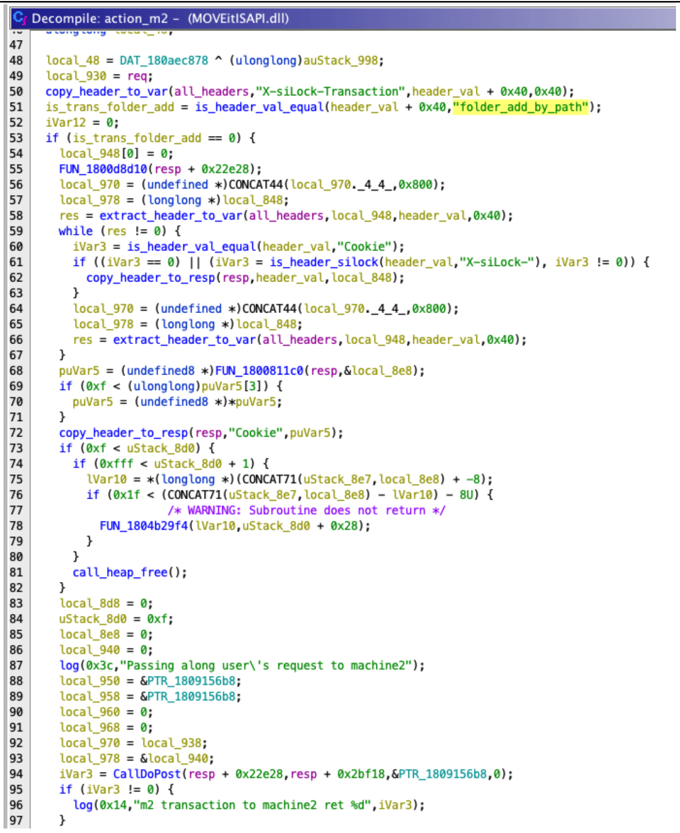
CYBER THREAT ALERT

MOVEit Transfer Vulnerability Exploited

(CVE- 2023-34362)

Summary	A well-known secure file transfer software, MOVEit is the center of high-profile hack that has been discovered recently as “Zero-Day MOVEit Transfer Vulnerability”. This potential vulnerability would lead to escalated privileges and potential unauthorized access to the environments. The method of attack is an SQL Injection to an unpatched MOVEit servers that allow threat actor to gain access and execute arbitrary code remotely. Hence, it is important to patch the bug with the latest security updates updated by Progress Software to block the exploitation attempts.
Technical Summary	A newly discovered LEMURLOOT webshell are being used by threat actors to execute the MOVEit Transfer vulnerability. This web shell masquerade filenames as “human.aspx”, which is a legitimate component in the MOVEit Transfer software. Before interacting with the LEMURLOOT webshell, a number of POST requests were made to the genuine “guestaccess.aspx” file, indicating SQL injection attacks were targeted at that particular file. LEMURLOOT also introduced a file named “human2.aspx” on the targeted device that acts as a backdoor for malicious commands and exploitation.
Technical Analysis	The function “SetAllSessionVarsFromHeaders()” from the MOVEit Transfer had a restriction that only localhost is allowed to route. The threat actor exploit the “/moveitisapi/moveitisapi.dll?action=m2”path which leads to the function at 0x180080920, dubbed action_m2. This function is responsible for parsing requests that contain the “action=m2” request parameter. The “action_m2” function takes requests, and forwards those requests on to the “machine2.aspx” endpoint only if the passed in header “X-siLock-Transaction” is equal to “folder_add_by_path”.



