ATTACK DEFENSE

by PentesterAcademy

Name	Crass: Hunting Sensitive Information
URL	https://www.attackdefense.com/challengedetails?cid=2151
Type	DevSecOps Basics: Sensitive Information Scan

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

The CRASS tool is used to find sensitive information from the source code of the application. It uses basic Unix commands to filter and search information.

A Kali CLI machine (kali-cli) is provided to the user with CRASS installed on it. The source code for three sample applications is provided in the home directory of the root user.

Objective: Use the CRASS utility to find sensitive information in the applications!

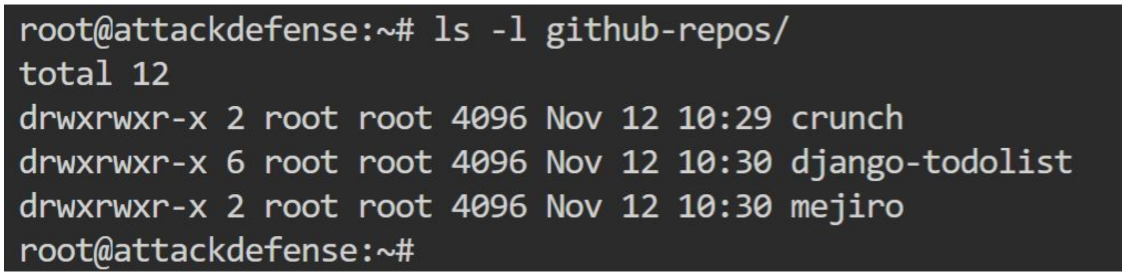
Instructions:

- The source code of applications is provided at /root/github-repos

Solution

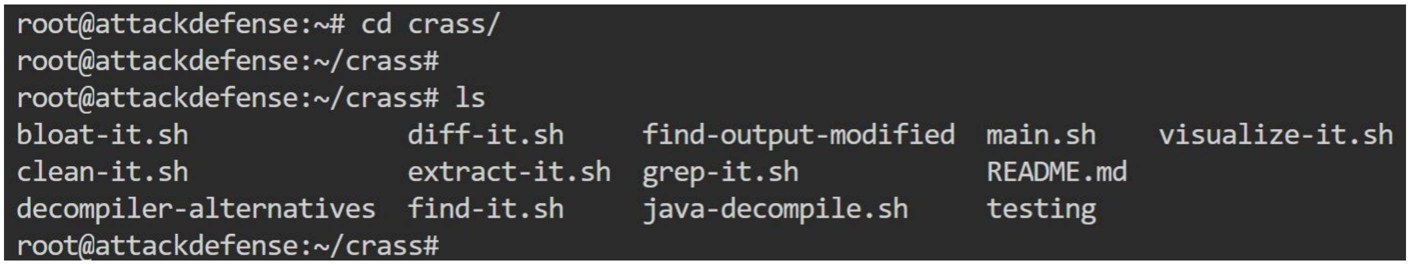
Step 1: Check the provided applications.

Command: ls -l github-repos/


```
root@attackdefense:~# ls -l github-repos/
total 12
drwxrwxr-x 2 root root 4096 Nov 12 10:29 crunch
drwxrwxr-x 6 root root 4096 Nov 12 10:30 django-todolist
drwxrwxr-x 2 root root 4096 Nov 12 10:30 mejiro
root@attackdefense:~#
```

Step 2: Navigate to the crass directory and check the files available.

Commands:
cd crass/
ls


```
root@attackdefense:~# cd crass/
root@attackdefense:~/crass#
root@attackdefense:~/crass# ls
bloat-it.sh              diff-it.sh     find-output-modified  main.sh    visualize-it.sh
clean-it.sh              extract-it.sh  grep-it.sh            README.md
decompiler-alternatives  find-it.sh     java-decompile.sh     testing
root@attackdefense:~/crass#
```

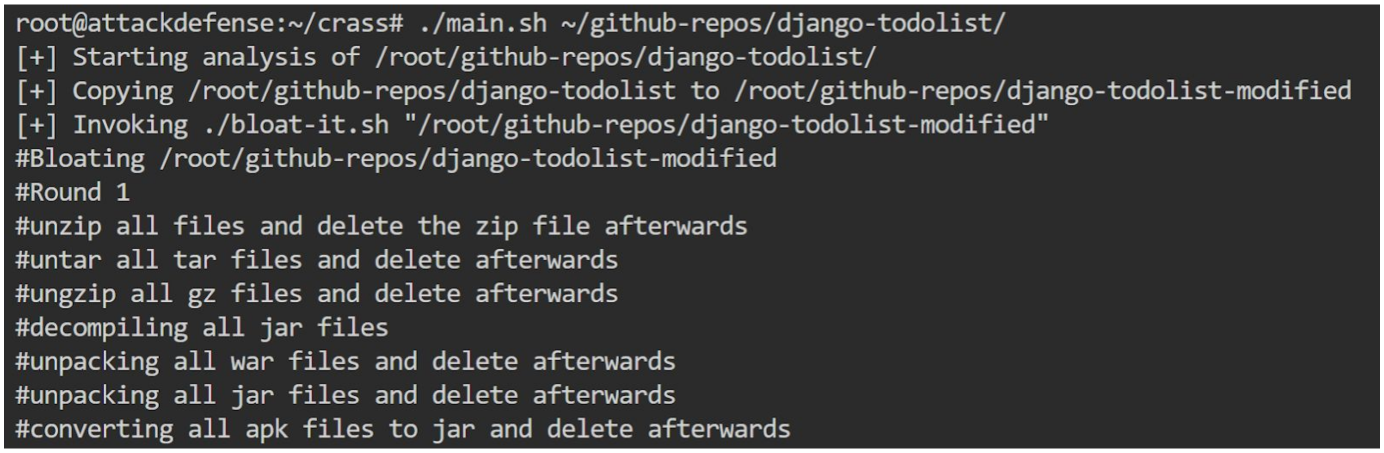
The main.sh will run while passing the target directory as an argument.

