Chapter 2

Steps for GSO review

from Global Security Organization Review for ADP Marketplace Apps Guide

Published on
Jul 26, 2021 11:42AM

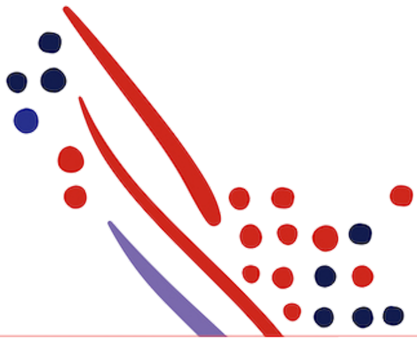
Last modified
Aug 03, 2021 12:41PM

Always Designing for People™

ADP Copyright Information

Published on
Jul 26, 2021 11:42AM

Last modified
Aug 03, 2021 12:41PM



Chapter Contents

Chapter 2

Steps for GSO review

- Steps Followed for GSO Review
- Types of testing
- Exit criteria
- Incremental integration review requirements
- Reassessment
- Annexure A
- Annexure B
- Cross-site scripting (XSS)
- File upload vulnerability
- Recommendations:
- Password complexity
- Clickjacking
- Obsolete protocol implementation
- Cross-site request forgery (CSRF)
- SQL injection
- Link injection
- Session expiration
- Annexure C

Chapter 2

Steps for GSO review

Steps Followed for GSO Review

Here is the process for partners undergoing review with GSO.

1. Partner provides a demo of the integration to the ADP Marketplace technical advisor and GSO.

2. Upon a successful demo, the ADP Marketplace technical advisor schedules a review of the integration with the ADP GSO.

3. GSO schedules an initial kickoff call with the partner to understand the scope of integration and data elements accessed.

4. Partner submits the required details through a shared ADP folder. These are #3, #4, and #5 in the pre-requisites section. GSO will provide access to this folder after the initial kickoff call.

5. Partner submits security documents listed below in Annexure A and penetration test requirements listed below in Annexure C.

6. GSO conducts review of security documentation, penetration test of application and source code analysis of connector piece and provides the feedback.

7. Partner resolves issues if any were identified by GSO and submits respective evidences as below:
   - Policy document if issue is related to security policies
   - Updated staging/test/development environment if the issue is related to application penetration testing
   - Updated source code of connector if issues is related to source code analysis
   - See the list of common issues GSO comes across. If these are known issues, the partner can work on these while they develop the integrated application as well.

8. After all critical and high issues identified are remediated successfully, GSO signs off on the application

Types of testing

Two major types of testing will be performed on your application for the integration with ADP APIs: Dynamic Application Security Testing (DAST) and Static Application Security Testing (SAST). Your application will be assessed on the below categories:

Category	***Assessments***
Authentication	Review of password strength, lockout, change function, logout, bypass, brute force, denial of service, and administrative 2-factor controls
Authorization	Review of auto-complete, forceful browsing, bypass, harvesting via error messages or account formats, and privilege escalation
Client-side issues	Review of signed applets and/or active controls where applicable
Cryptography	Review of valid code signatures and SSL certificates, key management, plaintext transport, algorithms, keys, and SSL ciphers
General application fitness	Review of conformance to ADP's policies, e-commerce security and compliance requirements, in addition to, general coding best practices
Information leakage	Review of indexing, robots.txt, default content, executables, URL encoding, caching, cross frame spoofing, error, help, and comment

	messages for personal, sensitive or system-related information
Input validation	Review of post and get commands, hidden elements, cross-site scripting, SQL injection, and malicious file handling
Session configuration	Review of HTTP commands including PUT, TRACE, TRACK, and DELETE
Session management	Review of cookies, predictability, manipulation, cloning, hijacking, page expiration, session timeout, IP hopping, session credentials, ID, and strength
System configuration	Review of web, application server, database, testing content, extensions, unhandled exceptions, and security patches
Penetration testing (also called pen testing)	Review of web application to find vulnerabilities that an attacker could exploit
Other	Review of additional security controls, logic flows, server executables (.CGI, .ASP, .PHP, Cold Fusion, PERL, etc.), bounds checking, stack-based buffer overflow. Additionally, review of hard-coded or unprotected credential disclosure in URLs, static or dynamic pages, accessible configuration files and/or code.

Info

The above is a minimal list only. Depending on your application and API, the list will be expanded to do the appropriate assessment.

Exit criteria

Below you'll find the remediation requirements for your integration depending upon the severity of the issue GSO discovers during testing.

