Security Hardening Guide for IBM WebSphere Portal

Index

Introduction

Index

Security is a fundamental requirement for most web applications. Organizations commonly demand detailed accounting of web application security, especially after mainstream media coverage of high-profile vulnerabilities or exploits. This guide instructs architects and administrators on evaluating and improving the security of applications based on IBM WebSphere Portal.

Security is a broad topic with many aspects for you to consider. This can become quite complex when applications integrate with other systems to deliver a solution. To simplify and organize your efforts, consider these several fundamental security concepts:

- **Authentication**: verifying a user's identity
  - Authentication answers the question, "*Who is this?"*
- **Authorization**: determining a user's right to access data
  - Authorization answers the question, "*What can he/she do?"*
- **Confidentiality**: preventing the unauthorized disclosure of data
  - Informally, *how to keep secrets*
- **Data integrity**: detecting and preventing unauthorized modification of data

- **Non-repudiation**: associating a user identity with an action
- **Trust**: reliance upon other components
    - Few web applications are entirely self-contained. Most rely on other systems to deliver at least some pieces of the security puzzle. Understanding the trust your application places in other systems will help you evaluate the security of your overall solution.
    - For example, WebSphere Portal trusts the underlying WebSphere Application Server with certain security functions like authentication and confidentiality. In turn, both may trust external LDAP servers to validate user credentials and determine group membership.

Each section in this guide introduces a common security requirement for, or security functionality of, WebSphere Portal (v8.5, though most recommendations apply to previous versions as well). Each section includes a *Recommended actions and considerations* section to guide you through evaluating and improving the security footing of your application.

WebSphere Portal is a platform upon which to build an application. The extent to which you may customize WebSphere Portal and associated security functions in WebSphere Application Server is nearly endless. Your approach to security hardening your application must encompass all of:

- WebSphere Portal
- WebSphere Application Server
- Custom code (themes, portlets, filters, etc.)
- Operating system (client and server)
- Browsers
- Web servers
- External security managers or other proxies
- Back-end applications, such as databases, LDAP servers, etc.
- Network security (including DNS, TCP/IP, etc.)
- Physical security

With the basics of this introduction in mind, proceed with these -

Recommended actions and considerations:

- Compile and prioritize the security requirements for your application.
- Diagram your application and its environment. Identify the major components.
- Identify component owners within your organization. Who is responsible for the security of, or otherwise has control over ...?:
    - the network, including firewalls
    - the web server
    - the LDAP server
    - any external security manager or other proxies
    - the database server
    - other back-end servers
    - any custom themes
    - any custom portlets
    - other custom code (authentication filters, vault adapters, etc.)

    - clients (operating systems, browsers)
- Read Advanced security hardening in WebSphere Application Server, Part 1 and Part 2 for an overview of security hardening.
    - Make note of topics you think might affect your application.
    - After following this *Security Hardening Guide for IBM WebSphere Portal*, circle back and address any remaining items.
    - Pay special attention to the introductory sections that address infrastructure.
- Read Web security concepts and considerations for IBM WebSphere Portal administrators for an overview of the main security components of WebSphere Portal.
- Review the IBM WebSphere Portal InfoCenter section on Security planning.
- Review the security considerations published by the Internet Engineering Task Force, W3, or other standard-setting body for any technology upon which your application relies. Primarily, consider:
    - HTTP (alt., W3)
    - Cookies
    - TLS / SSL
    - LDAP, particularly:
        - Protocol
        - Security Mechanisms
- Independently research web application security to identify other print and/or web resources to inform your evaluation. The author of this guide favors:
    - Open Web Application Security Project (OWASP)
    - The Web Application Hacker's Handbook (Stuttard, Pinto) – If you are new to web application security, this book provides a well-structured treatment of general web application security topics.

Install all available security fixes

Index

IBM implements IBM Secure Engineering practices to address common requirements for the security of IBM offerings. IBM posts important information regarding security vulnerabilities to the IBM Product Security Incident Response Blog.

IBM uses Security Bulletins when publicly disclosing security vulnerabilities. IBM publishes security bulletins for WebSphere Portal to the Support Portal. To help you evaluate and address each vulnerability, these security bulletins include CVE tracking numbers, CVSS scores, mitigations, and links to fixes.

Recommended actions and considerations:

- Install the most current maintenance for WebSphere Portal that is suitable for your application.
- Subscribe to notifications through the Support Portal to receive emails when security bulletins are published for new vulnerabilities. Optionally, also subscribe to the RSS feed for the IBM Product Security Incident Response blog.
    - It may not be practical for you to immediately install fixes on production systems.

Minimally, you should establish a procedure and schedule for evaluating vulnerabilities, installing security fixes, and testing your application. Consider using CVSS scores as the basis for your decisions on installation schedules.

