Information Security Laboratory

04-Backdoor
Sang Kil Cha
Backdoor

A means of access to a system in an illegitimate way (e.g., bypassing normal authentication)
Backdoor (cont’d)

```c
int auth(char* userid, char* passwd)
{
    DB* db = ...;
    if ( userExists( db, userid ) ) {
        if ( matchPassWord( db, userid, passwd ) )
            return 1; // Success (authenticated)
    } else if ( strcmp( userid, "secret_admin" ) == 0 ) {
        return 1; // Success (authenticated)
    } else {
        return 0; // Failure
    }
}
```
Benign Usage

Programmers may insert a backdoor routine to their own program, e.g., in order to troubleshoot.

Either it is benign or not, backdoors introduce a security hole.
How Can We Trust Software?
Can you trust this code?
Can you trust this code by analyzing the source code?
The answer is NO!

Ken Thompson
Reflections on Trusting Trust
CACM 1984
Trusting Trust
Trusting Software is Difficult

- You cannot trust code that you did not totally create yourself
- No amount of source-level verification or scrutiny will protect you from using untrusted code!
Stage 1: Self-Reproducing Program (a.k.a. Quine)

```c
char s[ ] = {
    \n, '0', 
    \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\n"");
    for(i=0; s[i]; i++)
        printf("\%d, \n", s[i]);
    printf("\%s", s);
}
```
Stage 2: C Compiler in C

\[\ldots\]
c = next( );
if(c != '\\')
    return(c);
c = next( );
if(c == '\\')
    return('\\');
if(c == 'n')
    return('n');
if(c == 'v')
    return('v');
\[\ldots\]
\[\ldots\]
c = next( );
if(c != '\\')
    return(c);
c = next( );
if(c == '\\')
    return('\\');
if(c == 'n')
    return('n');
if(c == 'v')
    return('v');
return(11);
\[\ldots\]
Stage 3: Trojan Horse

```c
void compile(char *s) {
    // ...
}
```

```c
void compile(char *s) {
    if (match(s, "login pattern")) {
        compile("login backdoor");
        return;
    }
    // ...
}
```
Stage 3: Trojan Horse (2)

```c
void compile(char *s) {
    // ...
}
```

```c
void compile(char *s) {
    if(match(s, "login pattern")) {
        compile("login backdoor");
        return;
    }
    if(match(s, "compiler pattern")) {
        compile("insert the backdoor");
        return;
    }
    // ...
}
```
Trusting Trust

This technique applies to *any* program-handling program such as an assembler, a loader, or hardware microcode, etc.

Analyzing source code is *not* enough!
What You See Is Not What You Execute*

```
#include <stdio.h>
int main (void)
{
    printf( "hi\n" );
}
```

* 2007 Gogul Balakrishnan, PhD Thesis
Binary Code Analysis is Crucial in Security!

Binary code is what we really execute!
Reverse Engineering (= Reversing)

Semantics

0101010101011111010101010101010001001001000111111101011111010010101000100101101000101101001010010011111010100000010101011000001000001011
IS-561: Binary Code Analysis and Secure Software Systems

Will be offered in 2018 spring (tentative)
Question?
Activity #3 Backdoor