John Huggins, KX4O
7317 Ridgedale Drive
Warrenton, VA 20186

August 13, 2019

Federal Communications Commission
445 12th Street, SW
Washington, DC 20554

Re: Addendum to previous exhibit[1] demonstrating off the air (OTA) monitoring of a Winlink email exchange using, in this example, the Pactor mode.

To whom it may concern:

Please find enclosed additional information detailing further progress made on the decoding of over the air transmissions between two Pactor/Winlink stations operating on 3588.5 MHz (dial) USB.

Some of my work files are available on my web site www.hamradio.me[2] so others may examine the data and provide their input for further improvements.

An online dialog capturing information exchange and dating the various moments of success described in this document is available on a QRZ thread[3] created specifically for this effort.

Sincerely,
John S. Huggins, kx4o

[1] https://ecfsapi.fcc.gov/file/1073182572879/KX4O_Demonstration_OTA_Winlink_Decoding.pdf

[2] https://www.hamradio.me/graphs/WinlinkTests/

[3] https://forums.qrz.com/index.php?threads/decode-off-the-air-winlink-message-request-for-programming-help.668470/

Table of Contents

Review of the previous partial solution

In my previous ECFS filing[1] I detail the successful parsing of data from an SCS DR-7400 Pactor modem using its built-in PMON mode available via the serial port console (through its USB connection). Message #2 was processed with the linux command as seen in the Filelist-Readme.txt file on my www.hamradio.me web site.[4]

```
cat 2019-07-27_Message_2_Modem.txt | awk -f parsedata.awk | xxd -r -p > 2019-07-27_Message_2_Modem.bin
```

The result is a binary file. The command…

```
hexdump -C 2019-07-27_Message_2_Modem.bin > 2019-07-27_Message_2_Modem.hex
```

