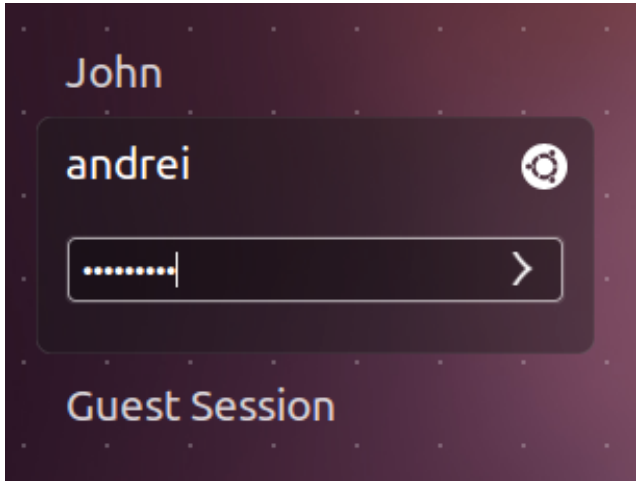
Reflections on Trusting Trust

Yan Huang

Do you Trust your Linux login?



Will your password be kept and later revealed to Eve?

Does it have a backdoor for a (powerful) adversary to sneak in?

Scrutinize its Source Code

```
class Class1 {

    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main(string[] args) {

        // Logon
        BTVLicenseManager licenseManager = new BTVLicenseManager();

        // put your valid license here
        string  networkLicense = "00000000-0000-0000-0000-000000000000-00000000";
        string  password;

        password = "";
        licenseManager.Logon( networkLicense, password );

        Console.WriteLine( "Logged on." );

        string  fullName = @"C:\Documents and Settings\rkuo.SNAPSTREAM\My Documents\My Videos\South Park-(Freak Strike)-2004-08-17-0.mpg";

        BTVLibrary library = new BTVLibrary();

        // Get properties
        PVSPropertyBag bag = library.GetMediaByFullName( fullName );

        // Print properties to the console
        Console.WriteLine( "Properties of {0}", fullName );
        foreach( PVSProperty prop in bag.Properties ) {
            Console.WriteLine( "Property: {0}, {1}", prop.Name, prop.Value );
        }

        // Put the PVSPropertyBag into a more friendly collection class.
        // It's a good idea for you to write a friendlier wrapper class that
        // would allow you to add and remove properties and cast back to
        // the PVSPropertyBag type on the fly.
        ArrayList aProperties = new ArrayList( bag.Properties );

        // Change the "EpisodeDescription" property
        foreach( PVSProperty prop in aProperties ) {
            if( prop.Name == "EpisodeDescription" ) {
                prop.Value = "The boys compete to appear on a talk show. (Edited by Beyond TV Framework)";
            }
        }

        // Create a new PVSPropertyBag with the edited property
        PVSPropertyBag newBag = new PVSPropertyBag();
        newBag.Properties = (PVSProperty[])aProperties.ToArray( typeof(PVSProperty) );

        // This method will edit the recording
        library.EditMedia( fullName, newBag );

        // Print properties to the console and verify the change
        Console.WriteLine( "Edited properties of {0}", fullName );
        foreach( PVSProperty prop in bag.Properties ) {
            Console.WriteLine( "Property: {0}, {1}", prop.Name, prop.Value );
        }

        // Pause so you can see the output, hit enter to continue
        Console.WriteLine( "Press any key to exit..." );
        Console.ReadLine();
        return;
    }
  }
}
```

