z/TPF Communications Security Enhancements

Jamie Farmer
z/TPF Software Engineer

IBM z/TPF
April 11, 2016

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

10 Minutes	**OpenSSL 1.0.2**
10 Minutes	**Unix Domain Sockets Support**
10 Minutes	**Reducing Socket Lock Contention**
5 Minutes	**MQ Client Enhancements**

OpenSSL 1.0.2

Problem Statement

- Current version of OpenSSL on z/TPF is 0.9.7c
  - This version is no longer being updated by the OpenSSL community
  - Known security vulnerabilities will not be fixed in this version

- Customers need to utilize new security standards recommended by the industry
  - The 0.9.7c version does not have the latest SSL versions and ciphers

Port of OpenSSL 1.0.2

- Upgrades z/TPF to use OpenSSL 1.0.2e
  - Previous version was 0.9.7c
  - All known security vulnerabilities resolved in latest version
- What's New in OpenSSL 1.0.2e
  - Transport Layer Security versions 1.1 and 1.2
  - Secure Hash Algorithm 256 – SSL ciphers using SHA256
  - Re-enabled SSL_renegotiate() to periodically create new cipher keys for long running SSL sessions
    - Previously disabled due to security vulnerability

What is No Longer Supported?

- Major security concerns with the older SSL versions and ciphers
  - SSL version 2
  - SSL version 3
  - RC2 and RC4 encryption algorithms
- Industry is strongly recommending NOT to use these versions and ciphers

OpenSSL on z/TPF Moving Forward

- Eliminated or isolated z/TPF unique modifications to the OpenSSL ported package
  - Making porting in new OpenSSL versions easier

- When new vulnerabilities are identified in OpenSSL that effect z/TPF
  - A new version can be quickly delivered

OpenSSL Performance Measurement

- Ran comparison tests between OpenSSL 0.9.7 vs. 1.0.2
- Performance Test Configuration
  - EC12 (2827-750)
  - Single dedicated I-stream
  - SSL Client/Server on two LPARs in same zSystem server communicating through a shared OSA card
  - All system level traces disabled
- All tests run with version TLS v1.0 and the AES128-SHA cipher

OpenSSL Performance Results

	OpenSSL 0.9.7		OpenSSL 1.0.2		
Message Size	CPU Utilization	Messages / Second	CPU Utilization	Messages / Second	Improvement Ratio
500	91	10,593	97.9	35,674	~3x
32,767	94.1	1,400	97.5	18,354	~12x

** Performance results may vary

OpenSSL Summary

- Provides the latest security standards recommended by the industry
- Greatly increases the performance of SSL on z/TPF

- APARs PJ43537 & PJ42982
  - As long as applications are using supported ciphers, no application changes are required.

Unix Domain Sockets

Problem Statement

- Many ported packages assume Unix Domain sockets are available on a platform
  - z/TPF ported some packages in the past that assumed this
    - Modifications were made to those packages because Unix Domain sockets was not supported
    - Over time has become problematic
      - Trying to change certain packages to not use Unix Domain sockets is very expensive.

TCP/IP Internet Sockets

- IPv4 Internet Addressing
  - Sockets created in AF_INET family
  - bind/connect to a IP address and a port number
  - Required to communicate with remote platforms

Unix Domain Sockets

- Sockets created in AF_UNIX family
- bind/connect to a file name
- Method for inter-process communication within a single node using sockets
  - Multiple processes can send and receive data on a Unix Domain socket simultaneously
  - Duplex Communication

Unix Domain Sockets on z/TPF

- Developed for porting applications to z/TPF
  - Requirement for Java implementation on z/TPF
- Externalized for use by z/TPF customer applications and porting efforts
- z/TPF command displays updated to display Unix Domain sockets
  - ZDTCP NETSTAT ←Display network statistics
  - ZSOCK DISPLAY ←Display socket block

Unix Domain Sockets File System Considerations

- Binding to a file name on z/TPF creates the file on the z/TPF file system
  - Responsibility of application to remove this file (same behavior as Linux/Unix)

- If loosely coupled, recommended that a processor unique file system is used
  - MFS, PFS, FFS
  - If you bind to a file name, and the file already exists an error is returned.

Unix Domain Sockets Example

Server Application – Process 1

```
int server_sock = socket(AF_UNIX, SOCK_STREAM,0);
…

struct sockaddr_un server_sockaddr;
server_sockaddr.sun_family = AF_UNIX;
strcpy(server_sockaddr.sun_path, "/tmp/unixFile");
len = sizeof(server_sockaddr);
unlink("/tmp/unixFile");
rc = bind(server_sock, (struct sockaddr *)
&server_sockaddr, len);
…

rc = listen(server_sock, backlog);
…

int client_sock = accept(server_sock, NULL, 0);
…
Send and Receive messages
```

Client Application – Process 2

```
int client_sock = socket(AF_UNIX, SOCK_STREAM, 0);
…

struct sockaddr_un server_sockaddr;
server_sockaddr.sun_family = AF_UNIX;
strcpy(server_sockaddr.sun_path, "/tmp/unixFile");
int rc = connect(client_sock,
                 (struct sockaddr *)&server_sockaddr, len);
…
Send and Receive messages
```

Unix Domain – socketpair()

- The socketpair() function can be used to easily create a pair of connected TCP UNIX domain sockets
  - The socketpair function issues the socket, bind, listen, accept and connect APIs on your behalf
  - Unnamed sockets – meaning no files are created and management of files is not required.
- Use to easily create a connection between two processes
  - Primary use case is when forking a child process with the standard fork() API
    - Parent / child process communication is required.

