WHITE PAPER FOR PUBLIC DISTRIBUTION

InformationRiskManagement

Security Testing Enterprise Messaging Systems

An IRM Research White Paper by

Andy Davis and Phil Huggins

Information Risk Management Plc 8th Floor Kings Building Smith Square London SW1P 3JJ
Tel: +44 (0)20 7808 6420 Fax: +44 (0)20 7808 6421 info@irmplc.com http://www.irmplc.com

Security Testing Enterprise Messaging Systems

All the large investment banks worldwide use enterprise messaging systems to transport data concerning billions of dollars worth of transactions daily. Therefore, the security of the messaging systems is critical to the continuity of the ongoing business of these companies. This paper discusses potential security weaknesses that may be present in messaging systems either as a result of software flaws, application design or the misconfiguration of services. It focuses on TIBCO Rendezvous, as an example of a commonly used enterprise messaging system. Recommendations are then presented which mitigate these security issues.

Introduction to messaging systems

An enterprise messaging system provides an infrastructure for asynchronous and loosely coupled application communication. Typically enterprise messaging systems support a wide variety of business-critical applications that have either a requirement for low latency or guaranteed delivery of messages.

These infrastructures are commonly shared by many applications across an enterprise and rely on the use of message channels to control the delivery of messages to the appropriate applications. These message channels are commonly either point-to-point or publish-and-subscribe connection types. The transmission of messages between different message channels is achieved through the use of message routers. Some messaging systems also provide more application services such as authentication, authorisation or message content translation.

Applications connect to the messaging systems either integrating a messaging client API into the application or by using a channel adaptor to transform data from the applications native format to the message format used by the messaging system.

The potential risk profile of a messaging system is diverse, they can range from attacks on the operating system infrastructure underlying the messaging system software, attacks on the network protocol transporting the messages, attacks on the custom channel adaptors providing interfaces to external systems, attacks through the messaging system endpoint API and attacks on the client application API exposed through the messaging interface.

IRM has been involved in a variety of proprietary messaging system evaluations including TIBCO Rendezvous and WebSphere MQ as well as a number of web services and JMS based enterprise service buses such as Sonic ESB and Oracle ESB.

Although the paper talks generically about the methodology used to test messaging systems, an example infrastructure using TIBCO Rendezvous (RV) is discussed in more detail.

Evaluation Approach

Evaluations of these systems are performed at three levels, infrastructure review, messaging system review and deployed application review. Each of these levels of evaluation requires different approaches and has different potential impacts on the messaging system under test.

Both the infrastructure review and the messaging system review are usually performed once for the messaging system in question, the application review is usually performed separately on each application that utilises the messaging system.

Infrastructure Review

The infrastructure review is usually performed out of business hours on a representative sample of the production environment. This level of the evaluation consists of a vulnerability scan and a build configuration review assessing the operating system robustness. This can also include reviews of network components such as routers and firewalls that impact on the security and availability of the messaging system.

It is important when planning this level of the evaluation to understand both the physical and logical system and network architectures. Similarly a clear understanding of the composition of the messaging system including numbers of devices in use, the number of types of components in use, the number of platforms in use and the number of third party or external systems that interface with the messaging system.

Given the sensitivity of the production environment and the criticality of applications usually deployed, it is important that techniques performed carry little or no risk of interruption of service.

The goal of this level of the evaluation is to identify the vulnerabilities in the underlying infrastructure that pose a risk to the messaging system. These are usually easily addressed through remedial configuration changes in production and amendments to build standards for future systems.

Messaging System Review

The messaging system review is often performed in a test environment. This level of the evaluation consists of a representative sample of messaging system components that have been configured as they would be in the production environment.

At this level of the evaluation it is important to understand the logical system architecture and the composition of the system including the types of components in use and the roles these components perform. The various phases associated with the messaging system review can be summarised as follows:

- System configuration enumeration
- Subject enumeration
- Message spoofing
- System vulnerability identification and exploitation

The output of this level of the evaluation is to identify vulnerabilities associated with both the configuration of the messaging systems and any associated with the patch level of messaging system software. Vulnerabilities relating to system configuration and patch levels can usually be easily addressed by the client by following the recommendations presented.