- Integrate security fixes delivered for default themes into your custom theme.
  - Theme developers initially copy a default theme to prevent fix packs from overwriting customization. If a fix pack includes security fixes for a default theme, then any custom theme based on that default theme would need to be rebuilt to incorporate the fixes.
  - Later versions of WebSphere Portal support custom themes based on certain default themes delivered with earlier versions. Monitor security bulletins for your current WebSphere Portal version and any older versions used in your custom themes.
- Similarly to custom themes, integrate security fixes for samples into any custom code based on those samples (e.g. Struts Portlet Framework samples, Content Template Catalog).
- Install current maintenance for all other software that is integrated with your application, including:
  - client browsers, including any plug-in or add-on like Java
  - the application server
  - operating systems (client and server)
  - web servers
  - databases
  - external security managers
  - LDAP servers
  - For IBM products, monitor the IBM Product Security Incident Response Blog or the Support Portal for that product.
- If you discover a new security vulnerability in WebSphere Portal, please report it to IBM and open a PMR to obtain a fix.

Enable HTTPS

Index

Clients communicate with WebSphere Portal via the Hypertext Transfer Protocol (HTTP), for which you should review these basic security considerations. One major consideration is protecting sensitive information while it is in transit.

Transport Layer Security (TLS, the successor to Secure Sockets Layer [SSL]) provides a means of securing communications over the Internet. *Implemented effectively*, TLS:

- Ensures confidentiality by encrypting communications to guard against eavesdropping
- Ensures confidentiality by guarding against man-in-the-middle attacks
- Helps ensure data integrity by guarding against replay attacks.

HTTP over TLS (HTTPS, commonly) allows web applications to leverage TLS in order to secure communications with clients. Since these communications include such sensitive information as application data and single sign-on (SSO) cookies, IBM recommends enabling HTTPS for WebSphere Portal. The InfoCenter documents how to do this.

The procedure in the InfoCenter addresses the following common requirements:

- Encrypting all communications between the browser and web server for content served from the personalized home (wps/myportal, by default).
  - The WebSphere Application Server plug-in forwards any requests received over HTTPS to the WCInboundDefaultSecure port of the application. Therefore, if the browser to web server communications are encrypted, the web server to application server communications will be encrypted as well, by default.
  - Refer to the WebSphere Application Server InfoCenter for instructions on securing the connection between the web server plug-in and application server web container.
- Encrypting the HTTP POST with the user credentials in the default login portlet. If you use a custom login portlet, work with the portlet developer to ensure that it similarly protects users' credentials.

Recommended actions and considerations:

- Enable HTTPS per the InfoCenter.
  - Additional IBM HTTP Server technote, for reference.
- Evaluate whether the common requirements listed above address all of the security requirements for your application. In particular:
  - Consider whether your application needs to use HTTPS for all communications between the browser through the web server and to WebSphere Portal. Doing so would protect sensitive data submitted or served under URLs other than .../wps/myportal (e.g. wps/portal, wps/um).
  - Consider especially whether users register to your site with the default Registration portlet (i.e. Selfcare). If so, serve it from a page protected by HTTPS to protect sensitive user data.
    - Do one of:
      1. Most simply (if the web server serves WebSphere Portal only), implement a rewrite rule on the web server to redirect all http://... to https://... .
         - For example:

```
<ifModule mod_rewrite.c>
RewriteEngine on
RewriteCond %{SERVER_PORT} =80
RewriteRule ^(.*) https://%{SERVER_NAME}%{REQUEST_URI}
[R]
</ifModule>
```

      2. Or disable the web server listening on port 80 (HTTP) altogether and listen only on port 443 (HTTPS).
      3. Alternatively (if the web server serves multiple web applications which require HTTP/non-SSL), configure a rewrite rule on the web server to redirect any http://<webserver>/wps/... to https://<webserver>/wps/...
         - For the case where your web server serves multiple web applications, do likewise for any other URLs that might serve sensitive data. In the latest versions of WebSphere Portal, most content is served under wps/* or a similar customized context. Check the URI Names in plugin-cfg.xml generated from WebSphere_Portal to see if any other URLs require similar rewrite rules for your application.
         - If you have one or several sensitive pages, but do not want to lock down all of wps/portal/*, then you may use friendly URLs and a security

constraint (in web.xml) or redirect rule on, for example, redirecting http://.../wps/portal/tlsprotected to https://.../wps/portal/tlsprotected, assuming friendly.redirect.enabled=true. If you are considering this approach, consider also whether you have any functional requirements for setting friendly.redirect.enabled=false.

- Block the WCInboundDefault port(s) with a firewall between the web server and WebSphere Portal. At the network level, allow communications only over WCInboundDefaultSecure.
  - Using firewalls for this purpose is recommended because at the time of this writing, the approach advocated by the WAS hardening guide to encrypting the web server to web container link is not feasible for WebSphere Portal.
  - Manually editing plugin-cfg.xml and removing:
    <Transport ... Protocol="http"/>
    From the server definitions would provide an additional guarantee (above and beyond the firewall) that the web server to web container link was encrypted. However, if you regenerated and redistributed the plugin, this would be overwritten.
    - Test your application if you choose this approach. The author tested this approach and observed good results with basic WebSphere Portal functionality, but not all of the Content Template Catalog worked as expected.
  - Providing that the firewall and/or plug-in routes exclusively to WCInboundDefaultSecure, you may optionally force HTTPS access through trusted web servers only. Such a requirement is not common. Refer to the examples and consider whether you have any similar requirements. Note that WCInboundDefault must remain open for WebSphere Portal, so restricting access at the operating system level would be important in any such implementation (i.e. guarding against nefarious requests to localhost).
  - Actual port numbers for WCInboundDefault, WCInboundDefaultSecure, etc. are available in the WebSphere Application Server Integrated Solutions Console > Application servers > WebSphere_Portal > Web container transport chains.

