

# Hardening Guidelines for Pramati Server

---

This paper provides you essential information about how to harden your Pramati Server environment. In addition to practical, hands-on configuration recommendations, this paper includes suggestions for combating other external threats to your server. While most server administrators can benefit from reading this paper, it also benefits developers writing applications for Pramati Server. As some procedures in this paper relate to security aspects discussed in the *Pramati Server Security Guide*, it may help to read that guide too.

The hardening guideline broadly cover the following components:

- Application
- Application Server
- VM
- OS
- Hardware

This paper is specific to application server aspects of hardening. You may need to refer documentation of other products for a complete hardening exercise. The following aspects are dealt in this paper:

- Denial of Service Protection
- Server Identity
- Access Controls

## Why Harden Server?

Enterprises depend on application servers to deliver data when they need it and how they need it, in a secure and reliable way. Data integrity, confidentiality, and availability are of paramount importance. One of the first steps to achieving this is to install and maintain the application server such that unauthorized access, unauthorized use, and disruptions in service is prevented.

## Who is Responsible?

Organizations usually have a server hardening policy as part of their security audit. The policy applies to all individuals who are responsible for the installation and running of server, the application that is deployed on the server, the data resources that will be accessed by the server, and the complete hardware ecosystem (network and servers).

## Useful Configuration Files

The following files are required during the hardening exercise for Pramati Server:

Table 1:

File	Use	Location
server-config.xml	Server configuration including security	{install_dir}/server/nodes/<node name>/config
web-config.xml	Web server configuration including DoS prevention, server masking, thread limits	{install_dir}/server/nodes/<node name>/config
web.xml	Web application configuration	{application_dir}/.../WEB-INF/
runserver.bat	Server startup file, takes parameters including properties to switch of server administration	{install_dir}/server/bin/
default-web.xml	Server masking	{install_dir}/server/nodes/<node_name>/config

## Changing Username and Password

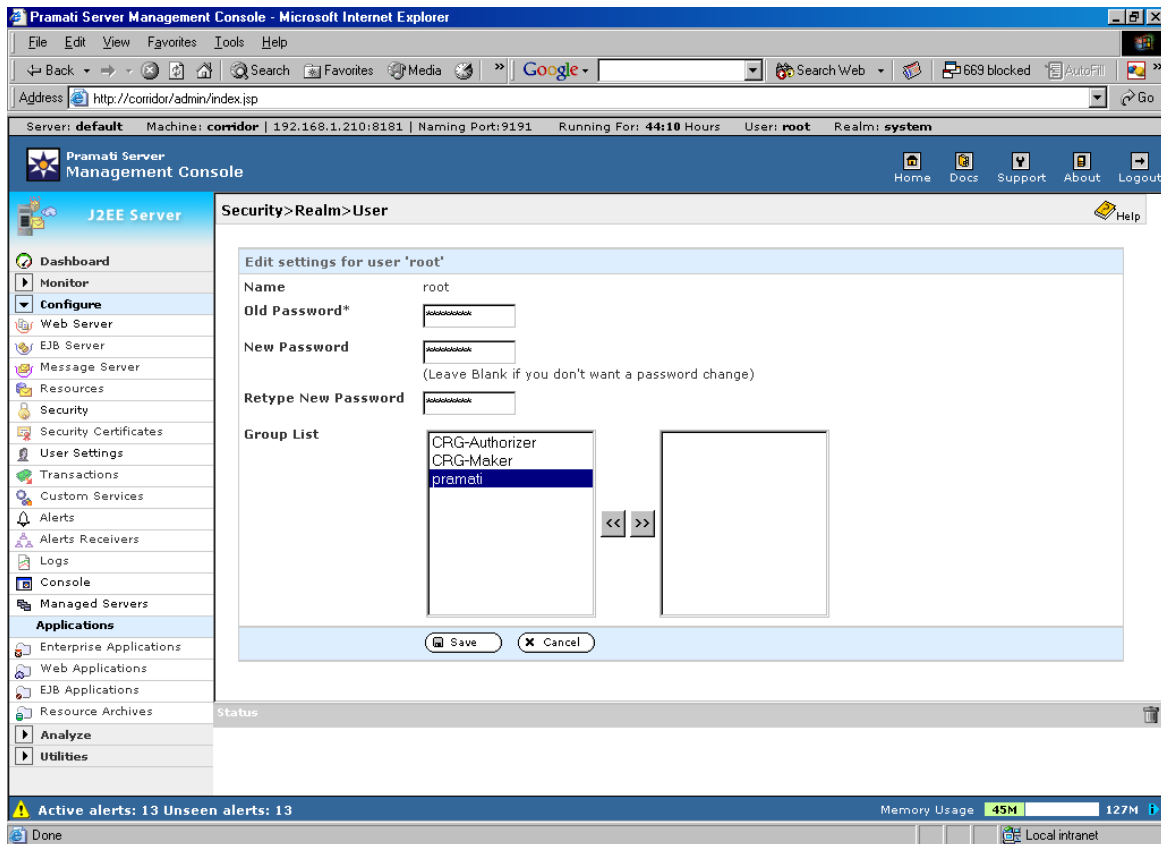
It is fair to assume that the default username and password for your Pramati Server is well known and, using it, anyone can access the Server Management Console with the administrator privileges. This gives malicious user administrative privileges to the Server. Moreover, IP-based restriction is not enabled for Server Management Console. This opens the default configuration server to attacks on the intranet.