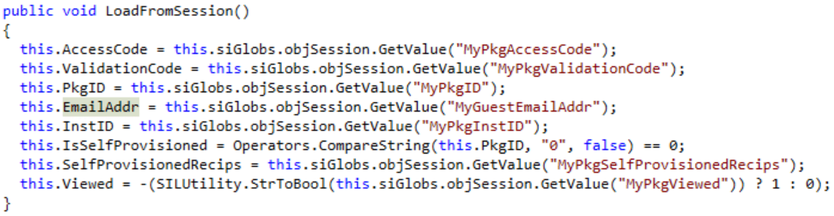
```

  local_48 = DAT_180aec878 ^ (ulonglong)auStack_998;
  local_930 = req;
  copy_header_to_var(all_headers,"X-siLock-Transaction",header_val + 0x40,0x40);
  is_trans_folder_add = is_header_val_equal(header_val + 0x40,"folder_add_by_path");
  iVar12 = 0;
  if (is_trans_folder_add == 0) {
    local_948[0] = 0;
    FUN_1800d8d10(resp + 0x22e28);
    local_970 = (undefined *)CONCAT44(local_970._4_4_,0x800);
    local_978 = (longlong *)local_848;
    res = extract_header_to_var(all_headers,local_948,header_val,0x40);
    while (res != 0) {
      iVar3 = is_header_val_equal(header_val,"Cookie");
      if ((iVar3 == 0) || (iVar3 = is_header_silock(header_val,"X-siLock-"), iVar3 != 0)) {
        copy_header_to_resp(resp,header_val,local_848);
      }
      local_970 = (undefined *)CONCAT44(local_970._4_4_,0x800);
      local_978 = (longlong *)local_848;
      res = extract_header_to_var(all_headers,local_948,header_val,0x40);
    }
    puVar5 = (undefined8 *)FUN_1800811c0(resp,&local_8e8);
    if (0xf < (ulonglong)puVar5[3]) {
      puVar5 = (undefined8 *)*puVar5;
    }
    copy_header_to_resp(resp,"Cookie",puVar5);
    if (0xf < uStack_8d0) {
      if (0xfff < uStack_8d0 + 1) {
        lVar10 = *(longlong *)(CONCAT71(uStack_8e7,local_8e8) + -8);
        if (0x1f < (CONCAT71(uStack_8e7,local_8e8) - lVar10) - 8U) {
                    /* WARNING: Subroutine does not return */
          FUN_1804b29f4(lVar10,uStack_8d0 + 0x28);
        }
      }
      call_heap_free();
    }
    local_8d8 = 0;
    uStack_8d0 = 0xf;
    local_8e8 = 0;
    local_940 = 0;
    log(0x3c,"Passing along user\'s request to machine2");
    local_950 = &PTR_1809156b8;
    local_958 = &PTR_1809156b8;
    local_960 = 0;
    local_968 = 0;
    local_970 = local_938;
    local_978 = &local_940;
    iVar3 = CallDoPost(resp + 0x22e28,resp + 0x2bf18,&PTR_1809156b8,0);
    if (iVar3 != 0) {
      log(0x14,"m2 transaction to machine2 ret %d",iVar3);
    }
```

A bug was found within those functions that the threat actor can trick the function to pass the request onto “machine2.aspx”. With entry to machine2.aspx, the “SetAllSessionVarsFromHeaders()” is able to pass a transaction of “session_setvars”.

Then, the “X-siLock-SessVar”will set the corresponding variable of the session in use to the arbitrary value such as “sysadmin”. This will provide access to set of many variables loaded in code paths but not the full sysadmin role.