…provides a clear view of the file contents as follows…

```
00000000  20 54 72 69 6d 6f 64 65  20 31 2e 33 2e 32 35 2e  | Trimode 1.3.25.|
00000010  30 20 57 61 73 68 2e 20  44 43 20 41 72 65 61 20  |0 Wash. DC Area |
00000020  47 61 74 65 77 61 79 2d  4c 65 65 40 6e 32 6c 65  |Gateway-Lee@n2le|
00000030  65 2e 63 6f 6d 20 2d 20  56 41 52 20 54 72 69 6d  |e.com - VAR Trim|
00000040  6f 64 65 20 31 2e 33 2e  32 35 2e 30 20 57 61 73  |ode 1.3.25.0 Was|
00000050  68 2e 20 44 43 20 41 72  65 61 20 47 61 74 65 77  |h. DC Area Gatew|
00000060  61 79 2d 4c 65 65 40 6e  32 6c 65 65 2e 63 6f 6d  |ay-Lee@n2lee.com|
00000070  20 2d 20 56 41 52 41 20  55 73 65 72 73 20 2d 20  | - VARA Users - |
00000080  55 70 64 61 74 65 20 33  2e 30 2e 32 20 72 65 71  |Update 3.0.2 req|
00000090  75 69 72 65 64 0d 0a 4b  4d 34 48 52 52 20 68 61  |uired..KM4HRR ha|
000000a0  73 20 31 31 38 20 6d 69  6e 75 74 65 73 20 72 65  |s 118 minutes re|
000000b0  6d 61 69 6e 69 6e 67 20  77 69 74 68 20 4e 32 4c  |maining with N2L|
000000c0  45 45 0d 5b 57 4c 32 4b  2d 35 2e 30 2d 42 32 46  |EE.[WL2K-5.0-B2F|
000000d0  57 49 48 4a 4d 24 5d 0d  3b 50 51 3a 20 37 30 31  |WIHJM$].;PQ: 701|
000000e0  31 36 35 33 37 0d 43 4d  53 20 76 69 61 20 4e 32  |16537.CMS via N2|
000000f0  4c 45 45 20 3e 0d 3b 46  57 3a 20 4b 4d 34 48 52  |LEE >.;FW: KM4HR|
00000100  52 0d 5b 52 4d 53 20 45  78 70 72 65 73 73 2d 31  |R.[RMS Express-1|
00000110  2e 35 2e 32 32 2e 30 2d  42 32 46 48 4d 24 5d 0d  |.5.22.0-B2FHM$].|
00000120  3b 50 52 3a 20 30 39 39  36 36 33 34 36 0d 3b 20  |;PR: 09966346.; |
00000130  4e 32 4c 45 45 20 44 45  20 4b 4d 34 48 52 52 20  |N2LEE DE KM4HRR |
00000140  28 46 4d 31 38 49 57 29  0d 46 43 20 45 4d 20 32  |(FM18IW).FC EM 2|
00000150  59 56 41 46 45 45 43 49  42 38 4a 20 39 30 33 20  |YVAFEECIB8J 903 |
00000160  35 39 31 20 30 0d 46 3e  20 38 42 0d 59 0d 59 0d  |591 0.F> 8B.Y.Y.|
00000170  65 3a 20 2f 2f 57 4c 32  4b 20 4d 79 20 73 65 63  |e: //WL2K My sec|
00000180  6f 6e 64 20 57 69 6e 6c  69 6e 6b 20 65 6d 61 69  |ond Winlink emai|
00000190  6c 00 30 00 02 fa e2 64  87 03 00 00 ec f5 7a 1c  |l.0....d......z.|
000001a0  6d 66 fb cb e2 e6 f4 ba  37 7c fc 4e 77 13 ad 99  |mf......7|.Nw...|
000001b0  cb 61 fb 40 3e 31 81 3d  e6 f7 8b bb b0 e1 d6 e0  |.a.@>1.=........|
000001c0  57 60 d7 f0 b0 a8 4f b6  b5 f0 02 ff 2c 42 fd cf  |W`....O.....,B..|
000001d0  f7 d4 0a 38 82 54 9b ca  2f df e6 5c ae be 2f 03  |...8.T../..\../.|
000001e0  a9 de 90 9e 1c 99 78 17  e3 92 ef c8 d1 ce 9b 1c  |......x.........|
000001f0  03 ee fb 59 7b ec e5 ca  7c f1 0e d6 0c 7f 62 ac  |...Y{...|.....b.|
00000200  9a af 29 57 ff b3 8a 77  fa f8 3a c6 85 f7 f0 a5  |..)W...w..:.....|
00000210  47 dd e8 13 16 8e 9c 4e  42 14 66 24 72 78 42 dc  |G......NB.f$rxB.|
00000220  bf 7c 58 40 eb 14 5c 22  83 45 02 57 2c 90 41 2d  |.|X@..\".E.W,.A-|
00000230  2d 5e b4 77 bf b1 8a 8a  98 91 ae 02 37 e5 f7 dd  |-^.w........7...|
00000240  0e 0c 84 9c 11 f8 81 61  13 41 c2 ec c1 42 f2 a5  |.......a.A...B..|
00000250  94 25 f5 46 88 b1 06 1a  7f 81 bb fe 9a c7 2e ba  |.%.F............|
00000260  24 f3 c3 43 f5 fd d2 2d  e4 60 f6 bb 43 52 7b e6  |$..C...-.`..CR{.|
00000270  85 82 f1 99 74 ae f6 0a  b4 48 80 35 c3 63 b1 f5  |....t....H.5.c..|
00000280  f5 de 8c 6e 88 b7 20 9d  1d 99 9b 00 50 61 b7 84  |...n.. .....Pa..|
00000290  02 fa 9f fb 3f 9d ae c5  f8 cd b5 10 cb 8a bd fe  |....?...........|
000002a0  ab dd e6 47 e6 d3 53 97  97 05 eb a9 5e f3 8e 61  |...G..S.....^..a|
```

[4] https://www.hamradio.me/graphs/WinlinkTests/Filelist-Readme.txt

```
000002b0  ea b8 22 a2 18 55 d6 6a  eb 7d 2a fc cc f9 61 da  |.."..U.j.}*...a.|
000002c0  da 07 57 f2 d5 76 0c 88  fa ee 69 97 f8 9e ee ce  |..W..v....i.....|
000002d0  6b 69 14 f5 57 44 b9 59  b6 80 b8 32 37 c8 7d 56  |ki..WD.Y...27.}V|
000002e0  b2 34 ff 09 b2 39 ed f1  5d 0c eb 74 84 7e 79 8f  |.4...9..]..t.~y.|
000002f0  3d 1a a6 61 d4 b9 3e 9e  d0 c7 ec 72 64 11 df 7d  |=..a..>....rd..}|
00000300  79 7c 52 0e db 9f 74 ce  62 1f 22 ec e3 e9 25 bf  |y|R...t.b."...%.|
00000310  c8 2c 82 18 e9 a9 cc 1b  78 b1 98 ec 44 9c 34 dc  |.,......x...D.4.|
00000320  e5 bb 9a 4f ce b7 b4 9c  6f b6 6a 14 5e 87 af 35  |...O....o.j.^..5|
00000330  e9 a7 61 f1 2b 61 d5 d9  e7 41 60 25 b5 70 91 19  |..a.+a...A`%.p..|
00000340  4a 42 4a d3 54 d1 5c 98  ca 1f b9 40 d1 45 37 8c  |JBJ.T.\....@.E7.|
00000350  05 5c dd 8c 9a dd fa 51  54 a3 c3 be 05 77 48 b5  |.\.....QT....wH.|
00000360  2c 2d 9a 37 f5 d5 6d 14  1e 9f 06 aa 25 51 31 c8  |,-.7..m.....%Q1.|
00000370  5e 9e fa 1e c9 3c ee a5  55 24 6d 6c 60 de aa d8  |^....<..U$ml`...|
00000380  7b ae be 95 0d d8 b0 8a  11 ca 26 98 02 5b 1a 62  |{.........&..[.b|
00000390  58 b1 34 81 6b 13 09 de  10 9a 67 e8 88 ef 8f 40  |X.4.k.....g....@|
000003a0  4b df 31 bd 4b 93 f5 50  c4 c6 14 80 4d 02 a6 fa  |K.1.K..P....M...|
000003b0  08 a3 a1 18 e3 5a 4a 8a  72 92 44 04 6d 24 18 59  |.....ZJ.r.D.m$.Y|
000003c0  d3 3d 01 78 8d 5a 97 a0  bb 72 b2 82 a6 82 86 dc  |.=.x.Z...r......|
000003d0  97 35 79 37 53 00 8f c6  95 61 5a 80 01 a2 07 03  |.5y7S....aZ.....|
000003e0  66 34 e5 88 cb 53 20 4f  e0 04 59                 |f4...S O..Y|
000003eb
```

