Bash Source Code Analysis

With Regards to the ShellShock Vulnerability

Written by Carter Yagemann

Analysis by Amit Ahlawat

Version 0.7

License

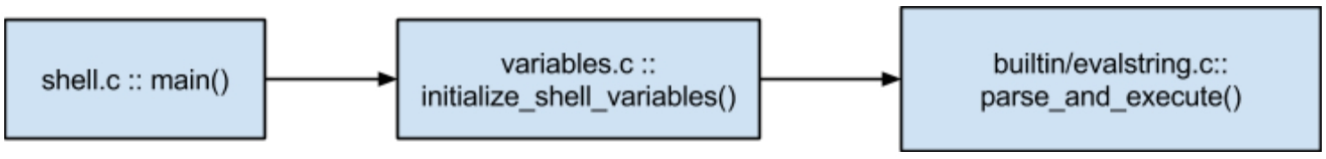
Introduction

The purpose of this document is to provide a brief analysis of the bash source code and identify areas of interest regarding the shellshock vulnerability. Over the course of this analysis, the files of interest will be:

`shell.c` `flags.c` `variables.c` `builtin/evalstring.c`

Control Flow Overview

The following is a diagram highlighting the flow of control of the bash program from the point where it begins execution:



shell.c :: main()

```
/*
 * uidget() determines whether the program's effective user and group match
 * its actual user and group
 */
running_setuid = uidget();
[…]

/* privileged mode is set using the p flag */
if (running_setuid && privileged_mode == 0)
     disabled_priv_mode();
```

shell.c :: uidget()

```
return (current_user.uid != current_user.euid) ||
     (current_user.gid != current_user.egid)
```

shell.c :: disable_priv_mode()

```
/* forces the program's uid and gid back to the user's */
setuid(current_user.uid);
setgid(current_user.gid);
current_user.euid = current_user.uid;
current_user.egid = current_user.gid;
```

variables.c :: initialize_shell_variables()

```
// iternate over all environment variables
for (string_index = 0; string = env[string_index++]; ) {
     […]
     /* if exported function, define it now */
     /* don't import functions from privileged mode */
     if (privmode == 0 && read_but_dont_execute == 0 &&
          STREQN (“() {“, string, 4)) {
          […]
          // this is where shellshock happens
          // parse function definition + execute extra commands
          parse_and_execute (temp_string, name,
               SEVAL_NONINT|SEVAL_NOHIST);
          […]
     }
}
```

builtin/evalstring.c :: parse_and_execute()

This method is complicated, but this is where the actual function declaration is parsed. It is within this parsing loop that bash is susceptible to executing extraneous code following a malformed function definition.