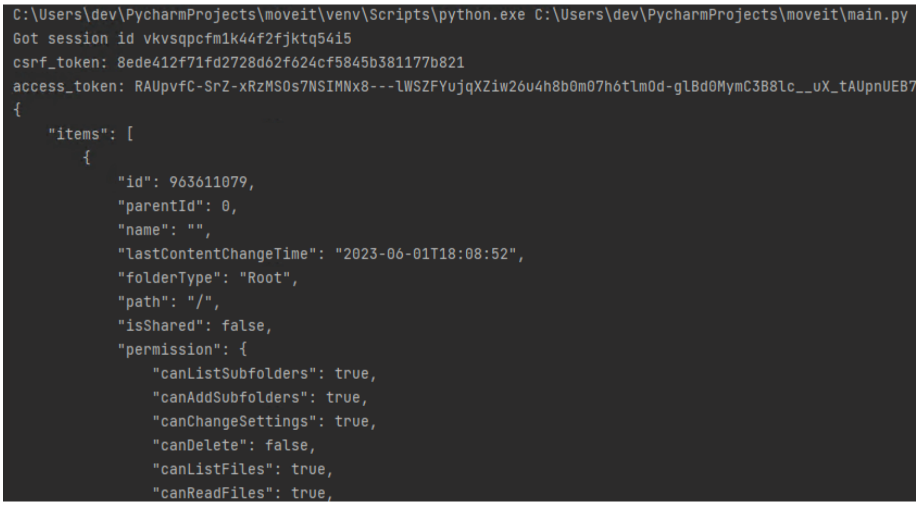
The threat actor will then manipulate the “guestaccess.aspx” handler in “SILGuestAccess”. The main function calls “this.m_pkginfo.LoadFromSession()”, which sets variables from session variables that we can now influence with “session_setvars”.



```
public void LoadFromSession()
{
  this.AccessCode = this.siGlobs.objSession.GetValue("MyPkgAccessCode");
  this.ValidationCode = this.siGlobs.objSession.GetValue("MyPkgValidationCode");
  this.PkgID = this.siGlobs.objSession.GetValue("MyPkgID");
  this.EmailAddr = this.siGlobs.objSession.GetValue("MyGuestEmailAddr");
  this.InstID = this.siGlobs.objSession.GetValue("MyPkgInstID");
  this.IsSelfProvisioned = Operators.CompareString(this.PkgID, "0", false) == 0;
  this.SelfProvisionedRecips = this.siGlobs.objSession.GetValue("MyPkgSelfProvisionedRecips");
  this.Viewed = -(SILUtility.StrToBool(this.siGlobs.objSession.GetValue("MyPkgViewed")) ? 1 : 0);
}
```

The part of the query can be exploit using SQL injection attack at "SelfProvisionedRecips" variable where it is split by a comma. The threat actor can inject SQL statement by avoid having commas to continue the execution. The threat actor can work around needing commas by reusing the SQL injection several times to do sequential statements such as INSERT then UPDATE. With all this information combined, the threat actor can read and write any data within the MOVEit database.

The threat actor then gain elevated permission by attacking the "/api/v1/auth/token"endpoint that is handled by "MOVEit.DMZ.WebAPI". The MOVEit Transfer application will reach out to the URL in the "amurl" field to retrieve the certificate that matches the given x5t signature to extract and validate that the JWT was in fact signed by the identity provider. The threat actor can use the SQL Injection from previous path to configure the database to trust their identity provider URL and inject an external token for the builtin sysadmin user. The threat actor then use SQL Injection to allow the sysadmin user to be able to login from any IP address.

