Compiler Trojan horse

Trojan horse

- Malware
- Looks Legitimate
- Has to executed by User
- Example: Command line trojan horse (Source: Wikipedia)

Compiler Trojan Horse

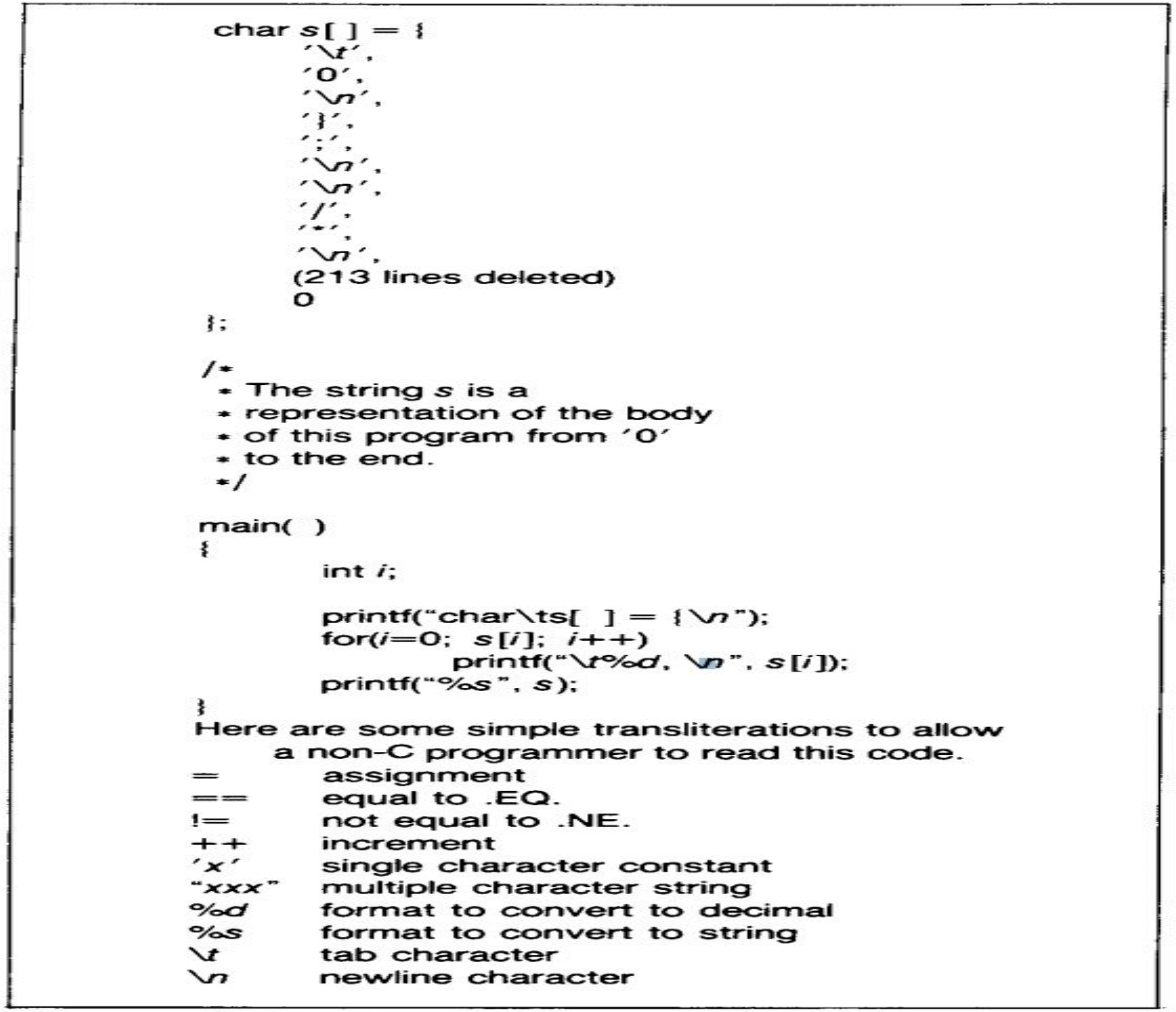
- Compiler is malicious
- Verify program safety
  - Check source code
  - Verify compiler source code
  - Verify compiler's compiler source code …..

