HP Fortify Scanner
Setup

- CSIF computer's should have the scanner already installed
- Command is “sourceanalyzer”
Problems

- “which sourceanalyzer” == 
  <path>/HP_Fortify/HP_Fortify_SCA_and_Apps_<version>/bin/sourceanalyzer

- “echo $PATH” == “/sbin:/bin:...:<sourceanalyzer path>”

- “export PATH=$PATH:<sourceanalyzer path>”
  - Insert this in “~/.bashrc” for it to be permanent
  - Note: root has a separate environment PATH so you can't use “sudo sourceanalyzer” (issue faced on Ubuntu 14.04)
Example Code

```c
#include <strings.h>
#include <stdio.h>
#define MAX_SIZE 128
void doMemCpy(char *buf, char *in, int chars){
    memcpy(buf, in, chars);
}
int main(){
    char buf[64];
    char in[MAX_SIZE];
    int bytes;
    printf("Enter buffer contents:\n");
    read(0, in, MAX_SIZE-1);
    printf("Bytes to copy:\n");
    scanf("%d", &bytes);
    doMemCpy(buf, in, bytes);
    return(0);
}
```
Run the analyzer

- Fortify version 4.21
- “sourceanalyzer gcc stackbuffer.c” will not output the same as the handout
- “sourceanalyzer -scan gcc stackbuffer.c”
Output

[D10CB5094B2FB1C2C6AC8AD7CADECA30 : low : Unchecked Return Value : semantic ]
stackbuffer.c(16) : read()

[4940AB43F66960894026F18AF2032001 : high : Buffer Overflow : dataflow ]
stackbuffer.c(7) : ->memcpy(2)
  stackbuffer.c(20) : ->doMemCpy(2)
stackbuffer.c(18) : <- scanf(1)
Reading Output

- [ID : security level : security problem : type of problem]
- Ex: [blah : low : Unchecked Return Value : semantic]
  
  Stackbuffer.c(16) : read()
  
  - The designer is not determining if the user is inputting the correct data. This could be a problem if the program requires all integers but the user might provide an input with an alphanumerical value or a new return address

- Indentation means that the problems are related

- Sequence of execution is from bottom to top
Reading Output cont.

- [ID : security level : security problem : type of problem]
- Ex: [blah : high : Buffer Overflow : dataflow]
  
  ```c
  stackbuffer.c(7): ->memcpy(2)
  stackbuffer.c(20): ->doMemCpy(2)
  stackbuffer.c(18): <-scanf(1)
  ```

- Since the problem type is “dataflow” the application uses arrows to represent the type of input
  - “<-” means input
  - “->” means pass to
• Stackbuffer.c(18): <- scanf(1)
• The “second” parameter input of function “scanf” has a security problem
• Code: scanf(“%d”, &bytes);
  – Sourceanalyzer numbers the parameters like “argv”
  – Zero parameter: “%d”
  – First parameter: &bytes
• The custom function “doMemCpy” then passes that value as the “third” parameter
• Then the function “memcpy” uses the “bytes” value to know how many bytes to copy from “in” to “buf”
Run Example Code

- Seg Fault
  
  
  ./a.out
  
  Enter buffer contents:
  
  aaa1234
  
  Bytes to copy:
  
  999
  
  Segmentation fault (core dumped)
Run Example Code

• Buffer overflow
  ./a.out
  Enter buffer contents:
  aaaaaaaaaa<new return address>
  Bytes to copy:
  <Bytes until return address location>
Multiple Files Program

• All the functions located one file so we used:
  – “sourceanalyzer -scan gcc stackbuffer.c”
• Some programs require multiple files
• The above command wont work
New Code Part 1

- Staticbuffer.c:
  
  ```c
  #include "headerfile.h"
  
  int main() {
    char buf[64];
    char in[MAX_SIZE];
    int bytes;
    printf("Enter buffer contents:\n");
    read(0, in, MAX_SIZE-1);
    printf("Bytes to copy:\n");
    scanf("%d", &bytes);
    doMemCpy(buf, in, bytes);
    return 0;
  }
  ```
New Code Part 2

- Memorycopy.c:
  ```c
  #include "headerfile.h"
  void doMemCpy(char* buf, char* in, int chars) {
    memcpy(buf, in, chars);
    printf("%s", buf);
  }
  ```

- Headerfile.h:
  ```c
  #include <string.h>
  #include <stdio.h>
  #define MAX_SIZE 128
  void doMemCpy(char* buf, char* in, int chars);
  ```
New Command

- “sourceanalyzer -b my_buildid make”
- “sourceanalyzer -b my_buildid -scan”

Notice that “my_buildid” was used twice, this is important because that is how Fortify references the just compiled code

Other options:
- “sourceanalyzer -b my_buildid -show-build-warnings” will list all warnings and errors that occurred during the compile process
- “sourceanalyzer -b my_buildid -export-build-session <new_file.mbs>” will make a mobile build of “my_buildid” for easier file movement