Web 应用服务器可扩展名字服务的设计和实现*)

陈宁江　范国闯
（中国科学院软件研究所　软件工程技术中心　北京100080）
（中国科学院软件研究所　计算机科学重点实验室　北京100080）

The Design and Implementation of Extensible Naming Service in Web Application Server

CHEN Ning-Jiang　FAN Guo-Chuang
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100080)
(Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract　Web application server provides the environment for transactional applications with a series of services (e.g. transaction, message, component container, etc.). Naming services provide a means for application to locate objects and act as essential support in application server. However, users may use various naming service implementations in their applications, so Web application server needs to support multiple naming systems in a flexible way. With analyzing the naming requirements of application server, this paper presents a naming service mechanism that supports multiple naming systems in application server, which based on JNDI and JMX technologies. The implementation of the mechanism in WebFrame application server is introduced. The mechanism has good manageability and openness and extensibility, and satisfies the requirements of Web application server.

Keywords　Web application server, Naming service, J2EE, JNDI, JMX

1　引言

在 Internet/Intranet/Extranet 环境中的企业级应用系统大多采用多层结构的应用模式。Web 应用服务器是为事务性 Web 应用提供一系列服务（如事务、消息、组件容器等）的运行环境，是一个创建、部署、运行、集成和维护多层分布式企业级应用的平台，它为企业级应用提供了关键的基础设施和应用框架。在应用服务器领域中，基于 J2EE 技术的应用服务器是近年来研究和开发的一个热点。J2EE[1]（Java 2 Platform Enterprise Edition）是 Sun 公司提出的开发、部署、运行和管理基于 Java 的分布式应用的标准技术体系，它为应用服务器的实现提供了一个完整的底层框架。J2EE 中的一系列技术规范，包括 EJB、JDBC、JNDI、JTS/JTA、JMS、JCA 等，为应用服务器提供了支持。

J2EE 应用服务器是一个复杂的计算机软件系统，其实现有许多问题要解决。其中，名字服务是应用服务器必须支持和实现的底层机制。在分布式企业应用系统中，名字服务提供了系统中的实体（如用户、机器、服务、组件和对象等）的名字、描述、位置、访问、管理和安全信息的统一描述方法，使系统中的实体与其名字相关联，从而通过名字可以方便地找到对应的实体。名字服务是使应用服务器中各种组件、服务和资源协同工作的核心服务。目前存在多种名字系统，如 DNS、NIS、文件系统、CORBA 名字系统、LDAP 等，不同系统的实现细节是有区别的。在实际应用中，会出现不同的用户需要使用不同名字系统的情况。如何将多个名字服务系统以统一的界面纳入 Web 应用服务器的框架中并进行管理，是作为中间件的应用服务器应当解决的重点问题。本文将就如何在 Web 应用服务器中实现多名字系统支持和满足 J2EE 规范对名字服务的需求进行研究，提出了一种名字服务支持机制。这种机制基于 J2EE 技术中的 JNDI（Java Naming and Directory Interface）规范，同时采用 JMX（Java Management Extensions）的管理体系，能够在应用服务器中有效地插入各种名字系统，实现了 J2EE 规范所要求的组件名字上下文功能，使用户以透明的方式访问底层的名字系统。

本文的第2小节阐述 Web 应用服务器对名字服务的需求；第3小节给出所设计的支持多名字系统的名字服务机制，说明其结构和组成；第4小节介绍该机制在自主开发的 WebFrame 应用服务器中的实现；最后是本文的小结。

2　Web 应用服务器对名字服务的需求

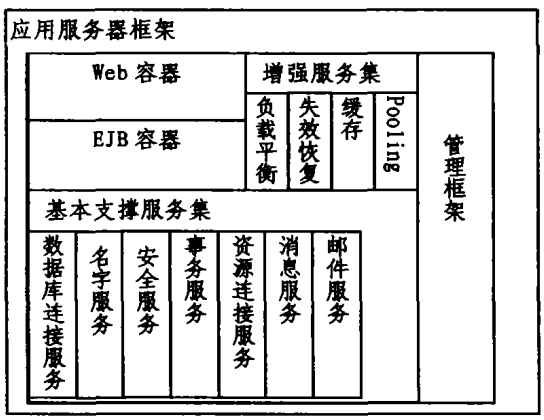

图1　Web 应用服务器参考结构

*)国家自然科学基金项目（69833030）；国家十五科技攻关计划（2001BA205A06）；国家高技术研究发展计划（863计划）（2001AA113010；2001AA414020；2001AA414310；2002AA413610）。陈宁江　博士研究生，主要研究网络分布计算。范国闯　博士研究生，主要研究网络分布计算。

