smbcmp

a network capture diff tool for SMB traffic

Aurélien Aptel <aaptel@suse.com>
SUSE
Who am I?

• Aurélien Aptel
• Work in SUSE, Samba Team
• Focus on SMB kernel client aka “cifs.ko”
  – Cifs-utils, Wireshark, Pike, …
What is this about?

• Different debugging approaches I use
• Some new features I worked on

• Mostly useful to developers
• But also for administrators, to diagnose network issues
Debugging is hard

• No silver bullet

• Some approaches work better than others for certain bugs

• SMB bugs
  – In client?
  – In server?
  – Both?
  – Specifications wrong?
  – Unspecified?

• Lot of possible failures
  – Goal: isolate as much as possible before digging in
Different versions: git bisect

- **Setup**
  - Find “good” commit
  - Find “bad” commit
- **Dichotomy**
  - Tries to find first bad commit
  - Checkouts intermediaries commits you can test
  - Search space divided by 2 at each step
  - $N$ commits $\rightarrow O(\log N)$ steps to determine first bad commit
  - Really powerful: 130k commits in 17 steps
- **Can be automated**
  - Reproduce script
    - Indicate if “good” or “bad” via the exit code
  - `git bisect run myscript.sh`
Code reading

• The inevitable code/doc-reading part
  – Reading the spec one time to get an idea of how it’s supposed to work at the protocol layer
  – Finding the corresponding codepath
  – Reading source code of the relevant functions
  – Look for bug, typos, and wrong logic wrt the specs
  – Repeat

• Amount of code to grok can be very big
  – Long process, easy to miss the bug
Different implementations

• Sometime there are no good commits or its very impractical to find
• Try different combination of servers/clients
  • Windows, samba, smbclient, cifs.ko
• Try writing a test client that only does the buggy steps
  – Samba torture test framework
  – Pike (https://github.com/emc-isilon/pike)
    • Clean, pure-python, SMB2/3 lib, with easily tweakable fields
    • Used to test SMB3 POSIX extensions (https://github.com/aaptel/pike/commits/smb3unix)
  – Microsoft has open-sourced a massive testing framework
    • https://github.com/Microsoft/WindowsProtocolTestSuites
Debugger

• Good tool but often impractical
• Breakpoints = timeouts
• Samba
  – Forks for user sessions
  – `set follow-fork-mode child`
    – `set detach-on-fork off`
• Kernel
  – Qemu gdb server
  – `qemu ... -s`
  – `gdb -ex 'add-auto-load-safe-path /'
    -ex 'target remote :1234' vmlinux`
Debugger

• Python helper funcs in kernel.git
• Kernel cannot be compiled without optimization
  – Out of order execution
  – dreaded <optimized out>
  – Inline code
  – Since GCC v4.8 '-Og'
    “kernel hacking: GCC optimization for better debug experience (-Og)”
• https://www.mail-archive.com/linux-kernel@vger.kernel.org/msg1707708.html
Logs

• Samba
  – smb.conf
    • Log level = 10
  – Smblog-mode for emacs :)
    • DEMO
Logs

• Samba
  – smb.conf
    • Log level = 10
    • Smblog-mode for emacs :)

• Kernel
  • echo 1 > /proc/fs/cifs/cifsFYI
  • echo 8 > /proc/sys/kernel/printk
  • echo 1 > /sys/module/dns_resolver/parameters/debug
  • echo "module cifs +p" > /sys/kernel/debug/dynamic_debug/control
  • echo 'file fs/cifs/* +p' > /sys/kernel/debug/dynamic_debug/control

• ftrace / trace-cmd
  • Record call graph
    • https://jvns.ca/blog/2017/03/19/getting-started-with-ftrace/
Kernel logs: ftrace

- Deeper strace
- Records call graph
  - `trace-cmd record -e all -p function_graph -F \
    mount.cifs //localhost/myshare /mnt -o ...
  - `trace-cmd report`

...
Kernel logs: ftrace

• System wide recording
• Filter for specific syscalls (mount 165, umount 166)

```
# trace-cmd record -e sys_enter -f id==165
Hit Ctrl^C to stop recording
^C
# trace-cmd report
mount.cifs-21482 [001] ...: sys_enter: NR 165 (...)
```
Kernel logs: ftrace

