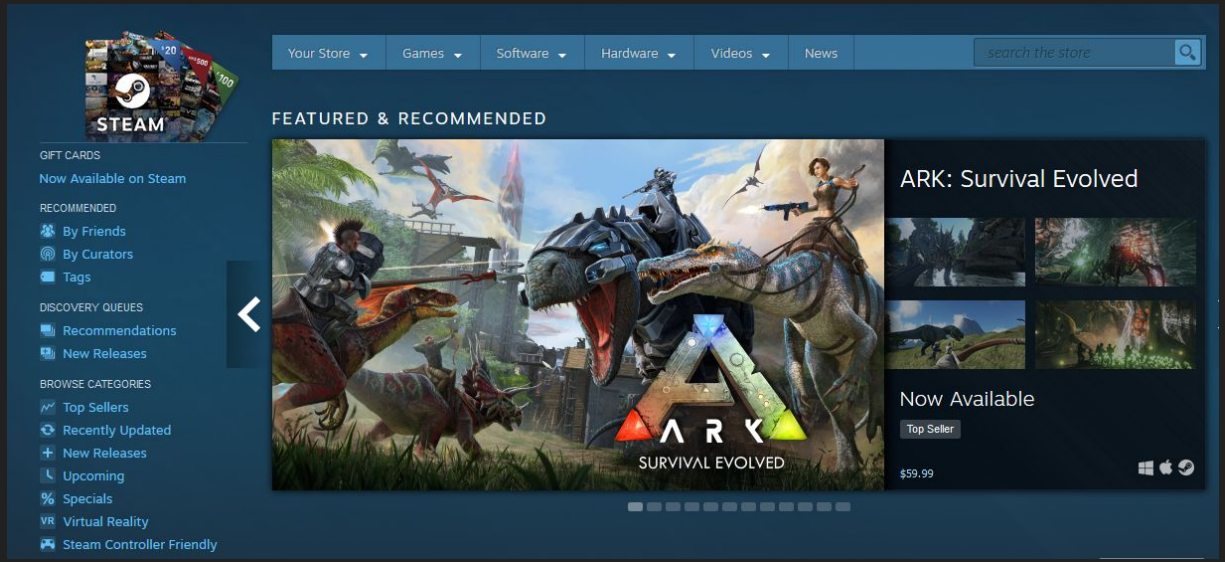
CWE Group Project

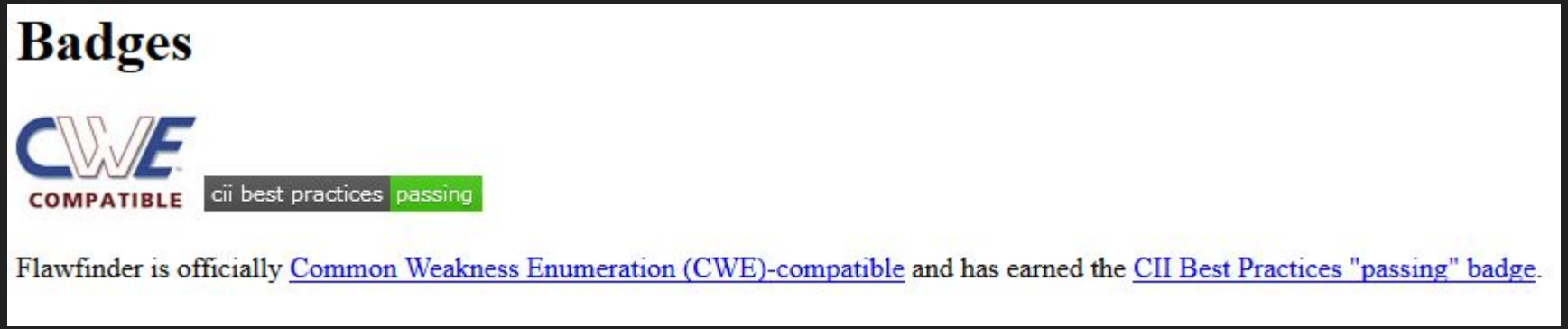
Omar Salih, Ben Olson, Taylor Scott, Kyle Brinker

