Mithibai College of Arts,
Chauhan Institute of Science &
Amrutben Jivanlal College of Commerce and Economics
(AUTONOMOUS)
Affiliated to University of Mumbai
NAAC Reaccredited 'A' Grade, CGPA : 3.57 (February 2016)
Granted under RUSA for Enhancing Quality & Excellence in select Autonomous Colleges
GRANTED UNDER FIST-DST & STAR COLLEGE SCHEME OF DBT, GOVERNMENT OF INDIA
BEST COLLEGE (2016-17), University of Mumbai

Practical-1 Static code analysis using open source tool Flaw-finder

Date: -12/07/2024 Submission Date: - 19/07/2024

Write-up: -

- ➢ Static code Analysis & Benefits
- ➢ Vulnerability
- ➢ Flaw-finder

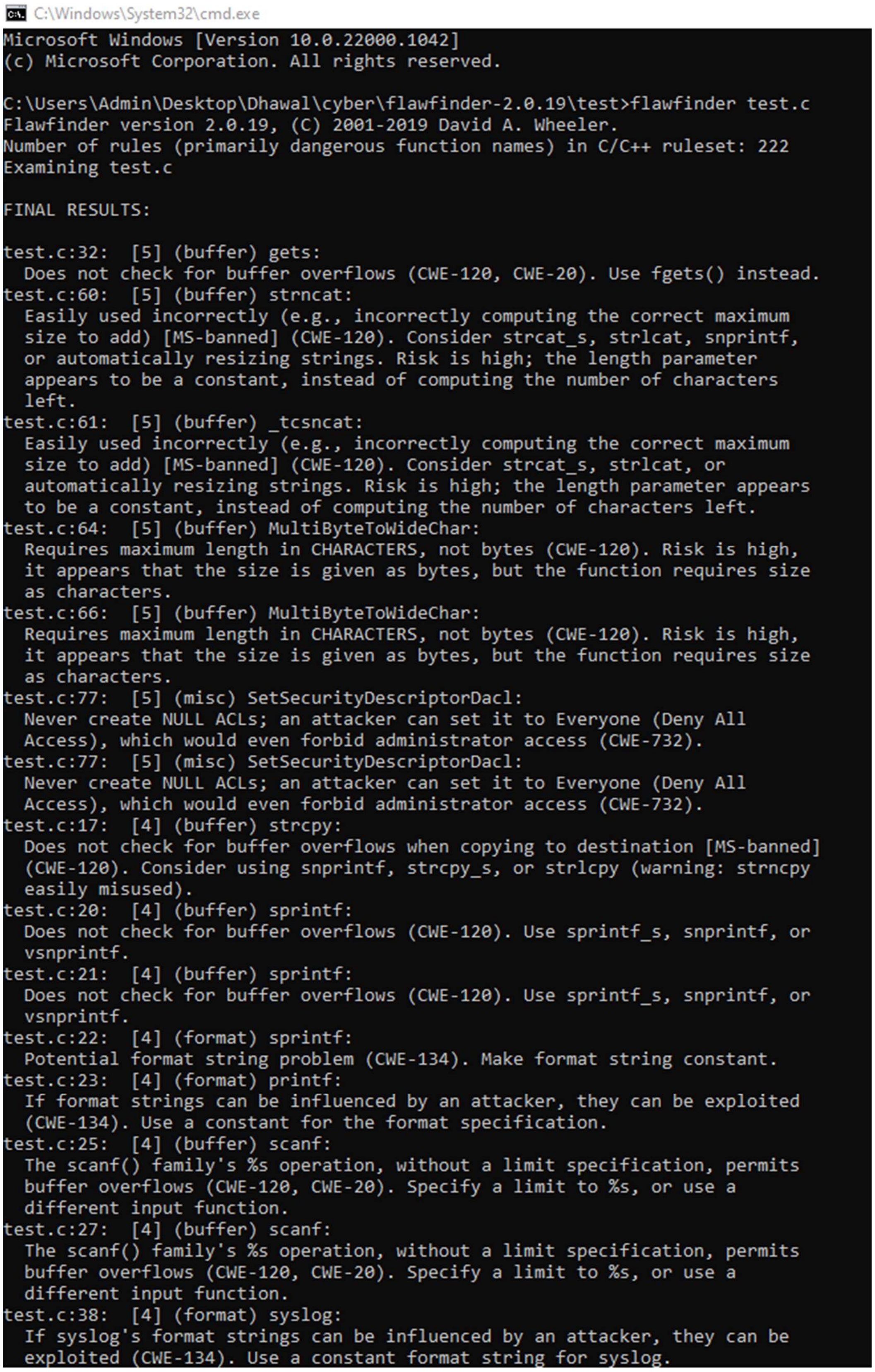
Implement static code analysis using open source tool Flaw-finder for the following:

- Buffer overflow
- String problem
- Race conditions, etc.

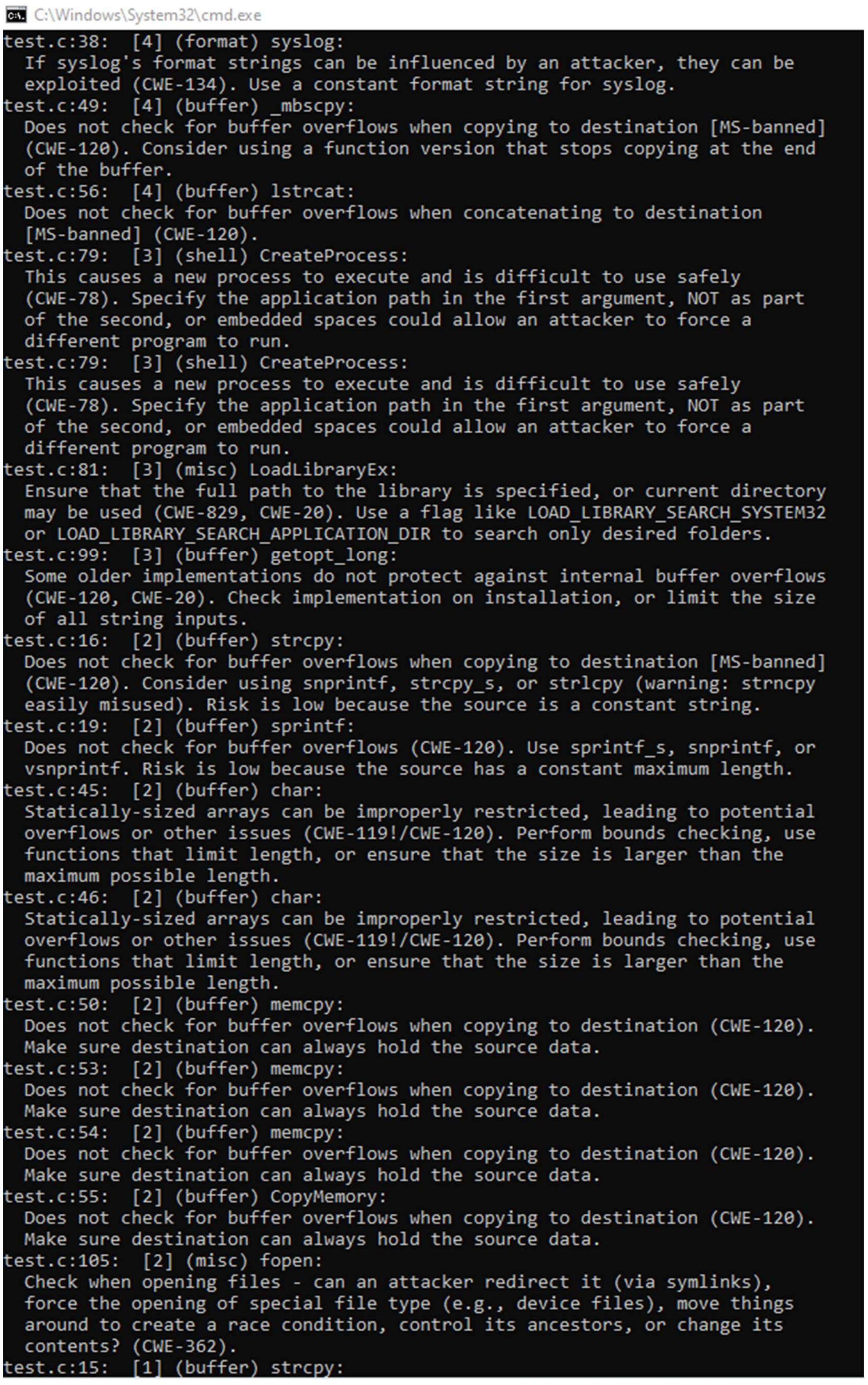
Code /Steps:

1. Download Flaw-finder
2. Set up the path where the Flaw-finder
3. Get the test file for the practical
4. Now open the terminal
5. Then check is the flaw-finder is available or not
6. Then run the command flaw-finder command
   - Flawfinder test.c

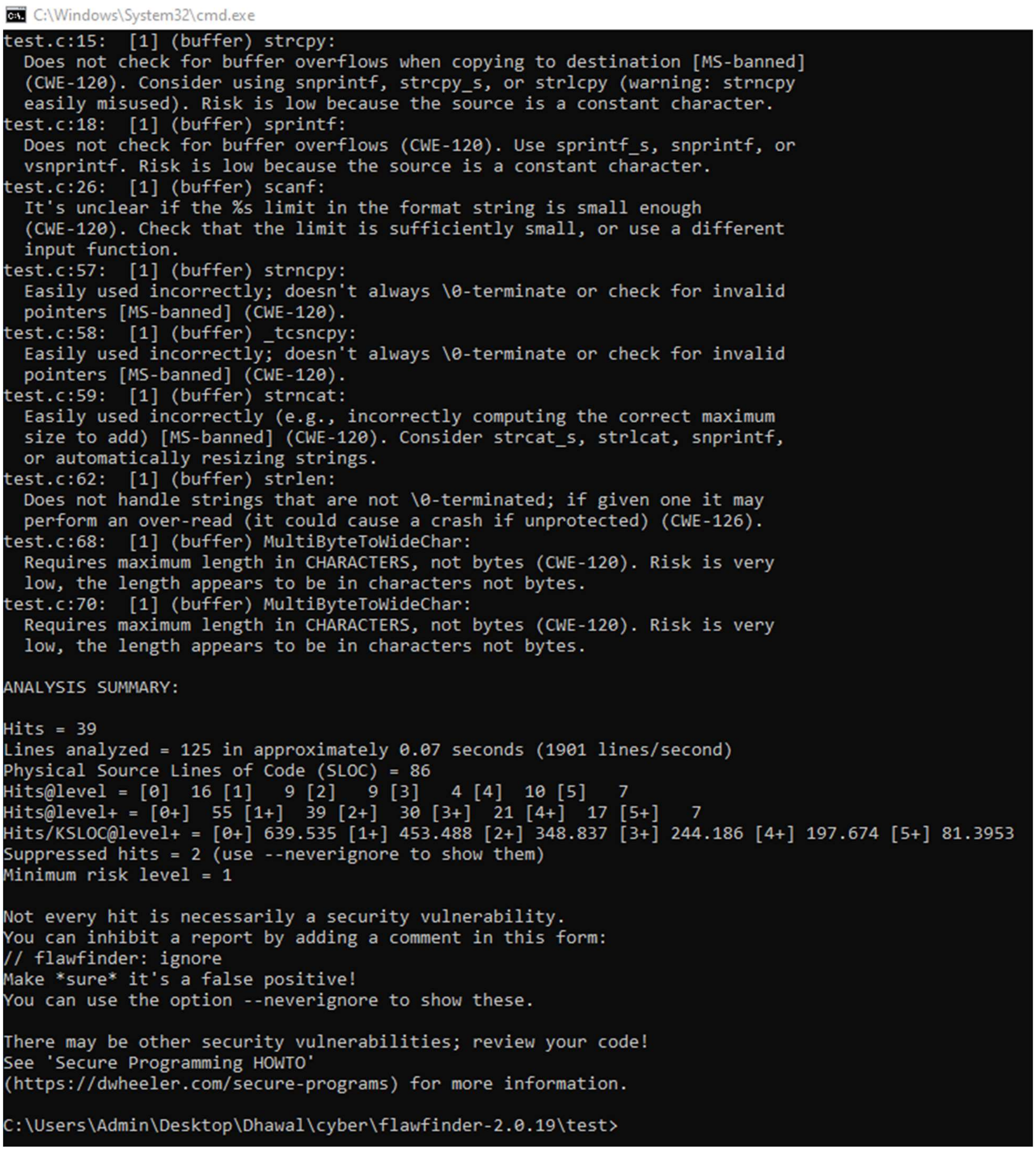
Output:


C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.22000.1042]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\Desktop\Dhawal\cyber\flawfinder-2.0.19\test>flawfinder test.c
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining test.c

FINAL RESULTS:

test.c:32:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
test.c:60:  [5] (buffer) strncat:
  Easily used incorrectly (e.g., incorrectly computing the correct maximum
  size to add) [MS-banned] (CWE-120). Consider strcat_s, strlcat, snprintf,
  or automatically resizing strings. Risk is high; the length parameter
  appears to be a constant, instead of computing the number of characters
  left.
test.c:61:  [5] (buffer) _tcsncat:
  Easily used incorrectly (e.g., incorrectly computing the correct maximum
  size to add) [MS-banned] (CWE-120). Consider strcat_s, strlcat, or
  automatically resizing strings. Risk is high; the length parameter appears
  to be a constant, instead of computing the number of characters left.
test.c:64:  [5] (buffer) MultiByteToWideChar:
  Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is high,
  it appears that the size is given as bytes, but the function requires size
  as characters.
test.c:66:  [5] (buffer) MultiByteToWideChar:
  Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is high,
  it appears that the size is given as bytes, but the function requires size
  as characters.
test.c:77:  [5] (misc) SetSecurityDescriptorDacl:
  Never create NULL ACLs; an attacker can set it to Everyone (Deny All
  Access), which would even forbid administrator access (CWE-732).
test.c:77:  [5] (misc) SetSecurityDescriptorDacl:
  Never create NULL ACLs; an attacker can set it to Everyone (Deny All
  Access), which would even forbid administrator access (CWE-732).
test.c:17:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
test.c:20:  [4] (buffer) sprintf:
  Does not check for buffer overflows (CWE-120). Use sprintf_s, snprintf, or
  vsnprintf.
test.c:21:  [4] (buffer) sprintf:
  Does not check for buffer overflows (CWE-120). Use sprintf_s, snprintf, or
  vsnprintf.
test.c:22:  [4] (format) sprintf:
  Potential format string problem (CWE-134). Make format string constant.
test.c:23:  [4] (format) printf:
  If format strings can be influenced by an attacker, they can be exploited
  (CWE-134). Use a constant for the format specification.
test.c:25:  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification, permits
  buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or use a
  different input function.
test.c:27:  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification, permits
  buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or use a
  different input function.
test.c:38:  [4] (format) syslog:
  If syslog's format strings can be influenced by an attacker, they can be
  exploited (CWE-134). Use a constant format string for syslog.
```