Application Review

The application review is usually performed in a test environment. This level of review consists of attempts to misuse the application in question.

At this level of the evaluation it is necessary to understand the service being delivered by the application, the business requirements that service must meet, the numbers of clients and servers that make up the application, the role each of these perform and the services or functions each of these provide, the languages they are written in and the sorts of systems they are deployed to.

Where sufficient documentation exists this is planned through a review of documents such as deployment maps, asset inventories, functional and non-functional design documentation, application use cases and client API definitions. Once the correct operation of the application is understood a threat modelling exercise is undertaken to create a series of 'misuse' cases which would then form the basis of the subsequent test. These misuse cases are implemented through a custom client written against the messaging system API.

Where detailed documentation is unavailable then a demonstration of the application is provided by a knowledgeable user. During this procedure the network behaviour of the application is monitored and the actions performed by the user noted. Application traffic is then replicated based on the observed use cases and the resulting application behaviour observed.

The output of this level of the evaluation is to identify vulnerabilities within the application utilising the messaging system, these vulnerabilities can include code flaws and functional logic flaws. These vulnerabilities are usually implementation flaws created by the application software developers. Time is then spent with development teams both in addressing identified vulnerabilities and in improving development standards to include appropriate levels of security.

The remainder of this paper will focus on the "Messaging System Review" section of the methodology with respect to testing a TIBCO RV application and underlying messaging infrastructure.

Testing the Messaging System Components of a TIBCO RV Application

There are specific nuances associated with each messaging system with respect to the methodology of security testing and therefore, only one, TIBCO Rendezvous (RV), is detailed in this paper to serve as an example.

Overview of RV Specifics

All application components that communicate via RV utilise the RV API, which includes functions to perform tasks such as sending and receiving messages, managing event queues and managing fault tolerance. The API is available in many common programming languages including C / C++ / PERL / .NET.

RV uses TIBCO's Subject-Based Addressing™ whereby applications do not need to understand the mechanics of transferring data across the network from component A to component B. If component B is listening for subject X and component A sends a message addressed with subject X, then component B will receive it.

All communication between application components is performed via RV daemons (rvd) which can either run on the same server as the application component or be accessed remotely. In addition, messages can be sent either point-to-point between application components or via multicast where multiple "subscribers" receive the same message. Various other daemons can also be used; examples include: rvrd (routing daemon to route RV traffic to another IP network), rvsd ("secure" version of rvd) and rvcache (caches data associated with subject names).

Information gathering

A large amount of information about the subject naming scheme, infrastructure configuration and application configuration can be gained using information gathering techniques. By default, all daemon processes run an HTTP-based administration interface, which can be accessed and managed using a web browser.

