Web Application Whitepaper

Prepared by Simone Quatrini and Isa Shorehdeli
Security Advisory EMEAR

css-adv-outreach@cisco.com | labs.portcullis.co.uk

6th September, 2017

1.0

General Release

1. Introduction

In this digital age, it is more important than ever for companies to protect their assets from unauthorised access. Too many times we have seen organisations not adequately protecting customer data, failing to prevent a malicious attack, or not adopting a responsible attitude to information security. These breaches in terms of confidentiality (limiting access to information to those who are authorised), integrity (to protect information from being modified without authorisation), and availability (ensuring information is accessible to authorised personnel at expected times), can have a devastating impact and can cause financial, economic, and reputational damage.
When developing web applications, it can sometimes be difficult to manage security with usability. While it is important to provide the best service possible to customers, web application security should not be forgotten about. A vulnerability in a web application can expose sensitive data or allow access to the back-end systems on which the application is running. Denial of Service (DoS) attacks can be used to take websites offline, disrupting the online business that many websites rely on.
The threat landscape of cyber security is constantly evolving, which is why it is important to take a defence-in-depth approach to information security. A company should not rely on one security feature to protect their organisation and data. Defence-in-depth not only prevents breaches, but can buy an organisation crucial time to react should an incident occur.
Cisco takes pride in providing customers with the best service possible, and this document aims to inform and educate on the issues that are seen most frequently in web applications, to enable our customers to better protect their important assets by adopting a healthier security posture.

2. Top Ten Issues From 2016

This section will detail the top ten most commonly seen web application issues that have been discovered by Security Advisory EMEAR during 2016.
We tested a total of 322 web applications and discovered 6798 issues throughout the year. Whilst none of these assessments resulted in a clean bill of health for the client, the level of exposure (whilst it varied) reflected the level of maturity of the organisation and application under test. Based on this data, we are now able to produce a list of the most commonly occurring issues of High, Medium, Low and Informational Severity.
Note that the same issue may appear in different categories due to the severity of the issue being changed for specific applications. The categories above may not necessarily represent the severity of an issue to a specific web application if there are other mitigating factors in place.

High severity:

- Web Application Vulnerable To Stored Cross-site Scripting
- Web Application Allows Arbitrary File Uploads
- Web Application Vulnerable To SQL Injection

- Web Application Implements Weak Access Control Lists (ACLs)
- Unsupported Version Of .NET Framework Installed
- Web Application Uses Insecure Transmission For Sensitive Content
- Out-Of-Date And Vulnerable Version Of PHP
- Web Application Allows Horizontal And Vertical Privilege Escalation
- Web Application Session ID Passed In URL
- Web Application Vulnerable To XML External Entity Attack

Medium severity:

- HTTP Strict Transport Security (HSTS) Not Enabled
- Web Application Can Be Embedded In Third-Party Websites
- Web Application Does Not Use 'secure' Cookies
- Web Application Includes Files From A Third-Party
- Web Application Implements Weak Password Policy
- Web Application Allows User Enumeration
- Web Application Vulnerable To Reflective Cross-site Scripting
- Out-Of-Date And Vulnerable Software Installed
- Web Application Available Over Unencrypted Channel
- Web Application Vulnerable To Cross-site Request Forgery Attacks

Low severity:

- 'X-Content-Type-Options' HTTP Header Not Specified
- Web Application Does Not Specify Content Security Policy (CSP)
- 'X-XSS-Protection' HTTP Header Not Specified
- 'X-Permitted-Cross-Domain-Policies' HTTP Header Not Specified
- Gratuitous Information Disclosure Within HTTP Response Headers
- HTTP Public Key Pinning (HPKP) Not Enabled
- Web Application Does Not Use 'HTTPOnly' Cookies
- Web Application Allows Concurrent Logins
- SSL Service Does Not Mandate Use Of Forward Secrecy
- Web Application Form AutoComplete Enabled

Informational severity:

- Web Application Does Not Display Last Successful And Unsuccessful Login
- Application Username Not Case-Sensitive
- HTTP GET And POST Request Methods Interchangeable
- Web Application Does Not Display Secure Use Policy
- Web Application Allows Client Browser To Cache Sensitive Information
- Web Server Does Not Use Custom Error Pages
- Default Apache Directories And Files Present
- Web Application Does Not Specify Character Set
- Users Unable To Log Out

