

Case Study: Low performance Accessing a SWITCH Webservice

As part of the Authentication and Authorization Infrastructure (AAI), SWITCH is running a webservice called WAYF (“Where Are You From”) that maps users to “Home Organizations”.

Some users report that connections to the WAYF server are very slow or stalling under certain conditions

- Source address is in a particular range
- Client is running Windows XP
- Client is using Internet Explorer

What would be a good methodology to debug this problem?

- Capture packets on both ends of a connection that exhibits the symptoms
- Try to identify artefacts that can affect TCP performance

Wireshark:

21	0.027070	130.59.31.249	192.168.1.136	SSLv3	Change Cipher Spec, Encrypted Handshake Message
22	0.000600	192.168.1.136	130.59.31.249	SSLv3	Application Data
23	0.012816	130.59.31.249	192.168.1.136	SSLv3	Application Data, [Unreassembled Packet [incorrect TCP checksum]]
24	0.000697	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
25	0.001007	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
26	0.000009	130.59.31.249	192.168.1.136	TCP	[TCP Dup ACK 25#1] 443 > 1636 [ACK] Seq=9150 Ack=786 Win=7504 Len=0
27	0.215462	130.59.31.249	192.168.1.136	SSLv3	[TCP Retransmission] Application Data,
28	0.198814	192.168.1.136	130.59.31.249	TCP	1636 > 443 [ACK] Seq=786 Ack=6230 Win=64240 Len=0
29	0.021243	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
30	0.000702	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
31	0.000306	130.59.31.249	192.168.1.136	TCP	[TCP Dup ACK 30#1] 443 > 1636 [ACK] Seq=12070 Ack=786 Win=7504 Len=0
32	0.418032	130.59.31.249	192.168.1.136	TCP	[TCP Retransmission] [TCP segment of a reassembled PDU]
33	0.160590	192.168.1.136	130.59.31.249	TCP	1636 > 443 [ACK] Seq=786 Ack=7690 Win=64240 Len=0

- Client is behind NAT
- Incorrect TCP checksums of *incoming* packets
- Some incoming packets have correct checksums

Why are bad checksums on incoming packets suspicious but less so on outgoing packets?

Investigating the Bad Checksums (1)



Corrupt outgoing checksums are to be expected if checksum offloading is being used, but incoming checksums should be correct. Try to identify outgoing packets on server by matching seq. numbers

```
▶ Frame 21 (4434 bytes on wire, 1600 bytes captured)
▶ Ethernet II, Src: Vmware 0d:05:67 (00:0c:29:0d:05:67), Dst: All-HSRP-routers_6c (00:00:0c:07:ac:6c)
▶ Internet Protocol, Src: 130.59.31.249 (130.59.31.249), Dst: 213.196.130.87 (213.196.130.87)
▼ Transmission Control Protocol, Src Port: 443 (443), Dst Port: 1636 (1636), Seq: 4770, Ack: 786, Len: 4
    Source port: 443 (443)
    Destination port: 1636 (1636)
    Sequence number: 4770 (relative sequence number)
    [Next sequence number: 9150 (relative sequence number)]
    Acknowledgement number: 786 (relative ack number)
    Header length: 20 bytes
▶ Flags: 0x0010 (ACK)
    Window size: 7504
    Checksum: 0x0b87 [unchecked, not all data available]
▶ Secure Socket Layer
    [Unreassembled Packet: SSL]
```

How can the server send such a large segment?

Investigating the Bad Checksums (2)



The server is using TCP Segmentation Offloading (TSO)

4434 byte Ethernet frame consists of

- 14 bytes Ethernet header
- 20 bytes IP header
- 20 bytes TCP header
- 4380 bytes payload = 3 regular 1460 byte TCP segments

Segmentation is performed by NIC.

What happens at the receiver with the corrupt packets?

Investigating the Bad Checksums (3)



Corrupt packets are discarded → triggers timeout at sender

21	0.027070	130.59.31.249	192.168.1.136	SSLv3	Change Cipher Spec, Encrypted Handshake Message
22	0.000600	192.168.1.136	130.59.31.249	SSLv3	Application Data
23	0.012816	130.59.31.249	192.168.1.136	SSLv3	Application Data, [Unreassembled Packet [incorrect TCP checksum]]
24	0.000697	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
25	0.001007	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
26	0.000009	130.59.31.249	192.168.1.136	TCP	[TCP Dup ACK 25#1] 443 > 1636 [ACK] Seq=9150 Ack=786 Win=7504 Len=0
27	0.215462	130.59.31.249	192.168.1.136	SSLv3	[TCP Retransmission] Application Data,
28	0.198814	192.168.1.136	130.59.31.249	TCP	1636 > 443 [ACK] Seq=786 Ack=6230 Win=64240 Len=0
29	0.021243	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
30	0.000702	130.59.31.249	192.168.1.136	SSLv3	Continuation Data, [Unreassembled Packet [incorrect TCP checksum]]
31	0.000306	130.59.31.249	192.168.1.136	TCP	[TCP Dup ACK 30#1] 443 > 1636 [ACK] Seq=12070 Ack=786 Win=7504 Len=0
32	0.418032	130.59.31.249	192.168.1.136	TCP	[TCP Retransmission] [TCP segment of a reassembled PDU]
33	0.160500	192.168.1.136	130.59.31.249	TCP	1636 > 443 [ACK] Seq=786 Ack=7690 Win=64240 Len=0

The retransmitted packet is a “normal” segment (1460 bytes TCP payload, 1500 byte IP frame) and arrives with correct checksum. The TCP connection can still progress, but very slowly.

What can we conclude from this?

Is TSO to Blame?



Hypothesis: Only packets sent with TSO are affected. The problem is expected to be on the server side.

Time to look more closely at this system.

- Linux
- Virtual host using KVM

How can we test the hypothesis?

Packet Capture on KVM Host



Virtual network adapter of a KVM guest are bridged on the KVM host through to the physical NIC.

Packet capture from the bridge on the KVM host shows that the checksums arrive corrupted from the KVM guest.

Workaround solution: turn off TSO on guest system.

Actual bug in KVM still under investigation.