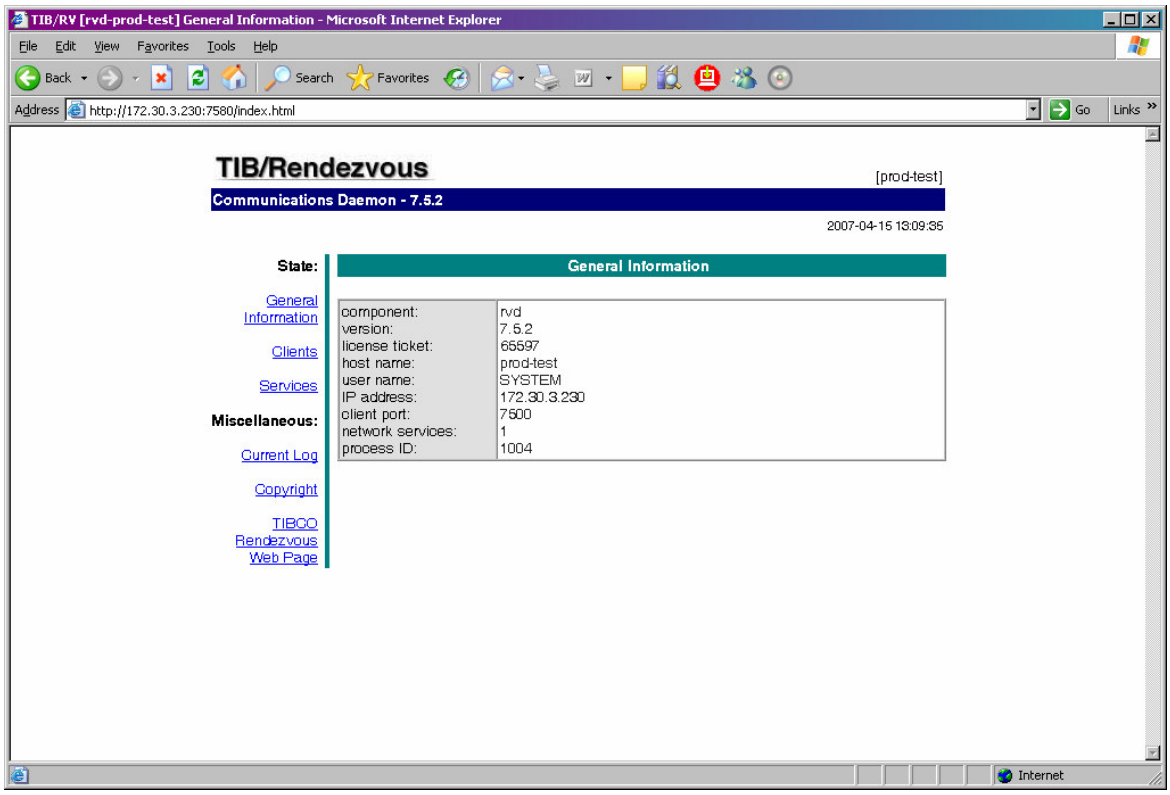

Figure 1. Access to the HTTP admin interface of the rvd daemon

Table 1 shows the default port numbers used for the HTTP administration interface on each daemon.

TCP Port Number	Rendezvous Daemons
7580	rvd, rvrd, rvsd, rvsrd
7581	rvcache
7680	rva

Table 1. Default TCP port numbers associated with HTTP administration interfaces

If access can be gained (by default no authentication is required, so, by access we mean TCP/IP network access) to the administration interface of any of the RV daemons, the following information can be obtained:

- **RV daemon version:** There are publicly known vulnerabilities in version 7.5.0 and earlier (CVE-2006-2830)
- **Application name:** This is the OS process name of the application component using RV
- **Operating system user:** The user that launched the application component is displayed, which could be used in an attempt to gain access to the underlying operating system via services utilising remote authentication mechanisms.

- **IP address details:** It may be the case that the RV infrastructure spans multiple IP networks using the rva daemon. The IP address details can be gathered and used to build up a network diagram of the RV infrastructure.

Recommendation:

All HTTP administration interfaces should be disabled on production systems by executing the rvd process using the *–no-http* option.

If administration interfaces must be used then a well-chosen username/password combination should be configured to protect access to the service. Furthermore, the administration interface should be forced to use SSL and should be bound to the local interface using the *–https 127.0.0.1:7585* option

Subject Enumeration

As already mentioned, RV uses TIBCO's Subject-Based Addressing™ as a mechanism to ensure that data is received by the correct clients. Subject names consist of "elements" separated by dots in the form of a subject name hierarchy e.g.

```
IRM.Sample_App.Userdata.Name
```

In order to add flexibility to the use of subjects, wildcards can be used:

The asterisk (*) is a wildcard character that matches any one element. The asterisk substitutes for whole elements only, not for partial substrings of characters within an element.

The greater-than symbol (>) is a wildcard character that matches all the elements remaining to the right.