图1给出了一个 Web 应用服务器的参考结构。在应用服务器的框架中，名字服务属于底层的基本支撑服务。应用服务器使用管理模块以统一的管理接口和管理风范来管理服务器中的各种服务。J2EE 平台对名字服务的要求，主要是要解决两个问题[1]：应用装配者和部署者应该不需存取应用的源代码就能够定制应用的业务逻辑；应用必须能够在它们的操作环境中存取所需资源，而不需知道这些外部资源信息在操作环境内是如何命名和组织的。

J2EE 针对名字服务提出了 JNDI 接口规范。JNDI 提供了一组标准的接口来访问不同类型的名字和目录服务系统。在逻辑上，JNDI 分为 API（Application Programming Interface）[2]和 SPI[3]（Service Provider Interface）两部分接口：应用程序通过 API 访问各种名字和目录系统；SPI 使开发者为特定的名字和目录系统编写服务提供者（service provider），使得各种名字/目录系统能够透明地加入到 JNDI 结构中，从而使 Java 应用程序能够通过 JNDI API 访问这些服务。JNDI 是其它 Java 编程模型和技术的组成部分，例如 JDBC 规范建议使用 JNDI 来存储数据源对象、Java 消息服务（JMS）规范建议通过 JNDI 来定位 JMS 所管理的对象（包括队列和主题）；在企业应用中，可以通过 JNDI 来为部署在应用服务器中的 EJB 和 Web 组件提供定位网络环境中的对象的方法。JNDI 简化了用户定位资源的操作，体现集中式管理信息资源的概念，满足企业应用易于管理的需求。应用服务器中的名字服务机制同样需要适应易管理性的要求，使不同的名字/目录系统能够方便地被使用和管理。

JNDI 在 J2EE 中的一个重要作用是将 J2EE 组件与它所被部署的环境相隔离[1]。使用应用组件环境（application component's environment）能让用户不用访问或改变应用组件的代码就可以定制组件。应用组件环境通常也称为企业名字上下文（Enterprise Naming Context，ENC）。ENC 是当应用服务器容器的控制线程与组件交互时仅能由该组件访问的本地环境。ENC 的目标是为组件提供一个隔离的、只读的名字空间。每一个组件定义自己的 ENC 环境内容，该环境必须与其它组件的 ENC 环境相隔离，即不能访问其它组件的 ENC 环境。J2EE 容器应当实现应用组件环境，将该环境作为 JNDI 名字上下文提供给应用组件实例。J2EE 规范中明确定义了 ENC 环境的名称“java:comp/env”，因此应用服务器必须提供实现“java:comp/env”名字上下文的方法。

总而言之，J2EE 应用服务器的名字服务支撑机制应当完全遵循 JNDI 规范，能够集成各种的名字系统并对它们进行管理，实现 ENC 环境，使应用通过 JNDI 接口以透明的方式调用 EJB、JMS 对象和 JDBC 数据源等各类资源。在实现时，应当考虑的质量因素包括名字服务模块的可扩展性、可管理性和开放性。

3 Web 应用服务器的名字服务支撑机制

基于上节所述，将名字服务部分从应用服务器的整体设计中抽取出来，可以得到如下的名字服务支撑结构（见图2）。该结构由名字服务提供者、名字服务器管理器、ENC 模块和应用服务器的管理服务器组成。

（1）JNDI 服务提供者　外部名字/目录系统通过 JNDI SPI 的包装能够容易地集成到 JNDI 体系中，并与应用服务器连接起来。JNDI 体系结构隔离名字服务使用者和服务实现者之间的关系，JNDI 服务提供者屏蔽各种名字/目录系统的实现细节，从而为客户提供一个一致的编程接口和访问接口。用户进行的名字操作可以委托给服务提供者进行，服务提供者再去与名字系统进行交互，从而使名字服务对客户透明。一般要为一种名字/目录系统提供一个服务提供者，用户可以按照 JNDI SPI 规范为所使用的名字/服务系统编写服务提供者，Sun 公司为 LDAP、文件系统、DNS、CORBA 等多种名字系统提供了相应的提供者实现[4]，这极大地方便了 JNDI 的应用。