Attack Stages:

1. Program generates exact copy of its source code
2. Perpetuating knowledge
3. Injecting backdoor login
4. Hiding

Reference: Reflections on Trusting Trust by KEN THOMPSON

Quine



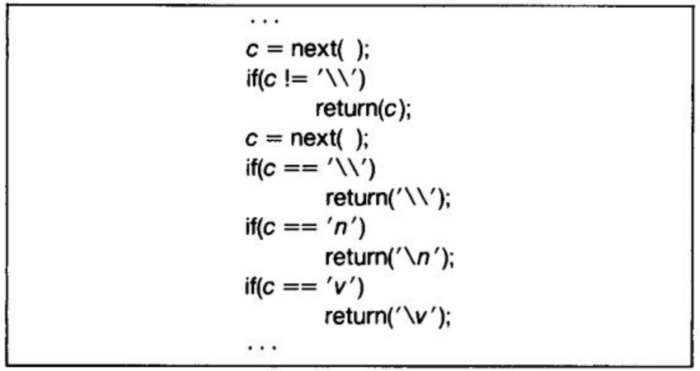
```
char s[ ] = {
        '\t',
        '0',
        '\n',
        '}',
        ';',
        '\n',
        '\n',
        '/',
        '*',
        '\n',
        (213 lines deleted)
        0
};

/*
 * The string s is a
 * representation of the body
 * of this program from '0'
 * to the end.
 */

main( )
{
        int i;

        printf("char\ts[  ] = {\n");
        for(i=0;  s[i];  i++)
                printf("\t%d, \n", s[i]);
        printf("%s", s);
}
```

Here are some simple transliterations to allow
a non-C programmer to read this code.

=	assignment
==	equal to .EQ.
!=	not equal to .NE.
++	increment
'x'	single character constant
"xxx"	multiple character string
%d	format to convert to decimal
%s	format to convert to string
\t	tab character
\n	newline character

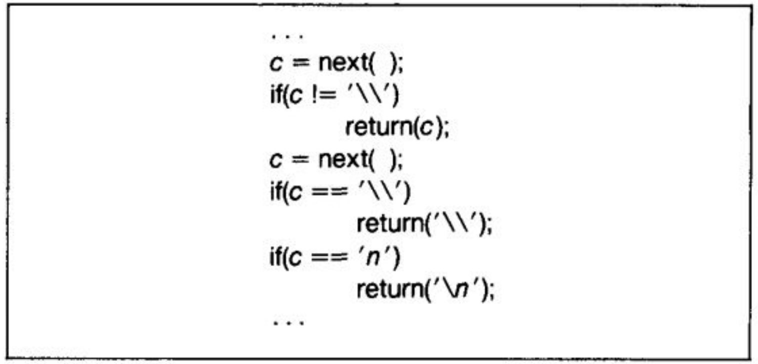
FIGURE 1.

Compiler understanding itself



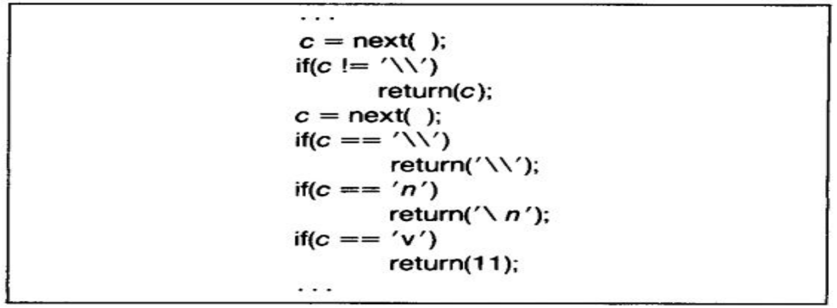
```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
...
```

FIGURE 2.1.



```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
...
```

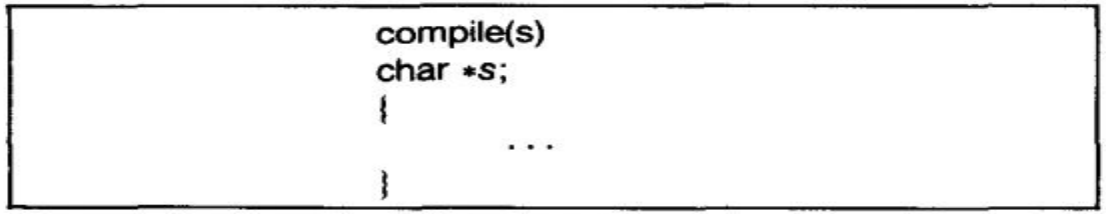
FIGURE 2.2.



```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\ n');
if(c == 'v')
        return(11);
...
```