Therefore, in order to enumerate the subject space used within an RV infrastructure wildcard subjects can be listened for in a structured way. IRM have a developed an RV testing tool called tibtool that can be used to perform this task. Figure 2 shows tibtool being used to listen for all subjects being sent within the RV infrastructure:

```
$ ./tibtool -function listen "*"
Starting Listener on subject "*"

----------------------------------------
New Message
----------------------------------------
Packet Size:            21
Number of Fields:       1
Hex Dump:

00 00 00 15 99 55 ee aa 05 44 41 54 41 00 08 05 61 6e 64 79 00
Subject:                IRM.Sample_App.Userdata.Name
Message:
{DATA="andy"}

----------------------------------------
New Message
----------------------------------------
Packet Size:            31
Number of Fields:       1
Hex Dump:

00 00 00 1f 99 55 ee aa 05 44 41 54 41 00 08 0f 74 68 69 73 5f
69 73 5f 61 5f 54 45 53 54 00
Subject:                test
Message:
{DATA="this_is_a_TEST"}
```

Figure 2. Using tibtool to listen for all subjects

Figure 3 shows tibtool listening for all subjects associated with IRM's sample application user data:

```
$ ./tibtool -function listen "IRM.Sample_App.Userdata*"
Starting Listener on subject "IRM.Sample_App.Userdata.*"

----------------------------------------
New Message
----------------------------------------
Packet Size:            21
Number of Fields:       1
Hex Dump:

00 00 00 15 99 55 ee aa 05 44 41 54 41 00 08 05 31 30 31 30 00
Subject:                IRM.Sample_App.Userdata.UID
Message:
{DATA="1010"}
```

Figure 3. Using tibtool to listen for subjects relating to application user data

Recommendation:

The "secure" equivalents for each of the daemons should be used to protect the confidentiality and integrity of the TCP data between client and daemon. The benefits of the secure daemons are:

- Only authorised clients can connect to a secure daemon
- Secure daemons restrict the combinations of network and service over which client transports can communicate
- Secure daemons limit the subject space that its clients can access using a white-list approach

By default, when an RV client sends a message to the network it is sent to a multicast group that can be captured from the network and the raw contents, in the RV "wire format" protocol, can be inspected.

> ***Recommendation:***
>
> Where appropriate, multicasting should be disabled using the *–no-multicast* option to protect the confidentiality and integrity of the data in transit across the network.

Attacks against TIBCO RV

Once the configuration and daemon versions associated with an RV infrastructure have been enumerated, the next phase is to perform exploitation attempts against the system. N.B. Based on the business criticality associated with most RV production systems, this phase must be performed in a test environment.

Remote Code Execution: A buffer overflow vulnerability (CVE-2006-2830) exists in the HTTP administration interfaces for RV daemons version 7.5.0 and below [1]. Publicly available exploit code is available to demonstrate this vulnerability, the impact of which is arbitrary remote code execution with the privilege level of that which the daemon was started (the exception to this is on a Unix-based system where, if the daemon was started as "root" then the code will be executed as "nobody").

Denial of Service: A remote Denial of Service vulnerability has been identified by IRM in the rvd daemon (version 7.5.2), which results in the exhaustion of operating system memory of the server on which the rvd process is running. The details of the vulnerability remain confidential while a patch is developed by TIBCO.

> ***Recommendation:***
>
> Ensure that the latest version of TIBCO RV is installed to reduce the risks associated with publicly known vulnerabilities such as those that may lead to either Denial of Service or remote code execution.

Degradation of Service: Sending to subjects with lead wildcards (for example, > or *.foo) can cause unexpected behaviour in some applications, and cause network instability in some configurations. [2]

> ***Recommendation:***
>
> To prevent potential Denial of Service attacks being launched which utilise lead wildcards, the *–no-lead-wc* option should be used when executing the rvd process.

Message spoofing: Although RV maintains availability of data sent between application components there are no mechanisms to protect either the confidentiality or integrity of data when it is sent between RV daemons. Therefore, attackers with access to the network infrastructure used to support RV can both capture and spoof traffic sent between daemons. The tool tibtool can easily be used to spoof RV packets.