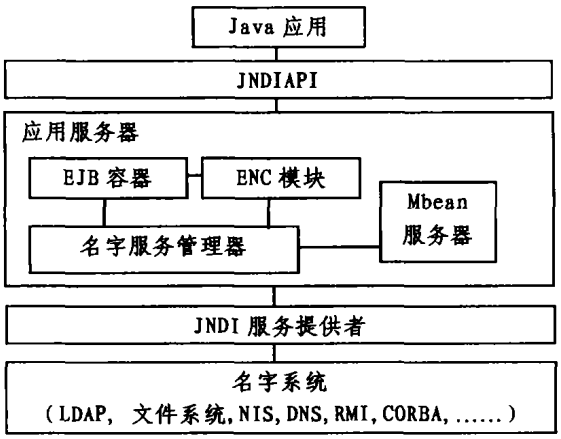

图2　应用服务器的名字服务结构

（2）名字服务管理器　是应用服务器中对名字服务进行管理的主要部分，其主要作用是实施对名字服务的管理活动（如属性配置、服务的启动/停止等），使名字系统对应用服务器以及部署在其中的应用组件和服务可用。名字服务管理器主要负责启动应用服务器中的 JNDI 名字服务，检查和激活应用服务器所使用的名字系统。当名字服务正常启动之后，应用服务器即可通过名字服务来访问对象。名字服务管理器还应创建 ENC 环境上下文，这通过与 ENC 模块交互完成。名字服务管理器作为一个服务由应用服务器的管理模块进行管理。应用服务器可以基于 JMX 规范及其技术建立管理框架。

（3）JMX Mbean 的管理方式　将一个服务集成进应用服务器的最佳途径是为该服务编写 JMX 的 Mbean。名字服务管理器以 JMX Mbean 的形式挂进应用服务器，融入整个应用服务器的管理体系中。JMX[5]（Java Management Extensions）是一种致力于解决分布式系统的管理问题的开放性标准，它对被管理资源的服务实现了抽象。JMX 定义了一个管理体系结构、API 和在管理规范下的各种管理服务，可以跨越一系列不同的异构操作系统平台、系统体系结构和网络传输协议，开发无缝集成的系统、网络和服务管理应用。Mbean 对象实现了 JMX 规范中定义的关于资源及其设备的接口。Mbean 服务器是 JMX 中的核心组件，为管理和控制 Mbean 提供服务，包含创建、注册、删除和访问 Mbean 所必需的方法。所有可管理的 Mbean 都是以插件方式插入到 Mbean 服务器中，已经注册到 Mbean 服务器中的 Mbean 组件即为可管理组件。

应用服务器中使用 JMX 服务来集成和管理各种定制服务，包括名字服务管理器。通过 Mbean 方法来控制名字服务的生命期，这样提供了清晰的服务接口，满足可管理性和可配置性的要求。按照 JMX 规范编写的名字服务器管理器在应用服务器启动时加载。

（4）ENC 模块　主要实现 ENC 环境的创建和使用，其中主要是提供 ENC 上下文工厂类，用于生成 ENC 上下文的实

例。ENC 模块重要的一点是要为组件提供私有的 ENC 环境，隔离对不同组件的 ENC 环境的访问。实现隔离的方法可以有多种，本文第4小节给出了一种方法。JNDI 定义了名为“nns”(nns 代表下一个名字系统 next naming system)的特殊形式的引用。这种引用包含一个类型为 nns 的地址，其值为名字系统部分处理的结果对象。该引用指示处理引用的对象工厂[3](ObjectFactory，对象工厂用于将存储在名字/目录系统中的数据转换为 Java 类型的对象，即使用存储在名字空间中的信息来创建对象)，当前正在处理的引用是用于获得下一个名字系统的上下文，对象工厂根据系统信息(部分处理的结果对象、已经解析的名字和剩余的名字等等信息)来动态地获得下一个名字系统的上下文。实现 ENC 环境时，首先使用当前名字系统来创建初始上下文，然后 ENC 上下文工厂(属于对象工厂类)以“java:comp”为名字、作为“nns”引用对象来绑定到当前上下文中。以后 EJB/Web 容器在部署 EJB 或者 Web 组件过程中建立组件环境时，可以利用已有的 ENC 上下文创建“java:/comp/env”组件上下文名字空间。