This is the same data seen in Appendix B of my original ECFS Exhibit.[1]

As demonstrated previously, the compressed information representing the email content begins at byte position 0x194 with the value of 0x02 as this was the only start point, after trial and error of various start positions, that produced a recognizable message after decompression with lzhuf.

The resulting binary file is the same file seen in Appendix C of my original ECFS Exhibit,[1] but is copied on the next page for reference.

The compressed email message encapsulated as FBB packets

```
00000000  02 fa e2 64 87 03 00 00  ec f5 7a 1c 6d 66 fb cb  |...d......z.mf..|
00000010  e2 e6 f4 ba 37 7c fc 4e  77 13 ad 99 cb 61 fb 40  |....7|.Nw....a.@|
00000020  3e 31 81 3d e6 f7 8b bb  b0 e1 d6 e0 57 60 d7 f0  |>1.=........W`..|
00000030  b0 a8 4f b6 b5 f0 02 ff  2c 42 fd cf f7 d4 0a 38  |..O.....,B.....8|
00000040  82 54 9b ca 2f df e6 5c  ae be 2f 03 a9 de 90 9e  |.T../..\../.....|
00000050  1c 99 78 17 e3 92 ef c8  d1 ce 9b 1c 03 ee fb 59  |..x............Y|
00000060  7b ec e5 ca 7c f1 0e d6  0c 7f 62 ac 9a af 29 57  |{...|.....b...)W|
00000070  ff b3 8a 77 fa f8 3a c6  85 f7 f0 a5 47 dd e8 13  |...w..:.....G...|
00000080  16 8e 9c 4e 42 14 66 24  72 78 42 dc bf 7c 58 40  |...NB.f$rxB..|X@|
00000090  eb 14 5c 22 83 45 02 57  2c 90 41 2d 2d 5e b4 77  |..\".E.W,.A--^.w|
000000a0  bf b1 8a 8a 98 91 ae 02  37 e5 f7 dd 0e 0c 84 9c  |........7.......|
000000b0  11 f8 81 61 13 41 c2 ec  c1 42 f2 a5 94 25 f5 46  |...a.A...B...%.F|
000000c0  88 b1 06 1a 7f 81 bb fe  9a c7 2e ba 24 f3 c3 43  |............$..C|
000000d0  f5 fd d2 2d e4 60 f6 bb  43 52 7b e6 85 82 f1 99  |...-.`..CR{.....|
000000e0  74 ae f6 0a b4 48 80 35  c3 63 b1 f5 f5 de 8c 6e  |t....H.5.c.....n|
000000f0  88 b7 20 9d 1d 99 9b 00  50 61 b7 84 02 fa 9f fb  |.. .....Pa......|
00000100  3f 9d ae c5 f8 cd b5 10  cb 8a bd fe ab dd e6 47  |?..............G|
00000110  e6 d3 53 97 97 05 eb a9  5e f3 8e 61 ea b8 22 a2  |..S.....^..a..".|
00000120  18 55 d6 6a eb 7d 2a fc  cc f9 61 da da 07 57 f2  |.U.j.}*...a...W.|
00000130  d5 76 0c 88 fa ee 69 97  f8 9e ee ce 6b 69 14 f5  |.v....i.....ki..|
00000140  57 44 b9 59 b6 80 b8 32  37 c8 7d 56 b2 34 ff 09  |WD.Y...27.}V.4..|
00000150  b2 39 ed f1 5d 0c eb 74  84 7e 79 8f 3d 1a a6 61  |.9..]..t.~y.=..a|
00000160  d4 b9 3e 9e d0 c7 ec 72  64 11 df 7d 79 7c 52 0e  |..>....rd..}y|R.|
00000170  db 9f 74 ce 62 1f 22 ec  e3 e9 25 bf c8 2c 82 18  |..t.b."...%..,..|
00000180  e9 a9 cc 1b 78 b1 98 ec  44 9c 34 dc e5 bb 9a 4f  |....x...D.4....O|
00000190  ce b7 b4 9c 6f b6 6a 14  5e 87 af 35 e9 a7 61 f1  |....o.j.^..5..a.|
000001a0  2b 61 d5 d9 e7 41 60 25  b5 70 91 19 4a 42 4a d3  |+a...A`%.p..JBJ.|
000001b0  54 d1 5c 98 ca 1f b9 40  d1 45 37 8c 05 5c dd 8c  |T.\....@.E7..\..|
000001c0  9a dd fa 51 54 a3 c3 be  05 77 48 b5 2c 2d 9a 37  |...QT....wH.,-.7|
000001d0  f5 d5 6d 14 1e 9f 06 aa  25 51 31 c8 5e 9e fa 1e  |..m.....%Q1.^...|
000001e0  c9 3c ee a5 55 24 6d 6c  60 de aa d8 7b ae be 95  |.<..U$ml`...{...|
000001f0  0d d8 b0 8a 11 ca 26 98  02 5b 1a 62 58 b1 34 81  |......&..[.bX.4.|
00000200  6b 13 09 de 10 9a 67 e8  88 ef 8f 40 4b df 31 bd  |k.....g....@K.1.|
00000210  4b 93 f5 50 c4 c6 14 80  4d 02 a6 fa 08 a3 a1 18  |K..P....M.......|
00000220  e3 5a 4a 8a 72 92 44 04  6d 24 18 59 d3 3d 01 78  |.ZJ.r.D.m$.Y.=.x|
00000230  8d 5a 97 a0 bb 72 b2 82  a6 82 86 dc 97 35 79 37  |.Z...r.......5y7|
00000240  53 00 8f c6 95 61 5a 80  01 a2 07 03 66 34 e5 88  |S....aZ.....f4..|
00000250  cb 53 20 4f e0 04 59                              |.S O..Y|
```