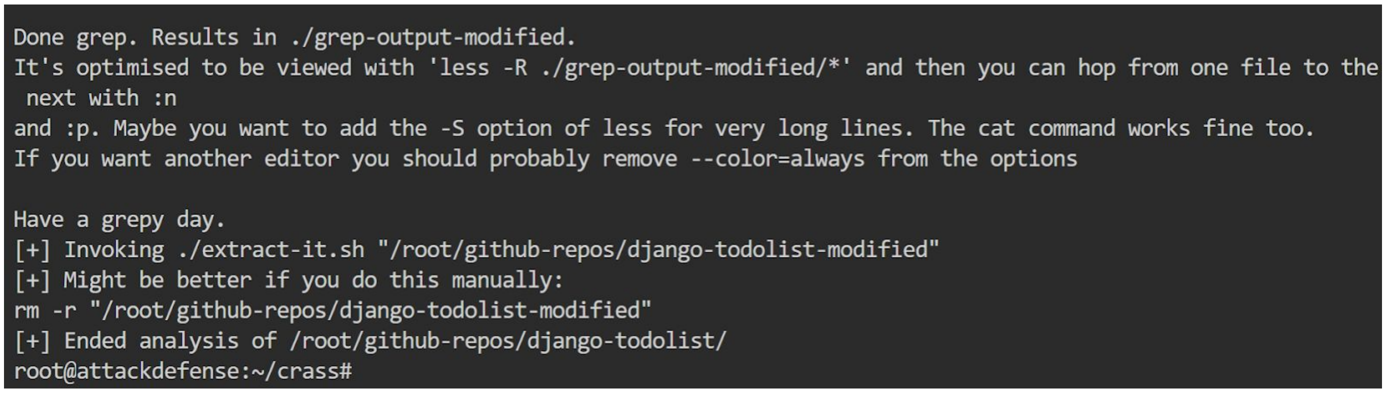
We will take one example at a time and run the tool on that.

Example 1: Django Todolist

Step 1: Run the crass tool on the django-todolist directory.

Commands:
./main.sh ~/github-repos/django-todolist/

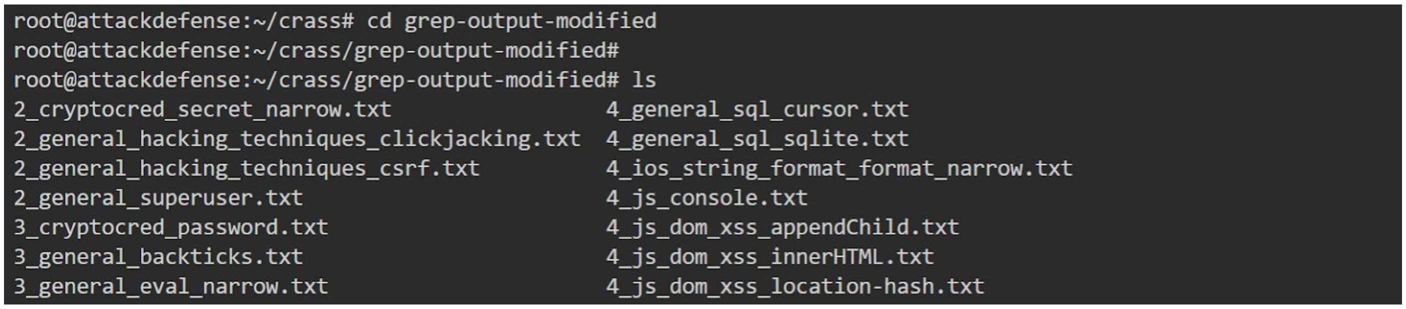

```
root@attackdefense:~/crass# ./main.sh ~/github-repos/django-todolist/
[+] Starting analysis of /root/github-repos/django-todolist/
[+] Copying /root/github-repos/django-todolist to /root/github-repos/django-todolist-modified
[+] Invoking ./bloat-it.sh "/root/github-repos/django-todolist-modified"
#Bloating /root/github-repos/django-todolist-modified
#Round 1
#unzip all files and delete the zip file afterwards
#untar all tar files and delete afterwards
#ungzip all gz files and delete afterwards
#decompiling all jar files
#unpacking all war files and delete afterwards
#unpacking all jar files and delete afterwards
#converting all apk files to jar and delete afterwards
```

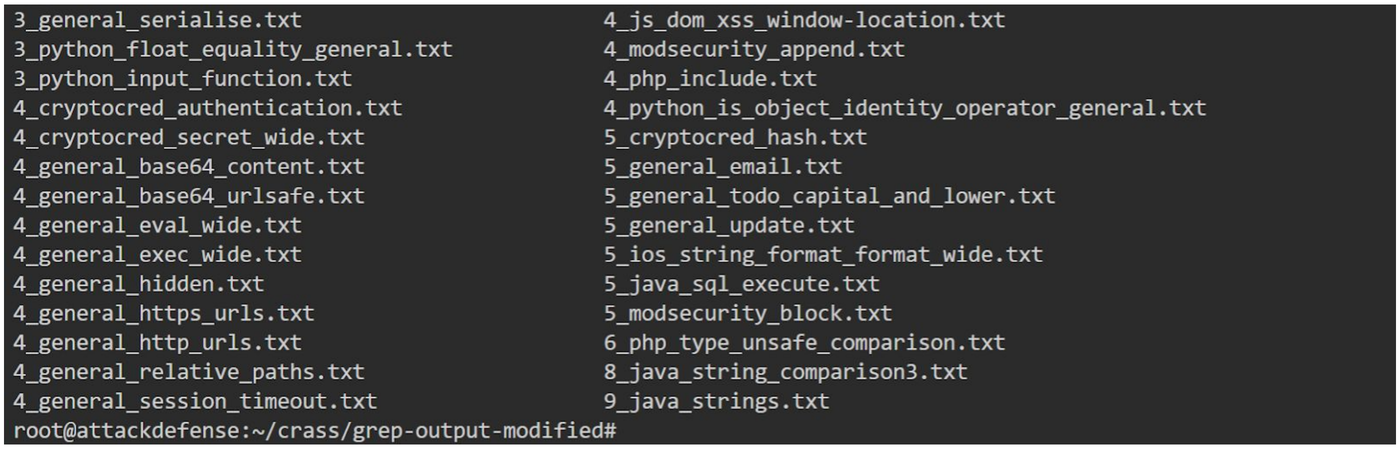

```
Done grep. Results in ./grep-output-modified.
It's optimised to be viewed with 'less -R ./grep-output-modified/*' and then you can hop from one file to the
 next with :n
and :p. Maybe you want to add the -S option of less for very long lines. The cat command works fine too.
If you want another editor you should probably remove --color=always from the options

Have a grepy day.
[+] Invoking ./extract-it.sh "/root/github-repos/django-todolist-modified"
[+] Might be better if you do this manually:
rm -r "/root/github-repos/django-todolist-modified"
[+] Ended analysis of /root/github-repos/django-todolist/
root@attackdefense:~/crass#
```

Crass has completed the scan and result is stored inside the ‘grep-output-modified’ directory

Step 2: Check the grep-output-modified directory.