综上所述，在 Web 应用服务器中提供名字服务支持时，首先需要为名字系统编写符合 JNDI SPI 规范的服务提供者；然后，按照 JMX Mbean 规范编写名字服务管理器，实现 Mbean 的管理接口(如初始化、启动等)；在应用服务器启动或运行期间，通过 Mbean 服务器对名字服务管理器进行服务的注册、配置、启动和生命期控制。名字服务管理器启动名字服务的主要过程如下：

1) 设置应用服务器所使用的名字系统(或者称为名字服务器)及其属性(名字服务器所在主机地址和端口)；

2) 在应用服务器特定的服务配置文件中设置名字服务管理器及其属性；

3) 应用服务器通过 Mbean 服务器装载名字服务管理器，在 Mbean 服务器中对其进行注册；

4) 名字服务管理器执行初始化操作，初始化名字服务环境；

5) 名字服务管理器判断名字服务器是否存在，激活名字服务器，设置有关系统属性；

6) 创建 ENC 环境工厂；

7) 名字服务器启动成功，开始监听名字服务请求。

4 实现

在中科院软件研究所自主开发的 Web 应用服务器 WebFrame 中实现了前文所述的名字服务支持机制。WebFrame 应用服务器中提供了一个 Mbean 服务器[6]，其细节在本文不作详述。在这里，以 WebFrame 缺省提供的 wfn 名字服务器为例介绍名字服务支持机制的实现。wfn 是一个用 Java 实现的基于 RMI 的小型名字服务器，它类似于 RMI 的简单名字工具 RMIRegistry。wfn 主要是改进了 RMIRegistry 的一些不足之处，如 wfn 可以支持多层次名字空间、为整个网络提供名字服务支持、增强了类装载的灵活性等，在 RMI 应用中可以作为 RMIRegistry 的替代者。

WebFrame 按照 JNDI SPI 规范为 wfn 名字服务器提供了一个名字服务提供者 wfnProvider，wfnProvider 执行与 wfn 名字服务器的连接，获得 wfn 名字服务器的 RMI 引用，通过 RMI stub/skeleton 机制，wfnProvider 将所有与名字服务相关的操作(如 lookup、bind、rebind 等)委托给与之关联的名字服务器实例完成。

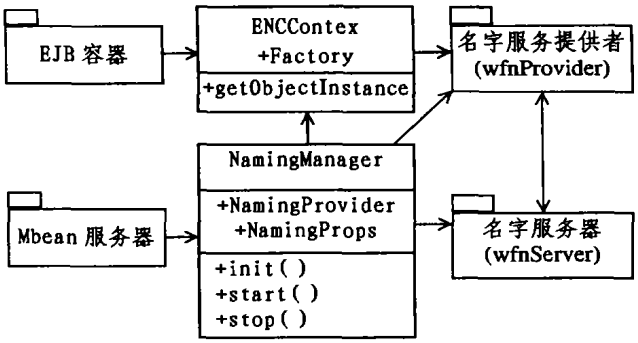

图3 WebFrame 中名字服务的实现

名字服务管理器(NamingManager 类)作为一个 Mbean 实现，在对名字系统属性的读/写方法之外，主要有 init()和 start()两个方法。

(1)init 方法：初始化所使用的名字系统。将名字服务配置信息(主要信息包括名字服务器的 URL 地址、使用的 ContextFactory 类等)读入系统，设置相关的系统属性值，所存取的文件包括标准的 jndi 配置文件(jndi.properties)和所使用的名字服务器的配置文件。

(2)start 方法：主要完成启动名字服务的工作。

①名字服务器的启动。根据所使用的名字系统类型及其已经设好的属性值，检查所要使用的名字系统是否已经存在或者有效，调用相应的启动模块启动或者激活名字服务器。不同的名字系统所做的启动工作会有所区别。例如对于 wfn 名字服务器，要完成如下工作：

a)创建远程名字服务器对象；

b)设置本地应用服务器所使用的名字服务器为上一步所创建的名字服务器对象；

c)通过给定的主机名和 Socket 端口，输出名字服务器的 Rmi Stub，使服务器可以接收发来的调用请求；

d)在给定的 Socket 端口上启动名字服务器的监听线程，监听名字服务的请求消息。

②将 ENC 上下文工厂绑定到当前名字空间中。首先创建初始名字上下文(InitialContext)，然后将“java:comp”上下文(即 ENCContextFactory 类)作为引用对象绑定到当前上下文中。