## Changing Default Root User Password During Installation

Pramati Server provides factory default username and password for root users:

Username: *root*

Password: *pramati*



The installer prompts you to change this password for the root user. The new password becomes the default to operate new instances of the server that you create.

## Changing Password of Server Instance

If you want to secure a particular instance of the Server, use the Management Console to change the root user password. For example,

*Username: root*

*Password: myserver123*

## Denial of Service (DoS) Protection

A DoS is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service. Examples include flooding a network preventing legitimate network traffic, disrupting connections between two machines preventing access to a service, preventing a particular individual from accessing a service and disrupting service to a specific system or person. Illegitimate use of resources may also result in Denial of Service. For example, an intruder may use your anonymous file transfer protocol (FTP) area as a place to store illegal copies of commercial software, consuming disk space and generating network traffic.

## Preventing DoS Attack

You can set up DoS attack prevention at five levels on Pramati Server:

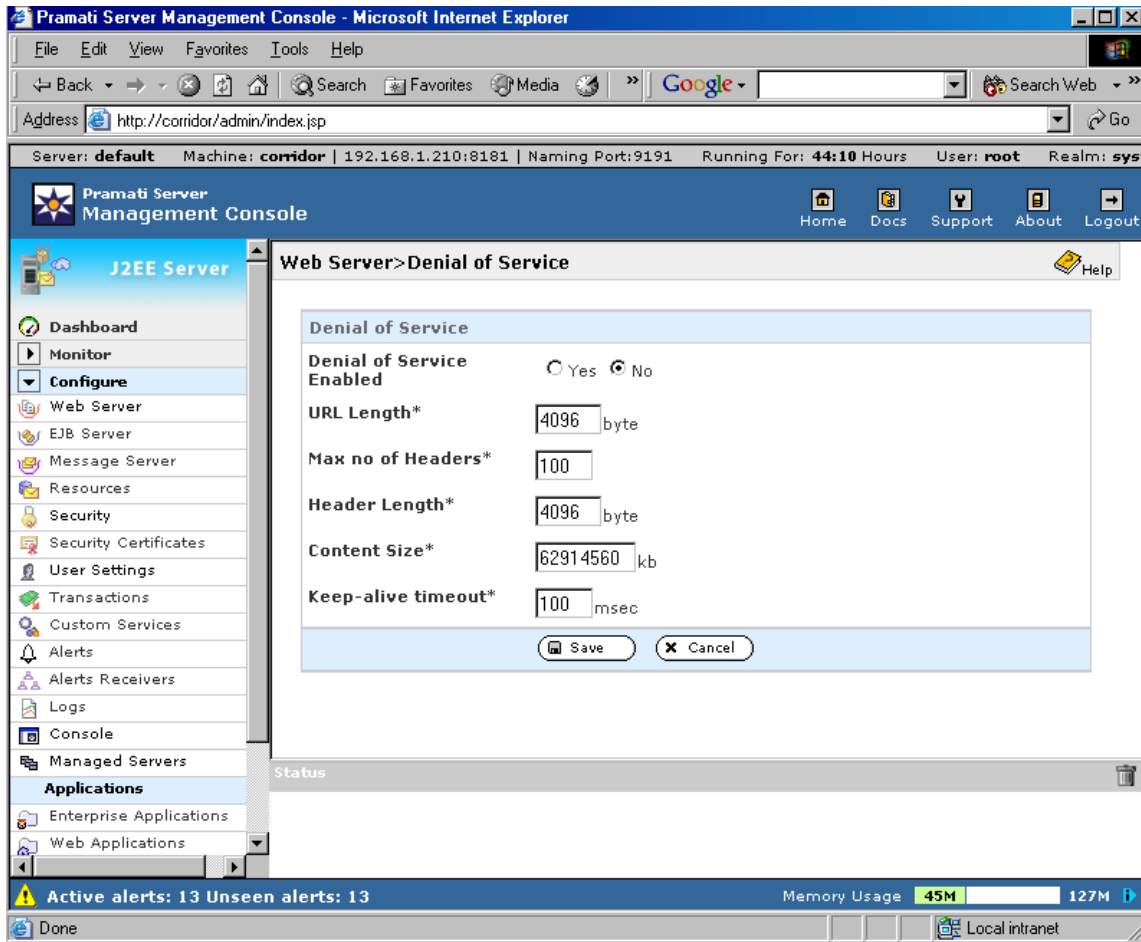
- Setting maximum URL length
- Setting maximum header key-value numbers
- Setting maximum header size
- Setting maximum body size
- Setting maximum keep-alive requests

While DoS can be prevented at the system or firewall level, Pramati Server provides DoS prevention features that can be configured in the `web-config.xml` using these tags:

```
<denial-of-service enabled="true">
  <uri-length-bytes>4096</uri-length-bytes>
  <header-limit numbers="100" length-bytes="4096"/>
  <body-size-bytes>10485760</body-size-bytes>
  <max-keepalive-requests>100</max-keepalive-requests>
  <send-http-response enabled="true"/>
</denial-of-service>
```