During online discussions with Gordon Gibby (KX4Z)[3], I learned the formatting of this data block conforms to a format born in the early FBBS system and used far and wide in similar systems. The red highlighting indicates boundaries between FBB data blocks. Gibby's research found data blocks begin with 0x02 followed by a length byte of the following data block. 0x04 marks the end of file followed by a form of single byte CRC for content verification. When I feed the above binary file to my copy of the lzhuf decode algorithm, I achieve decode as follows…

The partially decoded email message

MID: 2YVAFEECIB8J
Date: 2019/07/29 16:29
Type: Private
From: KM4HRR
To: KW4SHP
Subject: Re: //WL2K My second Winlink email
Mbo: KM4HRR
Body: 748

Fanatstic! Awesome stuff. Look slike it's working justr fine. Congrats!!!

73,
Brendan KM4HR

----- Message from KW4SHP sent 2019/07/28 2w<9A>6:4L --- M
Fassage frH<E2>H<ED> <89><E9>b<B3>lsP n ^O<89><F4><B6><8C>hQnU
<C1>I<89>^\^Rl9 9 <F4><BA>rW R:o.Y<B2>7
W1<D0>fyjL<AA>=<D0>]<95><FC>2<89> .:z<EC>^Ms:
g'w*c*sON <9A><C1>8^Y<8C>^Sf<A4>M 6<A7> ook slike it's working justr fine. Congrats!!!

73,
Br4SHP sent 2019 Fή7bX i C.^P<F2>^L <92> .4
^A7^E<89><82>9 f<A6>^]e<C5>/<9D> Py^^ n

The decode does not complete in a timely fashion and the program has to be killed, but this shows promise nonetheless. Interestingly, the uncompressing operation performs a hitherto thought impossible partial decode of the original content. While it is apparent the bulk of the message is sent to the recipient up to and including the "Message from KW4SHP line," it seems there should be more to the message. This was a reasonable milestone regardless, but begged for more reverse engineering.

Digging into lzhuf compressed file format

I studied what resources I could find concerning the format used by the lzhuf.c program. What we are using isn't quite the same as the more popular 1990s versions (not related to communication). The first clue is the location of the number 903 (0x0387) representing the unencoded/uncompressed file size as reported by the ASCII preamble of the full Winlink transfer file located just before the compressed message portion. As Gibby surmised after studying the lzhuf.c source code, the first four bytes should be this 0x0387 value as an unsigned integer in little endian format. I found other resources suggesting the first eight bytes are reserved for this size parameter and other unknown attributes followed by the compressed data stream starting on the ninth byte. Following this advice, I edited the binary file with a program called hexer (one of many choices out there for command line hex editing).