- Evaluate whether the accepted cipher suites meet your security requirements
  - At the web server – if IBM HTTP Server, via the SSLCipherSpec directive.
  - At the application server – via the Quality of Protection settings for all applicable SSL configurations.
  - Consider this especially if you have a requirement for FIPS.
- If you run XMLAccess from a remote system, use a secure connection.
- Only if you off-load SSL, per the procedure in this technote, evaluate whether this meets the security requirements for your application. Off-loading SSL terminates the secure tunnel at some proxy other than the web server. This means that anyone with access to the network between the SSL off-loader and WebSphere Portal could easily obtain sensitive data or otherwise alter these communications.
  - Consider this physical equivalent when making this decision: An armored car company must transport $1M. In doing so, they must traverse a road where a maintenance crew is restricting traffic while patching a pothole. Should everyone on the road maintenance crew have keys to the armored car?

Guard your cookies

Index

Cookies enable single sign-on (SSO) and session tracking in WebSphere Portal. Unauthorized parties could circumvent WebSphere Portal's security controls given access to:

LTPAToken2: Lightweight Third-Party Authentication: WebSphere Application Server relies on this cookie for Single Sign-On (SSO). WebSphere Application Server considers a client to be authenticated if it presents a valid LTPAToken2.

JSESSIONID: from the Java Servlet specification: WebSphere Portal's session management uses this cookie.

Note: These are the default names for these cookies, though you may customize them in later versions of WebSphere Portal and WebSphere Application Server.

Recommended actions and considerations:

- Enable HTTPS, as described above.
- Define the domain attribute value for these cookies as narrowly as your functional requirements permit.
  - Leave these unset, unless you have a specific requirement for SSO or sharing session identifiers. If no domain is set on a cookie, the browser should submit it only with requests to the server that set it.
  - If your functional requirements dictate that you set the domain, use the following paths in WebSphere Application Server Integrated Solutions Console:
    - LTPAToken2/domain: Global security > Single sign-on (SSO) > Domain name
    - JSESSIONID/domain: Application servers > WebSphere_Portal > Session management > Cookies > Cookie domain
- Set the Secure attribute on these cookies. This tells the browser to present these cookies only over HTTPS. In the WebSphere Application Server Integrated Solutions Console:
  - LTPAToken2/Secure: Global security > Single sign-on (SSO) > Requires SSL
  - JSESSIONID/Secure: Application servers > WebSphere_Portal > Session management > Cookies > Restrict cookies to HTTPS sessions
- Set the HTTPOnly attribute on these cookies, if your functional requirements permit. This tells the browser to disallow access to javascript and can help guard against certain types of cross-site scripting (XSS) attacks. In the WebSphere Application Server Integrated Solutions Console::
  - LTPAToken2/HTTPOnly: Global security > Single sign-on (SSO) > Set security cookies to HTTPOnly to help prevent cross-site scripting attacks
  - JSESSIONID/HTTPOnly: Application servers > WebSphere_Portal > Session management > Cookies > Set session cookies to HTTPOnly to help prevent cross-site scripting attacks
    - Some applications may require access to the session identifier on the client-side. If your custom javascript needs access to JSESSIONID, you should not set HTTPOnly on that cookie. Rather, rely on security integration and XSS protections to guard the session.
- Set the LTPAToken2 expiration as short as your functional requirements permit.

    - As long as a client presents a valid LTPAToken2, WebSphere Application Server considers the user authenticated.
    - Expiration is encoded into the LTPAToken2 value and is not part of the Set-Cookie directive.
    - The default LTPAToken2 expiration is 2 hours (120 minutes).
    - In the WebSphere Application Server Integrated Solutions Console > Global security > LTPA > LTPA timeout
- Set the inactive session timeout as short as your functional requirements permit.
    - The default is 30 minutes.
    - In the WebSphere Application Server Integrated Solutions Console > Application servers > WebSphere_Portal > Session management > Session timeout
- Disable LTPA interoperability mode.
    - Interoperability mode sets another cookie (LTPAToken, by default) with lesser encryption than LTPAToken2. LTPAToken is not as secure as LTPAToken2. It is generally used for SSO to legacy systems.
    - In the WebSphere Application Server Integrated Solutions Console > Global security > Single sign-on (SSO) > Interoperability mode
- Enable Security Integration (this is enabled by default in v8.5)
    - This tightly binds the user (LTPAToken2) with the session (JSESSIONID).
    - Application servers > WebSphere_Portal > Session management > Security integration
    - This is especially important if HttpSessionIdReuse has been enabled.
    - Certain errors may occur if an authenticated user tries to access an anonymous session. To guard against such errors, configure WebSphere Application Server to use available authentication data (enabled by default in v8.5):
        - In the WebSphere Application Server Integrated Solutions Console > Global security > Web security - General settings > Use available authentication data when an unprotected URI is accessed

The authentication mechanism

Index

How can users log in to your application? WebSphere Portal primarily relies upon WebSphere Application Server components for authentication. In turn, WebSphere Application Server may trust an external security manager/trust association interceptor pair or other mechanism (e.g. SPNEGO) to authenticate users.

