HCL Digital Experience

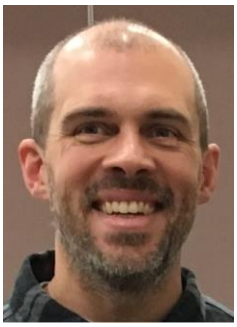
Security Hardening Guide
for
HCL Digital Experience

Jason Wicker – jason.wicker@hcl.com

Table of Contents

Author

This document was created by the following Subject Matter Expert:



Jason Wicker
HCL

Bio

Jason Wicker is a software engineer with HCL. He has worked on Digital Experience, focusing on web application security, since 2007. He graduated from the University of North Carolina with a B.S. in Applied Science - Computer Science in 1998.

Contact: jason.wicker@hcl.com

Introduction

Security is a fundamental requirement for most web applications. Organizations often 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 how to evaluate and improve the security of applications based on HCL Digital Experience.

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

- Authentication
  - *Who is this?*
  - Users provide credentials to identify themselves to the application.
- Authorization
  - *What can they do?*
  - Application determines what data the user can access and what functions they can perform on/with that data.
- Non-repudiation
  - *What have they done?*
  - Establish a record of actions against which user claims can be verified.
- Confidentiality
  - *Keep secrets*
  - Prevent unauthorized disclosure of data, in storage and in transit.
- Integrity
  - *Stop corruption*
  - Prevent unauthorized modification of data, in storage and in transit.
- Availability
  - *Be reliable*
  - Ensure that authorized users can access data, when needed.

Prerequisites

This guide assumes a general understanding of web application security.

Web application security is:

- Fundamental information security concepts,
- Considered from the perspective of a web application,
- Integrated into a comprehensive security plan via risk management.

Web applications are commonly the primary interface by which end users access information. Few are entirely self-contained, but are distinct components of broader, integrated information systems.

Such integrated components comprise a *trust chain*. Longer trust chains require greater effort to secure. However, the t*rust boundaries* which exist between components can help improve security by providing *defense-in-depth* – hardening each component can limit the scope and likelihood of any attack.

Each section in this guide introduces a common security requirement for, or security functionality of, HCL Digital Experience. Each section includes a *Recommended actions and considerations* section to guide you through evaluating and improving the security footing of your application.

Following this guide could require a months-long or open-ended project involving multiple teams that should result in an application-specific security hardening procedure that spans development, deployment, and maintenance. Regularly review this *Security Hardening Guide for HCL Digital Experience* for updates.

Recommended actions and considerations

- Independently research web application security to identify other resources to inform your evaluation. Especially consider:
  - OWASP
    - Top 10
    - Secure Coding
    - Application Security Verification Standard
    - Proactive Controls
    - Secure Development Lifecycle
  - National Institute of Standards and Technology
    - National Vulnerability Database
    - Computer Security Resource Center
  - MDN – HTTP
  - Google Browser Security Handbook
  - Google Web Security Fundamentals
  - Cybersecurity and Infrastructure Security Agency
    - Build Security In
    - Information Security
  - European Union Agency for Cybersecurity
  - Web Application Hacker's Handbook

Comprehensive Security Planning

HCL Digital Experience is a platform upon which to build an application. You can integrate it into a wide variety of digital ecosystems. The extent to which you may customize HCL Digital Experience and associated security functions in IBM WebSphere Application Server is nearly endless. Your approach to security hardening your application should encompass:

- HCL Digital Experience
- IBM WebSphere Application Server
- Custom code (themes, portlets, filters, etc.)
- Operating system (client and server)
- Container platform (if applicable)
- Web clients (e.g. browsers)
- Web servers and load balancers
- External security managers or other identity providers
- Back-end applications, such as databases, LDAP servers, etc.
- Network security (including DNS, TCP/IP, etc.)
- Physical security

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 and beyond your organization. Who is responsible for the security of, or otherwise has control over the following?
  - the network, including firewalls
  - the web server
  - the LDAP server
  - any external security manager or other proxies
  - any identity providers
  - the database server
  - other back-end servers (integrated via Web Application Bridge, WSRP, etc.)
  - 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 HCL Digital Experience*, circle back and address any remaining items.
  - Pay special attention to the introductory sections that address infrastructure.
- Read Web security concepts and considerations ... for an overview of the main security components of HCL Portal.
- 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
- SAML
- OpenID Connect

Install all available security fixes

HCL Software is committed to the safety and security of all our products and services. The HCL Software Product Security Incident Response Team (PSIRT) is a global team that manages the investigation and coordination of security vulnerability information related to HCL Software offerings. This team will coordinate with HCL Software product development teams to investigate and identify the appropriate response plan to security vulnerabilities in HCL products.

