



Adobe Systems Incorporated
Adobe Acrobat Connect Pro
June 15, 2009

1. ENGAGEMENT OVERVIEW

1.1 Overview

At the request of Adobe Systems Incorporated (Adobe), Neohapsis performed a blackbox security assessment of the Adobe Acrobat Connect Pro (AACP) on-premise deployment solution during the period of Q1 2009. The goal of the assessment was to evaluate the overall security posture of the AACP (version 7.0 sp3) application in a on-premise deployed environment. The assessment consisted of both manual and automated attempts to assess the design and implementation of the security mechanisms in use by AACP. In addition, Neohapsis also evaluated the physical and environmental controls as well as provided security policies and procedures which Adobe has employed to protect the production environment. A separate report addresses the assessment of the AACP in a hosted environment.

1.2 About Adobe Acrobat Connect Pro

AACP is web conferencing software that enables instant communication and collaboration through easy-to-use, easy-to-access online personal meeting rooms. AACP enables anyone using a web browser and the Adobe Flash® Player runtime to join a web meeting without having to download cumbersome software. Because the Adobe Flash Player is installed on more than 98 percent of Internet-connected computers worldwide, the experience of joining an online meeting is hassle-free.”.

1.3 About Neohapsis

Founded in 1997, Neohapsis helps organizations assess their critical business processes and build a consistent and sustainable risk management discipline to generate lasting value. Our heritage of providing superior IT risk management services and security consulting combined with our award winning Governance, Risk Management, and Compliance (GRC) technology enables organizations to move beyond discrete mitigation and compliance solutions to a comprehensive framework where risk can be transformed into information and opportunity.

1.4 Scope

In Q1 2009, Neohapsis assessed the AACP on-premise solution. Neohapsis consultants assessed both the overall design as well as the implementation of AACP. Assessment of the design focused on validating the existence of a sufficient feature to enforce a desired security policy. Assessment of the security implementation focused on the identification of vulnerabilities that would allow a malicious user to subvert a desired security policy. The vulnerability assessment primarily focused on common application vulnerabilities, including:

- Cross Site Scripting (XSS)
- Broken Authentication and Session Management
- Information Leakage and Improper Error Handling
- Injection Flaws

-
- Malicious File Execution
 - Insecure Direct Object Reference
 - Failure to Restrict URL Access
 - Insecure Cryptographic Storage
 - Insecure Communications

1.5 On-premise Solution Assessment Methodology

Neohapsis consultants used both manual and automated attack techniques in an attempt to bypass the intended functionality and secure design of the AACP on-premise application. This included an analysis of the application using the following components:

- spidering—attempts to identify application functionality by automated traversal of site hierarchy and permuting common variations on popular naming conventions
- manual fault injection—manual submission of malicious data to identify security vulnerabilities in request path
- automated fault injection (fuzzing)—automated submission of a range of malicious data to identify security vulnerabilities in request path
- known vulnerability testing—identification of vulnerabilities in the hosting platform (web server, etc.) using primarily automated analysis techniques
- Data correlation
 - Research vulnerabilities
 - Eliminate false positives
 - Investigate the extent of the findings