The main consideration from a WebSphere Portal perspective is the way in which the application initially receives user credentials, often via the default login portlet or a custom login portlet. WebSphere Portal also provides authentication filter interfaces which allow you to further customize the flow of authentication.

Recommended actions and considerations:

- Ensure that user credentials (user name and password, generally) are encrypted during transmission. Minimally, these should be encrypted between the browser and web server. Ideally, these should be encrypted from the browser to the WebSphere Portal server *and*

from WebSphere Portal to the user repository (LDAP server, generally).
    - Enable HTTPS for the default login portlet form POST or do the equivalent for any custom login portlet (see *Enable HTTPS,* above).
    - Secure communications between WebSphere Portal and the LDAP server (see the section on this topic, below).
- Evaluate whether your application requires the automatic login URL:
    - The automatic login URL is: `.../wps/portal/cxml/04_SD9ePMtCP1I800I_KydQvyHFUBADPmuQy?userid=userid&password=password`
    - The automatic login URL is suitable *only* for direct access by automated tools that cannot authenticate by any alternative means.
    - If PI13472 is installed, you may disable the automatic login URL altogether by setting custom property `authentication.isLoginUrlActive=false (boolean)` in the Authentication Service.
    - If your application requires the automatic login URL, or if you cannot install PI13472, then minimally block external access (to guard against certain spear-phishing attacks), by implementing a filter or rewrite rule on the web server, to the automatic login URL.
- WebSphere Portal offers only minimal controls over password strength (only length and valid characters), via the PUMA Validation Service.
    - Stricter controls would require custom code, most likely in the form of a custom registration/edit my profile portlet.
    - Alternatively, such controls could be enforced by the user repository itself, generally an LDAP. Discuss your options with the LDAP administrator.
- WebSphere Portal does not limit the number of times a user may submit the wrong password, so does not guard against brute force password trials.
    - Since several applications may share the same user repository, the best place to guard against this is in the user repository itself. Many LDAP servers provide a way to lock users out after a certain number of failed logins.
    - Optionally, a custom login portlet or custom authentication filter could help guard against this.
- WebSphere Portal neither supports nor prevents multiple concurrent logins. Consider implementing custom code to prevent multiple logins.
- The IBM Web Content Management (WCM) servlet comes with its own login.jsp. The WCM servlet will redirect the anonymous portal user to this login screen when using a myconnect URL. Usually, such URLs will be created only for authenticated users. Still, such URLs could be bookmarked or constructed independently of WebSphere Portal APIs.
    - Consider filtering or rewriting the following URL pattern at the web server: .../wps/wcm/webinterface/login/login.jsp.
    - This is not a security vulnerability. This is only a recommendation intended to limit exposures by blocking an extraneous path to log in to the system. You may be able to disable access to this JSP by following this technote.
- Refer to Advanced security hardening in WebSphere Application Server, Part 1 and Part 2 for most other authentication considerations.
    - Especially consider any trust association interceptors (TAI) for external security managers.

- Refer to third-party documentation for other authentication mechanisms (e.g. Tivoli Access Manager, CA SiteMinder, SPNEGO, etc.).

External security managers

Index

WebSphere Portal may be configured to rely upon external security managers to authenticate users and optionally authorize access to WebSphere Portal resources. If you use an external security manager, consider these -

Recommended actions and considerations:

- Trust association interceptor is a WebSphere Application Server interface. Refer to WebSphere Application Server documentation to ensure that the trust association interceptor is secure.
- Verify that logging off clears both ESM cookies and WebSphere cookies (e.g. LTPAToken2 and JSESSIONID) from the cookie store of the browser.
  - This blog post explores alternatives to an ESM logout page.
- Externalizing resources such that an external security manager (ESM) controls access/authorization is for administrative convenience. Favor managing access natively within WebSphere Portal if your business requirements allow it. If you must externalize access control, ensure that the communications link between WebSphere Portal and the ESM is secured. Refer to ESM documentation on hardening the ESM itself.
- The timeout.resume.session and persistent.session.level configuration settings are often used to avoid timing problems with ESM session invalidation. If you set these, recognize that they effectively override the WebSphere Application Server inactive session timeout. The onus is then on the ESM to enforce inactive session timeouts.

Initial access control settings

Index

WebSphere Portal assigns certain roles to certain users and groups, by default. This makes WebSphere Portal usable out-of-the-box, but may not meet the security requirements of your application.

Recommended actions and considerations:

- Review the Initial Access Control Settings section in the InfoCenter. Pay particular attention to the special user and group:
  - Anonymous Portal User
  - All Authenticated Portal Users
  - Note: This is a long list. Pay particular attention to the default role assignment where All Authenticated Portal User is assigned Privileged User@Home(label).

- Review the default permissions associated with Managed Pages.
- Compare these to the security requirements of your application.
- Update role mappings, role blocks, and/or group membership as needed, referencing the InfoCenter section Controlling access.