If any one of these rules is violated, the Web server discards the connection.

DoS prevention rules can be configured using the Management Console GUI too. To do so, select **Configure > Web Server > Denial of Service**. This displays the following window:



## Restricting Number of keep-alive Worker Threads

Worker threads enable concurrent processing by the server. Worker threads can be kept alive (that is, waiting on a connection for the next request) to improve performance. Keeping all threads alive maximizes performance but makes your server vulnerable. Rogue users can block all such threads by making them indefinitely wait for the next “elusive request” over the connection. As a result legitimate users get delays.

Reducing the number of worker threads that can be kept alive may affect performance but make your server more available to a wider number of users. This tag in `web-config.xml` specifies the number of keep alive worker threads:

```
<worker-thread-count max-keepAlive="150" max-active="-1">
200
</worker-thread-count>
```

In this case, at least 50 worker threads are always available to valid users. These are default values set when you install Pramati Server.

## Hiding Server Identity in Headers and Error Pages

Response sniffers available with common browsers can be used to show this information:

```
HTTP/1.1 200 OK
Server: Pramati Server
Date: Tue, 09 Nov 2004 10:41:40 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: Keep-Alive
```

Here you may want to mask the identity the Web server (“Pramati Server”). Publishing the type of Web server makes it vulnerable to hacking. Server identity also appears on default error pages. To mask the identity of the server, modify this tag in `web-config.xml` as `<display-server-name in-header="true" />`

The identity of the Server is masked on the error page by changing the `init-param` as `display-server-name-in-page` contained in the `PramatiDefaultStatusCodeServlet` in the `default-web.xml` configuration file.