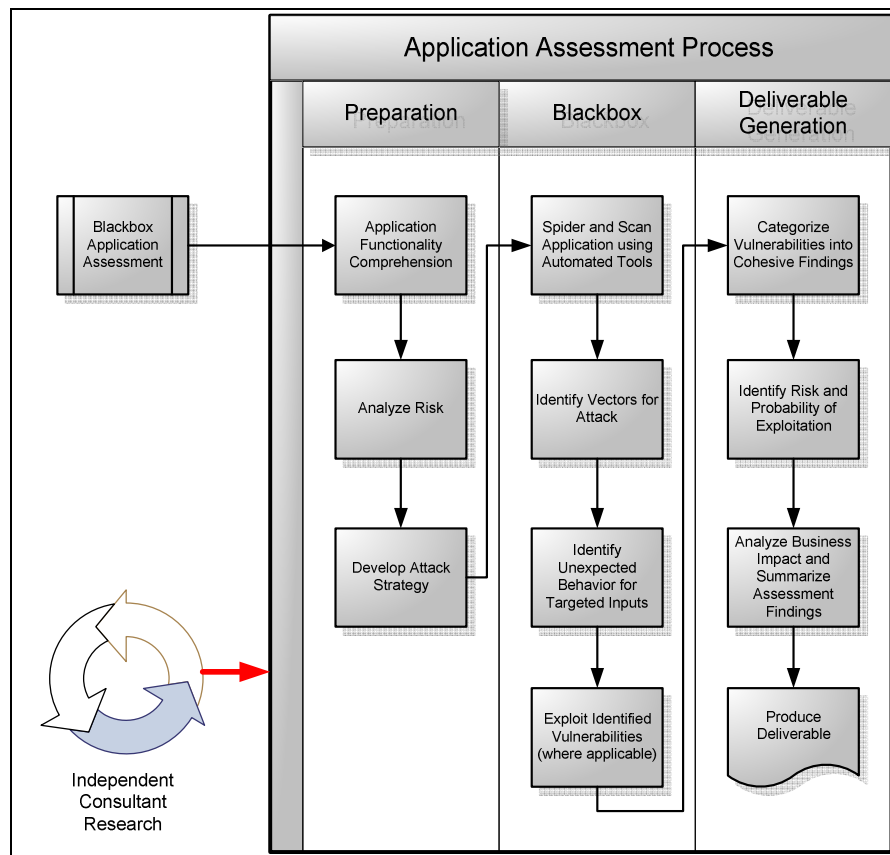


Figure 1 Blackbox Application Assessment Approach

2. SOFTWARE SECURITY ENGAGEMENT FINDINGS

AACP employs a variety of measures to secure its customer's communications and data. These security measures address the following categories:

- **User Authentication:** Users must be required to authenticate prior to accessing private content and meetings, and the authentication method itself must occur securely.
- **Password Management:** Users should be required to choose strong passwords and change them regularly.
- **Data Management:** Strong encryption must be used to secure communications and sensitive data stored in the database. Queries to the database must prevent malicious injection.
- **User Privileges:** Access to resources must be configurable and properly allow or restrict access to content and meetings.
- **Auditing and Logging:** For auditing purposes, potentially malicious use must be logged along with date, time and source information.

Each of the above security goals is implemented with a number of security features. Security features include those features whose explicit function is to enforce a security goal. As an example, the login component is a security feature. Neohapsis was able to validate that AACP provides a sufficient set of security features to implement effective control over the above stated goals.

However, security features are only the basis for a secure implementation. Any component within AACP that may affect the security posture of the application is security relevant. For example, AACP allows users to share content by uploading files. File uploading is not a security feature, but is security relevant, as failure to securely handle file uploads may lead to arbitrary code execution on the server. Therefore, beyond validating AACP provides a sufficient set of security features, Neohapsis' primary focus was validating that the security relevant features are implemented in a manner that does not allow a malicious user to subvert the desired security policy of Adobe and their customers.

Using a combination of automated and manual analysis, Neohapsis assessed AACP for common web application vulnerabilities. Section two details Neohapsis' findings for each vulnerability class under evaluation..

2.1 Cross Site Scripting

“XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.”

Using a combination of manual and automated testing, Neohapsis found AACP resilient against XSS attacks. The application validates untrusted user input using a combination of whitelist

and/or blacklist approaches. A whitelist ensures that untrusted user input conforms to an acceptable character set and format. A blacklist leverages a list of potentially malicious input to validate user input. As an example, AACP validates that email addresses conform to a specific format and character set and will not accept any input that does not match. Where a whitelist and/or blacklist is not used, the application encodes untrusted user input when displayed to the user. This prevents the browser from interpreting untrusted user input as valid HTML markup, thus mitigating the possibility of malicious JavaScript injection and execution.

2.2 Injection Flaws

“Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.”

Using a combination of manual and automated testing, Neohapsis found AACP to be resilient against injection-based attacks. Injection attacks can occur in a number of scenarios including SQL queries, LDAP queries, and XPATH queries. AACP makes extensive use of SQL queries throughout the application and Neohapsis did not identify any injection related vulnerabilities during the course of the assessment.

2.3 Malicious File Execution

“Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.”

Using extensive manual testing, Neohapsis found AACP to be resilient against malicious file execution based attacks. AACP mitigates this potential vulnerability using a defense in depth strategy that leverages both a secure design as well as a secure implementation. By design, AACP reduces the threat surface of the application by restricting the number of locations where users can upload files. In those locations where users may upload files, AACP rigorously validates their content type. In addition, AACP restricts uploaded files to a specific directory hierarchy and prevents directory traversal attacks that attempt to break out of this directory.

2.4 Insecure Direct Object Reference

“A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.”

Using a combination of manual and automated testing, Neohapsis found AACP to be resilient against direct object reference attacks. Applications that are vulnerable to direct object reference attacks often fail to leverage secure abstractions that prevent malicious users from interacting directly with low-level system operations. This can manifest itself in a number of ways, such as passing directory names, file names, or SQL queries in user parameters. Neohapsis did not identify any instances where AACP directly references a low-level construct, such as those just listed, in a user parameter. As an example, though AACP allows users to upload content, all file operations occur through an abstraction that prevents direct manipulation of the underlying file system.