Based on the previous partial decode, I suspected the starting point for the actual compressed data must be byte number nine. Thus, I moved the 0x0387 value to be at the start of the file (in little endian format) and padded the next six bytes with 00. The suspected first byte of compressed data was left at the ninth byte. Here are the first four lines from a hex dump of each file (before and after edits) for easy compare of the first line…

```
head -4 Message_2_Packet_All3.hex ztemptest.hex
==> Message_2_Packet_All3.hex <==
00000000 02 fa e2 64 87 03 00 00 ec f5 7a 1c 6d 66 fb cb |...d......z.mf..|
00000010 e2 e6 f4 ba 37 7c fc 4e 77 13 ad 99 cb 61 fb 40 |....7|.Nw....a.@|
00000020 3e 31 81 3d e6 f7 8b bb b0 e1 d6 e0 57 60 d7 f0 |>1.=........W`..|
00000030 b0 a8 4f b6 b5 f0 02 ff 2c 42 fd cf f7 d4 0a 38 |..O.....,B.....8|

==> ztemptest.hex <==
00000000 87 03 00 00 00 00 00 00 ec f5 7a 1c 6d 66 fb cb |..........z.mf..|
00000010 e2 e6 f4 ba 37 7c fc 4e 77 13 ad 99 cb 61 fb 40 |....7|.Nw....a.@|
00000020 3e 31 81 3d e6 f7 8b bb b0 e1 d6 e0 57 60 d7 f0 |>1.=........W`..|
00000030 b0 a8 4f b6 b5 f0 02 ff 2c 42 fd cf f7 d4 0a 38 |..O.....,B.....8|
```

The movement of the 0x0387 value to bytes at file address position 0x0 and 0x1 over wrote the “02 FA” thereby eliminating the first FBB header flag and length. The file’s byte index 0x8 (ninth byte) we see EC F5 7A 1C, etc. that allegedly begin the actual lzhuf compressed data.

Not shown above, I deleted the second FBB framing bytes "02 FA" along with the "02 5B" near the end of the file. I also deleted the 04 end of file marker and tried to remove the checksum byte value 59, but hexer would not let me remove the last byte for some reason. I moved on regardless. Here are links to the binary and hex dump of the resulting file.

- https://www.hamradio.me/graphs/WinlinkTests/ztemptest.bin
- https://www.hamradio.me/graphs/WinlinkTests/ztemptest.hex

