
“Computer maker Apple has released a security update to fix more than a dozen flaws in the Jaguar and Panther versions of its flagship Mac operating system.

“According to an advisory from Apple, the most serious flaw could permit remote attackers to execute arbitrary code and potentially take over a user’s system.”
Assignment due 2004.08.25: read foreword and preface of textbook.

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

Assignment due 2004.08.30: read the rest of Chapter 1.


Assignment due 2004.09.08: read textbook Chapter 7 pages 277–308.

Results of Gaim code inspection:
Can overflow keyword as follows.

\texttt{nm_process_new_data} reads bytes \texttt{6c 00 00 00} into \texttt{val}.
\texttt{nm_process_event}(...) reads bytes \texttt{00 00 00 00}.
\texttt{handle_receive_message} reads bytes \texttt{01 00 00 00} into \texttt{size};
\texttt{00} into \texttt{guid buffer};
\texttt{00 00 00 00} into \texttt{flags};
\texttt{41 00 00 00} into \texttt{size};
backslash and 64 more bytes into \texttt{msg}.
\texttt{rtf_parse} sees backslash,
calls \texttt{rtf_parse_keyword}.
\texttt{rtf_parse_keyword} copies 64 bytes into \texttt{keyword}, overflowing it.
These bytes are read from `conn`, an SSL-encrypted network connection to a Novell GroupWise server.

Whoever controls the server can overflow keyword. Unacceptable!

Even worse, the server forwards data from other people, so those people can overflow keyword.

Even worse, the server connection might have been intercepted by an attacker forging network packets. Gaim has an SSL-encrypted connection to the attacker; Gaim has no idea that the attacker isn’t the right server. The attacker can overflow keyword.
Alphabetic x86 machine language

Some machine-language instructions handling registers int *sp and int cx:

A (i.e., 0x41) means ++cx.
D means sp += 0.25.
I means --cx.
L means sp -= 0.25.
Q means *--sp = cx.
T means *--sp = sp.
Y means cx = *sp++.
hABCD means *--sp = 0x44434241.
kaDA means sp = ((int*)cx)[17] * 65.

hABCDhABCDYIQDYAQDYIIQDYAAQD means what?
Look at *bytes* on stack:

<table>
<thead>
<tr>
<th>stack</th>
<th>cx</th>
</tr>
</thead>
<tbody>
<tr>
<td>41 42 43 44</td>
<td>hABCD</td>
</tr>
<tr>
<td>41 42 43 44</td>
<td>hABCD</td>
</tr>
<tr>
<td>41 42 43 44</td>
<td>Y</td>
</tr>
<tr>
<td>41 42 43 44</td>
<td>44434241</td>
</tr>
<tr>
<td>41 42 43 44</td>
<td>44434240</td>
</tr>
<tr>
<td>41 42 43 44</td>
<td>44434240</td>
</tr>
<tr>
<td>41 42 43 44</td>
<td>44434240</td>
</tr>
<tr>
<td>42 43 44 44</td>
<td>41444342</td>
</tr>
<tr>
<td>42 43 44 44</td>
<td>41444343</td>
</tr>
<tr>
<td>43 43 44 44</td>
<td>41444343</td>
</tr>
<tr>
<td>41 44 41 42 43 44</td>
<td>42414441</td>
</tr>
<tr>
<td>46 41 42 43 44</td>
<td>43424146</td>
</tr>
<tr>
<td>41 42 43 44</td>
<td>43424146</td>
</tr>
</tbody>
</table>
Memory now has 40 43 41 46 rather than first 41 42 43 44.

Can vary number of A’s and I’s to store any four bytes into memory.

Can repeat to store (e.g.) 40 bytes.

Can use TYkaDA (i.e., *--sp = sp; cx = *sp++; sp = ((int*)cx)[17] * 65) to change where these bytes are stored; in particular, to store them at the end of the alphabetic payload.

So a fairly long alphabetic payload can take control of the computer.