- Usable without trace-cmd
- Fs-like API via /sys/kernel/debug/tracing

```bash
#!/bin/bash
set -v
d=/sys/kernel/debug/tracing

# set event and filter
echo sys_enter > $d/set_event
echo id==166 > $d/events/raw_syscalls/sys_enter/filter

# start/wait/stop tracing
echo 1 > $d/tracing_on
read -p "recording... press enter to stop"
echo 0 > $d/tracing_on

# print & clear
cat $d/trace
echo 0 > $d/trace
```

```plaintext
# tracer: nop
#
# TASK-PID  CPU#  TIMESTAMP FUNCTION
# umount-13991 [000] ...: sys_enter: NR 166 (...)
```
Network capture

• Wire log
• When applicable, network trace analysis is very effective
• Wireshark!
  – smb||smb2||dns||krb4
Network capture

• Wireshark decryption (3.0 and 3.11, CCM and GCM)
  – Requires wireshark 3.0.0 (28 feb 2019)

  – Samba (master)
    • Controls both client and server
    • smb.conf
      • debug encryption = yes
    • smbclient ... --option='debugencryption=yes' -e -mSMB3_11

  – Kernel (4.13+)
    • CONFIG_CIFS_DEBUG_DUMP_KEYS=y
    • Enable carefully!
Network capture

- Wireshark decryption (3.0 and 3.11)

```bash
$ smbclient //localhost/scratch --option='debugencryption=yes' \  -e -mSMB3 -U aaptel%aaptel -c quit
dump encryption: dumping generated session keys
Session Id    [0000] 26 48 BF FD 00 00 00 00
Session Key   [0000] 63 D6 CA BC 08 C8 4A D2 45 F6 AE 35 AB 4A B3 3B
Signing Key   [0000] 4E FE 35 92 AC 13 14 FC C9 17 62 B1 82 20 A4 12
App Key       [0000] A5 0F F4 8B 2F FB 0D FF F2 BF EE 39 E6 6D F5 0A
ServerIn Key  [0000] 2A 02 7E E1 D3 58 D8 12 4C 63 76 AE 59 17 5A E4
ServerOut Key [0000] 59 F2 5B 7F 66 8F 31 A0 A5 E4 A8 D8 2F BA 00 38
$ wireshark -ouat:smb2_seskey_list:2648BFFD00000000,63D6CABC08C84AD245F6AE35AB4AB33B \  -r capture.pcap
```
Network capture

- Wireshark decryption (3.0 and 3.11)

```bash
# mount.cifs //localhost/myshare /mnt -o vers=3.0,seal
# dmesg | grep CIFS
CIFS VFS: generate_smb3signingkey: dumping generated AES session keys
CIFS VFS: Session Id  31 00 00 54 64 1c 00 00
CIFS VFS: Session Key  5a 92 df 3f a4 a5 c2 52 46 06 05 e5 52 75 ca 0c
CIFS VFS: Signing Key   cb 7b 5d 7f d3 e5 21 68 74 3e 36 8f 12 da 2f 50
CIFS VFS: ServerIn Key  0a 47 11 de a8 7a 96 c2 c3 7f c5 82 3c ff ac 3f
CIFS VFS: ServerOut Key 48 81 e5 42 69 15 d1 a0 d0 70 ca 74 af f5 b3 ce

$ wireshark -ouat:smb2_seskey_list:31000054641C0000,5a92df3fa4a5c252460605e55275ca0c \ -r capture.pcap
```
Network capture

- Wireshark decryption (3.0 and 3.11)
Network capture

• Some other new changes in Wireshark SMB2 dissector:
  – Better parsing of compounded responses
  – Proper parsing of error contexts
  – Support for parsing reparse point data
    • NFS reparse tags (symlinks, block/char device, pipes, …)

  – Latest SMB3.1.1 changes:
    • Negotiate Context ID
    • Decompression of all compression types
      • LZ77, LZ77+HUFF, LZNT1
Network capture comparison

• Get a trace of a working case
• Get a network trace of the issue
• Look hard at both traces
  – try to see what the good client/server is doing that the bad one doesn’t (or vice versa)
  – Compare packets, fields, etc
Comparing network traces

- Open both traces side by side
- Expand the little handles
- Lots of them...
  - Nested
    - Into
      - Each
      - other