## Directory Listing

Directory Listing is specified in `default-web.xml`, using servlet mapping for `PramatiDefaultDirectoryListingServlet`. This servlet mapping is commented by default. So directory listing is OFF by default. No additional effort is required for securing access to the server directories. In any event, Server directory listing does not give a user access to directories outside of the doc-root specified for the application. Hence it is recommended to keep application files outside the server installation directory.

## Custom Directory Listing

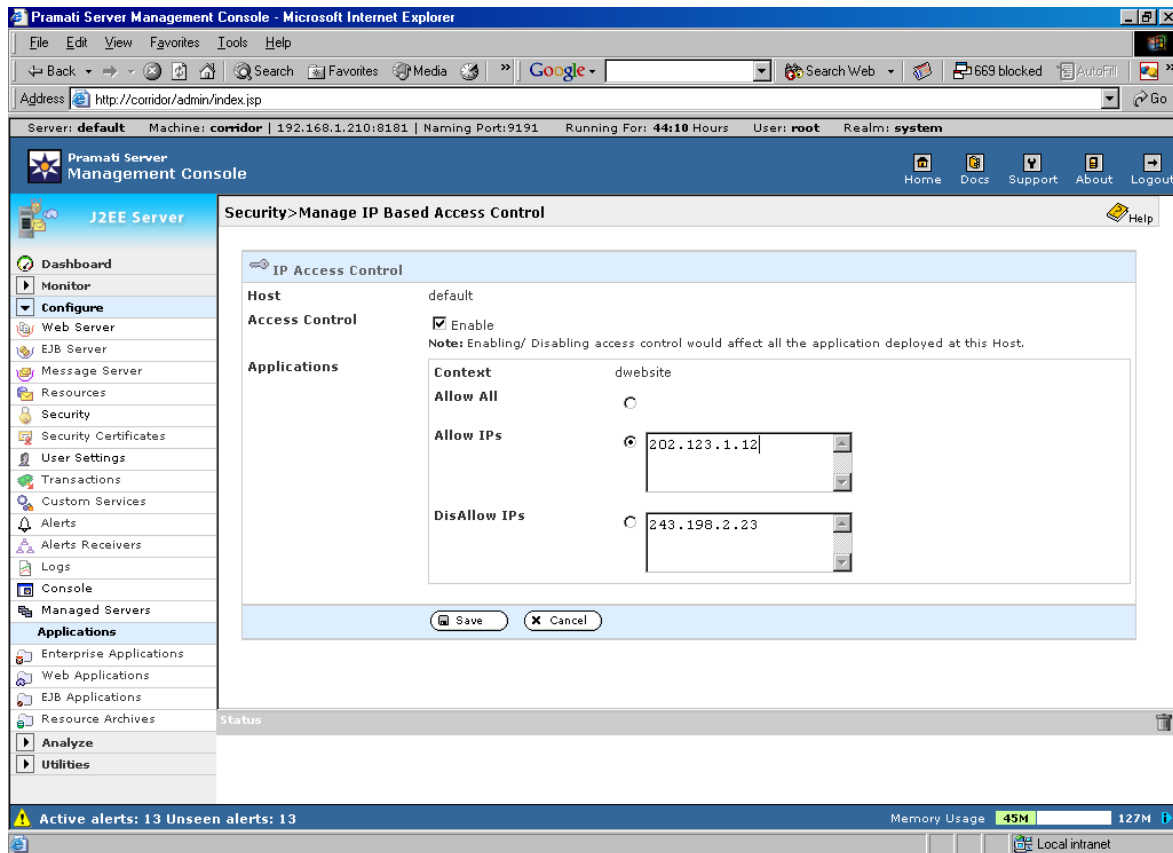
A custom directory listing servlet can be written by modifying the `servlet-class` tag under the `PramatiDefaultDirectoryListingServlet` in `default-web.xml`. The servlet class should be saved in: `/public_html/web-inf/classes` or inside a jar placed in `/public_html/web-inf/lib` directory.