Commands:
cd grep-output-modified
ls

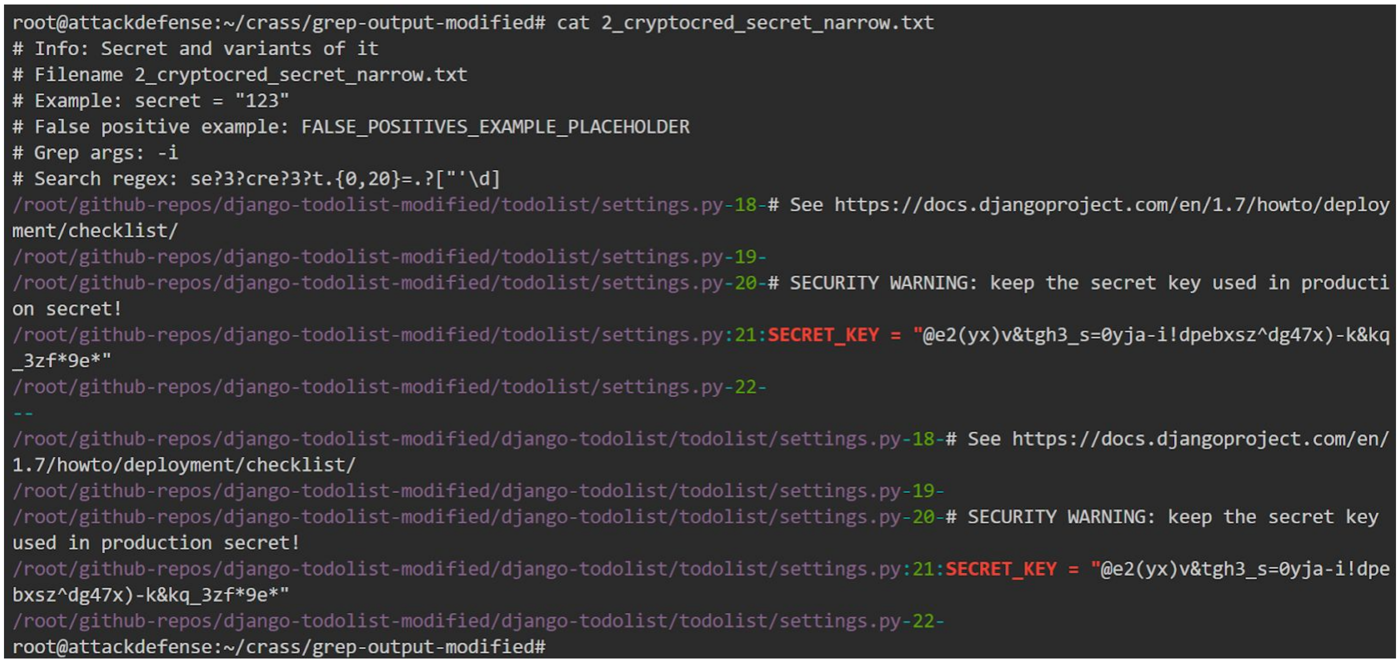

```
root@attackdefense:~/crass# cd grep-output-modified
root@attackdefense:~/crass/grep-output-modified#
root@attackdefense:~/crass/grep-output-modified# ls
2_cryptocred_secret_narrow.txt                    4_general_sql_cursor.txt
2_general_hacking_techniques_clickjacking.txt     4_general_sql_sqlite.txt
2_general_hacking_techniques_csrf.txt             4_ios_string_format_format_narrow.txt
2_general_superuser.txt                           4_js_console.txt
3_cryptocred_password.txt                         4_js_dom_xss_appendChild.txt
3_general_backticks.txt                           4_js_dom_xss_innerHTML.txt
3_general_eval_narrow.txt                         4_js_dom_xss_location-hash.txt
```



```
3_general_serialise.txt                       4_js_dom_xss_window-location.txt
3_python_float_equality_general.txt           4_modsecurity_append.txt
3_python_input_function.txt                   4_php_include.txt
4_cryptocred_authentication.txt               4_python_is_object_identity_operator_general.txt
4_cryptocred_secret_wide.txt                  5_cryptocred_hash.txt
4_general_base64_content.txt                  5_general_email.txt
4_general_base64_urlsafe.txt                  5_general_todo_capital_and_lower.txt
4_general_eval_wide.txt                       5_general_update.txt
4_general_exec_wide.txt                       5_ios_string_format_format_wide.txt
4_general_hidden.txt                          5_java_sql_execute.txt
4_general_https_urls.txt                      5_modsecurity_block.txt
4_general_http_urls.txt                       6_php_type_unsafe_comparison.txt
4_general_relative_paths.txt                  8_java_string_comparison3.txt
4_general_session_timeout.txt                 9_java_strings.txt
root@attackdefense:~/crass/grep-output-modified#
```

Step 3: Check the content of one of the files to get the information.

Command: cat 2_cryptocred_secret_narrow.txt



```
root@attackdefense:~/crass/grep-output-modified# cat 2_cryptocred_secret_narrow.txt
# Info: Secret and variants of it
# Filename 2_cryptocred_secret_narrow.txt
# Example: secret = "123"
# False positive example: FALSE_POSITIVES_EXAMPLE_PLACEHOLDER
# Grep args: -i
# Search regex: se?3?cre?3?t.{0,20}=.?["'\d]
/root/github-repos/django-todolist-modified/todolist/settings.py-18-# See https://docs.djangoproject.com/en/1.7/howto/deploy
ment/checklist/
/root/github-repos/django-todolist-modified/todolist/settings.py-19-
/root/github-repos/django-todolist-modified/todolist/settings.py-20-# SECURITY WARNING: keep the secret key used in producti
on secret!
/root/github-repos/django-todolist-modified/todolist/settings.py:21:SECRET_KEY = "@e2(yx)v&tgh3_s=0yja-i!dpebxsz^dg47x)-k&kq
_3zf*9e*"
/root/github-repos/django-todolist-modified/todolist/settings.py-22-
--
/root/github-repos/django-todolist-modified/django-todolist/todolist/settings.py-18-# See https://docs.djangoproject.com/en/
1.7/howto/deployment/checklist/
/root/github-repos/django-todolist-modified/django-todolist/todolist/settings.py-19-
/root/github-repos/django-todolist-modified/django-todolist/todolist/settings.py-20-# SECURITY WARNING: keep the secret key
used in production secret!
/root/github-repos/django-todolist-modified/django-todolist/todolist/settings.py:21:SECRET_KEY = "@e2(yx)v&tgh3_s=0yja-i!dpe
bxsz^dg47x)-k&kq_3zf*9e*"
/root/github-repos/django-todolist-modified/django-todolist/todolist/settings.py-22-
root@attackdefense:~/crass/grep-output-modified#
```

Note: There are multiple files which could show more information but, in this manual, only one file is being checked. You can check the other files and know more.