- Test Files Identified

During the testing we performed over the year, the following issues were identified as being the most commonly occurring across the range of differing severities:

- 'X-Content-Type-Options' HTTP Header Not Specified
- Web Application Does Not Specify Content Security Policy (CSP)
- 'X-XSS-Protection' HTTP Header Not Specified
- 'X-Permitted-Cross-Domain-Policies' HTTP Header Not Specified
- HTTP Strict Transport Security (HSTS) Not Enabled
- Web Application Can Be Embedded In Third-Party Websites
- HTTP Public Key Pinning (HPKP) Not Enabled
- Gratuitous Information Disclosure Within HTTP Response Headers
- Web Application Allows Concurrent Logins
- Web Application Does Not Use 'HTTPOnly' Cookies

Although the severity for most of these issues is relatively low, fixing these issues is fairly simple and can increase the security posture of an organisation.

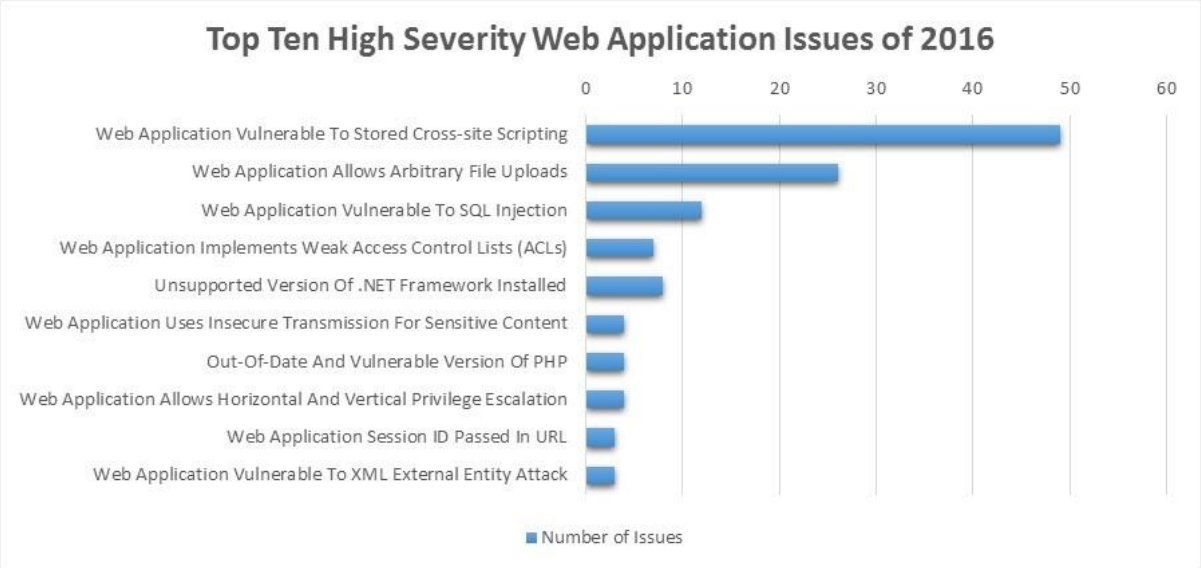

Figure 1: An Image Showing The Top Ten High Severity Issues Of 2016

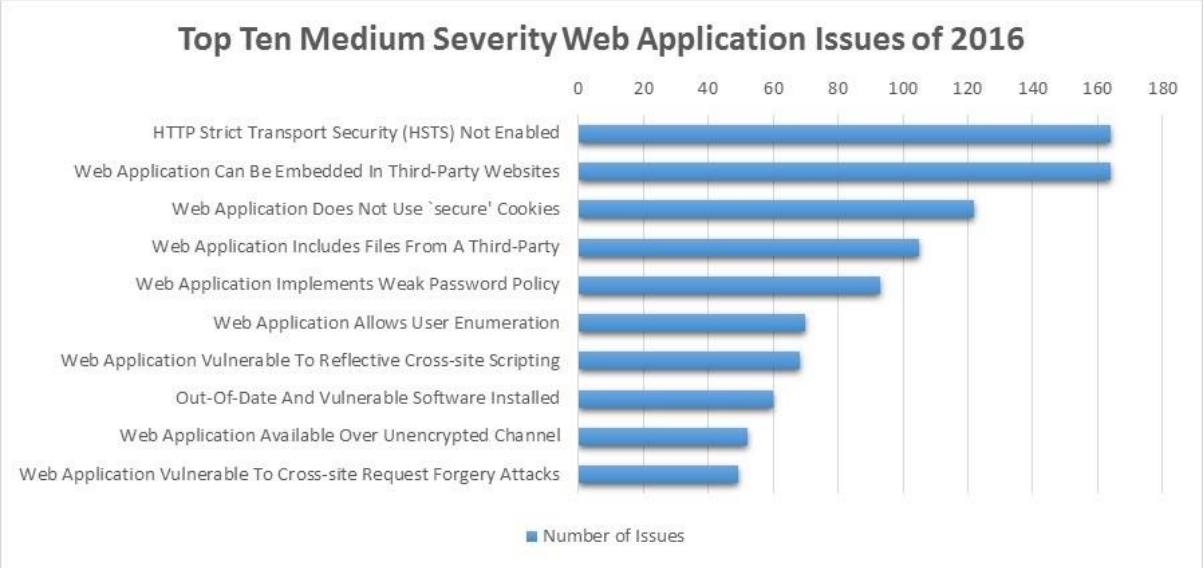

