linuxprivchecker / linuxprivchecker.py

jtpereyda add ps -w option to support systems where default width is concatenated 000fc51 · 4 years ago

Executable File · 649 lines (518 loc) · 36.3 KB

Code Blame Raw

```
#!/usr/bin/python

"""
###############################################################################################################
## [Title]: linuxprivchecker.py -- a Linux Privilege Escalation Check Script
## [Original Author]: Mike Czumak (T_v3rn1x) -- @SecuritySift
##  [Maintainer]: Michael Contino -- @Sleventyeleven
##-------------------------------------------------------------------------------------------------------------
## [Details]:
## This script is intended to be executed locally on a Linux box to enumerate basic system info and
## search for common privilege escalation vectors such as world writable files, misconfigurations,
## passwords and applicable exploits.
##-------------------------------------------------------------------------------------------------------------
## [Modification, Distribution, and Attribution]:
## You are free to modify and/or distribute this script as you wish.  I only ask that you maintain
## author attribution and not attempt to sell it or incorporate it into any commercial offering (as
## worth anything anyway :)
###############################################################################################################
TODO:
Add search for writable and/or missing library files
Add detection and enumeratation for systemd
Add search for accessiable ssh sockets
Add search for ssh keys
Add search for know access tokens
Expand Sudo support to include rules in sudoers.d
Add more high profile exploit checks (ie shellshock)
```

```
27      """
28
29      # conditional import for older versions of python not compatible with subprocess
30      try:
31          import subprocess as sub
32          compatmode = 0  # newer version of python, no need for compatibility mode
33      except ImportError:
34          import os  # older version of python, need to use os instead
35          compatmode = 1
36
37
38      def execute_cmd(cmddict):
39          """
40          Execute Command (execute_cmd)
41          loop through dictionary, execute the commands, store the results, return updated dict
42
43          :param cmddict: Dictionary of commands to execute and results
44          :return: The command Dictionary with the commands results included
45          """
46
47          for item in cmddict:
48              cmd = cmddict[item]["cmd"]
49              if compatmode == 0:  # newer version of python, use preferred subprocess
50                  out, error = sub.Popen([cmd], stdout=sub.PIPE, stderr=sub.PIPE, shell=True).communicate
51                  results = out.split('\n')
52              else:  # older version of python, use os.popen
53                  echo_stdout = os.popen(cmd, 'r')
54                  results = echo_stdout.read().split('\n')
55
56              # write the results to the command Dictionary for each command run
57              cmddict[item]["results"] = results
58
59          return cmddict
60
61
62      def print_results(cmddict):
63          """
64          Print Results (printResults)
65          Print results for each previously executed command, no return value
66
67          :param cmddict: Dictionary of commands to execute and results
68          :return: None
69          """
70
71          for item in cmddict:
72              msg = cmddict[item]["msg"]
```

```
            results = cmddict[item]["results"]
            print "[+] " + msg

            for result in results:
                if result.strip() != "":
                    print "    " + result.strip()
        print


    def enum_system_info():
        """
        Basic System Info (get_system_info)
        Enumerate Basic System Information by executing simple commands than saving the results

        :return: Dictionary of system information results
        """

        print "[*] GETTING BASIC SYSTEM INFO...\n"

        sysinfo = {
            "OS": {"cmd": "cat /etc/issue", "msg": "Operating System", "results": []},
            "KERNEL": {"cmd": "cat /proc/version", "msg": "Kernel", "results": []},
            "HOSTNAME": {"cmd": "hostname", "msg": "Hostname", "results": []}
        }

        sysinfo = execute_cmd(sysinfo)
        print_results(sysinfo)

        return sysinfo


    def enum_network_info():
        """
        Basic Network Info (get_network_info)
        Enumerate Basic Network Information by executing simple commands

        :return: Dictionary of Network information with results
        """

        print "[*] GETTING NETWORKING INFO...\n"

        netinfo = {
            "netinfo": {"cmd": "/sbin/ifconfig -a", "msg": "Interfaces", "results": []},
            "ROUTE": {"cmd": "route", "msg": "Route(s)", "results": []},
            "NETSTAT": {"cmd": "netstat -antup | grep -v 'TIME_WAIT'", "msg": "Netstat", "results": []}
          }
```

```
576                 # import sys for io redirection
```

```
577                 import sys
578
579                 class Logger(object):
580                     def __init__(self):
581                         self.terminal = sys.stdout
582                         self.log = open(args.outfile, 'a')
583
584                     def write(self, message):
585                         self.terminal.write(message)
586                         self.log.write(message)
587                 sys.stdout = Logger()
588
589         except ImportError:
590             print 'Arguments could not be processed, defaulting to print everything'
591             processsearches = True
592
593         # title / formatting
594         bigline = "================================================================================================
595         print bigline
596         print """
597             __    _                  ____       _       ________              __
598            / /   (_)___  __  ___  __/ __ \_____(_)   __/ ____/ /_  ___  _____/ /_____  _____
599           / /   / / __ \/ / / / |/_/ /_/ / ___/ / | / / /   / __ \/ _ \/ ___/ //_/ _ \/ ___/
600          / /___/ / / / / /_/ />  </ ____/ /  / /| |/ / /___/ / / /  __/ /__/ ,< /  __/ /
601         /_____/_/_/ /_/\__,_/_/|_/_/   /_/  /_/ |___/\____/_/ /_/\___/\___/_/|_|\___/_/
602
603         """
604         print bigline
605
606         # Enumerate Basic User Information
607         userinfo = enum_user_info()
608
609         # Enumerate Basic System Information
610         sysinfo = enum_system_info()
611
612         # Enumerate Basic Network Information
613         enum_network_info()
614
615         # Enumerate User History Files
616         enum_user_history_files()
617
618         # Enumerate Basic RC Files
619         enum_rc_files()
620
621         # Enumerate Basic Filesystem Information
```

```
622         driveinfo = enum_filesystem_info()
623 
624         # Enumerate List of all Cron jobs
625         enum_cron_jobs()
626 
627         # Enumerate Package and Process information
628         pkgsandprocs = enum_procs_pkgs()
629 
630         # Enumerate Possible Root/superuser packages or processes
631         enum_root_pkg_proc(pkgsandprocs, userinfo)
632 
633         # Emumerate Available Development Tools
634         devtools = enum_dev_tools()
635 
636         # Enumerate Possible Shell Escapes
637         enum_shell_esapes(devtools)
638 
639         find_likely_exploits(sysinfo, devtools, pkgsandprocs, driveinfo)
640 
641         if processsearches:
642             # Search for Insecure File/Folder Permissions
643             search_file_perms()
644 
645             # Search for files with potential credentials
646             search_file_passwords()
647 
648         print "Finished"
649         print bigline
```