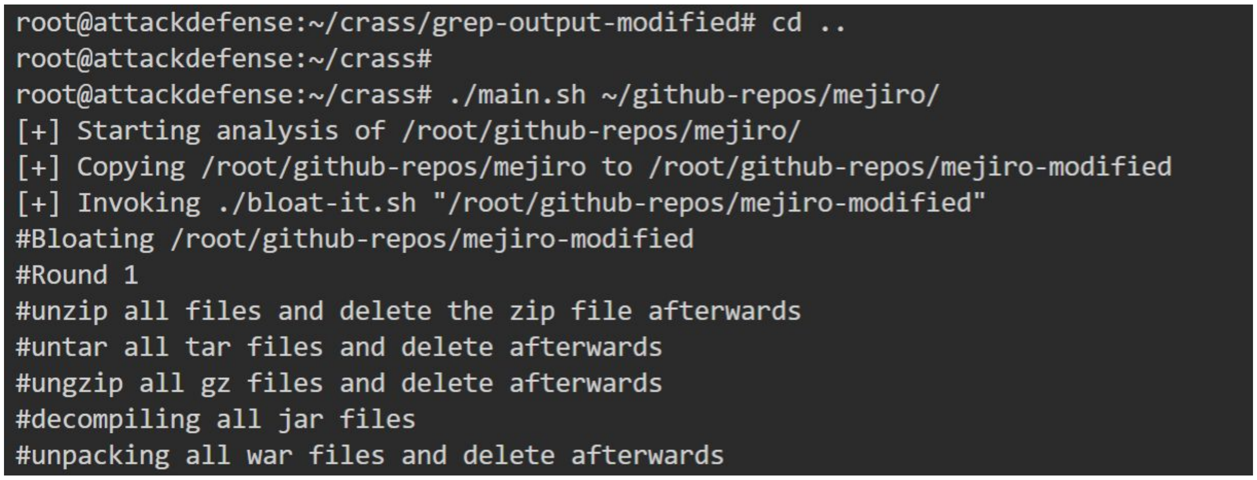
Issues Detected

- Secret key exposed in the source code.

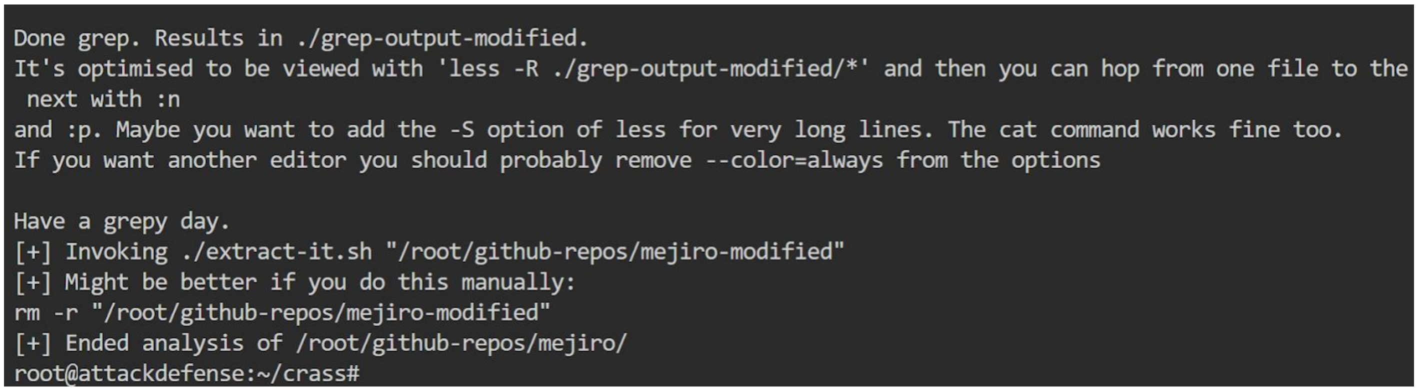
Example 2: Mejiro

Step 1: Change to the crass directory and run the tool on the mejiro project directory.

