Lecture 22 - Assembly Wrap

Stack Smashing/Stack buffer overflow

* No bounds checking for arrays in C.
* Stack - stores state information and local variables - trouble!

```
void echo ()
{
    char buf[8];
    gets (buf);
    puts (buf);
}
```

```
main ()
{

    Saved %ebp
    Locals
    args
    ret addr
    Saved %ebp
    Saved %ebx

    buf:
    4 15 6 7
    0 1 2 3

    echo's stack frame.
```

Warning: (C compiler).
"The gets function is dangerous and should not be used."

Solution: Use fgets.
`fgets(buf, n, stdin);`
`(n-1)` max characters to read.
Internet worm of Nov 1988, (or) Morris worm

- Computer Fraud & Abuse Act
- 3 years probation, 400 hours of community service, and a $10,500 fine.

Solutions

1. Stack Randomization

To insert exploit code, attacker needs to inject both code as well as a pointer to this code as part of the attack string.

To generate this pointer, need to know the stack address of the string in memory.

Idea

Vary the stack position from one run of the program to another.
2. Stack Corruption Detection.

 GCC - stack protector (to detect buffer overruns).

 Idea - store a special canary value in the stack frame between any local buffer and the rest of the stack state.

 Canary (guard value)
 random number for each run.

 check the "canary" for any buffer overflow!

 END OF ASSEMBLY!
Assembly Review

1. \( \text{movl } (\%\text{eax}), \%\text{esi} \) \( \Rightarrow \) \( \%\text{esi} \)
   \( \frac{p}{p} \)

2. \( \text{movl } (\%\text{esi}), \%\text{edi} \) \( \Rightarrow \) \( \%\text{edi} \)
   \( \star p \)

3. \( \text{movl } \%\text{edi}, (\%\text{ebx}) \) \( \Rightarrow \) \( q = \star p \)

4. \( a \) is in \(-0x4(\%\text{ebp})\)
5. \( b \) is in \(-0x8(\%\text{ebp})\)

\( \%\text{ebp} \) \( \downarrow \) \( \text{Saved } \%\text{ebp} \)

\( \frac{\&a}{a} \)
\( \frac{\&b}{b} \)

\( \text{movl } -0x4(\%\text{ebp}), \%\text{eax} \)
\( \text{movl } (\%\text{eax}), \%\text{eax} \)
\( \text{movl } \%\text{eax}, -0x8(\%\text{ebp}) \)

\( q = \star p \)
2. \[
\begin{align*}
\text{int } & \text{a[10];} \\
\text{int } & \text{xp = } \&\text{a[7];}
\end{align*}
\]

\[\text{A } p = \text{?}\]

\[7 \times 4 = 28 = 0x1c\]

\[\therefore \text{a[7] is at } 0x804900 + 0x1c = 0x80491c\]

\[\text{struct point } \{\]

\[\begin{align*}
\text{int } & \text{x;} \\
\text{int } & \text{y;}
\end{align*}\]

\[\}\text{struct point } \text{p[10];}\]

\[\text{int } \text{xp} = \&\text{p[7].y};\]

\[\text{p[6].y, p[6].x}\]

\[\text{ptr = } 0x804900 + 0x38 + 0x4\]

\[8 \times 7 = 56 = 0x38\]

\[\Rightarrow \text{ptr = } 0x80493c\]
3. \( \text{char} \ t = (\text{char}) \times > (\text{unsigned}) \ y; \)

**Functions**

\[
\text{char} \ f(x, \ y) = \text{int } x, \ \text{int } y;
\]

\( \text{x at } 0x8(\%\text{ebp}) \)

\( \text{y at } 0xC(\%\text{ebp}) \)

**Assembly**

\[
\begin{align*}
\text{movl} & \ 0x8(\%\text{ebp}), \ %\text{eax} \\
\text{movslb} & \ %\text{al}, \ %\text{edx} \\
\text{movl} & \ 0xC(\%\text{ebp}), \ %\text{eax} \\
\text{cmpl} & \ %\text{eax}, \ %\text{edx} \\
\text{seta} & \ %\text{al} \\
\text{movb} & \ %\text{al}, \ -0x4(\%\text{ebp})
\end{align*}
\]

**Alignment in structures!**