Securing Liferay Portal

Tomáš Polešovský
14. 01. 2014
tomas.polesovsky@liferay.com

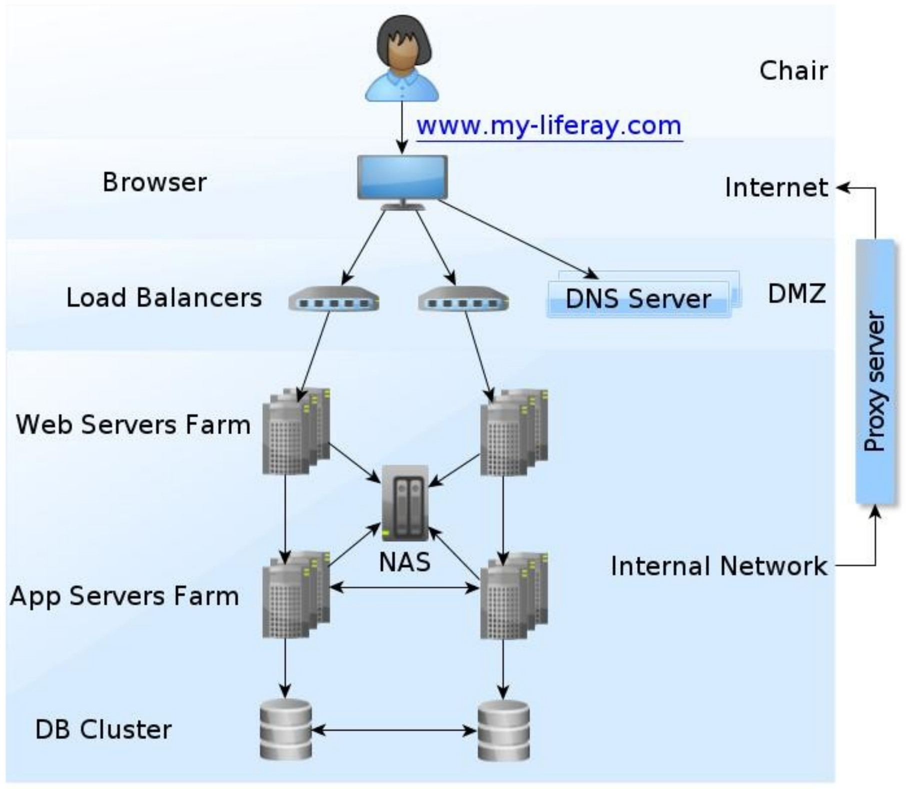
Securing Liferay Portal

- Agenda
  - Secure Installation
  - Securing and Hardening Configuration
  - Secure Liferay Development

