

## OT: (sort of) controlling the PC via SMS

---

*Source:* <http://sci.tech-archive.net/Archive/sci.electronics.design/2007-10/msg04163.html>

---

- *From:* Jan Panteltje <[pNaonStpealmtje@xxxxxxxxxx](mailto:pNaonStpealmtje@xxxxxxxxxx)>
  - *Date:* Mon, 22 Oct 2007 11:06:19 GMT
- 

As part of the alarm system I installed in the server room, I needed a way to send commands to the server (A linux server of course). The alarm system sends me a SMS when something happens, takes pictures of the room, and emails these to several web accessible email accounts. So even if the bad guys take the lot, the police will have their faces.

But sometimes it is needed to 're-arm' the system, or take some other action. So I wrote a small program that runs all the time in the background, and looks for a SMS from my cellphone (GSM).

This works from all over the world, but I tested only via Vodafone the Netherlands.

If you send a SMS like this (in the subject line):  
mysecret@xxxxxxxxxxxxx EMAIL do play mymusic.mp3  
to number 125, then the PC will play the music...  
Of course ANY command will be executed by the PC, this just being an innocent example..  
Vodafone Netherlands forwards SMS emails via gin.nl, and their emails have a fixed format.

So here is the program code in C:

You need to do a:

```
useradd mysecret
```

as root, to create a user that can receive email, and you need to have sendmail mail server running to accept emails (use a different name, nobody should know).

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <termios.h>
#include <sys/un.h>
#include <sys/file.h>
#include <pwd.h>
#include <errno.h>
#include <getopt.h>
#include <sys/io.h> // for glibc
#include <sys/types.h>
#include <unistd.h>
#include <ctype.h>
```

```
// your GSM number goes in the 000000
```

## OT: (sort of) controlling the PC via SMS

```
#define TRUSTED "+31000000000@xxxxxx"

// the name of the user that listens for the sms email, and its location on your PC.
// please define a user name that is less obvious then the 'mysecret' in this example.
#define EMAIL "/var/mail/mysecret"

// Please define a keyword that is less obvious then the 'do ' (space required) in this example, say 'getonwithit ',
// then you can send the text
// getonwithit rm mymusic.mp3

#define KEYWORD "do "

#define READ_SIZE 65535
#define TEMP_SIZE 4096

char *strsave(char *s) /* save char array s somewhere */
{
char *p;

p = malloc( strlen(s) + 1);
if(! p) return 0;

strcpy(p, s);

return p;
} /* end function strsave */

int readline(FILE *file, char *contents)
{
int a, c, i;

if(debug_flag)
{
fprintf(stderr, "readline(): arg file=%lu\n", (long)file);
}

for(i = 0; i < READ_SIZE - 1; i++)
{
while(1)
{
c = getc(file);
a = ferror(file);
if(a)
{
perror("readline():");
continue;
}
break;
}
}
```

## OT: (sort of) controlling the PC via SMS

```
if(feof(file) )
{
fclose(file);
contents[i] = 0; /* EOF marker */
return(EOF);
}
if(c == '\n')
{
contents[i] = 0; /* string termination */
return 1; /* end of line */
}
contents[i] = c;
}
/*
```

mmm since we are here, the line must be quite long, possibly something is wrong.

Since I do not always check for a return 0 in the use of this function, just to be safe, gona force a string termination.

This prevents stack overflow, and variables getting overwritten.

```
*/
contents[i] = 0; /* force string termination */
if(debug_flag)
{
fprintf(stderr,\
"readline(): line to long, returning 0 contents=%s\n", contents);
}
return 0;
} /* end function readline */
```

```
int main(int argc, char **argv)
{
int a;
FILE *fptr;
char temp[READ_SIZE];
char temp2[TEMP_SIZE];
char *email;
char *keyword;
int in_message_flag;
char *ptr;
char *command;
FILE *pptr;
int trusted_flag;
```

```
email = strsave(EMAIL);
if(! email)
{
fprintf(stderr, "test-sms-email: main(): could not allocate space for email, aborting.\n");
}
exit(1);
```

## OT: (sort of) controlling the PC via SMS

```
}

keyword = strsave(KEYWORD);
if(! keyword)
{
fprintf(stderr, "test-sms-email: main(): could not allocate space for keyword, aborting.\n");

exit(1);
}

while(1)
{
fptr = fopen(email, "r");
if(fptr)
{
in_message_flag = 0;
trusted_flag = 0;
while(1)
{
a = readline(fptr, temp); // closes file!
if(a == EOF)
{
in_message_flag = 0;

break;
}
// for debug only
// fprintf(stderr, "temp=%s\n", temp);

/* process lines */

ptr = strstr(temp, "From: ");
if(ptr)
{
fprintf(stderr, "Found From: \n");

if(strcmp(ptr + strlen("From: "), TRUSTED) == 0)
{
fprintf(stderr, "Trusted sender.\n");

trusted_flag = 1;
}
else
{
fprintf(stderr, "Sender incorrect.\n");
}

} /* end if From: */

if(trusted_flag)
{
```

## OT: (sort of) controlling the PC via SMS

```
/* look for ">--- Start message ---" */
if(strstr(temp, "Start message") )
{
fprintf(stderr, "Have start, temp=%s\n", temp);

in_message_flag = 1;
}

/* look for ">--- End message ---" */
if(strstr(temp, "End message") )
{
fprintf(stderr, "Have end, temp=%s\n", temp);

in_message_flag = 0;
}

if(in_message_flag)
{
/* parse the command lines */

ptr = strstr(temp, keyword);
if(ptr)
{
command = ptr + strlen(keyword);

fprintf(stderr, "test-sms-email: main(): executing %s\n", command);

pptr = popen(command, "w");
pclose(pptr);

#ifdef HAVE_SCRIPT
// this will only work if you have the gsm-alert script (ask me), and confirm the reception of the SMS, not
strictly needed.
/* confirm */
sprintf(temp2, "/usr/local/sbin/gsm-alert %s CONFIRMED", command);
fprintf(stderr, "executing %s\n", temp2);
popen(temp2, "w");
pclose(pptr);
#endif // HAVE_SCRIPT
}
} /* end if in_message_flag */

} /* end if trusted_flag */

} /* end while read lines from email */

a = unlink(email);
if(a < 0)
{
fprintf(stderr, "test-sms-email: main(): could not erase %s, aborting.\n", email);
```

## OT: (sort of) controlling the PC via SMS

```
exit(1);
}

trusted_flag = 0;
} /* end if email exists */

/* sleep 3 seconds */
usleep(3000000);
} /* end while check for email */

/* never here */
exit(0);
} /* end function main */
```

---

Have fun :-)

### WARNING:

<claimer>

In theory emails can be falsified with a false From: and content, and a user could try BAD THINGS.

Of course last time I killed a harddisk I said: Oh well, reinstall.

YOU DO MAKE BACKUPS DO YOU?

<end claimer>

USE AT YOUR OWN RISK.

THE PROGRAM GIVES FULL ROOT CONTROL.

Start with

myprogram &

to run it in background.

possibly start it automatically at boot.

.