تأملی در اعتمادکردن به اعتماد

محمد حدّادیان – mhadadian@ce.sharif.edu

یک تمرین برنامه‌نویسی

به‌عنوان یک دانشجوی دوره‌ی لیسانس، نوشتن برنامه‌ای که بتواند خودش را بازتولید کند، مفید خواهد بود. اگر چه در ابتدا، نوشتن چنین برنامه‌ای دشوار به‌نظر می‌رسد، امّا تفکر و درگیری با چالش‌های آن، توانایی فرد در «چگونه فکر کردن» را افزایش می‌دهد. در شکل زیر کد برنامه‌ای آمده است که یک برنامه‌ی خود-تولید-کننده را می‌سازد. در نوشتن این تمرین دو درس بزرگ برای ما وجود دارد، اول آن‌که هر برنامه‌ای که تصور کنید می‌تواند تولیدکننده‌ی این برنامه باشد (گفتیم می‌خواهیم برنامه‌ای بنویسیم که برنامه‌ای تولید کند که آن برنامه خودش را تولید کند). نکته‌ی دوم در این برنامه، این است که نه‌تنها یک برنامه‌ی خودتولیدکننده، که هر برنامه‌ی دیگری را می‌توان با این روش ساخت. حال تصور کنید یک برنامه‌ی پر استفاده‌ی شما که کارش را به‌درستی انجام می‌دهد، در پس زمینه برنامه‌ای تولید کند که یک کار مخرب بر روی سیستم شما اجرا کند. اعتمادها چه می‌شود؟

```
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

یک تمرین کامپایلرنویسی

حال اجازه دهید پا را فراتر گذاریم. به‌جای یک برنامه‌ی خودتولیدکننده، بیایید یک کامپایلر دگرتولیدکننده بنویسیم! نوشتن یک کامپایلر طبیعتاً با پیچیدگی‌های فراوانی روبروست. امّا تغییر یک کامپایلر چطور؟ فرض کنید که می‌خواهیم یک کاراکتر جدید به‌نام \v را به زبان C اضافه کنیم تا برای ما یک تَب عمودی چاپ کند. در حالت کلّی، کامپایلر زبان C این کاراکتر را نمی‌شناسد. امّا اگر مانند تصویر زیر آن را تغییر دهیم، می‌توانیم از کاراکتر موردنظر خود در برنامه‌های C استفاده کنیم. کامپایلر زبان C خود به زبان C نوشته شده است. پس ابتدا تغییر عکس زیر را در کامپایلر می‌دهیم تا کاراکتر را بشناسد و سپس با استفاده از کامپایلر، خودش را کامپایل می‌کنیم تا یک کامپایلر جدید داشته باشیم. به این ترتیب می‌توانیم در برنامه‌های خود به راحتی از تَب عمودی استفاده کنیم! حال عمل شناسایی و چاپ یک کاراکتر را، با یک فعالیت مخرب جایگزین کنید و کامپایلر جدید خود را در سطح اینترنت پخش کنید. چه می‌شود؟ حال شما توانایی نوشتن برنامه‌هایی دارید که کامپایلر به‌گونه‌ای متفاوت از آن‌چه کاربر هدف می‌پندارد، آن را اجرا خواهد کرد.

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

درس روز

در جهانی که کامپایلرها خودشان را تغییر می‌دهند، هیچ‌گاه نمی‌توانیم به کدی که خودمون آن را ننوشته‌ایم اعتماد کنیم!