Securing Liferay Portal

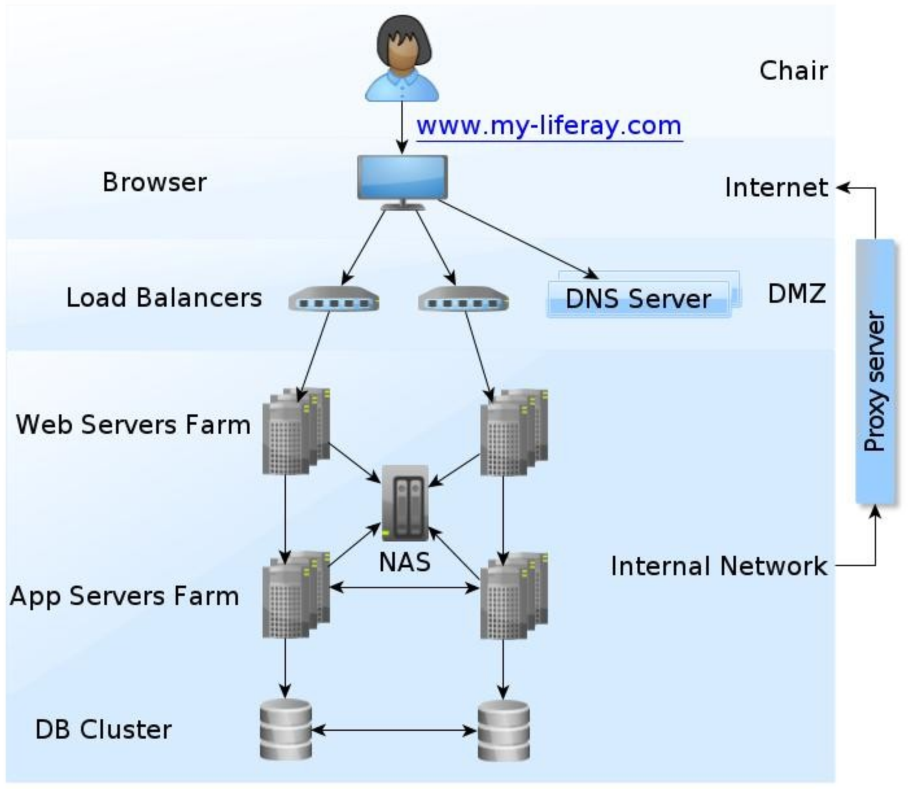
Installation

Liferay Deployment Security



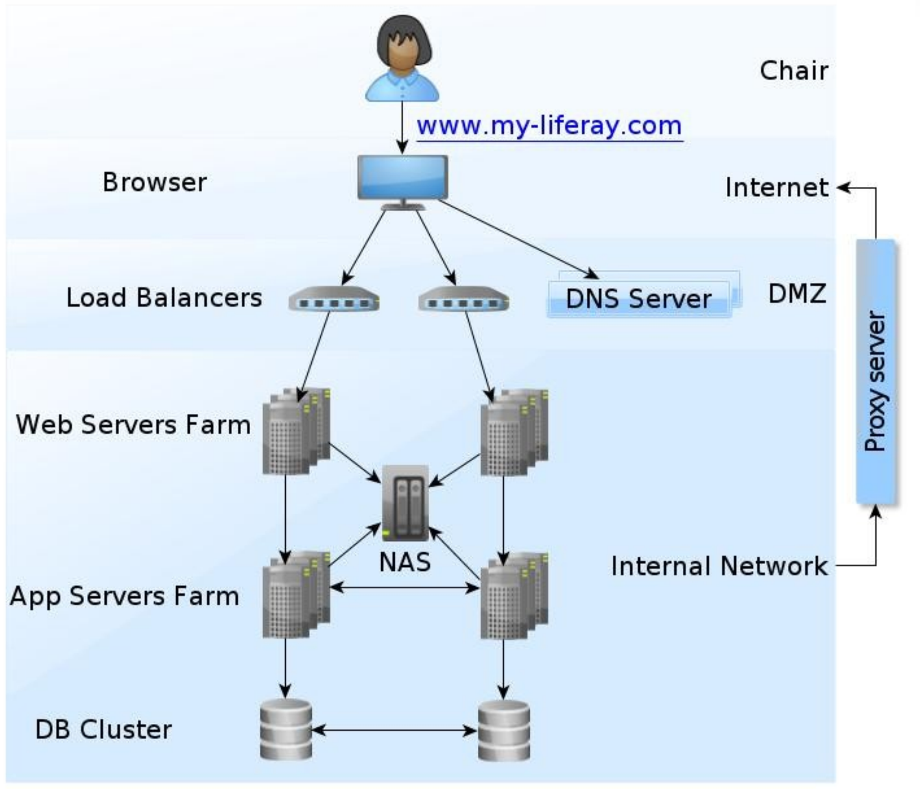
- Least Privilege
  - Each layer should see only the next one – firewalls on the boundaries
  - Going back through proxy - can filter out invalid outgoing requests
- Limit attack surface – open ports, visible IPs
- Beware of mixing environments
  - Test environment running in the same security boundary as production
  - Other applications running in the same DMZ / Internal network
  - Any compromised server is able to compromise other server in the boundary

