

MWR InfoSecurity Security
Advisory

IBM WebSphere MQ -
rriDecompress Remote
Denial of Service
Vulnerability

4th March 2010



Contents

1	Detailed Vulnerability Description	4
1.1	Introduction	4
1.2	Technical Background.....	5
1.3	Exploit Information	6
1.4	Dependencies	7
2	Recommendations.....	8

"rriDecompress" Remote Denial of Service

Package Name:	IBM WebSphere MQ
Date:	2010-01-15
Affected Versions:	WebSphere 7.0.0.2 on Windows are confirmed to be vulnerable. Other versions and platforms may also be affected by this issue.

CVE Reference	CVE-2009-3159
Author	A. Plaskett
Severity	High Risk
Local/Remote	Remote
Vulnerability Class	Denial of service
Vendor URL	http://www-306.ibm.com/software/integration/wmq/
Vendor Response	A patch is available from the following URL: http://www-01.ibm.com/support/docview.wss?uid=swg24024153
Exploit Details Included	Yes (although no exploit code is provided). Proof of concept code is available on request.

Overview:

The WebSphere MQ service can be used to transfer messages between systems and applications. A vulnerability was identified with the packet handling routines which would allow a malicious attacker to cause a denial of service condition.

Impact:

This vulnerability could be exploited to prevent or disrupt legitimate users from accessing a Queue Manager. It is also possible that the MQ service could be forced to produce large memory dumps potentially disclosing sensitive information or resource exhaustion.

Cause:

The vulnerability is caused by an integer underflow in the function 'rriDecompress' which could allow an attacker to pass a large value to a 'memmove' function size parameter.

Interim Workaround:

The use of SSL for authentication would mitigate this risk and restrict exploitation to those users in possession of a valid SSL client certificate. It is not expected that a security exit could prevent exploitation of this vulnerability, although only a limited number of exits have been tested at this point.

Solution:

The vendor supplied patches should be installed to resolve this issue. Links to the updated software can be found at the following location:
<http://www-01.ibm.com/support/docview.wss?uid=>

1 Detailed Vulnerability Description

1.1 Introduction

WebSphere MQ is an Enterprise level application that can be used to provide a unified platform for messaging within an organisation. IBM describes their technology as follows:

“WebSphere MQ provides an award-winning messaging backbone for deploying your enterprise service bus (ESB) today as the connectivity layer of a service-orientated architecture (SOA).”

Source: <http://www-306.ibm.com/software/integration/wmq/>

Communication with MQ services can be achieved in a number of ways and the technology requires a feature rich API and extensions in order to integrate with an Enterprise environment.

The main component of a WebSphere MQ instance is the Queue Manager. This is a process that manages the message queues and provides interfaces (known as channels) to the applications wishing to communicate with them. By default, a Queue Manager will listen on a network interface for incoming connections and process the data accordingly. A Queue Manager will accept any type of MQ data and begin processing it before determining whether the packet is authorised or has been received at the correct point within the application’s “state machine”. The result of this fact is that a large amount of MQ code is exposed to unauthenticated users.

Consequently, any vulnerability in the code used to parse the data after it has been passed from the network socket can potentially be exploited by an attacker. After receiving data from the network socket the MQ application will process and parse the data in various ways. The exact nature of this parsing is not within the scope of this document; however, it is important to note that a large amount of this activity occurs before a connection to the Queue Manager is fully established. Once the parsing has been completed MQ will check whether the state machine is setup such that the packet belongs to a session that has been correctly established with the Queue Manager at the application level.

1.2 Technical Background

An integer underflow vulnerability was identified within the function 'rriDecompress' which could lead to a large value being passed as the size parameter to the 'memmove' operation. This size would be larger than expected by the application and would result in an access attempt to unmapped memory, thus causing the process 'amqrmpa' to be terminated.

The following code is taken from the 'rriDecompress' function and illustrates that the ESI register is populated with a value obtained from the packet.

```
.text:4E8E5BA6      and     byte ptr [ebx+0Bh], 0FBh
.text:4E8E5BAA      mov     esi, [esp+80h+var_60]
```

The ESI register is loaded with the segment size of the packet from the TSHM section of the packet. This value will later be used in further operations before being passed to a call to the 'memmove' function. The processing continues as illustrated here:

```
.text:4E8E5BAE      xor     eax, eax
.text:4E8E5BB0      cmp     byte ptr [ebx+9], 4
.text:4E8E5BB4      mov     ecx, esi
.text:4E8E5BB6      setz   al
.text:4E8E5BB9      lea   eax, ds:2Ch[eax*4]
```