Commands:
cd ..
./main.sh ~/github-repos/mejiro/



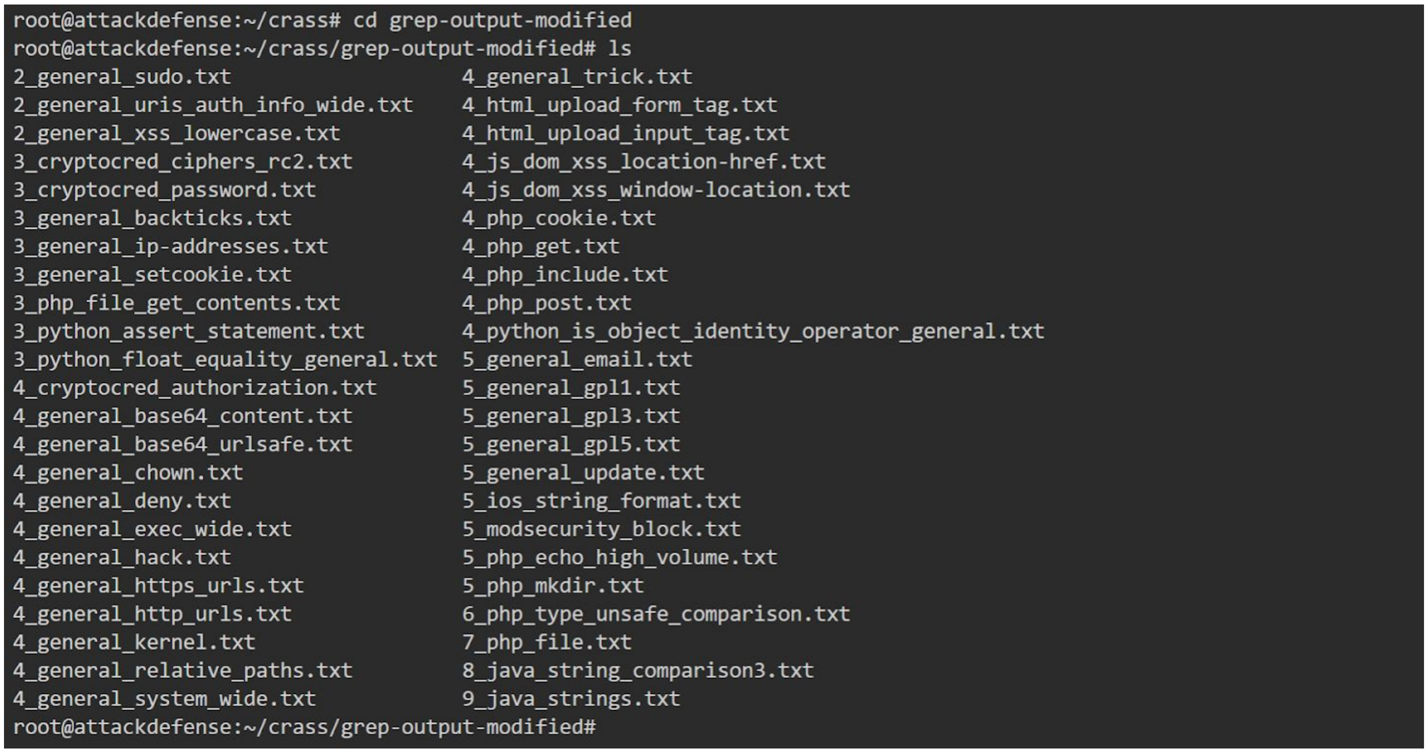
```
root@attackdefense:~/crass/grep-output-modified# cd ..
root@attackdefense:~/crass#
root@attackdefense:~/crass# ./main.sh ~/github-repos/mejiro/
[+] Starting analysis of /root/github-repos/mejiro/
[+] Copying /root/github-repos/mejiro to /root/github-repos/mejiro-modified
[+] Invoking ./bloat-it.sh "/root/github-repos/mejiro-modified"
#Bloating /root/github-repos/mejiro-modified
#Round 1
#unzip all files and delete the zip file afterwards
#untar all tar files and delete afterwards
#ungzip all gz files and delete afterwards
#decompiling all jar files
#unpacking all war files and delete afterwards
```



```
Done grep. Results in ./grep-output-modified.
It's optimised to be viewed with 'less -R ./grep-output-modified/*' and then you can hop from one file to the
 next with :n
and :p. Maybe you want to add the -S option of less for very long lines. The cat command works fine too.
If you want another editor you should probably remove --color=always from the options

Have a grepy day.
[+] Invoking ./extract-it.sh "/root/github-repos/mejiro-modified"
[+] Might be better if you do this manually:
rm -r "/root/github-repos/mejiro-modified"
[+] Ended analysis of /root/github-repos/mejiro/
root@attackdefense:~/crass#
```

Step 2: Check the grep-output-modified directory.

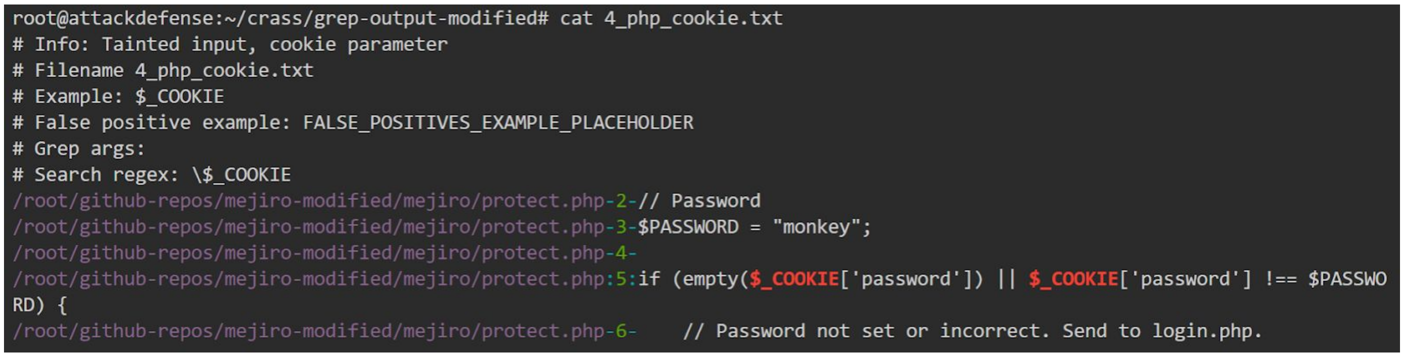
Commands:
cd grep-output-modified
ls



```
root@attackdefense:~/crass# cd grep-output-modified
root@attackdefense:~/crass/grep-output-modified# ls
2_general_sudo.txt                    4_general_trick.txt
2_general_uris_auth_info_wide.txt     4_html_upload_form_tag.txt
2_general_xss_lowercase.txt           4_html_upload_input_tag.txt
3_cryptocred_ciphers_rc2.txt          4_js_dom_xss_location-href.txt
3_cryptocred_password.txt             4_js_dom_xss_window-location.txt
3_general_backticks.txt               4_php_cookie.txt
3_general_ip-addresses.txt            4_php_get.txt
3_general_setcookie.txt               4_php_include.txt
3_php_file_get_contents.txt           4_php_post.txt
3_python_assert_statement.txt         4_python_is_object_identity_operator_general.txt
3_python_float_equality_general.txt   5_general_email.txt
4_cryptocred_authorization.txt        5_general_gpl1.txt
4_general_base64_content.txt          5_general_gpl3.txt
4_general_base64_urlsafe.txt          5_general_gpl5.txt
4_general_chown.txt                   5_general_update.txt
4_general_deny.txt                    5_ios_string_format.txt
4_general_exec_wide.txt               5_modsecurity_block.txt
4_general_hack.txt                    5_php_echo_high_volume.txt
4_general_https_urls.txt              5_php_mkdir.txt
4_general_http_urls.txt               6_php_type_unsafe_comparison.txt
4_general_kernel.txt                  7_php_file.txt
4_general_relative_paths.txt          8_java_string_comparison3.txt
4_general_system_wide.txt             9_java_strings.txt
root@attackdefense:~/crass/grep-output-modified#
```

Step 3: Check the content of one of the files to get the information.