HCL publishes Security Bulletins to disclose security vulnerabilities. These are available on the HCL Support portal. To help you evaluate and address each vulnerability, these security bulletins include CVE tracking numbers, CVSS scores, mitigations, and links to fixes.

HCL also posts important information regarding security vulnerabilities to the HCL PSIRT Blog. Subscribe to this blog to receive notifications about security vulnerabilities and remediation procedures.

Recommended actions and considerations

- Install the most current maintenance for HCL Digital Experience that is suitable for your application.
- Subscribe to the HCL PSIRT Blog to be notified when security bulletins are published for new vulnerabilities.
  - 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 decisions regarding 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 HCL Digital Experience support custom themes based on certain default themes delivered with earlier versions. Monitor security bulletins for your current HCL Digital Experience version and any older versions used in your custom themes.
- Similarly to custom themes, integrate any security fixes for samples in 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

  Refer to your vendors for these components regarding their practices for publishing security bulletins.
- If you discover a new security vulnerability in HCL Digital Experience, report it to HCL by opening a case.

Enable HTTPS

Clients communicate with HCL Digital Experience via 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 use TLS to secure communications with clients. Since these communications include such sensitive information as application data and single sign-on (SSO) cookies, HCL recommends enabling HTTPS for Digital Experience. The HCL Digital Experience Product Documentation documents how to do this.

The procedure in the Product Documentation 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 Knowledge Center 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 it similarly protects users' credentials.

Recommended actions and considerations

- Enable HTTPS per the Product Documentation.
  - Additional IBM HTTP Server technote, for reference.
- Evaluate whether the common requirements listed above address all 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 HCL Digital Experience. Doing so would protect sensitive data submitted or served under URLs other than .../wps/myportal (e.g. wps/portal, wps/um).
  - Especially consider 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 HCL Portal only), implement a rewrite rule on the web server to redirect all http://... Traffic 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), and if the context has not been removed, 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 HCL Digital Experience, most content is served under wps/* or a similar customized context. Check the URI Names in plugin-cfg.xml generated from HCL Digital Experience 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 for encrypting the web server to web container link is not feasible for HCL Digital Experience.
  - 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.
  - 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 is still open, 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 HCL Portal could easily obtain sensitive data or otherwise alter these communications.

Guard your cookies

Cookies enable single sign-on (SSO) and session tracking in HCL Digital Experience. Unauthorized parties could circumvent certain security controls given access to:

LTPAToken2: Lightweight Third-Party Authentication: IBM 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: The session management of Digital Experience uses this cookie.

Note: These are the default names for these cookies, though you may customize them in recent versions of HCL Digital Experience and IBM 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.
- Consider setting SameSite on these cookies as a mitigation against cross-site request forgery (CSRF/XSRF).
- Set the LTPAToken2 expiration as short as your functional requirements permit.
    - While 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
    - Test and verify the inactive session timeout. If it does not work as expected, check UseInvalidatedId.
- 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 identity (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
- Consider setting HTTPOnly for any other cookies your application might leverage, at the web container level.
- If the controls above prove insufficient for any application cookies, consider setting attributes at the web server (with mod_headers.so):

```
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

or:

```
Header set Set-Cookie HttpOnly;Secure
```

Authentication mechanism

How can users log in to your application? HCL Digital Experience primarily relies upon IBM 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. SAML, SPNEGO) to authenticate users.

The main consideration from an HCL Digital Experience perspective is the way in which the application initially receives user credentials, often via the default login portlet or a custom login portlet. HCL Digital Experience 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 Digital Experience 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.
  - You may disable the automatic login URL altogether by setting custom property isLoginUrlActive=false (boolean) in the Authentication Service.
  - If your application requires the automatic login URL, 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.
- HCL Digital Experience 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 (LDAP, generally). Discuss your options with the LDAP administrator.
- HCL Digital Experience 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. Discuss this with your LDAP administrator.
  - Optionally, a custom login portlet or custom authentication filter could help guard against this.
- HCL Digital Experience neither supports nor prevents multiple concurrent logins. If that is a requirement, 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 Digital Experience 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 recommendation is only 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 (OpenID Connect, SAML, Tivoli Access Manager, CA SiteMinder, SPNEGO, etc.).
- Carefully evaluate the security implications of Transient Users, if enabled for OpenID Connect, SAML, etc. integration.
    - Login modules can choose to treat these as *All Authenticated Portal Users* or can establish group membership for the purpose of access controls.
    - What implications are there for other applications in an LTPA-based SSO domain?

External security managers

HCL Digital Experience may be configured to rely upon external security managers to authenticate users and optionally authorize access to Digital Experience 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 with portlet code like the following, in doHeaders(). You could render this portlet on a logout page – this is a sample only:

```
Cookie[] cookies = request.getCookies();
        if (cookies != null)
        {
            for (Cookie cookie : cookies)
            {
                if (cookie.getName().equals("Cookie1") ||
cookie.getName().equals("Cookie2") ||
cookie.getName().equals("Cookie3"))
                {
                    cookie.setPath(PATH);
                    cookie.setDomain(DOMAIN);
                    cookie.setMaxAge(0);
                    cookie.setValue("");

                    response.addProperty(cookie);
                }
            }
        }
```

- Externalizing resources such that an external security manager (ESM) controls

access/authorization is for administrative convenience only. Favor managing access natively within HCL Digital Experience if your business requirements allow it. If you must externalize access control, ensure that the communications link between Digital Experience 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. Then, the onus is on the ESM to enforce inactive session timeouts.

Initial access control settings

HCL Digital Experience assigns certain roles to certain users and groups, by default. This makes HCL Digital Experience 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 Product Documentation.
  - Specifically, pay attention to the special user and group:
    - Anonymous Portal User
    - All Authenticated Portal Users
  - Note: This is a long list. Especially consider the default role assignment where All Authenticated Portal User is assigned Privileged User@Home(label).
  - Also, especially consider the default role mappings for All Authenticated Portal Users to have User@ on the virtual resource USERS and the virtual resource USER GROUPS. Do your security requirements permit disclosing one user's information to another user? Either through the application directly (portlets that use the PUMA API, like People Finder) or through the PUMA REST service?
- 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 Product Documentation section, Controlling access.

Portal Access Control

HCL Digital Experience 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 Web Content Manager libraries and content is managed independently of HCL Digital Experience 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 analyze how you use 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 HCL 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.
- Add users to these groups, as needed.
- Establish a process of periodically reevaluating group membership (e.g. as employees change departments or leave companies).
- 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. They will have equivalent access as the portal administrator, providing they are all members of the administrative group (the various *.admingroup property values in the Access Control Data Management Service).
- 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? See the Initial Access Control Settings section for more details.
- Per the Product Documentation, the base portal is associated with the default realm, by default, and the default realm must be a superset of the participating base entries of other realms. This means that users from any realm may be able to log in to the base portal. Especially consider which resources *All Authenticated Portal Users* can access in the base portal if you use virtual portals and realms or consider changing the default.
- Control access as tightly as possible on the live site (published site). If authors and reviewers need elevated access, grant such 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 with the LDAP server

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 HCL Digital Experience 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). HCL Digital Experience configuration tasks and associated procedures enable you to secure LDAP communications.

Recommended actions and considerations

- When you initially integrate HCL Digital Experience with the LDAP, ensure that LDAP communications are over TLS.
  - Refer to the Product Documentation 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 Product Documentation 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, which is an LDAP bind operation as that user identity). The LDAP server is entirely reliant upon HCL Digital Experience access controls to protect objects in the Directory Information Tree (DIT) to which the bind user has access. Ask the LDAP administrator to verify that access rights have been assigned to the bind user according to the principle of least privilege.

Impersonation

The impersonation feature in HCL Digital Experience 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 Digital Experience 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 Product Documentation is too broad. Assign the *Can Run As User* role per-user-group instead. For example, if you assign AdminGroup1 the role Can Run As User@GroupA, then any member of AdminGroup1 could impersonate any member of GroupA.
- Does your application participate in SSO with other applications via LTPA? If so, impersonation in Digital Experience 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

The Remember me cookie allows HCL Digital Experience 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 Digital Experience 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 to the Digital Experience 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.
- Set the Secure attribute on the RememberMe cookie.

Outbound HTTP connection

The same origin policies (W3, IETF, and JavaScript) of web browsers prevent one site from accessing the data of another site. Outbound HTTP Connection enables you to selectively circumvent same origin policies. For example, JavaScript on a Digital Experience page may request XML from another site through the HTTP Outbound Connection proxy (formerly, the AJAX Proxy).

Recommended actions and considerations

- What level of trust do you place in sites accessed through the HTTP Outbound Connection?
  - 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 to limit the scope of the exposed data.
- Which users/clients need access through Outbound HTTP Connection? Control access 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 back-end application accessed through Outbound HTTP Connection 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 Digital Experience and the other application (e.g. transmitted via HTTPS)?

Auditing

HCL Digital Experience 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

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 HCL Digital Experience does not store credentials, there is a smaller attack profile for the application.