ENCContextFactory 类实现了“java:comp”对象工厂，用于产生“java:comp”上下文对象，返回该对象的实例。在 WebFrame 中，使用线程上下文类装载器作为标识来区分不同组件的 ENC 环境。ENCContextFactory 所生成的上下文与线程类装载器相联系，用于存储上下文的映射表以类装载器作为关键字。ENCContextFactory 类中的主要方法是 getObjectInstance，由 JNDI 中的 ObjectFactory 接口继承而来，它在 ENCContextFactory 中用于返回 ENC 上下文。具体地说，当客户线程需要创建 ENC 环境时，以该线程上下文类装载器为关键字在上下文映射表中查找相应的 Context，如果查找结果为空，则创建该类装载器对应的 Context 对象，将它们绑定起来。在此，组件 ENC 环境的正确隔离依赖于每个组件所接收的与组件执行线程相联系的唯一的类装载器信息。EJB 容器建立组件应用环境时，先查找“java:/comp”上下文，然后在其下为组件创建“java:/comp/env”上下文，从而在容器中为组件建立分离的名字空间。

结束语 本文所设计的名字服务支持机制通过使用标准

的 JNDI 和 JMX 技术，能够有效地将多个名字系统插入到 J2EE 应用服务器框架中，实现了 J2EE 规范中对名字服务的要求。它具有如下特点：(1)开放性，遵循 J2EE 技术规范(包括 JNDI、JMX)的标准；(2)可管理性，应用服务器能够有效地管理名字服务的生命期，将名字服务纳入应用服务器的统一管理框架中；(3)可扩展性，能方便地扩展应用服务器所使用的名字系统，可集成多个名字系统；(4)可配置性，通过 Mbean 形式实现的名字服务管理器具有良好的可配置能力。这种机制在 WebFrame 应用服务器中得到了实现，证明是可行和有效的。

参 考 文 献

1 Sun Microsystems. Java2 Platform Enterprise Edition Specification(v1.3). 2001.7
2 Sun Microsystems. JNDI 1.2 API Specification. 1999.7
3 Sun Microsystems. JNDI 1.2 SPI Specification. 1999.7
4 http://java.sun.com/products/jndi/serviceproviders.html
5 Sun Microsystems. Java Management Extensions Instrumentation and Agent Specification (v1.0). 2000.7
6 中科院软件研究所软件工程技术中心. WebFrame 应用服务器技术白皮书. 2002.2
7 Perrone Paul J, Krishna C. Java Naming and Directory Service Interfaces. Distributed Computing, 1999,2(10):12～16
8 Lee R. The JNDI Tutorial. Available at: http://java.sun.com/products/jndi/tutorial/

(上接第162页)

Analysis Agent 的作用是判断当前收到的安全事件是一个新的攻击的开始，还是已有攻击的继续，若是新的攻击的开始则创建一个新的 Analysis Agent 进行分析，否则将安全事件送给已有的 Analysis Agent。Analysis Agent 的作用是对当前收到的安全事件，结合考虑该事件对应的攻击的历史状态来做出合理的响应决策。该 Agent 还检查以前做出的响应是否成功，如果失败则采用同种响应方式的其它实现版本，如果其它实现也失败则更换响应方式。Curtis 提出可以用三种尺度结合的方法判断安全事件是属于已有攻击还是新的攻击的开始，即时间尺度(安全事件之间的时间间隔)、会话标识尺度(如 IP 地址和用户名等)、攻击类型尺度。

Curtis[10]介绍了 AAIRS 的体系结构，AAIRS 的优点在于它良好的自适应性，而且在响应决策中考虑了环境因素的影响。但是 AAIRS 的响应决策没有基于成本模型，这使得系统可能产生过多的不必要的响应措施。

6. 响应的协同

现有的响应系统都是根据本地的安全事件信息，进行局限于本地的响应，而对于大规模网络而言，各响应系统之间有必要进行协同。这主要有以下两点好处：

(1) 各响应系统之间安全事件信息的共享有助于对攻击者行为做出更精确的判断，从而做出更合理的响应；

(2) 各响应系统之间响应措施的协同可以使总的损失代价达到全局的最小，如在离攻击者最近的边界控制器将攻击者 IP 隔离可以减小该响应措施对其他正常用户带来的额外损失。