Steam

We chose Valve's Steam source code for our software assurance target



FLAWFINDER



We used Flawfinder to analyze the code for flaws

FLAWFINDER: Results

- Flawfinder got 9444 hits in Steam’s code
- This is out of 2,021,042 lines of code
- Not all hits are necessarily vulnerabilities

```
ANALYSIS SUMMARY:

Hits = 9444
Lines analyzed = 3012544 in approximately 24.77 seconds (121644 lines/second)
Physical Source Lines of Code (SLOC) = 2021042
Hits@level = [0] 1348 [1] 1395 [2] 6916 [3] 222 [4] 903 [5]   8
Hits@level+ = [0+] 10792 [1+] 9444 [2+] 8049 [3+] 1133 [4+] 911 [5+]   8
Hits/KSLOC@level+ = [0+] 5.33982 [1+] 4.67284 [2+] 3.9826 [3+] 0.560602 [4+] 0.450758 [5+] 0.00395835
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://www.dwheeler.com/secure-programs) for more information.
```

Selected CWE’s

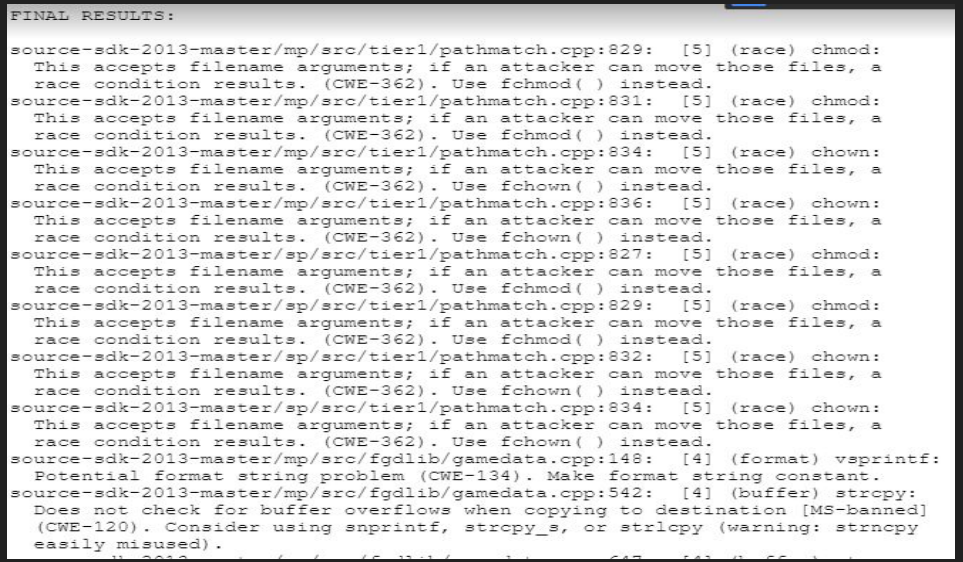
1. CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
    a. Medium likelihood of exploit, prevalent in C, C++, and Java languages

2. CWE-120: Does not check for buffer overflows when copying to destination
    a. High likelihood of exploit, prevalent in C, C++ languages

3. CWE-134: If format strings can be influenced by an attacker, they can be exploited
    a. High likelihood of exploit, prevalent in C, C++ languages

CWE 362

“The program contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently. “



```
FINAL RESULTS:

source-sdk-2013-master/mp/src/tier1/pathmatch.cpp:829:  [5] (race) chmod:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchmod( ) instead.
source-sdk-2013-master/mp/src/tier1/pathmatch.cpp:831:  [5] (race) chmod:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchmod( ) instead.
source-sdk-2013-master/mp/src/tier1/pathmatch.cpp:834:  [5] (race) chown:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchown( ) instead.
source-sdk-2013-master/mp/src/tier1/pathmatch.cpp:836:  [5] (race) chown:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchown( ) instead.
source-sdk-2013-master/sp/src/tier1/pathmatch.cpp:827:  [5] (race) chmod:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchmod( ) instead.
source-sdk-2013-master/sp/src/tier1/pathmatch.cpp:829:  [5] (race) chmod:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchmod( ) instead.
source-sdk-2013-master/sp/src/tier1/pathmatch.cpp:832:  [5] (race) chown:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchown( ) instead.
source-sdk-2013-master/sp/src/tier1/pathmatch.cpp:834:  [5] (race) chown:
  This accepts filename arguments; if an attacker can move those files, a
  race condition results. (CWE-362). Use fchown( ) instead.
source-sdk-2013-master/mp/src/fgdlib/gamedata.cpp:148:  [4] (format) vsprintf:
  Potential format string problem (CWE-134). Make format string constant.
source-sdk-2013-master/mp/src/fgdlib/gamedata.cpp:542:  [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).
```