Or write your own

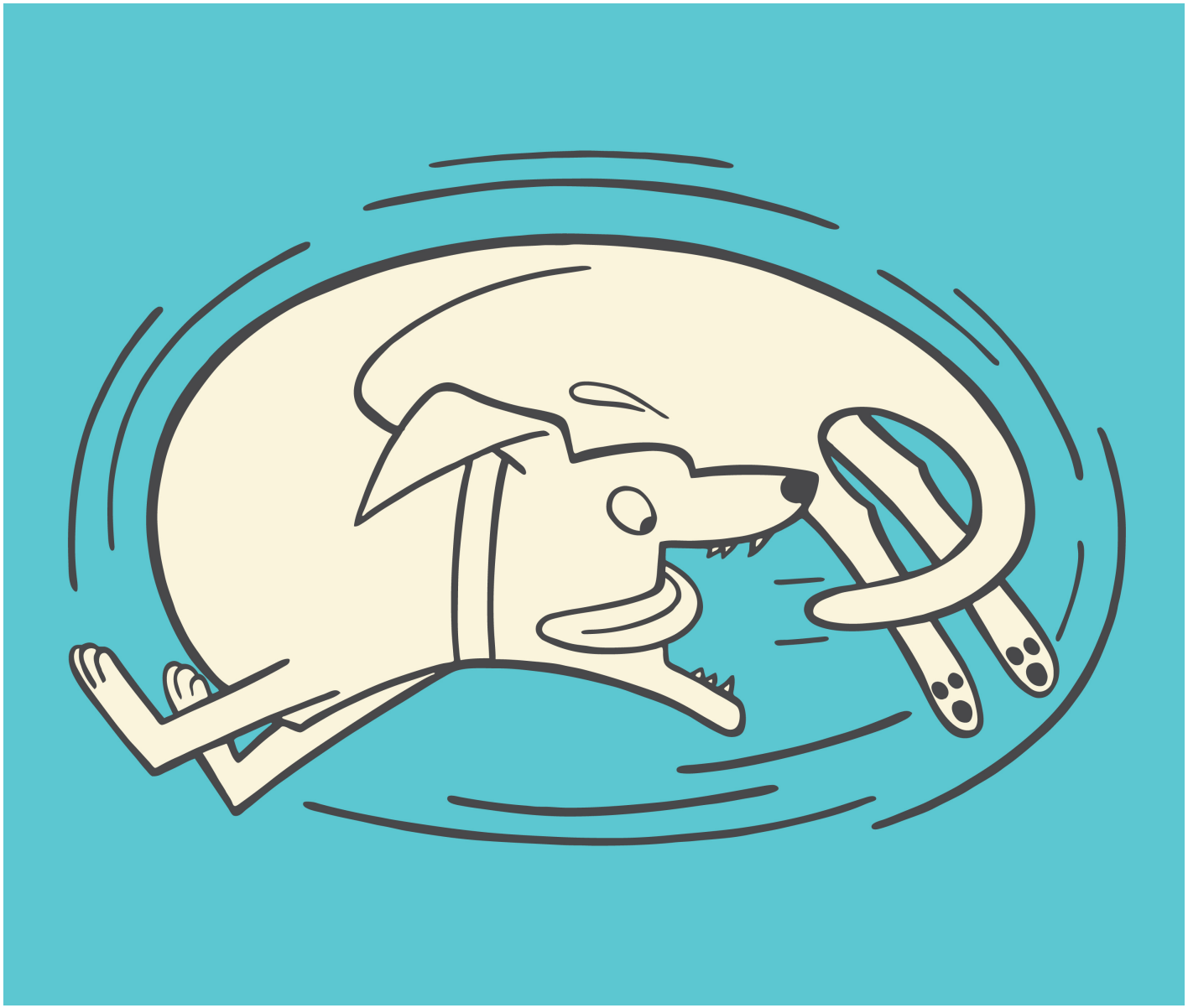
Does it solve the problem?

Need to further trusting the compiler/ interpreter/execution environment …

Thus need to look further at the source code of the compiler/interpretation/ execution environment!

Even Trickier

What program compiles the compiler source code that you (could have) spent your life to proofread?



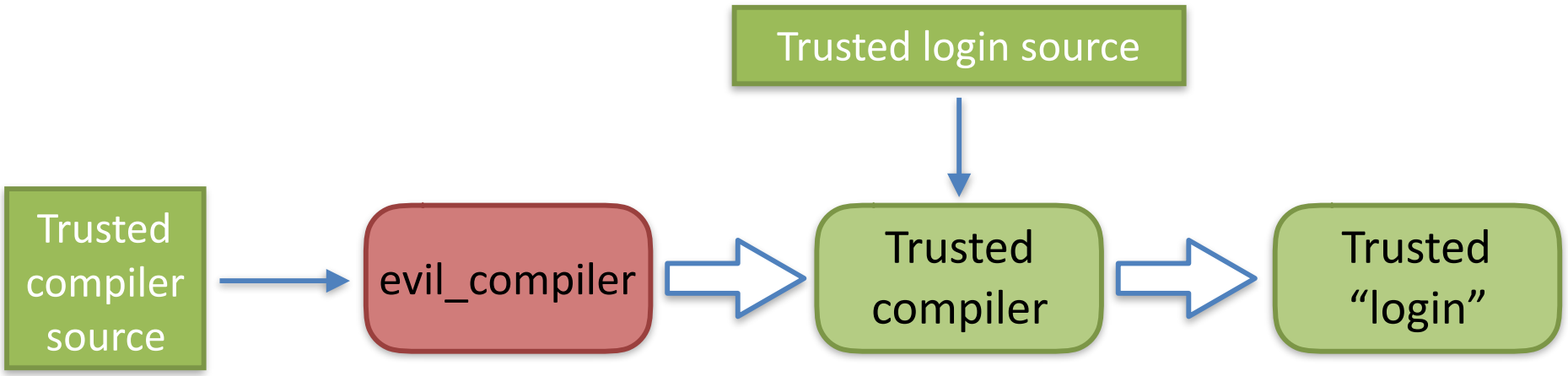
If you stop chasing your tail …

```
evil_compiler(src) {
  /* compile particular src for login program */
  if (match(src, login-pattern)) {
     compile(login-with—backdoor)
     return
  }
  .... /* compile other application src as usual */
}
```

evil_compile plants a backdoor into any pattern matched “login” program that it compiles; but compiles normally other program source code.