The next page contains the hex dump of the .bin file…

The reorganized compressed file

```
00000000  87 03 00 00 00 00 00 00  ec f5 7a 1c 6d 66 fb cb  |..........z.mf..|
00000010  e2 e6 f4 ba 37 7c fc 4e  77 13 ad 99 cb 61 fb 40  |....7|.Nw....a.@|
00000020  3e 31 81 3d e6 f7 8b bb  b0 e1 d6 e0 57 60 d7 f0  |>1.=........W`..|
00000030  b0 a8 4f b6 b5 f0 02 ff  2c 42 fd cf f7 d4 0a 38  |..O.....,B.....8|
00000040  82 54 9b ca 2f df e6 5c  ae be 2f 03 a9 de 90 9e  |.T../..\../.....|
00000050  1c 99 78 17 e3 92 ef c8  d1 ce 9b 1c 03 ee fb 59  |..x............Y|
00000060  7b ec e5 ca 7c f1 0e d6  0c 7f 62 ac 9a af 29 57  |{...|.....b...)W|
00000070  ff b3 8a 77 fa f8 3a c6  85 f7 f0 a5 47 dd e8 13  |...w..:.....G...|
00000080  16 8e 9c 4e 42 14 66 24  72 78 42 dc bf 7c 58 40  |...NB.f$rxB..|X@|
00000090  eb 14 5c 22 83 45 02 57  2c 90 41 2d 2d 5e b4 77  |..\".E.W,.A--^.w|
000000a0  bf b1 8a 8a 98 91 ae 02  37 e5 f7 dd 0e 0c 84 9c  |........7.......|
000000b0  11 f8 81 61 13 41 c2 ec  c1 42 f2 a5 94 25 f5 46  |...a.A...B...%.F|
000000c0  88 b1 06 1a 7f 81 bb fe  9a c7 2e ba 24 f3 c3 43  |............$..C|
000000d0  f5 fd d2 2d e4 60 f6 bb  43 52 7b e6 85 82 f1 99  |...-.`..CR{.....|
000000e0  74 ae f6 0a b4 48 80 35  c3 63 b1 f5 f5 de 8c 6e  |t....H.5.c.....n|
000000f0  88 b7 20 9d 1d 99 9b 00  50 61 b7 84 9f fb 3f 9d  |.. .....Pa....?.|
00000100  ae c5 f8 cd b5 10 cb 8a  bd fe ab dd e6 47 e6 d3  |.............G..|
00000110  53 97 97 05 eb a9 5e f3  8e 61 ea b8 22 a2 18 55  |S.....^..a.."..U|
00000120  d6 6a eb 7d 2a fc cc f9  61 da da 07 57 f2 d5 76  |.j.}*...a...W..v|
00000130  0c 88 fa ee 69 97 f8 9e  ee ce 6b 69 14 f5 57 44  |....i.....ki..WD|
00000140  b9 59 b6 80 b8 32 37 c8  7d 56 b2 34 ff 09 b2 39  |.Y...27.}V.4...9|
00000150  ed f1 5d 0c eb 74 84 7e  79 8f 3d 1a a6 61 d4 b9  |..]..t.~y.=..a..|
00000160  3e 9e d0 c7 ec 72 64 11  df 7d 79 7c 52 0e db 9f  |>....rd..}y|R...|
00000170  74 ce 62 1f 22 ec e3 e9  25 bf c8 2c 82 18 e9 a9  |t.b."...%..,....|
00000180  cc 1b 78 b1 98 ec 44 9c  34 dc e5 bb 9a 4f ce b7  |..x...D.4....O..|
00000190  b4 9c 6f b6 6a 14 5e 87  af 35 e9 a7 61 f1 2b 61  |..o.j.^..5..a.+a|
000001a0  d5 d9 e7 41 60 25 b5 70  91 19 4a 42 4a d3 54 d1  |...A`%.p..JBJ.T.|
000001b0  5c 98 ca 1f b9 40 d1 45  37 8c 05 5c dd 8c 9a dd  |\....@.E7..\....|
000001c0  fa 51 54 a3 c3 be 05 77  48 b5 2c 2d 9a 37 f5 d5  |.QT....wH.,-.7..|
000001d0  6d 14 1e 9f 06 aa 25 51  31 c8 5e 9e fa 1e c9 3c  |m.....%Q1.^....<|
000001e0  ee a5 55 24 6d 6c 60 de  aa d8 7b ae be 95 0d d8  |..U$ml`...{.....|
000001f0  b0 8a 11 ca 26 98 1a 62  58 b1 34 81 6b 13 09 de  |....&..bX.4.k...|
00000200  10 9a 67 e8 88 ef 8f 40  4b df 31 bd 4b 93 f5 50  |..g....@K.1.K..P|
00000210  c4 c6 14 80 4d 02 a6 fa  08 a3 a1 18 e3 5a 4a 8a  |....M........ZJ.|
00000220  72 92 44 04 6d 24 18 59  d3 3d 01 78 8d 5a 97 a0  |r.D.m$.Y.=.x.Z..|
00000230  bb 72 b2 82 a6 82 86 dc  97 35 79 37 53 00 8f c6  |.r.......5y7S...|
00000240  95 61 5a 80 01 a2 07 03  66 34 e5 88 cb 53 20 4f  |.aZ.....f4...S O|
00000250  e0 59                                            |.Y|
```

We hope I ran the lzhuf program with the “d” attribute on the above binary data…

```
./lzhuf d ztemptest.bin ztemptest.txt
```

The result was a successful decode of the entire email message…

- https://www.hamradio.me/graphs/WinlinkTests/ztemptest.txt

See the next page for the fully decoded email message of exactly 903 bytes in length.

Fully decoded Winlink message after decompression with lzhuf

```
MID: 2YVAFEECIB8J
Date: 2019/07/29 16:29
Type: Private
From: KM4HRR
To: KW4SHP
Subject: Re: //WL2K My second Winlink email
Mbo: KM4HRR
Body: 748

Fanatstic! Awesome stuff. Look slike it's working justr fine. Congrats!!!

73,
Brendan KM4HR

----- Message from KW4SHP sent 2019/07/28 23:41 -----

Message ID: FO40YS492PHY
Date: 2019/07/28 23:41
From: KW4SHP
To: KM4HRR
Source: KW4SHP
Subject: //WL2K My second Winlink email

Brendan:

Just completed my Winlink HT setup.

I purchased a mobilinkd TNC3 and attached it to my Baofeng BF-F8HP with a SlimJim in the attic.
I'm using Bluetooth from the TNC3 to my desktop PC running Winlink software.

Thanks for stoking my interest in this at Field Day.

My goal is to replace the PC with a Raspberry Pi using my Android Cell as a mouse/keyboard/display
via VNC over Wifi to the Raspberry.

Baby steps....

73
Steve Palmer
```

The previously undecipherable binary data after the second FBB framing packet header contains the remaining text from the original email below the response portion. It now appears to be entirely decoded. As well the lzhuf program terminated properly and created a file exactly 903 bytes long as you can see above and in the file on the web site...