Examine how your application leverages Portal Access Control

Index

WebSphere Portal provides granular access control on its resources. The access control model allows role mappings to propagate down the resource hierarchy, allows you to override this propagation, and allows group members to inherit role mappings from their groups.

Managed Pages provides methods of adjusting access controls during page authoring and publishing life cycle through projects and workflows.

Access to IBM Web Content Manager libraries and content is managed independently of WebSphere Portal resources.

Collectively, these provide a powerful and dynamic framework for managing access within your application. However, due to the complexity of the access control model, you should carefully evaluate how you leverage it.

Adhere to the principle of least privilege when assigning access rights. Assign each user or group only the minimal roles needed to do their job. For example, if a user needed a customized copy of a page, then the *Privileged User* role on the page would be appropriate. The *User* role would not be appropriate, since that would not permit customization. The *Administrator* role would not be appropriate, since that would allow the user to delete the page for all users.

Recommended actions and considerations:

- What functional roles do users have within your application? Define groups in your user repository that represent these functional roles. Map these groups to roles on WebSphere Portal and Web Content Manager resources, as needed. Such direct, simple associations make it easier to understand the role mappings, thus minimizing the potential for human error which could expose resources to unauthorized users.
- Do not share login IDs among different users. This is particularly important for auditing. For example, multiple administrators should have their own administrative accounts rather than sharing the wpsadmin ID.
- If you have a federated repository, and if users in a single functional role span multiple LDAPs, then consider using application groups to simplify group structures. Likewise, if you have a read-only LDAP.
- Study the resource hierarchy and role hierarchy and consider the implications of inheritance.
    - For example, if you grant *All Authenticated Portal Users* the *Privileged User* role on the *Home* label, they would inherit that role on all the child pages of this label by default. Are there any pages under the *Home* label to which you must restrict access? If so, you may need to implement role blocks.
    - For example, if you grant *Anonymous Portal User* the *User* role on the *virtual resource USERS* to enable People Finder on anonymous pages, do the access control checks in the PUMA REST service meet your security requirements?
    - For example, if you grant *All Authenticated Portal Users* the *Privileged User* role on the *virtual resource USERS,* do your security requirements permit disclosing one

user's information to another user? Either through the application directly (portlets that use the PUMA API) or through the PUMA REST service?

- Per the InfoCenter, the base portal must be associated with the default realm and the default realm must be a superset of the participating base entries of other realms. This means that users from any realm will be able to log in to the base portal. Pay particular attention to what resources *All Authenticated Portal Users* can access in the base portal if you use virtual portals and realms.
- Control access as tightly as possible on the live site (published site). If authors and reviewers need elevated access, as much as possible, grant this access through workflows and projects, and administer the site through Managed_Pages.
- Ensure that store.puma_default.disableACforRead=false in PUMA Store Service unless you have a specific business requirement for setting this to true. Validate any such business requirement against your security requirements.
    - Note that this setting affects access controls for users and groups only. It does not affect access controls for pages, portlets, or content.

Secure communications between WebSphere Portal and the LDAP server

Index

Though Lightweight Directory Access Protocol (LDAP) underlies many security functions, it is not an inherently secure protocol. If LDAP communications are not secured, anyone with access to the network between WebSphere Portal and the LDAP server could trivially obtain user names, passwords, or other sensitive data. The Internet Engineering Task Force advocates securing LDAP communications via TLS (SSL). WebSphere Portal configuration tasks and associated procedures enable you to secure LDAP communications.

Recommended actions and considerations:

- When you initially integrate WebSphere Portal with the LDAP, ensure that LDAP communications are over TLS.
    - Refer to the InfoCenter for general instructions.
    - For additional guidance on LDAP integration, refer to the LDAP Integration Guide.
- If you initially configured non-secure communications with LDAP (e.g. port 389), you may use the WebSphere Application Server Integrated Solutions Console to update the LDAP definition to use a secure connection instead (e.g. port 636).
    - For example, for an LDAP in a federated repository:
        - Global security > Federated repositories > *Repository Identifier* >
            - Port
            - Require SSL communications
            - SSL configuration
    - Refer to InfoCenter or LDAP Integration Guide, linked above, for instructions on getting the LDAP server's certificate.
- Recognize that nearly all requests to the LDAP server are done as the LDAP bind user (primarily through the WebSphere Application Server component, Virtual Member Manager [VMM] - assuming a federated repository, no ESM, and only custom code that uses PUMA for user/group requests; the only exception is the request to validate a user's password during authentication). The LDAP server is entirely reliant upon WebSphere Portal access controls to protect objects in the Directory Information Tree (DIT) to which

the bind user has access. Confirm with the LDAP administrator that access rights have been assigned to the bind user according to the principle of least privilege.

Impersonation

Index

The impersonation feature in WebSphere Portal enables users with sufficient access rights to act as a different user within the application, without actually having that user's credentials. Impersonation is most often used for:

- Authors to view the site as a regular user, while they are building the site,
- Reviewers (in the context of Managed Pages) to view the site as a regular user before approving updates,
- Help desk personnel to troubleshoot access control or other user-specific problems.

There are several major security concerns with impersonation which you should consider.