IDIP[11]是在 DARPA 的资助下，由波音公司、NAI 实验室和加州 Davis 分校计算机安全实验室共同研究的应用协议，它能将各种网络基础设施(如边界控制器、入侵检测系统、基于主机的响应器等)有机地集成在一起，从而实现以下功能：

(1) 协同追踪攻击源，并在最接近攻击源的边界控制器隔离攻击者 IP；

(2) 使用独立于设备的追踪和阻塞指示消息；

(3) 集中的报告和入侵响应的协同。

IDIP 系统被组织成两级的结构。第一级为 IDIP 团体，每一个 IDIP 团体是一个管理域，该域内所有的入侵检测和响应系统被发现协调点管理；团体由多个邻居组成，边界控制器是连接相邻邻居的设备，每个邻居内都有入侵检测和响应系统，入侵追踪就是从一个邻居通过边界控制器追踪到另一个邻居。对于一次攻击，每个邻居的 IDIP 节点会做出本地的响应，IDIP 将追踪攻击源，在攻击路径上的各 IDIP 节点都将做出响应，发现协调点综合各邻居的 IDIP 节点的信息，协调和纠正各 IDIP 节点的响应措施，从而达到全局最优的响应。

COMON 系统是在国家863计划通信主题重大项目资助下，由东南大学等高校研究和开发的分布式高速 IP 网络入侵监测系统。该系统的协同功能主要分为三个层次，最底层是一个通用安全传输平台，可以进行身份的鉴别、数据的加密传输、数据的完整性保护，秘钥的分配采用 PKI 机制。安全传输平台的上层是一个通用协同平台，它支持各协同点之间的基本协同操作，并提供了访问控制策略对各协同点之间的相互访问进行约束。协同平台之上是安全事件的综合和追踪功能，其中安全事件追踪通过向相邻协同点查询相关安全事件可以追溯到最接近攻击源的边界控制器，从而隔离攻击者 IP。试验表明，协同功能的引入大大增强了系统对入侵的检测和响应能力。

总结 随着攻击的复杂化和自动化，自动入侵响应系统将在保护系统安全、降低攻击对系统造成的损失方面起着重要的作用。自动入侵响应系统在成为实用的系统之前，必须具备安全性、合理性、实时性、自适应性和灵活性，但目前还没有一个 IDS 产品能达到所有这些要求。本文重点介绍了多种应用在自动入侵响应系统中的技术，其中成本敏感模型应当作为入侵响应决策的基础，意图识别技术和自适应技术增强了自动入侵响应系统的功能。这些技术的结合使我们能够构造一个功能强大、实用的自动入侵响应系统，从而为系统提供更有力的保护。

参 考 文 献

1 Center C C. CERT/CC Statistics for 1988. http://www.cert.org/stats/cert_stats.html, 2000
2 Center C C. CERT Coordination Center 1998. http://www.cert.org/annual_rpts/cert_rpt_98.html, 2000
3 Cohen F B. Simulating Cyber Attacks, Defenses, and Consequences. http://all.net/journal/ntb/simulate/simulate.html, 1999
4 Carver C A. Intrusion Response Systems: A Survey. http://faculty.cs.tamu.edu/pooch/course/CPSC665/Spring2001/Lessons/Intrusion_Detection_and_Response, 2000
5 Lee W. Toward Cost-Sensitive Modeling for Intrusion Detection and Response. Journal of Computer Security, 2002,10(1-2)
6 Lindqvist U, Jonsson E. How to Systematically Classify Computer Security Intrusions. IEEE Symposium on Security and Privacy, 1997
7 Geib C W, Goldman R P. Plan Recognition in Intrusion Detection Systems. IEEE, 2001
8 Kautz H A. A Formal Theory of Plan Recognition and its Implementation. University of Rochester, 1987
9 Carver C A. Limiting Uncertainty in Intrusion Response. IEEE, 2000
10 Carver C A. A Methodology for Using Intelligent Agents to provide Automated Intrusion Response. In: Proc. of the IEEE Systems, 2000
11 Schnackenberg D, Djahandari K, Sterne D. Infrastructure for Intrusion Detection and Response. In: Proc. of the DARPA Information Survivability Conference and Exposition, 2000