- https://www.hamradio.me/graphs/WinlinkTests/ztemptest.txt

This confirms there is an extra step to convert the FBB format with the packet framing bytes to the file/stream needed to feed my particular flavor of lzhuf. In other words, Winlink compresses the email message before parsing it into FBB packets. I previously got lucky the 02 FA and other info just happened to place the start of the actual compressed data at the 9th byte so I at least decoded the data in the first FBB packet up to the next 02 FA marker.

In my previous attempt to decode the message with lzhuf[1] the length value from the first four bytes 0x64E2FA02 told the program a ridiculously large uncompressed file size of 1,692,596,738 bytes… an obviously incorrect value and a good reason why it required termination.

Confirmation of the above findings are found by using the lzhuf program with the "e" option to compress a copy of George Washington's farewell address as stored on my web site.[5]

```
ls -l
-rw-rw-r-- 1 jhuggins jhuggins 81291 Aug  6 22:25 GW_Farewell_Speech.hex
-rw-rw-r-- 1 jhuggins jhuggins 16455 Aug  6 22:25 GW_Farewell_Speech.lzh
-rw-rw-r-- 1 jhuggins jhuggins 37033 Aug  6 22:24 GW_Farewell_Speech.txt
```

The key parameter to take away from the above file listing is the file size of the original .txt file of 37,033 bytes. The arbitrary extension .lzh is the text file after compression using lzhuf. A hex dump of this compressed file is found in the .hex file. Here are the first 4 lines of this hex dump.

```
head -4 GW_Farewell_Speech.hex
00000000  a9 90 00 00 00 00 00 00  e9 fc 7f 7f ef 9e 26 b3  |..............&.|
00000010  c7 ed ff fd 3e bf ac 10  0c 3b ac 0e 95 f6 15 a0  |....>....;......|
00000020  3b 7f 8b 4a fc df 85 85  5a 2f 67 65 a6 ee 2b dd  |;..J....Z/ge..+.|
00000030  d1 4a 7d 6b 33 aa d4 8a  5d 60 33 2e d6 4e 7c ab  |.J}k3...]`3..N|.|
```

The first four bytes a9 90 00 00 convert from little endian to the four byte unsigned integer 0x000090A9 or the decimal value 37,033… identical to the reported file size of the original text. The remaining four bytes are zero and data begins on the 9th byte as previously deduced. This more or less confirms what my copy of lzhuf expects to decode. Passing the compressed file back through lzhuf with the "d" option results in a file that hashes identically to the original text file.[6]

From this effort we learn the method to manipulate the values from the FBB portion of the Winlink message to a form decodable using the stock lzhuf compiled executable program.

All that said, there is the real possibility the version of lzhuf used in the Winlink system may include the necessary reorganizing code to handle the slightly different format of the first several bytes. More research is needed, but the overall point has been clearly made.

[5] https://www.hamradio.me/graphs/WinlinkTests/George_Washington_Speech/

[6] https://www.hamradio.me/graphs/WinlinkTests/George_Washington_Speech/hashes.txt

Why the "reverse engineering" approach

Readers of my previous ECFS Exhibit[1], QRZ forum posters and some folks in private communications, have wondered why I did not head straight to the abundantly available source code of various programs that deal with exactly what I did by hand above. I avoided these resources for as long as I could. Reasons include:

- Digging in deep to the actual binary trail provides a superb education of just how things are organized in the Winlink file transfer system. This includes the sent-in-the-clear Winlink header, FBB encapsulation of the compressed email content, and the form of compression itself.
- The trail from Pactor frames to FBB packets is blazed clearly to follow each step of the process. If nothing else this highlights the need to learn about framing and packetization. We really need an additional reminder of what level we are talking about: Pactor (Data/Modem Layer) or FBB (Application Layer). These layers are seldom explained separately in pithy and short prose targeting the non-technical audience in seats of power in government and legislature. There is a place for brevity,[7] but not at the expense of validity.[8]
- If I'm to challenge the broad-brush besmirching of Pactor or Winlink (or the combination) by erudite engineering professors of particular esteem, the long crawl through the binary steps is necessary to demonstrate any point.
- As a practicing electrical engineer, I find reverse engineering a rewarding experience. I might as well have some fun with this because it isn't paying any bills… unless the alleged award for successfully decoding a Winlink message over the air really exists[9] [10] [11] (I'm good on QRZ.com).