  - Though active credentials have been deprecated in v8.5, if you use Outbound HTTP to connect to back-end web applications, then you can configure the profile to pass SSO cookies through as an alternative.
- If you must use passive credentials, recognize that the default vault adapter only Base64 encodes and stores credentials in the HCL Digital Experience database (release and/or customization domain).
  - This encoding would not meet the security requirements for most organizations. You should either:
    - Use IBM Security Access Manager Global Sign-on (GSO) lockbox and the associated vault adapter instead.
      or:
    - Implement the vault adapter interface, com.ibm.portal.portlet.service.credentialvault.spi.VaultAdapter, 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. Understand that this password could be trivially obtained by anyone with access to the database.
- 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 (e.g. export.cipher, export.keyLength).

HTTP basic authentication

HCL Digital Experience provides a trust association interceptor (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. Basic authentication base-64 encodes the user ID and password and submits them with every request. Base-64 encoding is not secure and can be trivially decoded.

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).
- If you require the HTTP Basic Authentication TAI, tightly restrict which requests the TAI will attempt to authenticate, preferably with the configuration parameters:
  - userAgentWhiteList
  - urlWhiteList
- Note that WebDAV/HTTPS is not supported. Consider the security of your network when deciding where to run the WebDAV client, knowing that IDs and passwords may be transmitted while only base-64 encoded.
- Syndication relies on basic authentication, as provided by WebSphere Application Server. It can

function over SSL and should be configured to do so, considering the security of the network.

Custom code

HCL Digital Experience is a platform upon which to build custom applications. Digital Experience cannot guard against 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.

Recommended actions and considerations

- Always follow secure coding practices when developing and deploying custom code.
- Web application architects and developers should be trained on common risks and attack types.

After security misconfiguration, XSS is the most common flaw in web applications. Developers should be especially familiar with techniques for avoiding XSS:

- Ensure that all custom components HTML-encode or URL-encode markup they contribute to page source, where appropriate.
    - <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 theme developers should ensure that packaged libraries are well-maintained, repackaging as security vulnerabilities are fixed.
    - 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.
    - Custom portlet developers should ensure that packaged libraries are well-maintained.
    - Consider both server-side code (e.g. JSPs, Java classes) and client-side code (e.g. Javascript).
    - HTML-encoding output is not always sufficient to guard against XSS attacks. Consider whether any user input is inserted into custom Javascript.
    - Consider <c:url> and java.net.URLEncoder or similar tools for URL encoding.
- 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 HCL AppScan and related applications.
- Consider whether to set security.css.protection=true in the Configuration Service. This instructs

Digital Experience 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.

Containers

HCL Digital Experience 9.5 and future releases ship several docker containers, all of which are based upon the secure Red Hat UBI image and were built as non-root user.

Containers deployments may provide improved security, especially relating to physical security, availability, key management services, and CI/CD.

If you plan to deploy one or more of these, then consider the following –

Recommended actions and considerations

- Trust boundaries & responsibilities:
  - When moving from a traditional on-premise deployment to a hybrid or fully containerized deployment, you must clearly delineate responsibilities.
  - All cloud service provider responsibilities should be verified by the provider – do not only assume the scope of their role in security.
  - Establish responsiveness criteria from the service provider.
- Establish a procedure for asset management. In cloud environments, the simplicity of provisioning comes at a cost – provisioned resources may be forgotten, creating a larger attack surface. Ensure that assets are decommissioned when they are no longer needed.
- These images use the default administrator ID and password in the file based repository. Update the Portal and WebSphere Application Server administrative IDs and passwords so that they are not the defaults. Refer to the *Odds and Ends* section regarding recommendations against using the file repository in production.
- The HCL Digital Experience Operator container is only leveraged with version 9.5 for Red Hat OpenShift deployments and the best practices for Red Hat OpenShift security should be followed to ensure a secure deployment.
  - Basics
  - As with other fundamental technologies, the recommendations in this guide are not comprehensive. Independently research security hardening guidelines for Red Hat OpenShift.
- At the time of this writing, HCL Digital Experience API container tech preview should not be used in production, but only for development because client/API communication does not support HTTPS. When future releases support this, prior to promoting this to production, be sure to:
  - Configure HCL Digital Experience API container tech preview to connect to HCL Portal via HTTPS.
  - Configure HCL Digital Experience API container tech preview to be accessed via HTTPS.
- For more information on Docker container security, start here and continue researching independently.

Availability

Choose a topology that meets your business requirements for the availability of your application. Ensure that backup and recovery processes are in place to restore the application, if any catastrophe occurs.

Recommended actions and considerations

- On-premise solutions should consider implementing multiple clusters in separate data centers.
- For systems with a containerized architecture, replica sets ensure a stable number of pods.
- Backup and recovery procedures depend on the topology:
  - On-premise
  - Container
- Backup and recovery procedures should address:
  - File systems
  - Databases (online/offline)
  - User repositories
  - Downtime

Odds and Ends

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

Recommended actions and considerations

- Maintain a regular schedule for updating administrative passwords.
- Maintain a regular schedule for updating LTPA keys.
- Run Digital Experience as a 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 Digital Experience's database user adhere to the principle of least privilege. While administrative rights may be required during initial configuration, Digital Experience should not act as a database administrative user at run-time.
- Should your application prevent users from seeing Digital Experience 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 Digital Experience 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.
    - CVE-2015-1927 from WebSphere Application Server changes the default to be secure. Also see this bulletin.
- 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. Engage HCL Support for assistance, if needed.
- HCL recommends against using the file repository in a production environment. 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 Digital Experience to the least possible extent.
- Regarding cross-site request forgery (CSRF, XSRF), consider:
    - Setting the SameSite attribute on cookies (see *Guard your cookies* section).
    - Requests to Digital Experience 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 Digital Experience resources.
        - Digital Experience 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 Digital Experience.
- As you did with the Outbound HTTP Connection, identify all other sources of content on your Digital Experience pages. Similar considerations exist for the Web Application Bridge (WAB), WSRP, and ATOM feeds. Note that WAB uses iframes, to which modern browsers generally apply same origin policies, depending on X-Frame-Options.
- Evaluate whether the configuration of Digital Experience 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 Digital Experience will log out the user (i.e. expire LTPAToken2 and JSESSIONID from the cookie store of the browser). Other conditions may trigger implicit logouts as well. The configuration points to consider are:
    - uri.home.substitution
    - logout.user.onpublic

- Ensure that communications between Digital Experience and its database are secured. Refer to #29.
    - Encrypting these communications may require a JDBC Type 2 Driver to avoid certain problems, though type 2 drivers are deprecated in WebSphere Application Server 8.5.

  - Refer to JDBC driver documentation for details on its security configuration. For example, with DB2 and JDBC type 4, encryptionAlgorithm and ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY are of particular importance.
  - If encrypting this link is not feasible, otherwise ensure the security of the network.
- If any external search crawler authenticates to Digital Experience to traverse protected content, ensure that the user has read-only access to Digital Experience resources.
- Any search user should have read-only access to the application. Test the search application to verify functionality.
- There should be no confidential information in the unique name of any portal object (pages, portlets, etc). To check:
  - Export the configuration:
    - <portal>/bin/xmlaccess.sh -in <portal>/doc/xml-samples/Export.xml -out exportResults.xml -url http://<server>:<port>/wps/config
    - Search exportResults.xml for *uniquename*= and confirm that none of the unique names in the system include confidential or otherwise sensitive data.
- 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.
- To guard against various forms of host header attacks against the Digital Experience server, consider:
  - Setting host.name in WP ConfigService.
  - Specifying virtual hosts in WebSphere Application Server.
  - Similar configuration may be required on the web server, external security manager, or other proxies. For example, IBM HTTP Server can be configured to ignore the Host header by setting UseCanonicalName ON.
  - Such settings may limit your application in generating self-referential URLs. Users could be forced to use a specific host name (host.name) or set of host names (virtual hosts) when navigating the application. Consider which host names your application requires (e.g. direct access, virtual portals) when configuring these items.
- Utilize HTTP Headers as appropriate for your installation. Lists can be found here and here. Consider especially:
  - *X-XSS-Protection* : Digital Experience default content works with a setting of 'X-XSS-Protection: 1; mode=block'.
  - *X-Frame-Options* : Digital Experience default content works with a setting of 'X-Frame-Options: sameorigin'
  - *X-Content-Type-Options* : Digital Experience out of the box content works with a setting of 'X-Content-Type-Options: nosniff'
  - Implement via mod_headers (late processing), a servlet filter, or both.
- Verify the java.security file on your system.
- In the WebSphere Application Server Integrated Solutions Console, under *Security > Global Security > Authentication cache settings*, set *Use basic authentication cache keys* to **false**. Otherwise, the *checkPassword()* method of the LoginService interface could have unexpected results.
- Consider configuring HTTP/2 between browsers and the web server, to guard against HTTP Desync attacks.
- Web browsers may retain user IDs and passwords in memory. Generally, if browser memory is compromised, other attacks could easily be undertaken. Still, it would be best practice to close the browser to clear the memory, and you may choose to encourage users to do so on your logout page. You may also instruct users not to submit memory dumps for analysis. The design of the default Login portlet and WebSphere Application Server subcomponent VMM preclude any remediation, other than a custom login portlet and custom user repository that both

leverage salted hashing of passwords.
- Configure blacklists and whitelists appropriately for your applications, to guard against path traversal.
- In Digital Experience 9.5, the Practitioner Studio theme is intended for use in Practitioner Studio only. Do not assign this theme to pages which non-administrators can access. Do not use the Practitioner Studio theme as the basis for any custom themes.
- IBM WebSphere Application Server cannot block HTTP methods, so block any invalid or unwanted HTTP methods at the web server.
- Disable remote access to WCM Modules by setting the following in WCM WCMConfigService:

  ```
  connect.businesslogic.module.<modulename>.remoteaccess = false
  ```

  For example, for the delete libraries tool.

- Digital Experience introduced support for Content Security Policy (CSP) in CF192. Consider enabling this as a mitigation against cross-site scripting (XSS). While the onus remains on the application code that contributes markup, CSP offers defense-in-depth if any XSS vulnerabilities are present in application code.

Beyond the scope of this guide

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 as they relate 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
- Training users
  - Social engineering attacks
  - Installing software on clients (keystroke loggers, etc.)

Conclusion

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

Contributors

The author extends special thanks to the following people, who either contributed to or reviewed this guide for technical accuracy:

Alex Lang – Chief Architect, HCL Digital Experience
Thomas Hurek – Enterprise Architect, HCL Digital Experience
Ryan Wilson – Senior Solutions Architect, HCL Digital Experience
Howard Krovetz – Deputy General Manager, HCL Digital Experience
Travis Cornwell – Technical Support, HCL Digital Experience

Legal statements

This edition applies to versions 8.5, 9.0, and 9.5 of HCL Digital Experience, and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to HCL Technologies Ltd., you grant HCL Technologies Ltd. a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Disclaimers

This report is subject to the HCL Terms of Use (https://www.hcl.com/terms-of-use) and the following disclaimers:

The information contained in this report is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied, including but not limited to the implied warranties of merchantability, non-infringement, and fitness for a particular purpose. In addition, this information is based on HCL's current product plans and strategy, which are subject to change by HCL without notice. HCL shall not be responsible for any direct, indirect, incidental, consequential, special or other damages arising out of the use of, or otherwise related to, this report or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from HCL or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of HCL software.

References in this report to HCL products, programs, or services do not imply that they will be available in all countries in which HCL operates. Product release dates and/or capabilities referenced in this presentation may change at any time at HCL's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. The underlying database used to support these reports is refreshed on a weekly basis. Discrepancies found between reports generated using this web tool and other HCL documentation sources may or may not be attributed to different publish and refresh cycles for this tool and other sources. Nothing contained in this report is intended to, nor shall have the effect of, stating.

or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results. You assume sole responsibility for any results you obtain or decisions you make as a result of this report. Notwithstanding the HCL Terms of Use (https://www.hcl.com/terms-of-use), users of this site are permitted to copy and save the reports generated from this tool for such users own internal business purpose. No other use shall be permitted.