POS39-C. Use the correct byte ordering when transferring data between systems

Different system architectures use different byte ordering, either little endian (least significant byte first) or big endian (most significant byte first). IA-32 is an example of an architecture that implements little endian byte ordering. In contrast, PowerPC and most Network Protocols (including TCP and IP) use big endian.

When transferring data between systems of different endianness, the programmer must take care to reverse the byte ordering before interpreting the data.

The functions `htonl()`, `htons()`, `ntohl()`, and `ntohs()` can be used to transfer between network byte ordering (big endian) and the host's byte ordering. On big endian systems, these functions do nothing. They may also be implemented as macros rather than functions.

Noncompliant Code Example

In this noncompliant code example, the programmer tries to read an unsigned 32-bit integer off a previously connected network socket.

It is important to know the sizes of your data types lest they be different on architectures that are accessible over the network. Hence, we transfer a `uint32_t` rather than an `int`. For more information, see FIO09-C. Be careful with binary data when transferring data across systems.

```
/* sock is a connected TCP socket */

uint32_t num;

if (recv(sock, (void *)&num, sizeof(uint32_t), 0) < (int)sizeof(uint32_t)) {
  /* Handle error */
}

printf("We received %u from the network!\n", (unsigned int)num);
```

This program prints out the number received from the socket using an incorrect byte ordering. For example, if the value 4 is sent from a big endian machine, and the receiving system is little endian, the value 536,870,912 is read. This problem can be corrected by sending and receiving using network byte ordering.

Compliant Solution

In this compliant solution, the programmer uses the `ntohl()` function to convert the integer from network byte order to host byte ordering:

```
/* sock is a connected TCP socket */

uint32_t num;

if (recv(sock, (void *)&num, sizeof(uint32_t), 0) < (int)sizeof(uint32_t)) {
  /* Handle error */
}

num = ntohl(num);
printf("We recieved %u from the network!\n", (unsigned int)num);
```

The `ntohl()` function (network to host long) translates a `uint32_t` value into the host byte ordering from the network byte ordering. This function is always appropriate to use because its implementation depends on the specific system's byte ordering. Consequently, on a big endian architecture, `ntohl()` does nothing.

The reciprocal function `htonl()` (host to network long) should be used before sending any data to another system over network protocols.

Portability Details

- `ntohs()`, `ntohl()`, `htons()`, and `htonl()` are not part of the C Standard and are consequently not guaranteed to be portable to non-POSIX systems.
- The POSIX implementations of `ntohs()`, `ntohl()`, `htons()`, and `htonl()` take arguments of types `uint16_t` and `uint32_t` and can be found in the header file `<arpa/inet.h>`.
- The Windows implementations use `unsigned short` and `unsigned long` and can be found in the header file `<winsock2.h>`.
- Other variants of `ntoht()` and `htont()`, such as `ntohi()`/`htoni()` or `ntohll()`/`htonll()`, may exist on some systems.

Risk Assessment

If the programmer is careless, this bug is likely. However, it will immediately break the program by printing the incorrect result and therefore should be caught by the programmer during the early stages of debugging and testing. Recognizing a value as in reversed byte ordering, however, can be difficult depending on the type and magnitude of the data.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
POS39-C	Medium	Likely	Low	P18	L1

Automated Detection

Tool	Version	Checker	Description
Axivion Bauhaus Suite	6.9.0	**CertC-POS39**	
Klocwork	2018	**BYTEORDER.NTOH.RECV** **BYTEORDER.NTOH.READ** **BYTEORDER.HTON.SEND** **BYTEORDER.HTON.WRITE**	
Parasoft C/C++test	10.4.2	**CERT_C-POS39-a**	Use the correct byte ordering when transferring data between systems
Polyspace Bug Finder	R2019a	CERT C: Rule POS39-C	Checks for missing byte reordering when transferring data (rule fully covered)

Bibliography

[MSDN]	"Winsock Functions"
[Open Group 2004]	htonl, htons, ntohl, ntohs—Convert Values between Host and Network Byte Order