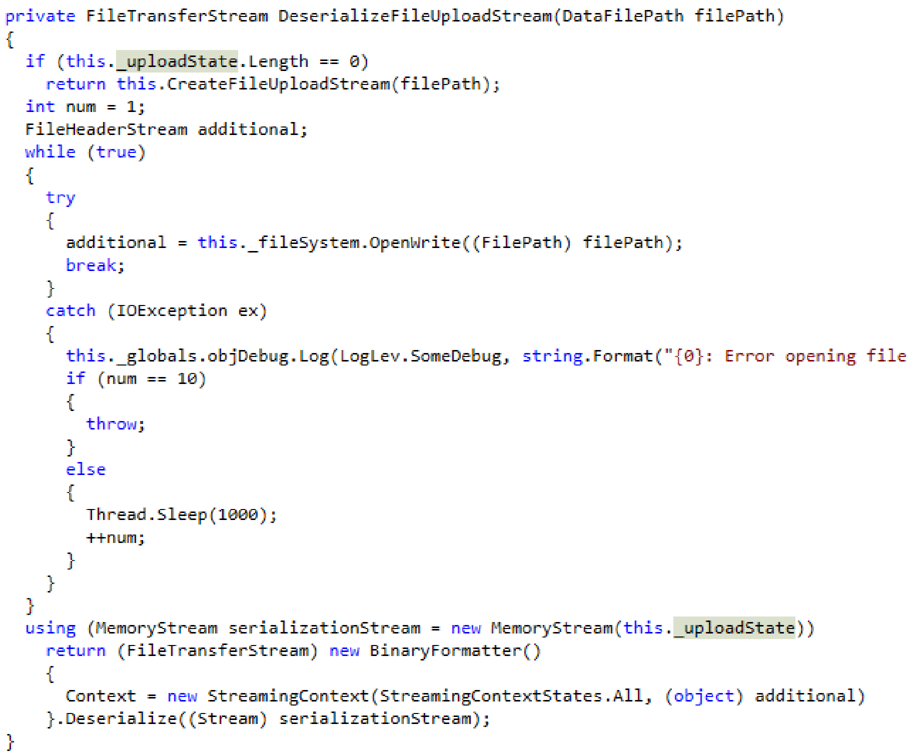


```
C:\Users\dev\PycharmProjects\moveit\venv\Scripts\python.exe C:\Users\dev\PycharmProjects\moveit\main.py
Got session id vkvsqpcfm1k44f2fjktq54i5
csrf_token: 8ede412f71fd2728d62f624cf5845b381177b821
access_token: RAUpvfC-SrZ-xRzMSOs7NSIMNx8---lWSZFYujqXZiw26u4h8b0m07h6tlm0d-glBd0MymC3B8lc__uX_tAUpnUEB7
{
    "items": [
        {
            "id": 963611079,
            "parentId": 0,
            "name": "",
            "lastContentChangeTime": "2023-06-01T18:08:52",
            "folderType": "Root",
            "path": "/",
            "isShared": false,
            "permission": {
                "canListSubfolders": true,
                "canAddSubfolders": true,
                "canChangeSettings": true,
                "canDelete": false,
                "canListFiles": true,
                "canReadFiles": true,
```

The threat actor then can obtain access token for the sysadmin user and use it to list files they have access to.

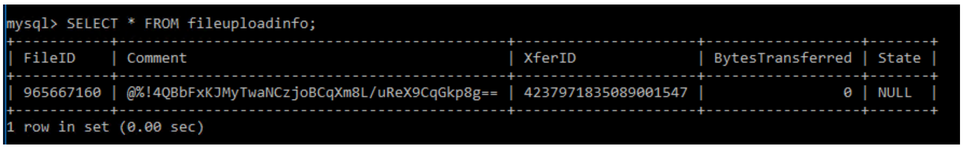
As for the remote code execution, the threat actor was observed exploiting the "/api/v1/folders","/api/v1/folders/<folder_id>/files?uploadType=resumable" and "/api/v1/folders/<folder_id>/files?uploadType=resumable&fileId=<file_id>" endpoints.

The next target for the threat actor would be the "._uploadState" variable. Examining where that variable is referenced in the .NET DLL, we can observe that the function DeserializeFileUploadStream() uses it to create a MemoryStream object and then immediately uses it in a call to BinaryFormatter().Deserialize(). This is a classic .NET deserialization vulnerability. Normally, the uploadState variable would not be under attacker influence, but because it is an SQL injection, the field from which that variable is set can be influence.



```
private FileTransferStream DeserializeFileUploadStream(DataFilePath filePath)
{
  if (this._uploadState.Length == 0)
    return this.CreateFileUploadStream(filePath);
  int num = 1;
  FileHeaderStream additional;
  while (true)
  {
    try
    {
      additional = this._fileSystem.OpenWrite((FilePath) filePath);
      break;
    }
    catch (IOException ex)
    {
      this._globals.objDebug.Log(LogLev.SomeDebug, string.Format("{0}: Error opening file
      if (num == 10)
      {
        throw;
      }
      else
      {
        Thread.Sleep(1000);
        ++num;
      }
    }
  }
  using (MemoryStream serializationStream = new MemoryStream(this._uploadState))
    return (FileTransferStream) new BinaryFormatter()
    {
      Context = new StreamingContext(StreamingContextStates.All, (object) additional)
    }.Deserialize((Stream) serializationStream);
}
```

The state value of the database was changed from NULL to the state value of the base64 encoded serialized .NET payload.



```
mysql> SELECT * FROM fileuploadinfo;
+-----------+---------------------------------------+---------------------+-----------------+-------+
| FileID    | Comment                               | XferID              | BytesTransferred | State |
+-----------+---------------------------------------+---------------------+-----------------+-------+
| 965667160 | @%!4QBbFxKJMyTwaNCzjoBCqXm8L/uReX9CqGkp8g== | 4237971835089001547 |               0 | NULL  |
+-----------+---------------------------------------+---------------------+-----------------+-------+
1 row in set (0.00 sec)
```

The payload can then be initiated and upload to execute remote code execution.

