Exchange machine account is a member of the Exchange Windows Permissions group
- The Exchange Windows Permissions group has WriteDACL access on the Domain object in Active Directory, an attackers can use these privileges to grant himself DCSync rights
- NTLM relay from HTTP to LDAP to escalate from a mailbox user to Domain Admin
New Attack Path

Attacker → RCE → LAB\Exchange1$ → Member Of (Exchange Trusted Subsystem)

Member Of → Add Member (Exchange Windows Permissions)
New Attack Path

Exchange Windows Permission group has privileges to create new OUs in the Active Directory.

The attacker can create a new OU and have full control on this OU.
New Attack Path

The attacker can move arbitrary User / Computer object (except adminCount=1) to the newly created OU
New Attack Path

The attacker can set arbitrary ACEs with \textit{InheritanceType set to All} on the newly created OU, these ACEs will inherit down to all descendent objects.

\textbf{Organizational Unit}

\textbf{EvilOU}

\textbf{Create a EvilOU}
GenericAll
\textit{InheritanceType: All}

\textbf{Moved to the EvilOU}

\textbf{Attacker}

\textbf{Victim}

The victim will inherit ACEs from the EvilOU

GenericAll

Take over users / computers with ShadowCredentials / RBCD attack
RCE on Domain Controllers

Domain Controller computers don't set adminCount by default

Move domain controller computers to the EvilOU

Set GenericAll on domain controller computers

RBCD / ShadowCredentials attack

RCE on domain controllers

Domain must have at least one DC running at least Windows 2012
RCE on Domain Controllers

DEMO: https://youtu.be/GsTfIAw5WFY
New Attack Path

The attacker can add himself to Group Policy Creator Owners group to gain privileges to create new GPOs.
New Attack Path

The attacker can create a new GPO and link it to the newly create OU, the GPO will take effect on the objects in the OU.

- Attacker
  - Create a EvilOU
  - Add a Computer / User Immediate Task to execute commands
  - Create a EvilGPO
  - Link to Group Policy Object

- EvilGPO
  - Execute commands on the victim computer as SYSTEM / the logon user
  - Wait for Group Policy update

- EvilOU
  - Moved to the EvilOU
  - Organizational Unit

- Victim

This attack path can also lead to remote commands execution as SYSTEM on domain controllers. No requirement for the domain controller version.
NTLM Relay

Combined with NTLM Relay?

- Need another vulnerability to trigger NTLM authentication of Exchange machine account over HTTP(s)
  - You can also perform RBCD attack on Exchange Servers, but this attack path can help you escalate to Domain Admin

- NTLM relay from HTTP to LDAP (just like what PrivExchange did) to add the attacker to the following high-privileged groups
  - Exchange Trusted Subsystem
  - Group Policy Creator Owners
Won’t Fix

- Microsoft won’t fix this privilege escalation method
- Apply Active Directory split permissions model (not enabled by default) to Exchange can protect your Active Directory

Exchange Organization

Specify the name for this Exchange organization:
First Organization

Apply Active Directory split permissions security model to the Exchange organization

The Active Directory split permissions security model is typically used by large organizations that completely separate the responsibility for the management of Exchange and Active Directory among different groups of people. Applying this security model removes the ability for Exchange servers and administrators to create Active Directory objects such as users, groups, and contacts. The ability to manage non-Exchange attributes on those objects is also removed.

You shouldn’t apply this security model if the same person or group manages both Exchange and Active Directory. Click ‘?’ for more information.
Conclusion & Takeaways

Vulnerabilities

● NTLM relay attack surface in Exchange Server cluster, attackers can achieve arbitrary mailbox takeover / remote code execution on your Exchange Servers with only a normal domain user / machine account.
● Privilege escalation methods from Exchange to Domain Admin that still works up to now and won’t be fixed.

Mitigations

● Enable Extended Protection to mitigate NTLM relay attack surface on Exchange Server.
● Keep your Exchange Servers and Windows Servers they are running on up-to-date.
● Apply Active Directory split permissions model to mitigate privilege escalation methods.
THANK YOU!

Tianze Ding (@D1iv3)