Yaakov Katz, Jerusalem Post, 2004.08.24:

“Police drill reveals security flaws in malls

“Tel Aviv District Police infiltrated dummy bombs into two central shopping malls and a hotel in the Kiryat Ono area on Aliyah Tuesday without being detected by security guards.

“Police, from the Mesubim Station in conjunction with cadets from the Israel Police officer’s course, carried out the drill at the Kiryat Ono Mall, Yehud’s Savyonim Mall and the Avia Hotel in the city. In all three cases, an undercover police officer succeeded in infiltrating a large dummy

explosives device inside a backpack into the establishments.

“Mesubim Region Police cheif [sic] Dep.-Cmdr. Ilan Mor said that police met with security officers from the three establishments Tuesday afternoon and that they are considering filing charges against them due to the ‘severe security failures.’ ”

Assignment due today: read foreword and preface of textbook.

Assignment due 2004.08.27: read textbook Chapter 1 pages 1–14, up to “The Trinity of Trouble.”

Some examples of bugs

Sendmail is a program that
accepts mail from local users,
accepts mail from the network,
delivers mail to local users,
delivers mail to the network.

1996.09.17 version: 14207 semicolons.
1999.02.04 version: 18085 semicolons.
2000.07.19 version: 26466 semicolons.
2001.09.08 version: 35171 semicolons.
2004.07.30 version: 38014 semicolons.

Sendmail's change log
reports a huge number of bug fixes,
including 58 "SECURITY" bug fixes.

What are some of the "SECURITY" bugs?

Bug fixed 1994.03.14:

```
int first, last;
register int i;
...
i = 0;
while (isdigit(*s))
  i = i * 10 + (*s++ - '0');
first = i;
...
if (first >= tTsize)
  first = tTsize - 1;
tTvect[first] = i;
```

Impact: Any local user
can take over the machine.

How? We'll study this later.

Sendmail FAQ editor Brad Knowles, 1996.02.08:

"Sendmail is actually one of the more secure processes on the machine. In fact, I understand that Eric has gotten a lot of complaints about his tightening security up too far, and breaking certain bits of functionality that used to work and that people liked."

Bug fixed 1996.09.17:

```
    while (*tz != '\0')
      *q++ = *tz++;
```

Impact: Any local user
can take over the machine.

Bug sort-of-fixed 1996.09.17:

```
a->q_uid = ...;
a->q_gid = ...;
pw = getpwnam(user);
if (pw != NULL) {
  a->q_uid = pw->pw_uid;
  a->q_gid = pw->pw_gid;
}
```

Impact: Any local user
can read and modify messages
to local mailing lists.

What's `getpwnam`?
What's the bug?
We'll see.

Bug fixed 1996.10.17:

```
h = res_search(host,...);
```

with Sendmail running `setuid`.

Impact: Any local user
can read and destroy local mail.

Bug allegedly fixed 1996.10.17:

```
m(...,char **x,...,int xlen)
{
  int nchar = 0;
  while (...) {
    ...
    if (++nchar > xlen) break;
    *(*x)++ = ...;
  }
}
char obuf[MAXLINE + 1];
char *obp = obuf;
while (...)
  m(...,&obp,...,MAXLINE);
```

Impact: Any user on the Internet can take over the machine.

Bug fixed 1996.10.18:

```
char obuf[MAXLINE + 1];
char *obp = obuf;
while (...)
  m(...,&obp,...,
    &obp[MAXLINE] - obp);
```

Impact: Any user on the Internet can take over the machine.

The fix:

```
m(...,&obp,...,
  &obuf[MAXLINE] - obp);
```

Bug fixed 1996.11.17:

```
execv(argv[0],argv);
```

with Sendmail running `setuid`.

Impact: Any local user
can take over the machine.

bug-of-the-month club: n.

[from "book-of-the-month club", a time-honored mail-order-marketing technique in the U.S.] A mythical club which users of sendmail(8) (the Unix mail daemon) belong to; this was coined on the Usenet newsgroup comp.security.unix at a time when sendmail security holes, which allowed outside crackers access to the system, were being uncovered at an alarming rate, forcing sysadmins to update very often. Also, more completely, *fatal security bug-of-the-month club*. See also *kernel-of-the-week club*.

Source: The Jargon File

Bug fixed 2003.03.29:

```
#define NOCHAR -1
register int c;
for (;;) {
  c = *p++;
  if (...)
    *q++ = '\\';
  ...
  if (c != NOCHAR)
    if (q > ...)
      break;
}
```

Impact: Any local user, and maybe any user on the Internet, can take over the machine.

How a typical computer's stack works

Each process (each running program)
has an array called the **stack**
and a variable called the **stack pointer**.

Stack stores function parameters,
other local variables, and return addresses.

When you call

```
zork(a,b,c);
```

the computer actually does

```
*--sp = c;
*--sp = b;
*--sp = a;
*--sp = target578;
goto zork;
target578: sp += 3;
```

When the `zork` function says

```
int x[10]; int y[10];
...
y[0] = x[8];
...
```

the computer actually does

```
sp -= 20;
...
sp[0] = sp[18];
...
sp += 20;
```

When the `zork` function says

```
return;
```

the computer actually does

```
goto *sp++;
```

Example:

```
void zork(void)
{
  return;
}

int main(int argc,char **argv)
{
  zork();
  zork();
  zork();
}
```

What computer actually does:

```
void zork(void)
{
  goto *sp++;
}
int main(int argc,char **argv)
{
  *--sp = t69; goto zork;
  t69: ;
  *--sp = t79; goto zork;
  t79: ;
  *--sp = t89; goto zork;
  t89: ;
}
```

(Don't try using `sp` and variable `goto` in your code; compiler won't allow it.)

Let's trace what this program does.

Assume original `sp` is `st+512`.

st[510]	st[511]	sp	
0	0	st+512	*--sp = t69;
0	t69	st+511	goto zork;
0	t69	st+511	goto *sp++;
0	t69	st+512	t69:
0	t69	st+512	*--sp = t79;
0	t79	st+511	goto zork;
0	t79	st+511	goto *sp++;
0	t79	st+512	t79:
0	t79	st+512	*--sp = t89;
0	t89	st+511	goto zork;
0	t89	st+511	goto *sp++;
0	t89	st+512	t89:

Example:

```
void zork(int a)
{
  int b;
  b = a + 5;
}

int main(int argc,char **argv)
{
  zork(3);
}
```

What computer actually does:

```
void zork(void)
{
  --sp;
  sp[0] = sp[2] + 5;
  ++sp;
  goto *sp++;
}

int main(int argc,char **argv)
{
  *--sp = 3;
  *--sp = t76;
  goto zork;
  t76: ++sp;
}
```