*Note: A JSP can be specified as the directory listing servlet by removing the `servlet-class` tag and adding a `jsp-file` tag in `default-web.xml`.*

## Custom Error Page

The `StatusCode` servlet is used as the default error page by the Server when no custom error page has been defined in the application `web.xml`. A custom servlet can override this servlet by modifying the `servlet-class` tag under `PramatiDefaultStatusCodeServlet` in `default-web.xml`. The servlet class should be present inside `/public_html/web-inf/classes` directory or inside a jar placed in `/public_html/web-inf/lib` directory.

*Note: The user can also specify a JSP as the error page servlet by removing the `servlet-class` tag and adding a `jsp-file` tag in `default-web.xml`.*



## Disabling Web Administration Applications

Production sites often like to configure bare, runtime servers with shell controls and stripped of any Web-based administration functions that may contain security holes. On Pramati Server, the administration applications are enabled by default. The following system property disables the Management Console as well as the Console Online Help application:

```
-Dcom.pramati.admin.disableconsoleapps=true
```

Place this property in the startup file, `runserver.bat` after the java command, like this: `java -Dcom.pramati.admin.disableconsoleapps=true -server . . .`

*Note: Instead of disabling Web-based administration, you may want to place IP-based access restrictions so that only certain workstations on the network are enabled to manage the server.*

## URL Access Constraints

This Pramati Server feature restricts access to certain URLs based on client IP address. For example, this can be used to restrict access to the administration application to workstations on the LAN. To do this, modify these tags in `web-config.xml`:

Table 2: URL access constraints tags in `web-config.xml`

Tag	Description
Access-constraint	Set denial access to a given url for particular IP address
entry-identifier	Set a unique identifier for each entry
allow-access	Set True or False value as per the access permission requirement

Table 2: URL access constraints tags in web-config.xml

Tag	Description
url-pattern	Set URL path for which the access constraint has to be set
Client-ip-address	IP address for which all above access constraint value is applicable. Can be specified in CIDR.

Here is a sample entry:

```
<access-constraint>
  <entry>
    <entry-identifier>LocalAdminAccess</entry-identifier>
    <allow-access>true</allow-access>
    <url-pattern>/admin/*</url-pattern>
    <client-ip-address>192.168.1.5/24</client-ip-address>
  </entry>
</access-constraint>
```

You may want to read more on CIDR patterns at <http://www.freesoft.org/CIE/Course/Subnet/>. URL restrictions can be configured from the Management Console too.

## ‘Nobody’ Role: Giving Safe Privileges

Administrators prefer to run the application server as OS user with least privileges (non-root user). This poses a problem in Linux environments where only a root user can open the port 80 required by the Web server.

*Note: On Linux, root user privileges are required to open ports less than 1024. Windows does not have this restriction.*

If the Web server does not need to open a port less than 1024, this is not a problem, provided the server has permissions to access to the files specified by its root context. On the other hand, starting the server with the root user privileges can compromise the entire machine, including the application server.

A solution is to create a dedicated user role for running the application server: the “Nobody” role. This role has privileges only on the app server installation directory, and no access to other systems in the box. A way of safely giving privileges to “Nobody” is by using IPTables.

## Using IPTables

This solution allows Pramati to run as an independent Web server, using a non-privileged user, and still respond to port 80 on the server. It requires Linux with a kernel v2.4.x and IPTables installed. IPTables also allow you to do nifty things like redirecting all incoming packets on, say, port 80 (the default www port) to another port such as port 8181 (Pramati Server default port). To do this redirection, two things are needed:

- 1 Ensure that the firewall allows incoming requests to port 8181.
- 2 Redirect packets from port 80 to port 8181.

The first step is only needed if you have configured your firewall to protect your machine. Use your firewall-configuration software to accept incoming packets of type TCP and destination-port 8181. Essentially, you need a copy of the standard rule for port 80, using port 8181.

The second step is to instruct IPTables to redirect all packets that arrive on port 80 to port 8181.