[7] https://youtu.be/wCKzJjaIp7k?t=16

[8] https://www.thoughtco.com/oversimplification-and-exaggeration-fallacies-3968441

[9] http://www.la3f.no/faste/digi/winlink/ExpressTutorial1130a.pdf

[10] https://ecfsapi.fcc.gov/file/1111110314487/FCC%20EX%20PARTE%2016-239%20Eric%20Burger%20Nov%2011%202018.pdf

[11] https://ecfsapi.fcc.gov/file/1210493007587/Reply%20to%20arsfi%20rebuttal%20.pdf

Examples of other partially decoded messages

Some examples of partially decoded messages are available for review on my web site in a separate subdirectory.[12] As of this writing there are two examples to step through, but as more emerge from my experimental detection process, they will be added here.

These partially decoded messages are separated in their own directory to highlight the important notion you can, with lzhuf at least, get at least part of the message so long as you have at least the first FBB frame of data to start with.

Unfortunately, both example email messages consume most of the first good packets with an almost endless parade of To: addresses before missing an FBB frame and losing decompression sync. Arguments can be made this approach towards sending bulk email might better be handled in a way that doesn't increase the email headers so greatly with correspondingly longer usage of the RF bandwidth. It's not really my place to say, but if we are all concerned about efficiency, here's a place to focus. Try an email reflector approach for these mass informational emails.

The important takeaway is I decoded messages containing missed Pactor packets, hence munged FBB packets, well enough to critique Winlink's transfer time lengthening mechanism for announcement style messages.

[12] https://www.hamradio.me/graphs/WinlinkTests/Examples_of_Partial_Decodes/

Method to determine if you have the data required for a full or partial decode

Here is a list of steps I go through to determine file validity and if it can handle a full or partial decode of a sequence of binary data pulled from a PMON capture using the parsedata.awk[13] script and written to a binary file.

1. Look past the ASCII part of the file and search for the hex values of 0x003000.
2. Check to see if the next byte is the FBB packet start value of 0x02.
3. If yes, examine the value of the next byte and store the value of the FBB packet length. In my examples it always is 0xFA or 250 unless it's the very last packet, but might be much less if this really is a single FBB packet message. Assume nothing. Record the value as FBBPacketLength.
4. Look ahead in the file FBBPacketLength.
5. Examine the value at the new byte location.
    a. If the new byte equals 0x02, go to step 3.
    b. If the new byte equals 0x04, you've reached the end of file. If you wish, calculate the modulo 8 sum of the bytes including the value of the next byte after the 0x04.
    c. If the new byte equals anything other than 0x02 or 0x04, you probably have missing data resulting from missing a Pactor frame so you can stop here.

To obtain a file suitable to feed to the lzhuf decoder, apply the steps outlined earlier to remove the FBB packet framing bytes and reorganize the header. So long as you have the first FBB packet, you should get a decode of the first portion of the email message.

From practice to production

Please understand this meticulous processing and stepping through binary files, while educational as a proof of concept experiment, is most certainly not useful in a production environment. All of this should eventually be written as a program or program suite that will be of use to as many potential volunteer monitors as possible. Hopefully readers with a programming background will see the light, so to speak, and be able to take this process to a more final form. The complete system should be ready to also support other modems such as VARA, ARDOP, etc. if those modems can provide a capability similar to the SCS PMON utility.

[13] https://www.hamradio.me/graphs/WinlinkTests/parsedata.awk

Conclusion

It's abundantly clear a great many protagonists for either side of this particular debate have partial knowledge how any of this actually works. This is especially so when discussing compression as many think the modem performs this role when in reality the modem is transparent and the Winlink application (or BPQ32[14] or DTN[15]) perform the compression role. Despite my telecommunications background, I also benefited from learning the baby steps of the data exchange to understand the larger picture. There's A LOT to take in so no one should feel ashamed for not knowing the specifics at first.

While many combine their discontent for Pactor and Winlink into one over simplified rejection of both, it's vitally important to break things down into their relevant components. To do so brings crisp focus on the actual benefit and issues of the pieces. Hopefully the work I performed provides understanding that:

- Compressed Winlink messages over Pactor may, in reality, be monitored by anyone, and
- Partial decode is possible as well thanks to the ability of lzhuf to function so long as the initial data is intact.

This is a reasonable step forward as proof of concept. My telecommunications and engineering background certainly made this task easier for me, but, as you can see, it just wasn't that difficult a task to perform. I hope to further the work to yield a functioning monitoring tool for those with interest in monitoring the Winlink system. Perhaps this work will encourage others with better programming skills than myself to take these concepts to the next level. It's clear to me this capability will be warmly received by all parties in the debate although it does seem neither side was much interested in a monitoring solution before this year's maelstrom.

The Winlink monitorability nut is cracked. Privacy on radio Winlink no longer exists… and that's a very good thing for all parties.

[14] http://www.cantab.net/users/john.wiseman/Documents/

[15] http://nts-digital.net/mw/index.php/Main_Page