Liferay Deployment Security



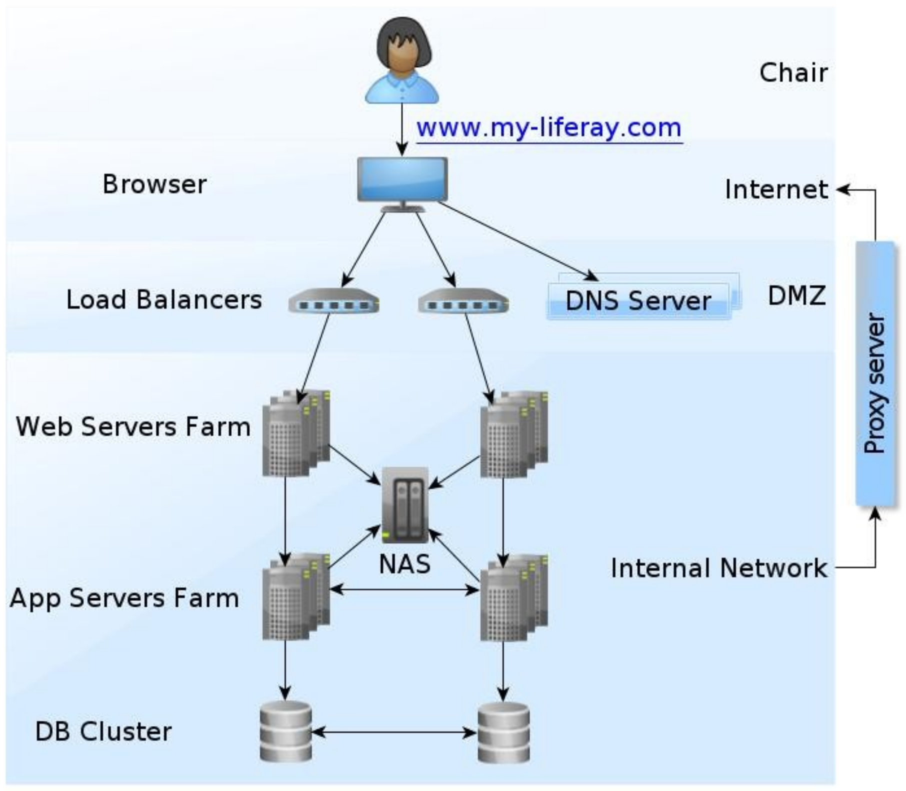
- DMZ layer
  - The only visible to the outside
  - DNS – port 53
    - DNSSEC
    - Open Resolver, Cache Poisoning, More issues
  - Load Balancers – port 80, 443
    - HTTPS endpoints
    - IDS/IPS - Security filters that helps firewalls
    - DDoS protection
  - Other open ports should have remote IP check (allow SSH only from whitelisted IPs)

Liferay Deployment Security



- Internal Network
  - Web Servers
    - Enable only required modules – no PHP, SSI, CGI, open proxy, status pages
    - Enable ModSecurity
    - Don't deploy any applications, only serve static content
    - If required, whitelist allowed IP addresses
    - Avoid wildcard / default virtual hosts
      - Prevents Host header attacks

Liferay Deployment Security



- Internal Network
  - App Servers, DB
    - Undeploy all unnecessary applications
    - Management console should be available only to specific IP addresses outside internal network
  - NAS
    - Should be accessible only from required servers
    - Should contain only data related to the portal

Liferay OS Level Security

- Olaf's Blog post
- Don't use passwords, only public key certificates
- Use separate temp location, preferably inside Liferay installation
- Use own user account for Liferay JVM process
  - The user should not be able to login into OS
  - Should NOT have write access except:
    - Temp locations
    - /deploy directory … only temporarily during the deploy
  - Read access only for Liferay installation directory
  - No execution - start portal as a service
  - Everything should be owned by a different user
- Run deployment + backup scripts under different account, don't leave the scripts together with Liferay installation to be readable by Liferay user account
- Uninstall any other unnecessary applications, mainly those with open ports

Securing Liferay Portal

End of Secure Installation

Questions?

Next:
Securing and Hardening Configuration

Liferay Configuration

- Many configuration options in portal.properties
- Don't disable global configuration options
  - We mostly provide whitelists
- Roles and permissions
  - Keep them organized, they can easily go complex
  - Don't rely on the built-in roles unless you know what permission they have, delete otherwise
  - No default permissions for User role, everything should be assigned by custom roles
  - Keep least privilege principle
- Public communities and organization pages
  - Limit public access only to minimum necessary portlets and pages, the more portlets are accessible the bigger is the attack surface
- Configure default user policy
  - Keep it usable
  - Account lock-out
    - Can have low values (3 tries each 1/5 minut) with good password rules … no words, big password space
    - Without password rules the values must be much more restrictive

Liferay Hardening

- Disable unwanted entry points
  - AtomServlet, AxisServlet, DisplayChartServlet, FacebookServlet, GoogleGadgetServlet, NetvibesServlet, PollerServlet, RemotingServlet, SharepointDocumentWorkspaceServlet, SharepointServlet, SharepointWebServicesServlet, SoftwareCatalogServlet, WebDAVServlet, WidgetServlet, XmlRpcServlet
  - AuditFilter, CacheFilter, CompoundSessionIdFilter, DynamicCSSFilter, ETagFilter, FragmentFilter, GZipFilter, HeaderFilter, IgnoreFilter, MonitoringFilter, CASFilter, NtlmFilter, NtlmPostFilter, OpenSSOFilter, ThemePreviewFilter, ThreadDumpFilter, UrlRewriteFilter, SharepointFilter,