Command: cat 4_php_cookie.txt



```
root@attackdefense:~/crass/grep-output-modified# cat 4_php_cookie.txt
# Info: Tainted input, cookie parameter
# Filename 4_php_cookie.txt
# Example: $_COOKIE
# False positive example: FALSE_POSITIVES_EXAMPLE_PLACEHOLDER
# Grep args:
# Search regex: \$_COOKIE
/root/github-repos/mejiro-modified/mejiro/protect.php-2-// Password
/root/github-repos/mejiro-modified/mejiro/protect.php-3-$PASSWORD = "monkey";
/root/github-repos/mejiro-modified/mejiro/protect.php-4-
/root/github-repos/mejiro-modified/mejiro/protect.php:5:if (empty($_COOKIE['password']) || $_COOKIE['password'] !== $PASSWO
RD) {
/root/github-repos/mejiro-modified/mejiro/protect.php-6-    // Password not set or incorrect. Send to login.php.
```

Note: There are multiple files which could show more information but, in this manual, only one file is being checked. You can check the other files and know more.

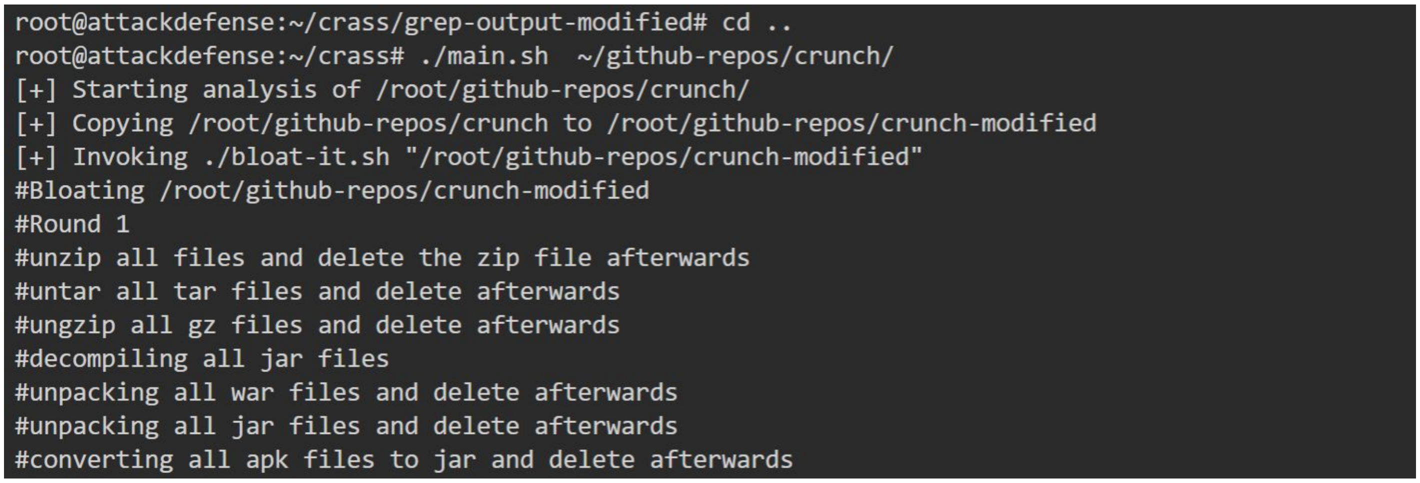
Issues Detected

- Hardcoded credentials found

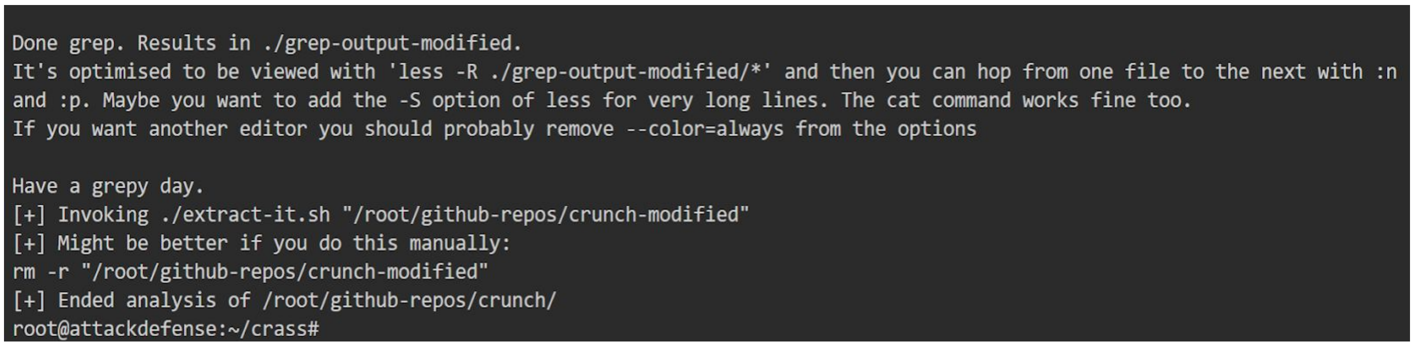
Example 3: Crunch

Step 1: Change to the crass directory and run the tool on crunch project directory.