Indicator of Compromise (IOC)	<u>IP Address</u> - 5.252.23.116 - 5.252.25.88 - 84.234.96.104 - 89.39.105.108 - 138.197.152.201 - 148.113.152.144 - 198.12.76.214 - 209.97.137.33 - 209.222.103.170 - 104.194.222.107 - 146.0.77.141 - 146.0.77.155 - 146.0.77.183 - 162.244.34.26 - 162.244.35.6 - 179.60.150.143 - 185.104.194.156 - 185.104.194.24 - 185.104.194.40 - 185.117.88.17 - 185.162.128.75 - 185.174.100.215 - 185.174.100.250 - 185.181.229.240 - 185.181.229.73 - 185.183.32.122 - 185.185.50.172 - 188.241.58.244 - 193.169.245.79 - 194.33.40.103 - 194.33.40.104 - 194.33.40.164

	- 198.27.75.110 - 206.221.182.106 - 209.127.116.122 - 209.127.4.22 - 45.227.253.133 - 45.227.253.147 - 45.227.253.50 - 45.227.253.6 - 45.227.253.82 - 45.56.165.248 - 5.149.248.68 - 5.149.250.74 - 5.149.250.92 - 5.188.86.114 - 5.188.86.250 - 5.188.87.194 - 5.188.87.226 - 5.188.87.27 - 5.34.180.205 - 62.112.11.57 - 62.182.82.19 - 62.182.85.234 - 66.85.26.215 - 66.85.26.234 - 66.85.26.248 - 79.141.160.78 - 79.141.160.83 - 84.234.96.31 - 89.39.104.118 - 91.202.4.76 - 91.222.174.95 - 91.229.76.187 - 93.190.142.131

Folder Path

- C:\Windows\TEMP\[random]\[random].cmdline

HTTP Request

- POST /moveitisapi/moveitisapi.dll
- POST /guestaccess.aspx
- POST /api/v1/folders/[random]/files
- GET /human2.aspx

User Agent

- Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/105.0.5195.102+Safari/537.36

Domain

- dojustit[.]mooo[.]com
- http[:]//hiperfdhaus[.]com
- http://jirostrogud[.]com
- http://qweastradoc[.]com
- http://qweastradoc[.]com/gate.php
- http://connectzoomdownload[.]com/download/ZoomInstaller.exe
- http://zoom[.]voyage/download/Zoom.exe
- http[:]//guerdofest[.]com/gate.php

SHA 256 Hash

- 0b3220b11698b1436d1d866ac07cc90018e59884e91a8cb71ef8924309f1e0e9
- 110e301d3b5019177728010202c8096824829c0b11bb0dc0bff55547ead18286
- 1826268249e1ea58275328102a5a8d158d36b4fd312009e4a2526f0bfbc30de2
- 2ccf7e42afd3f6bf845865c74b2e01e2046e541bb633d037b05bd1cdb296fa59
- 58ccfb603cdc4d305fddd52b84ad3f58ff554f1af4d7ef164007cb8438976166

