woftp 2.6.0

```c
Void logUser(char* user) {
    char buf[2048];
    snprintf(buf, sizeof(buf), user);
    ....
}
```

Attacker provides user = “XXXX<0x12345E78>%0x12345D78d%n<shellcode>”
XXX – junk
<0x12345E78> - target
%0x12345D78d – value
%n – write

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>user</td>
</tr>
<tr>
<td>2</td>
<td>Return address ➔ 0x12345E78 ( after %n)</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0x12345E78</td>
</tr>
<tr>
<td>5</td>
<td>“XXXX”</td>
</tr>
<tr>
<td>6</td>
<td>user</td>
</tr>
<tr>
<td>7</td>
<td>2048</td>
</tr>
<tr>
<td>8</td>
<td>&amp;buf</td>
</tr>
<tr>
<td>9</td>
<td>ARGP</td>
</tr>
<tr>
<td>10</td>
<td>Counter ➔ 0x12345D80 (after value)</td>
</tr>
</tbody>
</table>

From 1 to 5 : logUser frame.
From 6 to 10 : snprintf stack frame

Format String Bug detection/prevention
- education
  - mistakes
- Automatic code auditing
Automatic Bug Fixing

System defenses: prevent damage

- Format String Bug.

```c
int printf(char* fmt, ...); \1
char* getenv(char*); \2
int main(...) {
  char *s, *t;
  s = getenv("Foo");
  t = s;
  printf(t);
}
```

\(\Rightarrow\) Big Idea: data flow analysis

```c
char * s = "hey";
printf(s); \rightarrow no bugs (It depends on where the data comes from)
```

Could any dangerous inputs flow to the format arg of `printf`?

Bug finding Tools.
- Require annotations
  - 1: int printf($untainted char* fmt,…);
  - 2: $tainted char* getenv(char *);
- False positives: warning, but no bugs.
  - Complete (precise) – no false positives
- False negatives: bug, but no warning.
  - Sound – no false negatives

Getenv-ret \( \leq \) s \( \leq \) t \( \leq \) fmt
| ref \( \leq \) ref \( \leq \) ref \( \leq \) ref
| Stainted char = char = char = $untainted char

Contradiction \( \Rightarrow \) potential error

Vector < Object
$untainted < Stainted
$untainted char < $tainted char

How do we detect (test) for false negatives?
- Theorem
- Testing
- Comparison

CQual is, in theory, sound.
Pick two:
- sound
- complete
- terminates

CQual result:

<table>
<thead>
<tr>
<th></th>
<th>Warning1</th>
<th>Warning2</th>
<th>bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>muh</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>cfengine</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>bftpd</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>total</td>
<td>19</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Current state of the art: 20% FP rate

Attacker needs 1 bug!
Defender needs ALL bugs!