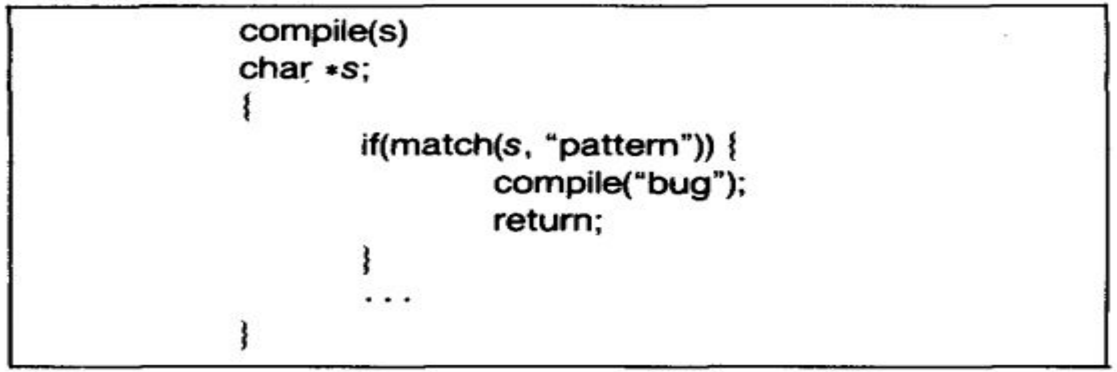
FIGURE 2.3.

Attack



```
compile(s)
char *s;
{
        . . .
}
```

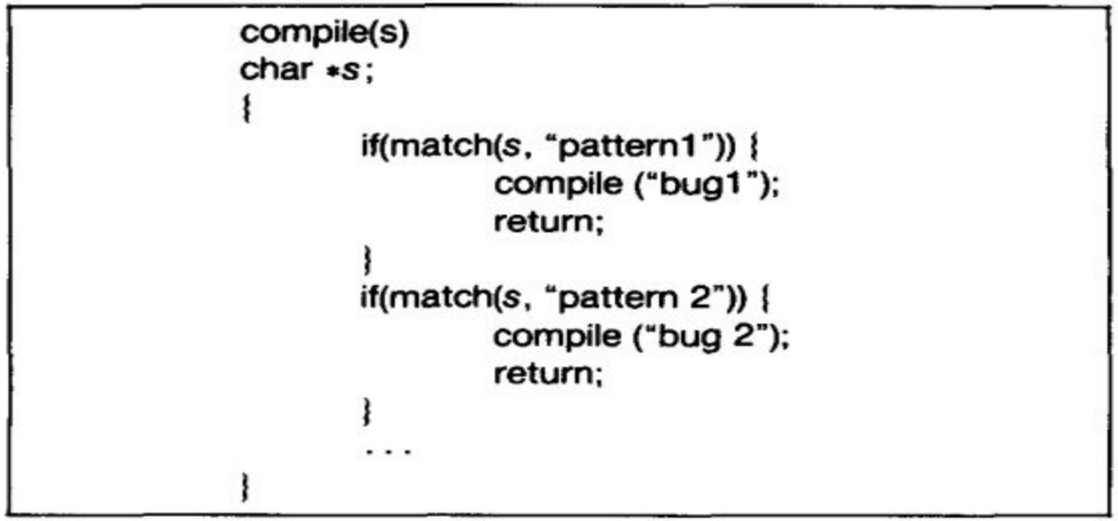
FIGURE 3.1.



```
compile(s)
char *s;
{
        if(match(s, "pattern")) {
                compile("bug");
                return;
        }
        . . .
}
```

FIGURE 3.2.

Hiding



```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        . . .
}
```

FIGURE 3.3.

- Bugged binary from bugged source – A
- Remove bugs from source and recompile it with A – B
- Recompile B’s source code with B
- Now B has both trojans but B’s source code is not malicious.

Historic attacks and preventive measures

Defenses:

- Reverse engineer compiler machine code (hard)
- Create a minimal compiler that can fool the Trojan compiler[John McDermott, Naval Research Laboratory, 1988]
- Verifiable builds to correspond build with source
- Disassembler (Not perfect)
- Diverse Double Compiling (DDC)

Attacks:

- W32/Induc-A infection of Delphi Compiler (propagated for over a year)