Recommendation:

To prevent arbitrary message capture and spoofing, the application using RV as a transport should include its own mechanisms to protect both the integrity and confidentiality of the data.

Brute force attack: If RV secure daemons are in use then tibtool can be used to perform a dictionary-based attack against any identified users. Figure 4 shows tibtool in "dictionary attack" mode:

```
$ ./tibtool -function bf -users /dictionaries/unames.txt -pwds /dictionaries/passwords.txt

Attempting Secure Daemon Username/Password connect...

Verbose Output Suppressed
Using username dictionary /dictionaries/unames.txt

Trying "administrator" with passwords from /dictionaries/common_passwords.txt...None
Trying "admin" with passwords from /dictionaries/common_passwords.txt...None
Trying "root" with passwords from /dictionaries/common_passwords.txt...None
Trying "test" with passwords from /dictionaries/common_passwords.txt...None
Trying "temp" with passwords from /dictionaries/common_passwords.txt...None
Trying "tibco" with passwords from /dictionaries/common_passwords.txt...TIBRV_OK=true

!Valid Username/Password Pair Found: tibco/password123
```

Figure 5. Using tibtool to perform a dictionary attack against an RV secure daemon

Recommendation:

To prevent dictionary attacks against secure RV daemons, certificates should be used instead of username / password combinations. If for any reason username / password combinations must be used then the RV logs should be regularly monitored to identify any brute force attempts against the service.

Conclusion

The aim of this whitepaper was to discuss how Enterprise Messaging Systems can be tested from a security perspective to identify weaknesses as a result of software flaws, application design or misconfiguration. The example used to highlight the primary security issues was TIBCO Rendezvous; however, any of the commonly used messaging systems could have been discussed. As detailed in this paper, there are a range of potential software flaws and configurations of TIBCO RV that may lead to security exposures; however, these can be addressed by the implementation of the recommendations presented.

References

[1] http://www.tibco.com/resources/mk/rendezvous_security_advisory.txt

[2] TIBCO Rendezvous Administration Guide

About the Authors

Andy Davis is the IRM Chief Research Officer where his responsibilities include managing the technical research programme and the software security team. Andy also performs scenario-based penetration testing, software security analysis and bespoke security consultancy for a range of clients throughout Europe, North America and South East Asia, including UK Government Departments, Law enforcement, CNI (Critical National Infrastructure) and financial institutions. Andy is a certified CESG CHECK Scheme team leader and CLAS consultant with six years of commercial consultancy experience at IRM and ten years of Government Information Security experience gained from working at GCHQ in Cheltenham. Four of these years were spent operationally seconded to other areas within the UK Intelligence Community, which included the participation in several software security research placements at NSA.

Phil Huggins is the Chief Technical Officer for IRM. He is a trainer, speaker and professional security consultant. Phil consults with enterprise organisations on how to embed security within their organisations and bridges the gap between technical security risks and business. In over 10 years as a security consultant Phil has gained varied experience providing expert security advice within the government, financial, transport and media sectors. His areas of expertise include Security Evaluation, Risk Analysis, Security Design and Incident Response & Forensics. Phil was employed by the Ministry of Defence as the team leader in an IT Security Health Check Team providing penetration testing to UK military and government networks as well as major financial institutions. Phil has also provided application security testing for a wide range of global clients. Since joining IRM Phil has been the lead security consultant for a multi-million pound security evaluation programme for a critical national infrastructure client. Phil has authored custom risk analysis methodologies to align IT security with normal business risk management.

About IRM

Information Risk Management Plc (IRM) is a vendor independent information risk consultancy, founded in 1998. IRM has become a leader in client side risk assessment, technical level auditing and in the research and development of security vulnerabilities and tools. IRM is headquartered in London with Technical Centres in Europe and Asia as well as Regional Offices in the Far East and North America. Please visit our website at www.irmplc.com for further information.