Socketpair Example

Socketpair() creates a pair of sockets

Fork a child process

Child closes one end of pipe and sends message to parent

Parent closes other end of pipe and received message from child

```
int sockets[2];     /* integer array for sockets to be returned in */
int rc;
pid_t pid;
…

rc = socketpair(AF_UNIX, SOCK_STREAM, 0, sockets);
…

pid = fork();
…

if (pid == 0){ /* this is the child */
   close(sockets[0]); /* close the parent's socket */
   rc = write(sockets[1], CHILDDATA, sizeof(CHILDDATA)); /* write to parent */
   child_process_work(sockets[1]); /* do work on socket in child process */
}

else { /* this is the parent */
   close(sockets[1]); /* close the child's socket*/
   rc = read(sockets[0], buf, sizeof(buf));
   do_parent_process_work(sockets[0]); /* do work on socket in parent process */
}
```

Unix Domain Sockets Summary

- Easier for IBM and customers to port applications to z/TPF
- Alternate approach to inter-process communications using sockets as the transport mechanism

- APAR PJ43020 provides Unix Domain Socket support

Reducing Socket Lock Contention

Problem Statement

- For integrity a single core lock –"the socket block lock" was created for TCP/IP processing on z/TPF

- Over time, the contention on the single socket block lock has increased.
  - Increase in TCP/IP traffic
  - Increase in the number of I-streams
  - Processor technology changes in the industry has increased the penalty of lock contention

Socket Block Lock Solution

- Effort underway to reduce contention on the socket block lock
  - Eliminate the socket block lock from some critical TCP/IP paths

- Phased approach
  - Phase 1: Eliminate socket block lock from certain APIs
  - Phase 2: Update TCP/IP send API processing to do most mainline processing, for example, constructing packets without holding the lock

Phase 1: Read Only APIs

A typical customer application may look like this

1.	aor invoked	← New application ECB created to process inbound message
2.	ioctl()	← Set socket controls
3.	getsockopt()	← Get socket options
4.	getpeername()	← Get remote socket information
5.	getsockname()	← Get local socket information
6.	read()	← Read the next application message
7.	tpf_tcpip_message_cnt	← Increment inbound message counts
8.	send()	← Send the application reply
9.	tpf_tcpip_message_cnt	← Increment outbound message counts
10.	aor to read next message	← This ECB exits and when the next message comes for this socket, AOR completes, a new ECB is created that will start at step 1.

Phase 1: Performance Testing

- Created a driver modeled after a typical customer TCP/IP application
  - Measured the throughput/utilization with and without the Phase 1 enhancement
    - Measured the time spent spinning on the socket block lock
    - No application logic - driver reads message and immediately sends a response.
    - Varied the number of I-streams
    - Varied whether I-streams are shared or dedicated

Phase 1 Performance Results

	Before Changes		After Changes			
Number of Istreams	Utilization	Messages / Second	Utilization	Messages / Second	Throughput Improvement	Spin Lock Reduction
8 dedicated	61.9	43,579	64.40	58,247	28%	33%
16 shared	31.2	39,267	28.8	50,740	40%	45%

Significant reduction in lock reduction
Up to 40% improvement

** Performance results may vary

Phase 2: Reduce Socket Lock Hold Time During Send Processing

- Phase 2 project is currently underway
- Will consist of the following
  - Update TCP/IP send API processing to do most mainline processing, for example, constructing packets without holding the lock
  - Optimize processing of send/read/AOR APIs to reduce the socket block lock hold time
    - Reduce SVC calls
    - Eliminate MALOC calls

Socket Lock Contention Summary

- z/TPF performance and throughput improvements
- Preparation for future workload growth

- Phase 1 delivered in September 2015
  - APAR PJ43441
- Phase 2 in development
  - APAR PJ43697

- No application migration considerations
- No tuning or configuration required

MQ Client Maximum Message Size

Problem Statement

- The z/TPF MQ client support has a maximum message size of 30,000 bytes

- If an MQ application has a message to send to or from z/TPF that is greater than 30,000 bytes
  - Requires breaking up the message into 30,000 byte chunks before sending
  - Requires the reassembling the 30,000 byte chunks back into a contiguous message

Increased Maximum Message Size For MQ Client

- The maximum message size for an MQ client channel has been increased to 4 megabytes
  - Use the ZMQID command to update the channel definition
  - Delivered with APAR PJ43145 in May 2015

- Does not require any application changes to apply the APAR

Summary

- OpenSSL 1.0.2 (PJ43537 & PJ42982)
  - Provides the latest security standards and greatly increases the performance of SSL on z/TPF
- Unix Domain Socket (PJ43020)
  - Alternate approach to inter-process communications making it easier for IBM and customers to port applications to z/TPF
- Socket Lock Contention Enhancement (PJ43441 & PJ43697)
  - z/TPF performance and throughput improvements and preparation for future workload growth
- MQ Client Message Size Enhancement (PJ43145)
  - Increased usability of MQ client support on z/TPF

Thank you!

Questions or comments?

Trademarks

- IBM, the IBM logo, ibm.com and Rational are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Notes

- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
- Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.