Severity	**Remediation requirement**
Critical/high	You MUST fully remediate all issues before the app is moved to production.
Medium	You MUST submit a remediation plan with a timeline provided to ADP, followed by evidence of remediation completion.
False positives	If you determine that a finding or condition does not exist and may be a false positive, you will dispute the issue in writing along with any evidence or proof supporting their position. The dispute and information should be submitted to the ADP risk assessor. The risk assessor will review internally, and should they agree, the false positive will be acknowledged/confirmed via email as being addressed. In some cases, the assessor may ask for additional evidence or require a meeting to discuss further.

Incremental integration review requirements

No.	**Use case**	**Action to be performed**
1	Usage of more APIs (Worker) in addition to Phase I APIs (Worker) for same SOR (ADP Workforce Now®/ADP Vantage HCM®/RUN Powered by ADP®) used and no changes to UI of core application	Perform the source code analysis of new connector piece during the reassessment cycle.
2	Usage of same APIs (Worker) of phase I integration but for	Perform the source code analysis of new connector piece

	different SOR (ADP Workforce Now/ADP Vantage HCM/RUN Powered by ADP) and no changes to UI of core application	during the reassessment cycle.
3	Usage of more APIs (Worker) in addition to Phase I APIs (Worker) for different SORs (ADP Workforce Now/ADP Vantage HCM/RUN Powered by ADP) and no changes to UI of core application	Perform the source code analysis of new connector piece during the reassessment cycle.
4	Usage of different module APIs (Deductions) in addition to Phase I APIs (Worker) for different SORs (ADP Workforce Now/ADP Vantage HCM/RUN Powered by ADP) and addition of modules to core application.	No changes with regards to dynamic test as there are UI changes. Perform the source code analysis of new connector piece during the reassessment cycle.

Reassessment

Reassessment is a part of ADP's Global Third-Party Risk Management program and is required by ADP's 112 – Global Vendor Assurance Policy, which mandates that vendors/partners be assessed for each service provided prior to engaging and are reassessed on a cyclical basis. The GSO team is responsible for assessing the security controls of partners who have been granted access to ADP systems, information assets and/or customer information. The intent of the security assessment is to gain confidence in the partner's general computing controls for the services provided to ADP. The timelines are as follows:

Vendor/partner tier	Time frame
Tier 1	12 months
Tier 2	18 months
Tier 3	24 months

Annexure A

If your integration reads/writes sensitive information (e.g., SSN, date of birth, salary info) from/to ADP, then you must fill in a Third-Party Assurance Questionnaire (which is based on ISO 27001 security standard) and share supporting security artifacts. Alternatively, you can share your organization's security compliance reports, like ISO 27001 Certificate and Statement of Applicability or Latest SOC 2 Type II report.

If your integration reads/writes basic information (e.g., first name, last name, work email address, work phone) from/to ADP, then following security documentation must be shared:

Your high-level Information security policy should include but is not limited to:

- Objective of Information security
- Organization of information security
- Human resource security
- Asset management
- Access control
- Cryptography
- Physical and environmental security
- Operations security
- Communications security
- System acquisition, development and maintenance
- Supplier relationships
- Security Incident management
- Business continuity and disaster recovery management

Your human resource security policy should include but is not limited to:

- Prior to employment (roles and responsibilities, background screening, terms and conditions of employment)
- During employment (security awareness training and disciplinary actions)
- Termination or change of employment (responsibilities for termination or change of employment, return of assets, removal of access rights)

Your secure software development life cycle procedure should include but is not limited to:

- Software development life cycle
- Security in SDLC
- Security awareness training to software developers
- OWASP methodology
- Software security testing (penetration testing and source code analysis)
- Separation of development, testing and production environments

Your incident management procedure should include but is not limited to:

- Reporting and classifying an incident
- Communication of incident to stakeholders (external and

Annexure B

Below are some of the most common issues identified by GSO. If these are known issues within your integration, you should proactively resolve them in your core application when you develop the integrated application as well.

Cross-site scripting (XSS)

Finding: Cross-site scripting (XSS) refers to a client-side code injection attack wherein an attacker can inject malicious scripts into an application through parameters, hidden fields, cookies or other client-side inputs into a legitimate website or web application. XSS is among the most rampant of web application vulnerabilities and occurs when a web application makes use of un-validated or un-encoded user input within the output it generates.

By leveraging XSS, an attacker does not target a victim directly. Instead, an attacker would exploit vulnerability within a website or web application that the victim would visit, essentially using the vulnerable website as a vehicle to deliver a malicious script to the victim's browser.

