



## OWASP TOP TEN

The OWASP Top Ten provides a powerful awareness document for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are. Project members include a variety of security experts from around the world who have shared their expertise to produce this list. The U.S. Federal Trade Commission strongly recommends that all companies use the OWASP Top Ten and ensure that their partners do the same. In addition, the U.S. Defense Information Systems Agency has listed the OWASP Top Ten as key best practices that should be used as part of the DOD Information Technology Security Certification and Accreditation (C&A) Process (DITSCAP). For more details on this project, please click [here](#).

Boomi's security team has reviewed the Top Ten very carefully, and below are our responses as to how Boomi prevents each of these security flaws.

Vulnerability	Description	Remediation
A1 - Cross Site Scripting (XSS)	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.	The underlying API is REST based, which relies on proper XML encoding of inputs. Data is XML encoded when delivered to the client.
A2 - 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.	No dynamic queries are produced in the backend. We use Hibernate as our data access layer, and only execute statements through appropriate setting of replacement parameters.
A3 - 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.	Files that are uploaded are first validated for conformity/correctness. The majority is stored for download by the clients, and the files are never executed remotely.
A4 - 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.	All API calls are authenticated not only for username/password but for context as well. We have an account hierarchy structure which allows parent accounts to access child account information. Therefore, each request is first authenticated to make sure the caller is valid, but then also authorized to make sure that the caller should be able to access the object they are requesting. This is a core concept in our application and in our security model.

Vulnerability	Description	Remediation
A5 - Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.	Pre-authenticated requests are mitigated by including a timestamp with every request and validating the timestamp on the server. We utilize AWS ( <a href="http://docs.amazonwebservices.com/AmazonS3/latest/index.html?RESTAuthentication.html">http://docs.amazonwebservices.com/AmazonS3/latest/index.html?RESTAuthentication.html</a> ) for our authentication. Each request contains a timestamp of the request in a digested form, which is then validated with the time of the server. The request is allowed to be within a certain set of minutes of the server time. Anything outside this range is considered an invalid request and unauthorized.
A6 - 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.	Errors are sanitized before delivering to the end-user. Real errors are logged in the back-end only.
A7 - 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.	User passwords are not stored in our application, instead a digested form is stored. When users log in they receive a token and a secret key. The browser is expected to use the token and the secret key to produce digests of the APIs they want to use in accordance with the AWS security style. All requests are time-sensitive, and are carried over SSL. The secret key is only delivered once during the login session.
A8 - 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.	All accounts in our system are assigned a unique public/private key pair based on the x509 certificate standard. All passwords and sensitive data are encrypted using this standard. More information on password security can be found here: <a href="http://help.boomi.com/display/BOD/Password+Encryption+Security">http://help.boomi.com/display/BOD/Password+Encryption+Security</a>
A9 - Insecure Communications	Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.	All data is encrypted using SSL over HTTP with a certificate signed by a trusted Certificate Authority. Any attempts to use regular HTTP are either rejected or redirected to the HTTPS server.
A10 - 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.	In addition to authorization provided in A4, all requests are authorized based on account features and licensing. This is a core concept in our application and in our security model.