```
test.c:38:  [4] (format) syslog:
  If syslog's format strings can be influenced by an attacker, they can be
  exploited (CWE-134). Use a constant format string for syslog.
test.c:49:  [4] (buffer) _mbscpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using a function version that stops copying at the end
  of the buffer.
test.c:56:  [4] (buffer) lstrcat:
  Does not check for buffer overflows when concatenating to destination
  [MS-banned] (CWE-120).
test.c:79:  [3] (shell) CreateProcess:
  This causes a new process to execute and is difficult to use safely
  (CWE-78). Specify the application path in the first argument, NOT as part
  of the second, or embedded spaces could allow an attacker to force a
  different program to run.
test.c:79:  [3] (shell) CreateProcess:
  This causes a new process to execute and is difficult to use safely
  (CWE-78). Specify the application path in the first argument, NOT as part
  of the second, or embedded spaces could allow an attacker to force a
  different program to run.
test.c:81:  [3] (misc) LoadLibraryEx:
  Ensure that the full path to the library is specified, or current directory
  may be used (CWE-829, CWE-20). Use a flag like LOAD_LIBRARY_SEARCH_SYSTEM32
  or LOAD_LIBRARY_SEARCH_APPLICATION_DIR to search only desired folders.
test.c:99:  [3] (buffer) getopt_long:
  Some older implementations do not protect against internal buffer overflows
  (CWE-120, CWE-20). Check implementation on installation, or limit the size
  of all string inputs.
test.c:16:  [2] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused). Risk is low because the source is a constant string.
test.c:19:  [2] (buffer) sprintf:
  Does not check for buffer overflows (CWE-120). Use sprintf_s, snprintf, or
  vsnprintf. Risk is low because the source has a constant maximum length.
test.c:45:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
test.c:46:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
test.c:50:  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination (CWE-120).
  Make sure destination can always hold the source data.
test.c:53:  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination (CWE-120).
  Make sure destination can always hold the source data.
test.c:54:  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination (CWE-120).
  Make sure destination can always hold the source data.
test.c:55:  [2] (buffer) CopyMemory:
  Does not check for buffer overflows when copying to destination (CWE-120).
  Make sure destination can always hold the source data.
test.c:105:  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move things
  around to create a race condition, control its ancestors, or change its
  contents? (CWE-362).
test.c:15:  [1] (buffer) strcpy:
```



C:\Windows\System32\cmd.exe

```
test.c:15:  [1] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused). Risk is low because the source is a constant character.
test.c:18:  [1] (buffer) sprintf:
  Does not check for buffer overflows (CWE-120). Use sprintf_s, snprintf, or
  vsnprintf. Risk is low because the source is a constant character.
test.c:26:  [1] (buffer) scanf:
  It's unclear if the %s limit in the format string is small enough
  (CWE-120). Check that the limit is sufficiently small, or use a different
  input function.
test.c:57:  [1] (buffer) strncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for invalid
  pointers [MS-banned] (CWE-120).
test.c:58:  [1] (buffer) _tcsncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for invalid
  pointers [MS-banned] (CWE-120).
test.c:59:  [1] (buffer) strncat:
  Easily used incorrectly (e.g., incorrectly computing the correct maximum
  size to add) [MS-banned] (CWE-120). Consider strcat_s, strlcat, snprintf,
  or automatically resizing strings.
test.c:62:  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated; if given one it may
  perform an over-read (it could cause a crash if unprotected) (CWE-126).
test.c:68:  [1] (buffer) MultiByteToWideChar:
  Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is very
  low, the length appears to be in characters not bytes.
test.c:70:  [1] (buffer) MultiByteToWideChar:
  Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is very
  low, the length appears to be in characters not bytes.

ANALYSIS SUMMARY:

Hits = 39
Lines analyzed = 125 in approximately 0.07 seconds (1901 lines/second)
Physical Source Lines of Code (SLOC) = 86
Hits@level = [0]  16 [1]   9 [2]   9 [3]   4 [4]  10 [5]   7
Hits@level+ = [0+]  55 [1+]  39 [2+]  30 [3+]  21 [4+]  17 [5+]   7
Hits/KSLOC@level+ = [0+] 639.535 [1+] 453.488 [2+] 348.837 [3+] 244.186 [4+] 197.674 [5+] 81.3953
Suppressed hits = 2 (use --neverignore to show them)
Minimum risk level = 1

Not every hit is necessarily a security vulnerability.
You can inhibit a report by adding a comment in this form:
// flawfinder: ignore
Make *sure* it's a false positive!
You can use the option --neverignore to show these.

There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.

C:\Users\Admin\Desktop\Dhawal\cyber\flawfinder-2.0.19\test>
```