Recommended actions and considerations:

- Do you need to use impersonation in your application? If not, consider disabling impersonation altogether.
- If your application requires impersonation, do your security requirements permit impersonating any arbitrary user? Should users be able to impersonate the WebSphere Portal administrator (e.g. wpsadmin)? Should they be able to impersonate the CEO?
  - If not, the role mapping *Can Run As User@USERS (virtual resource)* suggested in the InfoCenter is too broad. Assign the *Can Run As User* role per-user-group instead.
- Does your application participate in SSO with other applications via LTPA? If so, impersonation in WebSphere Portal may circumvent the security controls of those other applications. Consider limiting the scope of user impersonation in your environment.
- Do your security requirements permit impersonating users without their consent? If not, then the default impersonation portlet is not sufficient. Consider developing a custom portlet instead.
- If impersonation is enabled in your environment, enable auditing for impersonation events. As a deterrent, inform anyone with *Can Run As User@X* that impersonation events are auditable.

Remember me cookie and step up authentication

Index

The Remember me cookie allows WebSphere Portal to identify and optionally authenticate users based on a persistent cookie. Step up authentication enforces stricter authentication requirements for particularly sensitive resources.

Recommended actions and considerations:

- Enable the remember me cookie only if your application requires it.
- J2EE authentication allows users to authenticate to WebSphere Portal with only a remember me cookie (i.e. without presenting any other credentials, such as user name and password). Carefully consider your security requirements before enabling this

feature. Leave it disabled unless you have specific functional requirements for it.
    - Does your application participate in an SSO realm with any other applications via LTPA? If so, consider restricting the domain of the LTPA token on WebSphere Portal server. Other applications likely could not distinguish between an LTPA token issued via remember me versus normal authentication methods.
- If J2EE authentication is enabled, review the resource hierarchy and assign any particularly sensitive resources an authentication level higher than *Identified*. Step-up authentication will then force users to submit credentials to access those resources.

AJAX proxy

Index

The same origin policies (W3, IETF, and Javascript) of web browsers prevent one site from accessing the data of another site. The AJAX Proxy enables you to selectively circumvent same origin policies. Javascript on a WebSphere Portal page may request XML from another site through the AJAX Proxy.

Recommended actions and considerations:

- What level of trust do you place in sites accessed through the AJAX Proxy?
    - What assumptions do your scripts make about this data? Could an XSS attack be implemented from the other site (e.g. a DOM-based attack)?
- Which portlets need to access the back-end application? If only one or few portlets, then consider using application scoped profiles (alt. v8) to limit the scope of the exposed data.
- Which users/clients need access through the AJAX proxy? Control access (alt. v8) as tightly as your functional requirements permit to limit the security exposure inherent in circumventing same origin policies. Control access via:
    - Access policies
    - IP filtering
- If the application accessed through the AJAX Proxy requires authentication, how is that achieved? SSO via cookies? Credentials from a credential vault slot? HTTP Basic authentication? Are the credentials and/or cookies protected on the network path between WebSphere Portal and the other application (e.g. transmitted via HTTPS)?

Auditing

Index

WebSphere Portal allows you to audit certain administrative events, such as impersonation, mentioned above. Most of the auditable events are related to user management and access control.

Recommended actions and considerations:

- Enable audit logging for all events for which non-repudiation is a security requirement in your application. Doing so allows you to associate a user identity with the action.
- As a deterrent, inform your users/administrators that auditing is enabled.
- Certain actions, like modifying page layout, are not auditable by default. However, if you

sufficiently restrict access and leverage Managed Pages as described above, you could implement auditing in custom workflows to log these events.

Credential Vault

Index

The Credential Vault allows portlets to access other applications using credentials obtained from the vault. These may be HTTP headers on the current request (active [deprecated in v8.5]; e.g. LTPA token) or credentials stored in the vault (passive; e.g. user name and password, TLS certificate).

Recommended actions and considerations:

- To the extent possible, use SSO methodologies to access back-end applications. If WebSphere Portal does not store credentials, there is a smaller attack profile for the application. Note that active credentials have been deprecated in v8.5.
- If you must use passive credentials, recognize that the default vault adapter only Base64 encodes and stores credentials in the WebSphere Portal database (release and/or customization domain).
  - This encoding would not meet the security requirements for most organizations. You should either:
    - Use Tivoli Access Manager Global Sign-on (GSO) lockbox and the associated vault adapter instead.
      or:
    - Implement the vault adapter interface with sufficient encryption and specify your class in the Credential Vault Service
    - You may reasonably make an exception for system credentials, providing sufficient controls over the database. For example, if you use the credential vault only for storing credentials to enable syndication.
- Set export.enforceSSL to ensure that any XMLAccess requests to export credential vault slots are secured.
  - Similarly, adjust other configuration properties as-needed to meet your security requirements.

HTTP Basic Authentication

Index

WebSphere Portal provides a trust association intercepter (TAI) for HTTP Basic Authentication to allow simple clients, such as WebDAV, to connect. HTTP Basic Authentication is not as secure as form-based login via HTTPS with LTPA for SSO.