This is done using the following command (as root): `iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j REDIRECT --to-port 8181`

The above command tells IPTables to redirect all TCP packets with a destination port 80, coming in on network interface eth0, to the port 8181 (on the same machine). You may want to add this rule to files used by your firewall configuration so that it is added automatically on booting.

*Note: This will not redirect local requests, since these bypass pre-routing chain. Thus, any browsers or other client software running on the Server itself will either have to connect directly to port 8181, or something else must be used to redirect these local requests. While some servers might not require local requests to be redirected, note that this could affect local search engine software and such.*

## HTTPS for Critical Pages

Pages that ask for your password, account number, credit card number or any other sensitive information, need to be secured through SSL (HTTPS). Users are authenticated and authorized before they can enter the site. To enable this, the Web server is configured in two steps:

- 1 Enabling SSL on the web server (accept HTTPS requests).
- 2 Specifying secured pages that require authentication.

*Note: By default, Pramati Server is not enabled for SSL.*

### Enabling HTTPS on Web Server

Modify the `web-config.xml` tag `protocol` to enable HTTPS. Set the `ssl-enabled` tag under `protocol` to `true` to enable HTTPS.

```
<protocol http-enabled="true" ssl-enabled="true"/>
```

Specify the HTTPS port in the `<https-port>` tag under `server-config.xml`. The default port is 443.

### Specifying Pages That Need Authentication

Create a resource collection of pages that will be accessed over HTTPS. Do this in `web.xml`

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>SecuredLogins</web-resource-name>
    <description>All accesses to login page over HTTPS</description>
    <url-pattern>/signin/login.jsp</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

## Using Custom Error Pages

On hitting a HTTP error like 404, a default page is returned by the server. This page uses a standard template that may give away the server identity (even if you have masked the identity of the server as discussed earlier). Instead, you can write a custom page and return that to the browser in case of an error. Such a page could be a sitemap or the homepage of the Web application. This

is good programming practice. The application developer must modify the `web.xml` in the `/web-inf` directory of the application. You can throw a custom error page for two types of exceptions:

- HTTP error
- Java runtime

Here is a sample entry:

```
<error-page>
  <error-code>404</error-code>
  <location>404error.jsp</location>
</error-page>
<error-page>
  <exception-type>java.lang.NullPointerException</exception-type>
  <location>npe.jsp</location>
</error-page>
```

## Preventing Cross-site Scripting

Dynamically generated HTML pages can introduce security risks if inputs are not validated either on the way in or on the way out. Malicious script can be embedded within input that is submitted to Web pages and appear to browsers as originating from a trusted source. This problem is referred to as a cross-site security scripting issue. This is avoided through good programming practices. Application developers must avoid using query parameters directly to generate HTML text or SQL queries. For further information, see <http://www-106.ibm.com/developerworks/security/library/s-csscript/>.

## Working With NAT Firewalls

Network Address Translation (NAT) or IP masquerading implies that the IP of Server host is mapped to a global IP outside the firewall, so that any request arriving at the global IP is transparently directed to the local IP.

For Pramati Server to receive calls from the global IP across the firewall, point

`java.rmi.server.hostname` to the global IP while starting Server. For example, if the user's global IP is 192.12.1.143, start the Server using: `java -Djava.rmi.server.hostname=192.12.1.143 com.pramati.Server.`

## FAQs

### On Security Procedures

**Q. What is the OS level access required for the user-id used by the clients accessing the site, say a separate guest account with minimum privileges could be created?**

A. Clients accessing the site need no OS-level access to the server at all. The application itself manages the authentication and authorization of users based in information available in the application database.

**Q. What Web-server user-id is required for Internet users?**

A. The user-id `root` in Pramati Server is not the same as the OS root in Unix systems. This is the default admin user in Pramati Security domain. The password of this user can be changed or another user can be added. For the application, if the standard J2EE security model is used, then the users (internet, internal, or admin) can be configured in the application server and assigned to various groups. Access privileges are set on the groups. Pramati Server uses JAAS framework and enables plugging in various security repositories.