Recommendations: Sanitize all end-user data, especially those that will be incorporated into any HTML. Solutions to prevent XSS include:

- **Enforce strict whitelists for all user input.** This is the most effective technique. (e.g., if a variable should only contain 5 alphanumeric characters, return an error if it doesn't meet that requirement).
- **HTML-encode all potentially dangerous characters.** At a minimum, ensure the following characters are properly encoded in input and output: <> & ' ' ( ) # % ; -
- **Filter bad characters from user input and output.** However, be advised that blacklisting is a game of catchup every time a new XSS attack string is discovered.
- **Consider not displaying user-supplied input** if it is not critical to application functionality.

File upload vulnerability

Finding: File upload capability allows a web user to send a file from their file to the web server. If the web application that receives the file does not carefully examine it for malicious content, an attacker may be able to use file uploads to execute arbitrary commands on the server. Arbitrary command execution may allow an attacker access to the server with the permissions of the web server or script engine user. Additionally, it may be possible to upload viruses or other malware.

Recommendations:

- Limit the types of files that can be uploaded not only by extension but also by content.
- Check file extensions — specify a white list of allowed values and verify against it. You should avoid allowing the upload of scripts, executable binaries and source code such as exe, com, pl, sh, bin, java, class, php, jsp, xlsm, xlsb, docb, etc.
- Check file content (e.g., MIME type) in accordance of the allowed file types above.
- Scan for viruses using scanners (Clam AV recommended). If you accept compressed archives be sure the expand them and scan the files in them. Some AV vendors have built-in support for this but require configuration to enable.
- Only allow newer versions of MS Office files, which do not contain macros (e.g., xlsx, docx). If older versions must be uploaded, check for the existence of macros within the Office files.
- See reference link for code samples:
    1. http://www.codeproject.com/Articles/100629/Removing-Macro-from-Microsoft-Word-Word
    2. http://www.rgagnon.com/javadetails/java-detect-if-xls-excel-file-contains-a-macro.html
- Do not allow the end user to control the name of the file or location where it is stored on the server.
- Do not use the existing name of the user's file.
- Do not derive the filename from the username, session ID or other variables known to the user.
- Do not place the file in a directory accessible by web users. It is preferable for this location to be outside of the document root.
- Ensure that strict permissions are set on both the uploaded file and the directory in which it is located.
- Do not grant execute permissions.
- Allow read permissions only for the processes that require access (such as the script engine).

Password complexity

Findings:

- **Case 1:** Application passwords do not utilize a character from each of the required character sets and/or utilize a character from an unapproved character set. Passwords are required to have at least three of the four groups:
    - Uppercase letter
    - Lowercase letter
    - Numeric character
    - Special characters including - [! " # $ % & '( ) * + , - ./ : ; < = > ?@ [\ ] ^ _ ` { | } ~]
- **Case 2:** Passwords may not contain more than three (3) repeating or sequential characters as well as any personal identifiers or usernames.
- **Case 3:** In order to meet the required credential entropy for any particular assurance level, passwords MUST have password with minimum length of 8 characters.
- **Case 4:** Password management policy is not implemented properly in the application. It's not checking against the minimum password history requirements. Application is even accepting most recently used password as a new password.
- **Case 5:** Application does not trigger temporary account lockout for five minutes, even after three consecutive invalid password attempts.

Recommendations: Passwords are the most commonly used credential to validate the authenticity of a user and, in order to provide sufficient authentication assurance, these must be governed by policies dictating their complexity as mentioned below:

Password policy area	**Assurance level**
Minimum credential entropy	30 bits
Minimum password length	8 characters
Minimum password alphabet	94 characters
Minimum password composition rules	Dictionary exclusion check AND at least three of the following: • Uppercase letter

	• Lowercase letter • Number • Special character
Maximum password lifetime	90 days
Minimum password history	Cannot match last 6 passwords
Maximum consecutive invalid password attempts	3
Minimum login lockout	5 minutes

In order to meet the required credential entropy for an assurance level, passwords must contain at least 8 characters and should not exceed 256 characters. For all assurance levels, a password alphabet of 94 characters must be used. The 94-character password alphabet is case sensitive, based on the basic ASCII printable characters, and includes:

- Uppercase A to Z
- Lowercase a to z
- Numbers 0 to 9
- Special characters including - [! “ # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~]

In addition to the banned password list (dictionary exclusions), passwords MUST NOT contain:

- No more than 3 sequential characters (ascending or descending)
- No more than 3 identical chars in a row

Personal information that MUST NOT be part of the password includes:

- The user’s user ID
- The user’s government-issued identifiers (Social Security number, Social Insurance Number, etc.)
- The user’s date of birth
- The user’s family name (i.e., surname, last name) or personal name (i.e., first name)
- The user’s phone number or zip code

The user must not be allowed to enter an invalid password for more than 3 times consecutively and when the threshold for consecutive invalid password attempts has been reached, the account should temporarily be locked for the period of 5 minutes.