Recommended actions and considerations:

- Disable the HTTP Basic Authentication TAI if you are not using WebDAV (e.g. to manage custom themes). Do so by setting enabled=false.
  - Consider enabling this TAI on an as-needed basis (e.g. when deploying themes, installing maintenance).
    - Modular themes require the HttpBasicAuthTAI to deploy via WebDAV clients or

the webdav-deploy-zip-file task.
        - Cumulative fix upgrades will fail if the TAI is not enabled.
        - The Portal servers must be restarted to load any TAI configuration changes.
- If you require the HTTP Basic Authentication TAI, tightly restrict which requests the TAI will attempt to authenticate, preferably with the configuration parameters:
    - userAgentWhiteList
        - The cumulative fix installation process uses one or more user-agents with "Java" in it.
    - urlWhiteList
- Note that WebDAV/HTTPS is not supported. Consider the security of your network when deciding where to run the WebDAV client.

Preventing vulnerabilities in custom code

Index

As stated in the introduction, WebSphere Portal is a platform upon which to build custom applications. WebSphere Portal cannot guard against any and all potential security vulnerabilities that could be introduced by custom code. Application architects should take steps during development and test to avoid such vulnerabilities. Since custom themes, custom portlets, and other custom components all contribute to the page source, each bears some responsibility in guarding against attacks on the application.

A comprehensive treatment of web application security is beyond the scope of this guide. As suggested in the introduction, application architects should seek out additional resources on web application security.

Application architects should be aware of and pay particular attention to guard against the following (*this is not a comprehensive list – only a starting point*):

Cross-site scripting

Cross-site scripting (XSS) is a type of attack on a web application where either nefarious scripts are injected into page content, or vulnerabilities in the application's scripts are used against the application itself. There are three basic types of XSS:

- Persistent: an application stores user input and later uses it to build the page source.
- Reflected: user input (e.g. the requested URL) is immediately reflected in page source.
- DOM-based: scripts on the page are exploited and run in an unintended manner to attack the user/session.

Request forgery

With request forgery, a user unwittingly makes a request to the application. The browser submits authentication and session cookies automatically, which may bypass the security controls of the application. There are two main types:

- Cross-site request forgery (CSRF / XSRF): request originates from an external site; the same origin policies of modern browsers generally force these to be one-way attacks. JSON hijacking is a sub-type of XSRF.
- On-site request forgery (OSRF): request originates from the site itself, most often

implemented via XSS.

Injection
Certain vulnerabilities allow attackers to submit code as user input and have it run on the application server or back-end server. The most common of these are SQL injection and LDAP injection. Pay particular attention to these if any custom portlets access their own databases or make calls directly to the LDAP.

Redirection
In redirection attacks, the application redirects users from the URL they requested to some other URL, perhaps to another site entirely. These could be used for spear phishing attacks, in particular. If user input (especially the requested URL itself) is the basis for any redirection in custom code (e.g. login filters), inspect that code and consider whether it could be leveraged in a redirection attack.

Recommended actions and considerations: (These focus on XSS.)

- Ensure that all custom components HTML-encode anything they contribute to page source.
  - <c:out> with escapeXML from the JSP standard tag library (JSTL) is a good tool for HTML-encoding output from JSPs.
  - Custom theme developers should confirm that they HTML-encode all output.
  - Custom theme developers should confirm that all custom scripts have been reviewed for DOM-based vulnerabilities.
  - Custom portlet developers should confirm that they HTML-encode all output.
  - Custom portlet developers should confirm that all custom scripts have been reviewed for DOM-based vulnerabilities.
  - Consider both server-side code (JSPs, Java classes) and client-side code (Javascript).
  - HTML-encoding output is not sufficient to guard against any and all XSS attacks. Consider especially whether any user input is inserted into custom Javascript.
- Ensure that all custom components follow other best-practices to guard against XSS. Here are a few references:
  - W3
  - OWASP, with links to cheat sheets
  - Prevent a cross-site scripting attack
- Scan your application with an automated web security tool. Consider using IBM Security AppScan and related applications.
- Consider whether to set security.css.protection=true in the Configuration Service. This instructs WebSphere Portal to HTML-encode certain user input before passing it along to custom portlets.
  - For example, < and > would be converted to *&lt* and *&gt*.
  - This can cause problems in some portlets which do not expect this encoding.
  - *This setting offers only minimal protection against XSS.* The preferred method is to validate output, as described above.

Odds and Ends

Index

Consider the following several items which do not directly relate to the categories above.