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>14:25:11.348881</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>SMB2</td>
<td>296</td>
<td>Session Setup Response...</td>
</tr>
<tr>
<td>65</td>
<td>14:25:11.348739</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>SMB2</td>
<td>432</td>
<td>Session Setup Request...</td>
</tr>
</tbody>
</table>

SMB2 (Server Message Block Protocol version 2)

- SMB2 Header
  - Session Setup Response (0x01)
    - StructureSize: 0x0000
    - Session Flags: 0x0000
      - .... .... ...0 = Guest: False
      - .... .... ...0. = Null: False
      - .... .... ...0.. = Encrypt: False
      - Blob Offset: 0x00000048
      - Blob Length: 152
  - Security Blob: 4e544c4d5353500000200000014000100038000000035028ae2...

- NTLM Secure Service Provider
  - NTLMSSP identifier: NTLMSSP
  - NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002)
  - Target Name: LINUX-0E2K
    - Length: 20
Comparing network traces

• Eventually you switch to a different packet and the click-dance starts again
• Impractical for multiple reasons
  – Your index hurts
  – You skip expanding some fields because “it’s never going to be different here”
    • Until it does…
  – Your 133t h4cker eyes might just miss a difference
    • whitespace, caps, slash directions, flags..?
  – Some differences are false positives
    • Timestamps, random GUID, hashes, ...
Automating the comparison

• Wireshark is great...
• Would be nice to interact with it programmatically
• API?
  – Not really :(  
  – Tshark: text output
    • Also json and xml output
  – Also a daemon version sharkd
    • Undocumented?
tshark

tshark -r smb3-aes-128-ccm.pcap -Y smb2

1 ... 10.160.64.139 → 10.160.65.202 SMB2 172 Negotiate Protocol Request
2 ... 10.160.65.202 → 10.160.64.139 SMB2 318 Negotiate Protocol Response
3 ... 10.160.64.139 → 10.160.65.202 SMB2 190 Session Setup Request, NTLMSSP_NEGOTIATE
4 ... 10.160.65.202 → 10.160.64.139 SMB2 318 Session Setup Response, Error: STATUS_
5 ... 10.160.64.139 → 10.160.65.202 SMB2 430 Session Setup Request, NTLMSSP_AUTH, User:
SUSE\administrator
6 ... 10.160.65.202 → 10.160.64.139 SMB2 142 Session Setup Response

...
**tshark**

```
tshark -r smb3-aes-128-ccm.pcap -Y smb2 -V
```

Frame 1: 172 bytes on wire (1376 bits), 172 bytes captured (1376 bits) on interface 0
- Interface id: 0 (unknown)
- Encapsulation type: Ethernet (1)
- Arrival Time: May 17, 2017 12:02:16.523633000 CEST

...  


...  

**SMB2 (Server Message Block Protocol version 2)**

- **SMB2 Header**
  - Server Component: SMB2
  - Header Length: 64
  - Credit Charge: 0
  - Channel Sequence: 0
  - Reserved: 0000
  - Command: Negotiate Protocol (0)
  - Credits requested: 2
  - Flags: 0x00000000

  ---- ---- ---- ---- ---- ---- ---- ---- 0 = Response: This is a REQUEST
  ---- ---- ---- ---- ---- ---- ---- ---- 0. = Async command: This is a SYNC command
smbcmp

• First prototype in emacs
  – https://github.com/aaptel/elshark
• Moved to Python script using curses
  – Calls tshark in the background
• 2 modes
  – Single trace
    • aka curses-wireshark (summaries + details)
  – Diff traces
    • Show 2 summaries
    • Diffs the detailed output

  – Get it at https://smbcmp.github.io
    • Available for Linux (SUSE, Redhat, Ubuntu, Debian, ...)
    • And Windows!
Google Summer of Code

• Accepted *Google Summer of Code* project this year
• “Improving smbcmp” by Paul Mairo Rufus

• Worked on using Wireshark XML output
  – More precise diff
  – Ignore fields
• A new GUI version of the tool (wxWidget)
• Search in summaries
Running on Windows (bis)
Future work

• Smbcmp
  – Deeper analysis
  – Better ignore rules
  – UI improvements
  – …
• Qemu record/replay