2.5 Information Leakage and Improper Error Handling

“Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.”

Using a combination of manual and automated testing, Neohapsis found AACP generally resilient against information leakage attacks. Information leakage attacks can occur in a many number of ways. Some of the more common problems include insecure exception and error handling.

Insecure exception handling occurs when an application fails due to some unforeseen situation and the application returns an error message that, while potentially useful to an application developer, may reveal details about the internal implementation of the application that can be leveraged by a malicious user. Neohapsis found AACP mitigates this attack by returning useful error messages that reveal minimal information to a malicious user. Neohapsis identified a finite number of instances where an exception was thrown and the details were returned to the user. However, this was atypical, and the identified instances did not reveal any sensitive user or critical system information.

In contrast to insecure exception handling, insecure error handling occurs when an anticipated error occurs and the application explicitly discloses more information than necessary to the user. As an example, many applications differentiate between an invalid username and an invalid password during login, providing different error messages for each. However, this may allow an attacker to brute-force valid usernames, which itself may be valuable. AACP securely handles this scenario, and returns the error, “Invalid user or password. Please try again.” Similarly, throughout AACP, Neohapsis did not identify the disclosure of any potentially sensitive information due to insecure error handling.

2.6 Broken Authentication and Session Management

“Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.”

Using a combination of manual and automated testing, Neohapsis found AACP to be resilient against authentication and session management attacks.

In addition to following the secure design and implementation practices detailed in the other findings, an authentication system must also implement a number of secure operational practices. AACP uniquely identifies a user based on a combination of username and password. The user transmits their username and password over a secured SSL/TLS connection, mitigating the possibility of disclosing their credentials during transmission. In addition, AACP allows for the customization and enforcement of various aspects of password management including setting password lifetime duration, enforcing a minimum password length, and enforcing a minimum password complexity.

Once authenticated, secure session management protects an authenticated user from unauthorized users attempting to perform actions on their behalf. Secure session management must provide security for the entire lifetime of the session: from the initial authentication, throughout the duration of the user's session, until the user logs out of the application. AACP uses a combination of best practices to implement each of these phases. First, AACP generates a cryptographically strong session cookie for each user upon visiting the site. This cookie contains approximately seventy-one bits of entropy, minimizing the chance that a malicious user will be able to predict a user's session cookie. Throughout the user's session, AACP protects the

sensitive session cookie by encrypting all communications over SSL/TLS. Finally, upon logout, AACP invalidates the session cookie, both on the client and server, preventing a malicious user from replaying prior requests.

2.7 Insecure Cryptographic Storage

“Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.”

Using a combination of manual and automated testing, Neohapsis was unable to identify any evidence of plaintext storage of sensitive user and system information. Within AACP, the most sensitive information stored by the application is related to user passwords. AACP does not provide any facility to retrieve a forgotten password. Either a user must submit a request to receive an email with a password reset link, or they must contact the system administrator to reset the password on their behalf. Neohapsis was able to validate the use of password hashes, rather than plaintext or reversibly encrypted passwords, within the database. Storing hashed passwords makes the recovery of plaintext passwords more difficult for a malicious user. Using hashed passwords is a good security practice and is an essential component of using secure cryptographic storage for password management.

2.8 Insecure Communications

“Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.”

Using a combination of manual and automated testing, Neohapsis found AACP to be resilient against insecure communication attacks. Insecure communication attacks typically involve a malicious user “sniffing” sensitive user information while the data is in transit from the user’s browser to the communicating server. AACP uses SSL/TLS to prevent the disclosure of sensitive information to other users. Moreover, AACP can be configured to require the use of SSL/TLS, and will not accept requests over an unprotected connection. Finally, AACP follows best practices by setting the “Secure” flag on all sensitive session cookies, mitigating the possibility of accidental disclosure over an insecure connection.

2.9 Failure to Restrict URL Access

“Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly.”

Using a combination of manual and automated testing, Neohapsis found AACP to be resilient against unauthorized URL access attacks. The AACP groups users into various roles such as Administrators, Authors, and Meeting Hosts. Based on this role, AACP either grants or denies access to various features within the application. Neohapsis conducted numerous tests to attempt submitting requests under a user role that should not have access to the corresponding functionality; all such tests failed. AACP uses a site-wide authorization scheme that validates each request against a role based access control policy. All unsuccessful request submissions either respond with an “unauthorized” message or simply forward the user to the login page.

3. CONCLUSION

Neohapsis found the design and implementation of AACP resilient to attack under the evaluation criteria detailed in section two. It is the opinion of Neohapsis that Adobe has prioritized security during the software development lifecycle, and as a result, has implemented a product that continually strives to address information security best practices.