Commands:
cd ..
./main.sh ~/github-repos/crunch/



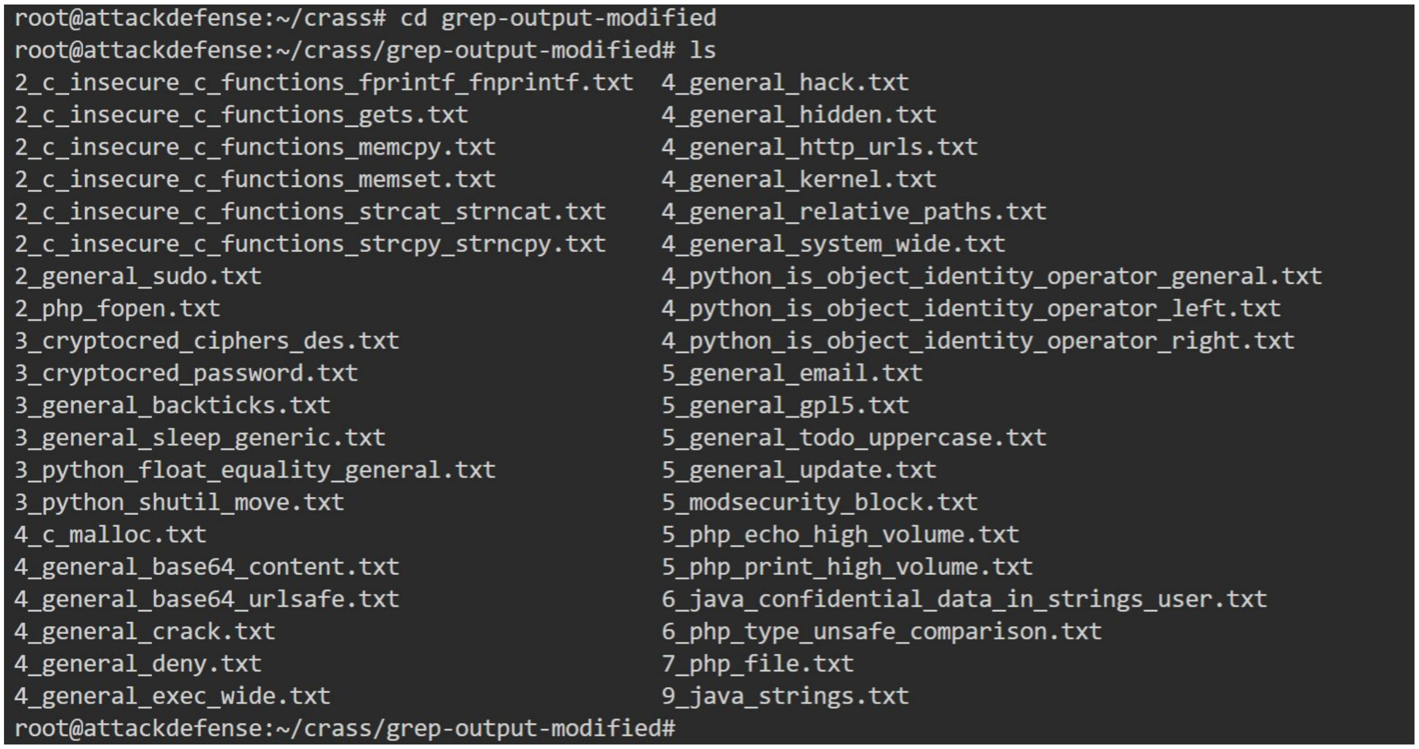
```
root@attackdefense:~/crass/grep-output-modified# cd ..
root@attackdefense:~/crass# ./main.sh  ~/github-repos/crunch/
[+] Starting analysis of /root/github-repos/crunch/
[+] Copying /root/github-repos/crunch to /root/github-repos/crunch-modified
[+] Invoking ./bloat-it.sh "/root/github-repos/crunch-modified"
#Bloating /root/github-repos/crunch-modified
#Round 1
#unzip all files and delete the zip file afterwards
#untar all tar files and delete afterwards
#ungzip all gz files and delete afterwards
#decompiling all jar files
#unpacking all war files and delete afterwards
#unpacking all jar files and delete afterwards
#converting all apk files to jar and delete afterwards
```



```
Done grep. Results in ./grep-output-modified.
It's optimised to be viewed with 'less -R ./grep-output-modified/*' and then you can hop from one file to the next with :n
and :p. Maybe you want to add the -S option of less for very long lines. The cat command works fine too.
If you want another editor you should probably remove --color=always from the options

Have a grepy day.
[+] Invoking ./extract-it.sh "/root/github-repos/crunch-modified"
[+] Might be better if you do this manually:
rm -r "/root/github-repos/crunch-modified"
[+] Ended analysis of /root/github-repos/crunch/
root@attackdefense:~/crass#
```

Step 2: Check the grep-output-modified directory.