Clickjacking

Finding: OWASP defines clickjacking, also known as a "UI redress attack," as "when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both." The attacker can use clickjacking attacks to steal user credentials, perform unwanted actions on a logged-in user's account and more.

Recommendations: Set "X-Frame-Options" HTTP response header to indicate whether or not a browser can render a page in a frame. Below are the three possible values for X-Frame-Options:

- **DENY:** The page cannot be displayed in a frame, regardless of the site attempting to do so.
- **SAMEORIGIN:** The page can only be displayed in a frame on the same origin as the page itself.
- **ALLOW-FROM url:** The page can only be displayed in a frame on the specified origin.

Obsolete protocol implementation

Finding: This refers to a situation in which the application reveals usage of obsolete version of SSL protocols like TLSv1.0 and TLSv1.1 which are obsolete from June 2018.

Recommendations: Ensure that the application is configured to use the secure version of SSL protocol (TLS v1.2 and above).

Cross-site request forgery (CSRF)

Finding: Cross-site request forgery is a vulnerability in which malicious input from a user may be viewed or rendered by another user in the application. This input may be in the form of an image or a script that accesses a site to which the victim user is known to have authenticated. The reason this vulnerability is called a "request forgery" is that the malicious user will include in the image or script a request to a site for which the user is authenticated, therefore executing the forged request with the authority of the victim user. For this to work, the target site must fully authorize requests using a cookie.

Recommendations: The most robust way to defend against CSRF attacks is to include a CSRF token along with each state-changing request, which should be unpredictable with high entropy, tied to the user's session, and needs to be strictly validated in every case before the relevant action is executed.

SQL injection

Finding: The application suffers from SQL injection due to input validation problems and unsafe development practices. The instances found enable an unauthenticated user to dump the contents of the database and will likely lead to full system compromise. This is due to several factors: insufficient input validation of user-supplied data, use of string concatenation to build SQL statements, and improper exception handling.

Recommendations: Validate all user-supplied input prior to processing. Only accept known good input rather than attempting to identify bad input. For example, for a username, only accept the values [A-Z][a-z][0-9]. When parameters are passed via GET or POST, validation should happen after URL decoding. Do not build up SQL statements via string concatenation in the application server. Instead, use stored procedures or prepared SQL statements. All exceptions and errors should be caught and logged for later review by an administrator or developer.

Link injection

Finding: This refers to a scenario that allows a user to inject the link of any website in the application. This attack can be used to inject malicious content into the page, redirect user to dangerous websites etc.

Recommendations: Ensure that the application does not provide any such feature to the user. If user input can't be avoided, sanitize the input by creating a list of trusted URLs (lists of hosts or a regex). Force all redirects to first go through a page notifying users that they are going off your site, and have them click a link to confirm.

Session expiration

Finding: The application does not properly terminate user sessions upon users performing explicit logout action.

Recommendation: Ensure that the session is expired on the server side once the "Logout" button is clicked.

Annexure C

URL: Please share the staging/test/development environment URL for your core application.

Credentials: Please share two sets of test credentials for Company 1 (ADP) for each user role (e.g., manager, HR, employee) and map the test user accounts with any of the following email IDs (which we use for testing) in order to reset a password or unlock user accounts (if any account gets locked during testing). As well, please provide the related security questions and answers.

Alice.adpappsec@adp.com

Bob.adpappsec@adp.com

Cathy.adpappsec@adp.com

David.adpappsec@adp.com

Emily.adpappsec@adp.com

Fred.adpappsec@adp.com

Gina.adpappsec@adp.com

Harry.adpappsec@adp.com

You'll also need to share one set of credentials for Company 2 (non-ADP) to check if we can access the Company 1 (ADP) data from Company 2 (non-ADP) accounts and vice versa.

User roles: Share all different user roles (e.g., employee, manager) available for this application, along with their scope and the variations between them.

Workflow: Share the demo or workflow document of your core application, which should cover functionalities, tabs, user roles, etc.

Performing automated scan: GSO will need your permission to perform automated scans on your application. During automation scans, a large number of HTTP requests with malicious content will be sent to the server so that the scanner can identify vulnerabilities. Our testing team works from 12:30 am EST to 9:30 am EST; let us know if we can perform the automated scans during this time frame.

Source code of connector piece: You must share the source code of the connector piece developed for integration with ADP APIs. The source code must be uploaded to the shared ADP folder.