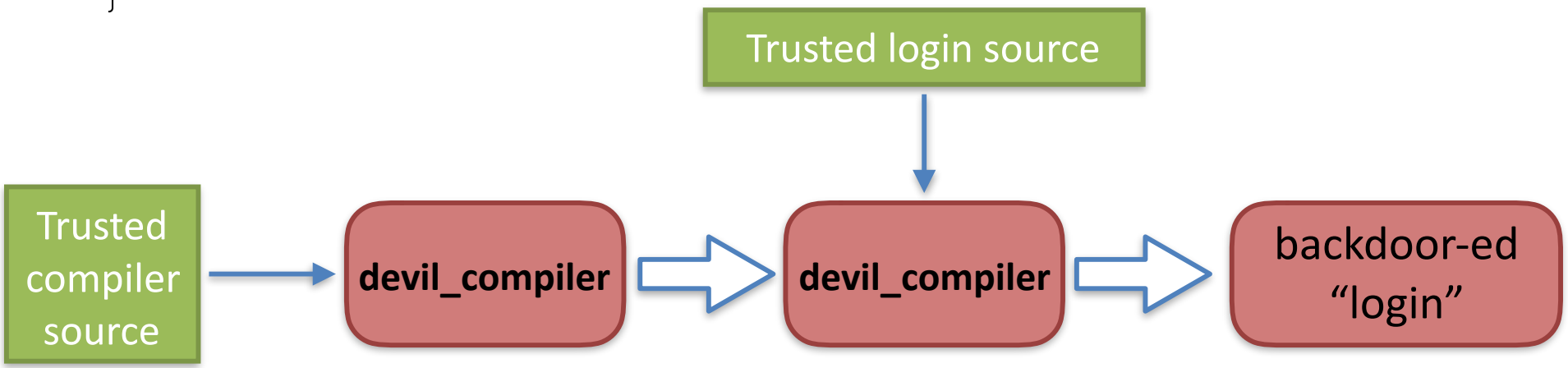
If you stop chasing your tail …

```
evil_compiler(src) {
  /* compile particular src for login program */
  if (match(src, login-pattern)) {
     compile(login-with—backdoor)
     return
  }
  .... /* compile other application src as usual */
}
```

