Danny Brierley, Local London, 2004.09.30:

“BA flight diverted after security scare

“A flight bound for London was forced to make an emergency landing in Amsterdam following a security alert.

“The British Airways flight, which flew from Berlin, was said to have been escorted to Holland by fighter jets.

“A spokesman for the airline said the diversion had to be made after a threat was received.

“The scare comes days after a plane bound for New York from Athens was diverted to Stanstead after Greek newspapers received threats.”

Finding exploitable buffer overflows: recap

1. Look for an interesting target program on, e.g., `www.sourceforge.net`. Choose program that inspects data from an untrusted source: e.g., program that reads file from network and converts it to another format.

2. Find buffer overflow that receives untrusted input. May need quite a bit of effort: read the program; also try various inputs, watching the program in the debugger. Work backwards from unchecked array accesses; work forwards from untrusted inputs.

3. Create input that smashes return address (or other interesting pointer). Often very easy, but sometimes need to create complicated input.

4. Modify input so that function returns (or uses other smashed pointer). Often very easy, rarely very difficult.

5. Modify input to place payload at smashed return address. Usually very easy. Can usually copy payload from another target.

Reporting a security hole

Identify problem: "There is a remotely exploitable security hole in LaTeX2RTF, at least in the current version, 1.9.15."

Summarize data flow: "For example, an attacker runs the C program below and saves the output as `foo.tex`. The attacker sends `foo.tex` to you by email. You feed `foo.tex` through LaTeX2RTF on an x86-compatible computer running FreeBSD. (The documented purpose of LaTeX2RTF is to convert LaTeX files to RTF files; users are not told that they must not run LaTeX2RTF on network data.) Result: LaTeX2RTF removes all of your files in the current directory."

Explain how to install the program: "To see the attack in action under FreeBSD 4.10, first download LaTeX2RTF:

```
mkdir $HOME/tmp
cd $HOME/tmp
wget http://
  umn.dl.sourceforge.net
  /sourceforge/latex2rtf
  /latex2rtf-1.9.15.tar.gz
gunzip latex2rtf-1.9.15.tar
tar -xf latex2rtf-1.9.15.tar
cd latex2rtf-1.9.15
```

Change the `PREFIX` line in `Makefile` to `PREFIX=$(HOME)/tmp`. Then type

```
make install
```

to compile and install the program."

Explain how to run the attack: "Now save the attack program below as `attack.c`, and run it to create `foo.tex`:

```
cd $HOME/tmp
gcc -o attack attack.c
./attack >foo.tex
```

Finally, feed `attack.out` through LaTeX2RTF:

```
bin/latex2rtf <foo.tex >foo.rtf
```

This runs `rm *`, a command specified by the attacker."

Summarize the bug: "Here's the relevant bug in the program: `strcpy(expanded,macro_piece)` in `expandmacro()` fails to check for enough buffer space for a copy of `macro_piece`."

Homework procedures

Put credits at top of report: "This security hole was discovered by George W. Bush and John Kerry."

On class FreeBSD machine, send report:

```
mail homework < bug1
```

I'll check that program is deployed; attack involves reasonable (even if uncommon) user behavior; attack works; attack is new. If security hole was discovered by n-member team, each member receives $1/n$ credit. Rediscoveries receive no credit.

I'll handle public notification. Do *not* attempt to notify public yourself.

The full-disclosure debate

Charles Tomlinson, "Rudimentary treatise on the construction of locks," 1853:

"A commercial, and in some respects a social, doubt has been started within the last year or two, whether or not it is right to discuss so openly the security or insecurity of locks. Many well-meaning persons suppose that the discussion respecting the means for baffling the supposed safety of locks offers a premium for dishonesty, by showing others how to be dishonest. This is a fallacy. Rogues are very keen in their profession, and already know much more than we can teach them respecting their several kinds

of roguery. Rogues knew a good deal about lockpicking long before locksmiths discussed it among themselves, as they have lately done. If a lock—let it have been made in whatever country, or by whatever maker—is not so inviolable as it has hitherto been deemed to be, surely it is in the interest of honest persons to know this fact, because the dishonest are tolerably certain to be the first to apply the knowledge practically; and the spread of knowledge is necessary to give fair play to those who might suffer by ignorance. It cannot be too earnestly urged, that an acquintance with real facts will, in the end, be better for all parties.

"Some time ago, when the reading public was alarmed at being told how London milk is adulterated, timid persons deprecated the exposure, on the plea that it would give instructions in the art of adulterating milk; a vain fear—milkmen knew all about it before, whether they practiced it or not; and the exposure only taught purchasers the necessity of a little scrutiny and caution, leaving them to obey this necessity or not, as they pleased. . . .

"The unscrupulous have the command of much of this kind of knowledge without our aid; and there is moral and commercial justice in placing on their guard those who might possibly suffer therefrom. We

employ these stray expressions concerning adulteration, debasement, roguery, and so forth, simply as a mode of illustrating a principle—the advantage of publicity. In respect to lock-making, there can scarcely be such a thing as dishonesty of intention: the inventor produces a lock which he honestly thinks will possess such and such qualities; and he declares his belief to the world. If others differ from him in opinion concerning those qualities, it is open to them to say so; and the discussion, truthfully conducted, must lead to public advantage: the discussion stimulates curiosity, and curiosity stimulates invention. Nothing but

a partial and limited view of the question could lead to the opinion that harm can result: if there be harm, it will be much more than counterbalanced by good."

How quickly should software-security-hole information be made public? What if we *do* manage to teach the rogues something they don't know about roguery?

Short-term view: Public disclosure hurts users and programmers by forcing emergency upgrades. Hide as much information as possible, as long as possible.

Long-term view: Public disclosure creates incentive to write and use secure programs. Disclose as much information as possible, as quickly as possible.