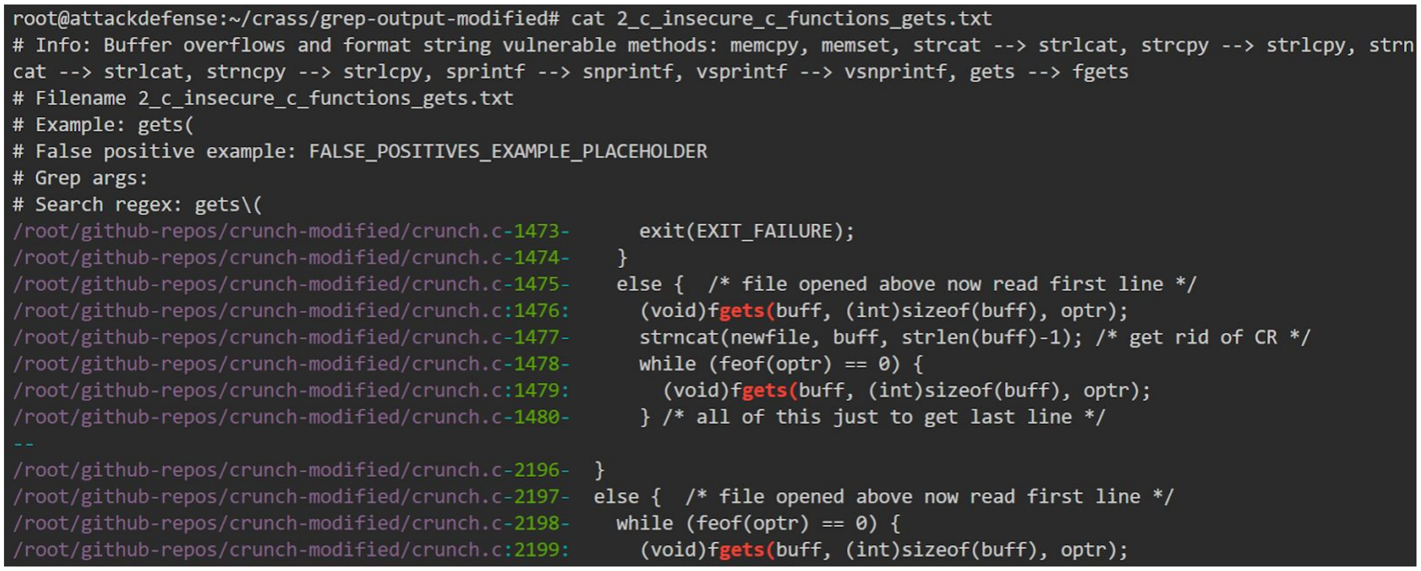
Commands:
cd grep-output-modified
ls



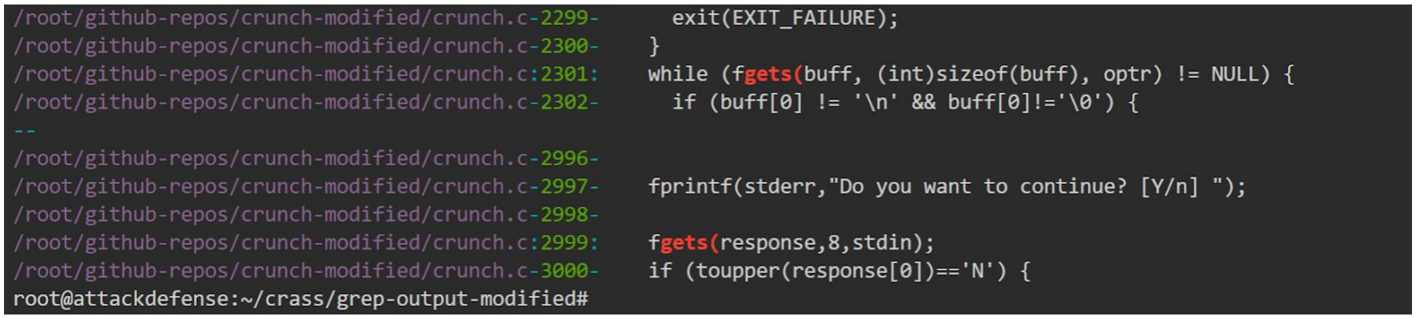
```
root@attackdefense:~/crass# cd grep-output-modified
root@attackdefense:~/crass/grep-output-modified# ls
2_c_insecure_c_functions_fprintf_fnprintf.txt  4_general_hack.txt
2_c_insecure_c_functions_gets.txt              4_general_hidden.txt
2_c_insecure_c_functions_memcpy.txt            4_general_http_urls.txt
2_c_insecure_c_functions_memset.txt            4_general_kernel.txt
2_c_insecure_c_functions_strcat_strncat.txt    4_general_relative_paths.txt
2_c_insecure_c_functions_strcpy_strncpy.txt    4_general_system_wide.txt
2_general_sudo.txt                             4_python_is_object_identity_operator_general.txt
2_php_fopen.txt                                4_python_is_object_identity_operator_left.txt
3_cryptocred_ciphers_des.txt                   4_python_is_object_identity_operator_right.txt
3_cryptocred_password.txt                      5_general_email.txt
3_general_backticks.txt                        5_general_gpl5.txt
3_general_sleep_generic.txt                    5_general_todo_uppercase.txt
3_python_float_equality_general.txt            5_general_update.txt
3_python_shutil_move.txt                       5_modsecurity_block.txt
4_c_malloc.txt                                 5_php_echo_high_volume.txt
4_general_base64_content.txt                   5_php_print_high_volume.txt
4_general_base64_urlsafe.txt                   6_java_confidential_data_in_strings_user.txt
4_general_crack.txt                            6_php_type_unsafe_comparison.txt
4_general_deny.txt                             7_php_file.txt
4_general_exec_wide.txt                        9_java_strings.txt
root@attackdefense:~/crass/grep-output-modified#
```

Step 3: Check the content of one of the files to get the information.

Command: cat 2_c_insecure_c_functions_gets.txt



```
root@attackdefense:~/crass/grep-output-modified# cat 2_c_insecure_c_functions_gets.txt
# Info: Buffer overflows and format string vulnerable methods: memcpy, memset, strcat --> strlcat, strcpy --> strlcpy, strn
cat --> strlcat, strncpy --> strlcpy, sprintf --> snprintf, vsprintf --> vsnprintf, gets --> fgets
# Filename 2_c_insecure_c_functions_gets.txt
# Example: gets(
# False positive example: FALSE_POSITIVES_EXAMPLE_PLACEHOLDER
# Grep args:
# Search regex: gets\(
/root/github-repos/crunch-modified/crunch.c-1473-      exit(EXIT_FAILURE);
/root/github-repos/crunch-modified/crunch.c-1474-    }
/root/github-repos/crunch-modified/crunch.c-1475-    else {  /* file opened above now read first line */
/root/github-repos/crunch-modified/crunch.c:1476:      (void)fgets(buff, (int)sizeof(buff), optr);
/root/github-repos/crunch-modified/crunch.c-1477-      strncat(newfile, buff, strlen(buff)-1); /* get rid of CR */
/root/github-repos/crunch-modified/crunch.c-1478-      while (feof(optr) == 0) {
/root/github-repos/crunch-modified/crunch.c:1479:        (void)fgets(buff, (int)sizeof(buff), optr);
/root/github-repos/crunch-modified/crunch.c-1480-      } /* all of this just to get last line */
--
/root/github-repos/crunch-modified/crunch.c-2196-  }
/root/github-repos/crunch-modified/crunch.c-2197-  else {  /* file opened above now read first line */
/root/github-repos/crunch-modified/crunch.c-2198-    while (feof(optr) == 0) {
/root/github-repos/crunch-modified/crunch.c:2199:      (void)fgets(buff, (int)sizeof(buff), optr);
```



```
/root/github-repos/crunch-modified/crunch.c-2299-        exit(EXIT_FAILURE);
/root/github-repos/crunch-modified/crunch.c-2300-      }
/root/github-repos/crunch-modified/crunch.c:2301:     while (fgets(buff, (int)sizeof(buff), optr) != NULL) {
/root/github-repos/crunch-modified/crunch.c-2302-        if (buff[0] != '\n' && buff[0]!='\0') {
--
/root/github-repos/crunch-modified/crunch.c-2996-
/root/github-repos/crunch-modified/crunch.c-2997-     fprintf(stderr,"Do you want to continue? [Y/n] ");
/root/github-repos/crunch-modified/crunch.c-2998-
/root/github-repos/crunch-modified/crunch.c:2999:     fgets(response,8,stdin);
/root/github-repos/crunch-modified/crunch.c-3000-     if (toupper(response[0])=='N') {
root@attackdefense:~/crass/grep-output-modified#
```

Note: There are multiple files which could show more information but, in this manual, only one file is being checked. You can check the other files and know more.

Issues Detected

- Insecure function 'gets' has been used in the application.

Learnings

Perform Sensitive Information Scan using the CRASS tool.

References:

- Django-todolist (https://github.com/rtzll/django-todolist.git)
- Mejiro (https://github.com/dmpop/mejiro.git)
- Crunch (https://github.com/crunchsec/crunch.git)