Liferay Hardening

- Disable URLs accessible from internet:
  - Access to administration interface (Control Panel)
  - Access to JSON WS API listing (/api/jsonws) and unused services
  - Use different domain/origin for local static files (CSS, JS, Flash, …, documents?)
    - Prevents XSS by exploiting local files vulnerabilities
    - Disallow access to these files from the portal domain, be careful with “//”, “../” and path parameters /path;param=value/path;param=value/ when creating the URL whitelist
- Disable unused authentication / SSO
- Disable new user accounts at all, if possible
  - Don't forget these are created also by SSOs
- Disable user private pages where users can use own portlets
- Use remote staging
  - You can disable or delete omni-admin account in the live DB
  - You don't need any high-privileged accounts in production
- Deploy AntiSamy plugin to prevent XSS by content creator accounts

Liferay Monitoring

- Use IDS to monitor attack attempts
- Use Liferay Auditing framework + EE plugin
- Hopefully, in the near future, use Liferay Cloud Services for monitoring
- Correctly rotate logs so that exception stack traces don't overwrite important log events
- Create scripts to review logs
  - Filter out common exceptions
  - Filter out false positives
- Be prepared to act responsibly when you spot something strange
  - Can be a bug, can be an attack
  - Be prepared, you need to know what to do, who to contact
  - Disaster Recovery plan can help

Securing Liferay

End of Securing and Hardening Configuration

Questions?

Next:
Securing Development

Secure Liferay Development

- Liferay provides many classes for securing most of the common vulnerabilities, use them
- `HtmlUtil` to prevent XSS in different contexts
- `HtmlUtil#escapeXPath*` – prevent XPath injection
- `AuthTokenUtil#checkCSRFToken`
- `FileUtil#createTempFile*` - prevent file system related issues
- `PortalUtil#escapeRedirect` - prevent open redirects
- `StringUtil#random*` - insecure but random enough strings
- `PwdGenerator#getPassword`, `SecureRandomUtil` – cryptographically strong pseudorandom output, optimized for performance
- `PasswordEncryptorUtil` – verification and creation of strong password hashes, currently configured to use PBKDF2PasswordEncryptor
- `EncryptorUtil` – en/decryption, currently configured to use AES-ECB (not suitable for plain-texts > 16 bytes … block size)
- `DigesterUtil` – SHA-1 hashes, good for file checksums

Common Liferay Security Vulnerabilities & Countermeasures

- XSS
  - Escape all user originated output in portlets
  - HtmlUtil, <aui-input>, <search-container-row escapeModel=true>
- Authorization
  - Use *LocalService only for entities you won't display / manipulate with
  - In remote services implementation don't forget to check all related permissions
- Authentication
  - Don't use Servlets unless you secure them
- CSRF
  - Add CSRF token check when serveResource modifies state/entities
  - Render phase is only for display, updates should be done in action phase

Common Liferay Security Vulnerabilities & Countermeasures

- File Uploads
  - Check file name – Validator.isFileName, Validator.isFileExtension
  - Temporary files create using FileUtil.createTempFile or FileUtil.createTempFileName
  - Use FileUtil.createTempFolder when you need directories
  - Limit file size, Limit number of stored files, unless you implement Megaupload
- File Downloads
  - Use HTTP Header `Content-Disposition: attachment`
  - Only in specific cases (images) use `Content-Disposition: inline`
    - Check magic bytes to be sure you don't offer flash files

Common Liferay Security Vulnerabilities & Countermeasures

- Open Redirect
  - Sanitize redirects with PortalUtil.escapeRedirect
- Arbitrary/Remote Code Execution
  - Don't use Runtime.exec() and ScriptingUtil
  - Use only when (1) input is originated from administrator or (2) you are really really, I mean really sure it is safe
- Storing Credentials
  - Use PBKDF2 or BCrypt – Use PasswordEncryptorUtil
  - Don't echo back credentials into HTML forms, redirect on validation errors
- Log wisely
  - configure each new log to use INFO
  - log what you can with DEBUG
  - Consider auditing framework - AuditRouterUtil.route(auditMessage);

Securing Liferay Portal

That's all for now.

Thank you.

Questions?