Example Language: **C**

```
void f(pthread_mutex_t *mutex) {
  pthread_mutex_lock(mutex);
  /* access shared resource */


  pthread_mutex_unlock(mutex);

}
```

CWE 362 Consequences

When a race condition allows multiple control flows to access a resource simultaneously, it might lead the program into unexpected states, possibly resulting in a crash.

When a race condition is combined with predictable resource names and loose permissions, it may be possible for an attacker to overwrite or access confidential data.

CWE 120

The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.

```
source-sdk-2013-master/mp/src/vgui2/vgui_controls/TextImage.cpp:221:  [4] (buffer) wcscpy:
  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.
source-sdk-2013-master/mp/src/vgui2/vgui_controls/Tooltip.cpp:57:  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely
  (CWE-78). try using a library call that implements the same functionality
  if available.
source-sdk-2013-master/mp/src/vgui2/vgui_controls/Tooltip.cpp:118:  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely
  (CWE-78). try using a library call that implements the same functionality
  if available.
source-sdk-2013-master/mp/src/vgui2/vgui_controls/TreeView.cpp:148:  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely
  (CWE-78). try using a library call that implements the same functionality
  if available.
```

Example Language: **C**

```
char last_name[20];
printf ("Enter your last name: ");
scanf ("%s", last_name);
```

CWE 120 Consequences

Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. This can often be used to subvert any other security service.

Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop.

CWE 134

```
source-sdk-2013-master/mp/src/public/XUnzip.cpp:674:  [4] (format) fprintf:
  If format strings can be influenced by an attacker, they can be exploited
  (CWE-134). Use a constant for the format specification.
source-sdk-2013-master/mp/src/public/XUnzip.cpp:675:  [4] (format) fprintf:
  If format strings can be influenced by an attacker, they can be exploited
  (CWE-134). Use a constant for the format specification.
source-sdk-2013-master/mp/src/public/XUnzip.cpp:676:  [4] (format) fprintf:
  If format strings can be influenced by an attacker, they can be exploited
  (CWE-134). Use a constant for the format specification.
source-sdk-2013-master/mp/src/public/XUnzip.cpp:3448:  [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).
source-sdk-2013-master/mp/src/public/XUnzip.cpp:3988:  [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).
source-sdk-2013-master/mp/src/public/XUnzip.cpp:4098:  [4] (buffer) _tcscpy:
  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.
```

Example Language: **C**

```c
#include <stdio.h>

void printWrapper(char *string) {

  printf(string);

}

int main(int argc, char **argv) {

  char buf[5012];
  memcpy(buf, argv[1], 5012);
  printWrapper(argv[1]);
  return (0);

}
```

The software uses a function that accepts a format string as an argument, but the format string originates from an external source.

CWE 134 Consequences

When an attacker can modify an externally-controlled format string, this can lead to buffer overflows, denial of service, or data representation problems.

It should be noted that in some circumstances, such as internationalization, the set of format strings is externally controlled by design. If the source of these format strings is trusted then the external control might not itself pose a vulnerability.

CWE 134: Mitigation

1. Choose a language that is not subject to this flaw.

2. Ensure that all format string functions are passed a static string which cannot be controlled by the user and that the proper number of arguments are always sent to that function as well.

3. Heed the warnings of compilers and linkers, since they may alert you to improper usage.

CONCLUSION

- Everything has a vulnerability, weakness, or bug that can be exploited.
- BE CAREFUL WITH WHAT YOU DO!!!!!!!!!!
- Questions?