The EAX register is initially zeroed by the XOR operation. The segment type value read from the TSHM section of the packet (EBX+9) is then used to compare against the constant value 4. The ECX register is then loaded with the size of the packet from the ESI register (after it has been converted from a TSHM segment to a TSH segment). The register state resulting from this operation is such that the AL register is then not set. This results in the constant value 0x2C being loaded into the EAX register by the LEA instruction.

```
.text:4E8E5BC0      sub     ecx, eax
.text:4E8E5BC2      add     ebx, eax
.text:4E8E5BC4      sub     ecx, 4
.text:4E8E5BC7      push   ecx                ; size_t
.text:4E8E5BC8      lea   edx, [ebx+4]
.text:4E8E5BCB      push   edx                ; void *
.text:4E8E5BCC      push   ebx                ; void *
.text:4E8E5BCD      call  ds:_imp_memmove
```

The value held in EAX (constant size 0x2C) is then subtracted from the attacker controlled ECX register (previously populated by a value derived from the TSH segment size). If the attacker controlled ECX register is less than 0x2C then an integer underflow occurs which results in a large number. The value is then decreased even further when the subsequent sub instruction occurs. The ECX value is then passed to the 'memmove' call which causes a trap (Access Violation) to occur because it attempts to read beyond the extent of the memory that is mapped. This results in a denial of service of the 'amqrmpa' process as it is terminated at this point.

1.3 Exploit Information

To exploit this vulnerability a packet must be crafted which triggers a call to the 'rriDecompress' function.

The following MQ trace output shows the call stack for reaching this function.

```

00003143 16:40:59.966764 1524.3 RSESS:000001 -----{ cciProcessUserData
00003144 16:40:59.966774 1524.3 RSESS:000001 -----{ cciProcessAsyncRcv
00003145 16:40:59.966784 1524.3 RSESS:000001 -----{ rriServerAsyncRcv
00003146 16:40:59.966793 1524.3 RSESS:000001 -----{ ccxGetUserData
00003147 16:40:59.966803 1524.3 RSESS:000001 -----} ccxGetUserData
(rc=OK)
00003148 16:40:59.967073 1524.3 RSESS:000001 -----{ rriDecompress
00003149 16:40:59.967103 1524.3 RSESS:000001 Just removing CSH for
SegmentType : 0x5
0000314A 16:40:59.967712 1524.3 RSESS:000001 -----{ xcsFFST
0000314B 16:40:59.967737 1524.3 RSESS:000001 !! - ErrorCode :- 20006119
Numeric Insert1 :- 00000000 (0) Numeric Insert2 :- 00000000 (0)
0000314C 16:40:59.967748 1524.3 RSESS:000001 !! - Access Violation at address
01AFD000 when reading
    
```

A malicious status packet (0x5 segment type) can be constructed which will be handled by the 'rriDecompress' function.

The following fields are those which can be set in the packet to trigger the 'rriDecompress' function in an unexpected manner. All other data should be set according to the format of a valid Status packet.

Field Name	Value
TSHM segment length	0x2D (Becomes 0x25 after conversion)
TSHM unknown field 1	0x01
TSHM segment type	0x05
TSHM Reserved flag	0x14

By sending the above packet it was possible to cause an underflow to occur in the following code:

```

.text:4E8E5BC0      sub     ecx, eax ; eax = 0x2C 0x25-0x2C = 0xffffffff
.text:4E8E5BC2      add     ebx, eax
.text:4E8E5BC4      sub     ecx, 4      ; ecx = 0xffffffff9 - 4 = ecx =
0xffffffff5
.text:4E8E5BC7      push   ecx          ; size_t   ecx = 0xffffffff
.text:4E8E5BC8      lea    edx, [ebx+4]
.text:4E8E5BCB      push   edx          ; void *
.text:4E8E5BCC      push   ebx          ; void *
.text:4E8E5BCD      call   ds:_imp_memmove
    
```

It is expected that this issue is not further exploitable because of the size of the buffer which is allocated (as this is much larger than the size an attacker could reliably control and overflow). However, because of the inherently dangerous nature of memory corruption vulnerabilities and the fact that the only mitigating control

available in this instance is the use of SSL, it is recommended that this issue should be addressed in an appropriate manner.

1.4 Dependencies

Initial testing indicated that it would be possible to prevent this specific, vulnerable code path being executed by the use of a security exit. However, the security exit tested 'amqrspin.dll' was prone to another issue which would cause an earlier access violation to happen which would also cause the process 'amqrmppa' to terminate. Consequently, this issue is rated as high risk as an unauthorised attacker could cause the 'amqrmppa' process to terminate regardless of whether a security exit had been implemented or not. The use of SSL for authentication would mitigate this risk and restrict exploitation to those users in possession of a valid SSL client certificate.



2 Recommendations

It is recommended that all users install the appropriate security patches released by the vendor in response to this issue. Links to the updated software can be found at the following location:

<http://www-01.ibm.com/support/docview.wss?uid=swg24024153>

MWR InfoSecurity
St. Clement House
1-3 Alencon Link
Basingstoke, RG21 7SB
Tel: +44 (0)1256 300920
Fax: +44 (0)1256 844083
mwrinfosecurity.com