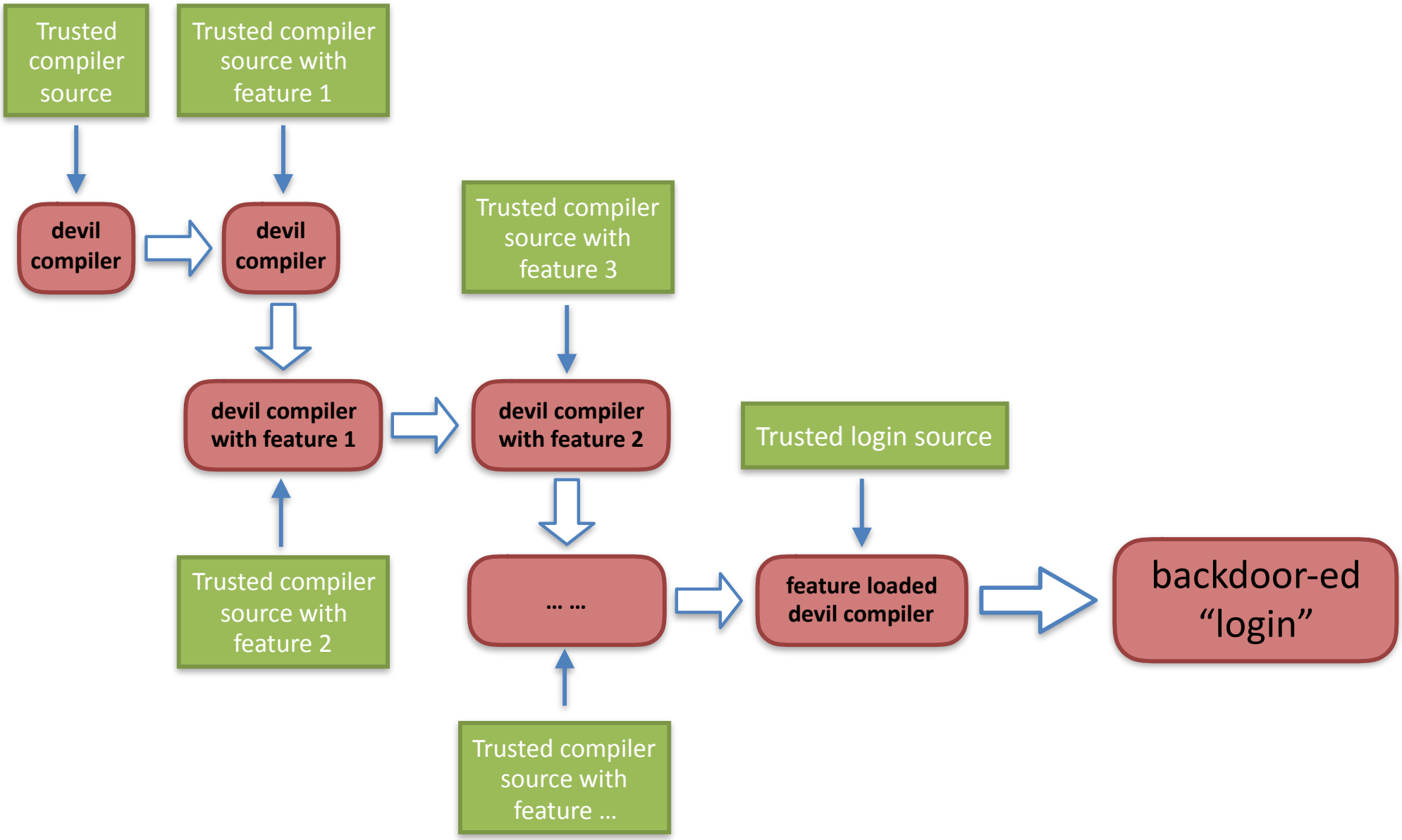


evil_compile can be *devil* …

```
devil_compiler(src) {
  /* compile particular src for login program */
  if (match(src, login-pattern)) {
     compile(login-with—backdoor)
     return
  }
  if (match(src, compiler-pattern)) {
    compile (myself)
    return
  }
  .... /* compile other application src as usual */
}
```



*devil*ness never decay



How to embed *exactly* “myself”?

```
devil_compiler(src) {
  /* compile particular
  if (match(src, logi
     compile(login-w
     return
  }
  if (match(src, compi  pattern)) {
    compile (myself)
    return
  }
  .... /* compile other application src as usual */
}
```

It reminds me of programs outputting exactly themselves …

How to output “myself”, exactly?

```
#include <stdio.h>
void main(){printf(“myself”);}
```

```
myself
```

Live Demo

```
#include <stdio.h>

int main(){
  char s[]="#include <stdio.h>%c%cint main(){%c  char s[]=%c
%s%c;%c  return printf(s,10,10,10,34,s,34,10);%c}";
  return printf(s,10,10,10,34,s,34,10,10);
}
```

```
#include <stdio.h>

int main(){
  char s[]="#include <stdio.h>%c%cint main(){%c  char s[]=%c
%s%c;%c  return printf(s,10,10,10,34,s,34,10);%c}";
  return printf(s,10,10,10,34,s,34,10,10);
}
```

Live Demo — Alternatives

```
#include <stdio.h>

char* s[]={"#include <stdio.h>\n\nchar* s[]={",};\n\nvoid print_string_literal(char *s){\n  putchar(34);
\n  while(*s!=0) {\n    switch (*s) {\n    case 10: printf(\"\\\\n\"); break;\n    default: putchar(*s);
\n    }\n    s++;\n  }\n  putchar(34);\n}\n\nint main(){\n  printf(\"%s\", s[0]);\n
print_string_literal(s[0]);\n  putchar(',');\n  print_string_literal(s[1]);\n  printf(\"%s\", s[1]);\n
return 0;\n}"};

void print_string_literal(char *s){
  putchar(34);
  while (*s!=0) {
    switch (*s) {
    case 10: printf("\\n"); break;
    default: putchar(*s);
    }
    s++;
  }
  putchar(34);
}

int main(){
  printf("%s", s[0]);
  print_string_literal(s[0]);
  putchar(',');
  print_string_literal(s[1]);
  printf("%s", s[1]);
  return 0;
}
```

Trust, but only cautiously …

"The moral is obvious. You can't trust code that you did not totally create yourself. (*Especially code from companies that employ people like me.*)"