**Q. How can the default access privileges required for the XML file, which includes the Server user-id and password, be modified?**

A. Access privileges to this file can be restricted to the administrator who will be starting Pramati Server.

- In order to change the Root password using the Pramati Server Management Console, click **Configure > Security**. Click the **Settings** option provided against the realm name, the password for which has to be changed. This leads you to the **Security > Realm** screen.
- Under the **Groups** field, click **administrator**. This takes you to the **Security > Realm > Group** screen. Click **Settings** for the user **root**.
- Enter the new password for the system realm. This enables you to change the root password. Here, the administrator can also manage all admin users.
- Information regarding various levels of permissions that can be given to various groups and users is available in the Security section of the *Pramati Server Administration Guide*.

**Q. What logs are available through Pramati Server and how can the logs required be enabled? What access settings are recommended for the log file?**

A. There is two types of logging that can be configured on Pramati Server- Web activity logging and Server internal logging.

Web activity logging generates logs in W3C format to monitor activity on the Web applications. The location and size of Log files can be configured using the `web-config.xml`.

Server internal logging can be set to default—which is, only severe errors and messages.

## **On Security Audit**

### **Q. What is the process behind auditing security?**

A. Based on the domain specific solution, security audit may involve one of these processes.

- Enumerating system vulnerabilities.
- Detailed report generation.
- Suggestions for fixes and necessary patches.

There are various tools available in the market for performing one or few of these tasks. Most of the tools have a friendly interface for detecting and managing vulnerabilities. But very few application servers provide built-in tools for scanning vulnerabilities as the exercise is out of scope for the server.

### **Q. What are the key benefits of Security Audit?**

A. The the key benefits of Security Audit are:

- Vulnerability assessment for all the components.
- Breach analysis and report generation

### **Q. How often do I need to perform security audit?**

A. With the amount of traffic and impregnability of business, it can vary from monthly to every two days.

### **Q. How does Pramati Server aid in the auditing process?**

A. Pramati Server logs in information about authentication and authorization including results and failures in a dedicated audit log file. This information can be used by third party tools or the network administrator to assess the security breaches and estimate the damages if any.

### **Q. Can I install the Web server in the same partition as the OS?**

A. Yes, you can. But if the Web server vulnerabilities are exploited, files in the same partition can be easily accessed. Critical system files like cmd.exe (command interpreter for Windows 2000/Windows NT) can be used to carry out unauthorized activities on the Web server like gaining access to the database, source codes of the server side includes etc. Hence ensure that Web server is installed on a different partition than the operating system. Also, the content files should be installed on a separate partition.

### **Q. Why do I need to set up access control for Web server folders?**

A. Users having full default permissions will be able to read the source code contained in script files (like JSP files) and may be able to execute these scripts in unauthorized method. Source code could contain sensitive information like database location, table structure, username/password etc. Hence, Web server folder access should be restricted to a separate user who starts the Web server and has permissions only on the specific folder.

**Q. Is installing Pramati Administration Service absolutely necessary?**

A. No. Pramati Administration Service is used only for accessing the clusters. If the server is installed in a standalone mode, the Service may not be required.

**Q. Why is Web access logging important for a server?**

A. Unless the Web access is logged, it will not be possible to detect security breaches or DoS attacks. Also, if rotation of log files is not implemented in the server, new logs will overwrite the old log entries once file size reaches the specified limit.

Every URL access is logged on the Web server. Depending on the configuration set, this log file can contain information like client IP address, the requested Web object, and date/time, which can be used for analyzing an attack on the Web server. Archival policy should be formulated and implemented in the server. Access to write such log file must be restricted to the user who has started the Web server. Access to read log files should be restricted to different user, who will review the logs.

**Q. Why is reviewing of logs important for analyzing breaches?**

A. Any undesirable activity on the server will remain unnoticed as the logging information increases. Hence the security logs should be checked periodically by the administrator or automated with the use of auditing tools.

**Q. Why should I audit Web server specific files?**

A. If malicious changes are made to the Web server specific files, it will remain undetected. Appropriate audit logs should be enabled to track changes made to these files. Logs should be monitored to ensure that unauthorized changes are detected immediately.