- Maintain a regular schedule for updating administrative passwords.
- Maintain a regular schedule for updating LTPA keys.
- Run WebSphere Portal as non-root user.
- Restrict access to log files at the operating system level.
- Work with the database administrator to ensure that the rights assigned to WebSphere Portal's database user adhere to the principle of least privilege. While administrative rights may be required during initial configuration, WebSphere Portal should not act as a database administrative user at run-time.
- Should your application prevent users from seeing WebSphere Portal or WebSphere Application Server error messages (e.g. stack traces associated with HTTP 500 errors)? Should your application serve custom, generic error pages instead? If so, consider using a proxy server and special error page application.
- Review who can access the Configuration Wizard and adjust its security settings as needed. Security for the Configuration Wizard is entirely separate from WebSphere Portal security (it has its own profile). Manage these security settings via WebSphere Application Server administrative tools.
- When running configuration tasks, favor specifying passwords in properties files rather than at the command line. If you specify passwords in properties files, be sure to delete these passwords after running the configuration tasks. Restrict access to these properties files with operating system controls (e.g. chmod).
    - Note: ConfigEngine properties files do not currently support encrypted passwords.
- As a general precaution, apply the property com.ibm.ws.webcontainer.disallowserveservletsbyclassname=true (which disables the serving of servlets by classname at the application server level) according to the description in the flash related to WAS APAR PK52059. Application developers should explicitly disallow serving servlets by classname.
- Should administrators run XMLAccess through the web server? If not, filter out .../wps/config and .../wps/config/* requests at the web server (if customized, substitute your actual custom context root for “wps”).
- In the spirit of the recommendation in the WebSphere Application Server Hardening Guide to not run samples in production, consider disabling unused applications.
    - The technote linked above is for WebSphere Portal v7. Monitor the WebSphere Portal support site for updates to this technote.
- IBM recommends against using the file repository in a production evironment. If you choose to use the file repository, consider access controls on fileRegistry.xml and the strength of encryption of passwords stored in this file.
- Use caution if your application leverages Cross Origin Resource Sharing (CORS). Extend the trust domain of WebSphere Portal to the least possible extent.
- Regarding cross-site request forgery (CSRF, XSRF), consider:
    - Requests to WebSphere Portal REST services to modify resources require HTTP

PUT, DELETE, or POST (with XML data) which the same origin policies of modern browsers should prevent other sites from compromising. HTTP POST with no data would be allowed, but any returned data could not be processed by the other sites' scripts and such a POST would not modify WebSphere Portal resources.
        - WebSphere Portal uses these REST services for some functionality, including certain elements of the default theme.
        - Some organizations' security policies restrict the HTTP verbs their web servers support. If your web server is so restricted, tunnel such requests by setting x-method-override.enabled.
    - A different site could induce HTTP GET requests, but same-origin policies of modern browsers should prevent the site's scripts from accessing any data returned.
    - Action identifiers are encoded in portlet action URLs, which are tightly bound to the current session. Monitor SystemOut.log for indications of attempts at actual replay attacks. Any such attempts should be blocked by the state preprocessor of WebSphere Portal.
- As you did with the AJAX proxy, identify all other sources of content on your WebSphere Portal pages. Similar considerations exist for the Web Application Bridge, WSRP, and ATOM feeds.
- Evaluate whether the configuration of WebSphere Portal meets your requirements for implicit logouts. The term *implicit logout* may refer to when an authenticated user (valid LTPAToken2) requests unprotected content (e.g. .../wps/portal/...) - with certain configurations WebSphere Portal will log out the user (i.e. expire LTPAToken2 and JSESSIONID from the cookie store of the browser). Other conditions may affect implicit logouts as well. The configuration points to consider are:
    - uri.home.substitution (InfoCenter reference)
    - logout.user.onpublic

- Ensure that communications between WebSphere Portal and its database are secured.
    - Encrypting these communications requires JDBC Type 2 Driver, which is deprecated in WebSphere Application Server 8.5.
    - If encrypting this link is not feasible, ensure the security of the network.
- If any external search crawler authenticates to WebSphere Portal to traverse protected content, ensure that the user has read-only access to WebSphere Portal resources.
- To guard against Clickjacking, add the X-Frame-Options header via a web server directive. If this is not feasible with your web server of choice, then consider implementing defensive scripts in your custom theme.

Beyond the scope of this guide

Index

When evaluating the security of your application, also consider the following items which are beyond the scope of this guide. Some of these items, particularly those relating to network and infrastructure, are addressed in the security hardening guide for WebSphere Application Server.

- Basic system security
    - physical access

    - operating system access
    - disaster recovery
- Basic application security
    - inducing outages, hangs, or crashes – effectively equivalent to denial of service (DoS)
- Security for front-end and back-end applications
    - External security manager
    - database
    - LDAP
- Network / web attacks
    - DoS / DDoS – Denial of Service / Distributed Denial of Service – sometimes called flood attacks.
    - Firewalls
- Cloud (2)
- Training users
    - Social engineering attacks
    - Other software running on client systems (keystroke loggers, etc.)

Closing

Index

Administrators, architects, and developers must consider many factors when evaluating the secuirty of their web application. WebSphere Portal and WebSphere Application Server can be leveraged to address many security requirements. WebSphere Portal and WebSphere Application Server both provide extension points to allow custom code to extend security functionality as well.

About the author

Index

Jason Wicker is a Software Engineer with IBM Software Group. He has served in technical support for WebSphere Portal, focusing on security and administration, since 2007. He joined IBM after graduating from the **University of North Carolina** with a B.Sc. in Applied Science - Computer Science in 1998. Jason can be reached at jwicker@us.ibm.com.

Contributors

Index

The author extends special thanks to the following contributors who provided input to, technical review of, and feedback on this guide:

Daniel Blum – Security Architect, IBM WebSphere Portal
Donald Jones – Chief Programmer, IBM WebSphere Portal
Sascha Schefenacker – Developer, Security Team; IBM WebSphere Portal
Travis Cornwell – Support for Security, Administration, and Run-time; IBM WebSphere Portal