	- 98a30c7251cf622bd4abce92ab527c3f233b817a57519c2dd2bf8e3d3ccb7db8 - a8f6c1ccba662a908ef7b0cb3cc59c2d1c9e2cbbe1866937da81c4c616e68986 - b5ef11d04604c9145e4fe1bedaeb52f2c2345703d52115a5bf11ea56d7fb6b03 - cec425b3383890b63f5022054c396f6d510fae436041add935cd6ce42033f621 - ed0c3e75b7ac2587a5892ca951707b4e0dd9c8b18aaf8590c24720d73aa6b90c - 0b3220b11698b1436d1d866ac07cc90018e59884e91a8cb71ef8924309f1e0e9 - 110e301d3b5019177728010202c8096824829c0b11bb0dc0bff55547ead18286 - 1826268249e1ea58275328102a5a8d158d36b4fd312009e4a2526f0bfbc30de2 - 2ccf7e42afd3f6bf845865c74b2e01e2046e541bb633d037b05bd1cdb296fa59 - 58ccfb603cdc4d305fddd52b84ad3f58ff554f1af4d7ef164007cb8438976166 - 98a30c7251cf622bd4abce92ab527c3f233b817a57519c2dd2bf8e3d3ccb7db8 - a8f6c1ccba662a908ef7b0cb3cc59c2d1c9e2cbbe1866937da81c4c616e68986 - b5ef11d04604c9145e4fe1bedaeb52f2c2345703d52115a5bf11ea56d7fb6b03 - cec425b3383890b63f5022054c396f6d510fae436041add935cd6ce42033f621 - ed0c3e75b7ac2587a5892ca951707b4e0dd9c8b18aaf8590c24720d73aa6b90c

	File Name - human2.aspx - human2.aspx.lnk - huamn2.aspx.[random].compiled
Affected Version	- MOVEit Transfer 2023.0.0 (15.0) - MOVEit Transfer 2022.1.x (14.1) - MOVEit Transfer 2022.0.x (14.0) - MOVEit Transfer 2021.1.x (13.1) - MOVEit Transfer 2021.0.x (13.0) - MOVEit Transfer 2020.1.x (12.1) - MOVEit Transfer 2020.0.x (12.0) or older
Fixed Version	- MOVEit Transfer 2023.0.2 (15.0.2) - MOVEit Transfer 2022.1.6 (14.1.6) - MOVEit Transfer 2022.0.5 (14.0.5) - MOVEit Transfer 2021.1.5 (13.1.5) - MOVEit Transfer 2021.0.7 (13.0.7) - Special Patch (For MOVEit Transfer 12.1) - Must upgrade to supported version (For MOVEit 12.0 and older)
Conclusion	This attack is a sophisticated attack that most organization cannot prevent due to it being a Zero-Day attack and the vulnerability has just been exploited. However, this does not mean that organization cannot minimize the damage of this attack. Organization should focus on building resilience and mitigating the risks, since this attack have been tried and tested more than many have realize.
Recommendation	FIRMUS recommend organization to (by order) - Disable all HTTP and HTTPs traffic to the MOVEit environment - Delete Unauthorized Files and User Accounts - Review any new files created in “C:\MOVEitTransfer\wwwroot\ directory” and “C:\Windows\TEMP\[random]\” directory with a file extension of [.]cmdline - Remove any unauthorized user accounts - Remove all active sessions

	- Review logs for unexpected download from unknown IP - Review logs for large number of files downloaded - Review IIS logs for “GET /human2.aspx” - Review Azure logs for unauthorized access to Azure Blob Storage Keys - Reset Service Account Credentials - Apply the latest MOVEit patch from Progress Software - Enable all HTTP and HTTPs traffic back to MOVEit environment - Monitor network, endpoints and logs for IoCs mentioned above
Additional Best Security Practices	FIRMUS recommend organization to - Update network firewall rules - Update remote access policies - Allow inbound access from trusted entities only - Enable multi-factor authentication
Article Source	- https://www.mandiant.com/resources/blog/zero-day-moveit-data-theft - https://www.techrepublic.com/article/zero-day-moveit-vulnerability/ - https://www.rapid7.com/blog/post/2023/06/01/rapid7-observed-exploitation-of-critical-moveit-transfer-vulnerability/ - https://www.bleepingcomputer.com/news/security/exploit-released-for-moveit-rce-bug-used-in-data-theft-attacks/ - https://nvd.nist.gov/vuln/detail/CVE-2023-34362 - https://www.horizon3.ai/moveit-transfer-cve-2023-34362-deep-dive-and-indicators-of-compromise/ - https://community.progress.com/s/article/MOVEit-Transfer-Critical-Vulnerability-31May2023