

Class notes for CSE 509

2/15/07

woftpd 2.6.0

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

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

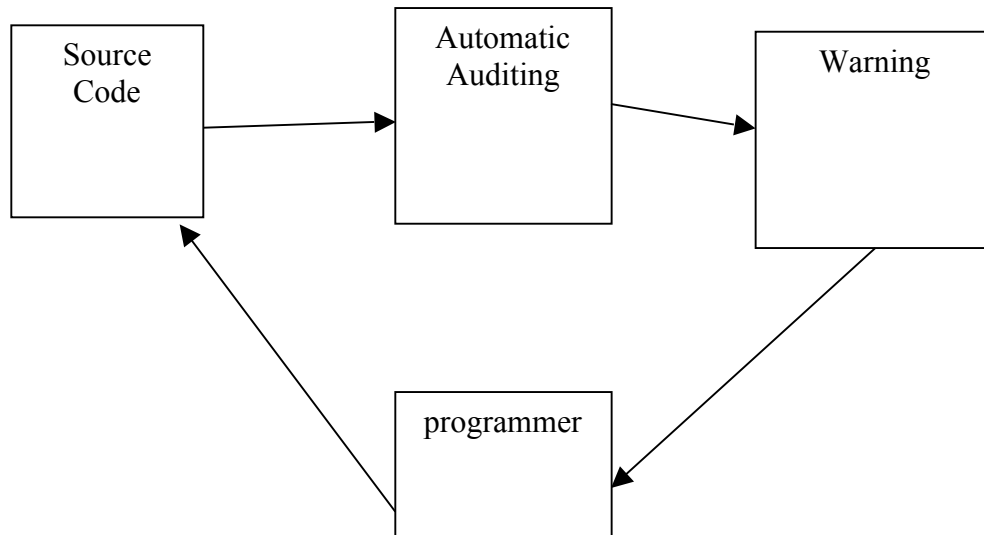
| |
|--|
| 1. user |
| 2. Return address → 0x12345E78 (after %n) |
| 3. |
| 4. 0x12345E78 |
| 5. "XXXX" |
| 6. user |
| 7. 2048 |
| 8. &buf |
| 9. ARGV |
| 10. Counter → 0x12345D80 (after value) |
| |
| |

→ 0x12345678

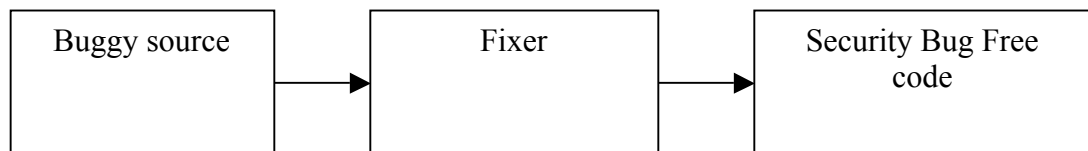
From 1 to 5 : logUser frame.
From 6 to 10 : snprintf stack frame

Format String Bug detection/prevention

- education
 - o mistakes
- Automatic code auditing



- Automatic Bug Fixing



- System defenses : prevent damage

Format String Bug.

```

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

⇒ Big Idea : data flow analysis

```

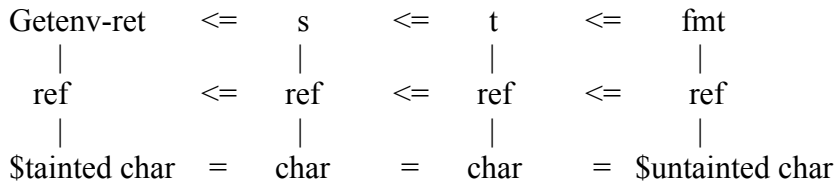
char * s = "hey";
printf(s);
  
```

→ 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
 - o **1** : int printf(\$untainted char* fmt,...);
 - o **2** : \$tainted char* getenv(char *);
- false positives : warning, but no bugs.
 - o Complete (precise) – no false positives
- False negatives : bug, but no warning.
 - o Sound – no false negatives



Contradiction → potential error

Vector < Object

\$untainted < \$tainted

\$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 :

| | Warning1 | Warning2 | bugs |
|----------|----------|----------|------|
| muh | | | 1 |
| cfengine | | | 1 |
| bftpd | | | 1 |
| total | 19 | 5 | 3 |

Current state of the art : 20% FP rate

Attacker needs 1 bug!

Defender needs ALL bugs!