Figure 2: An Image Showing The Top Ten Medium Severity Issues Of 2016

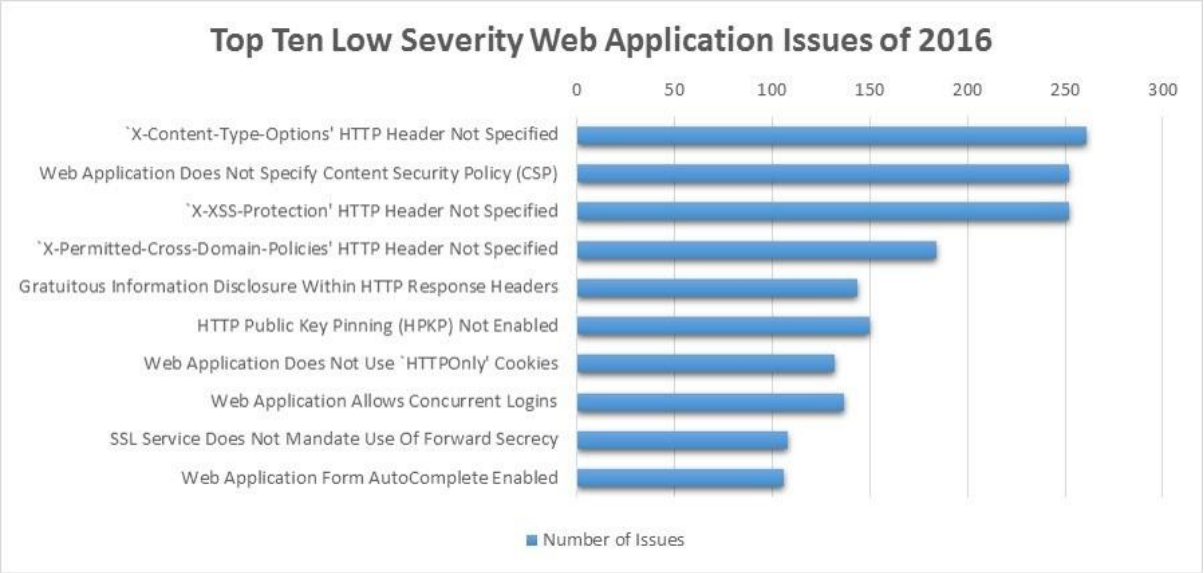

Figure 3: An Image Showing The Top Ten Low Severity Issues Of 2016

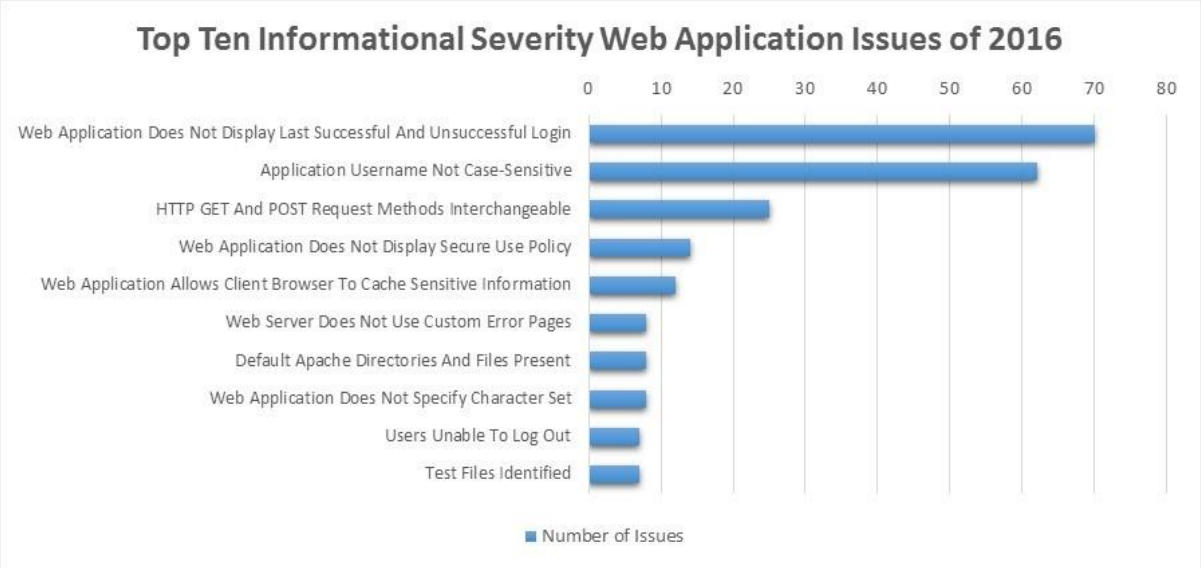

Figure 4: An Image Showing The Top Ten High Severity Issues Of 2016

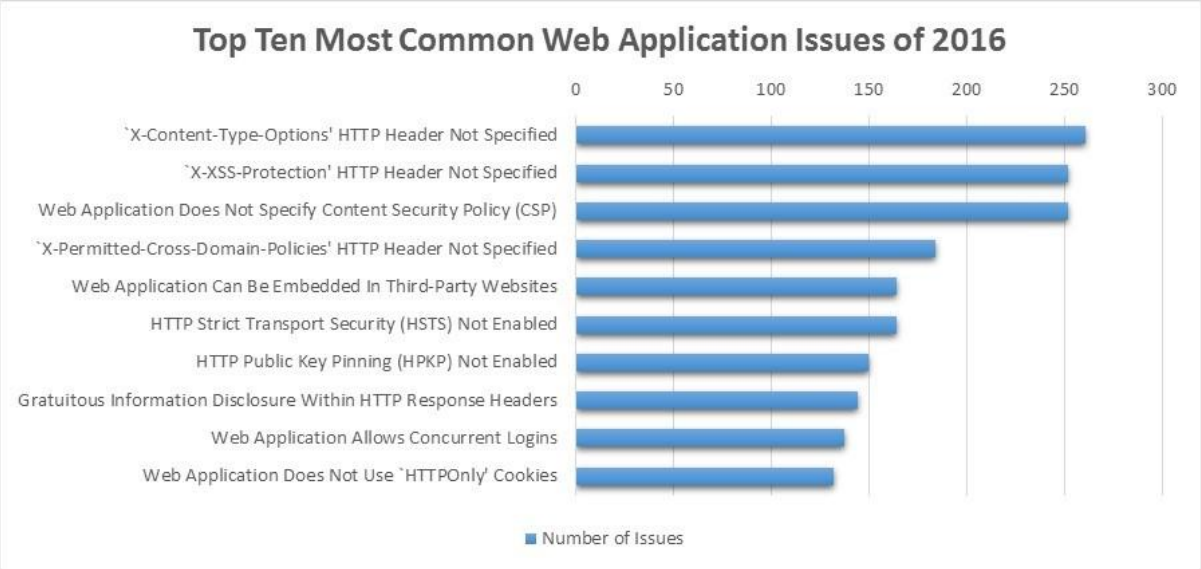

Figure 5: An Image Showing The Top Ten Issues Of 2016

2.1. High Severity Issues

2.1.1. Web Application Vulnerable To Stored Cross-site Scripting

Cross-site Scripting (XSS) occurs when a browser processes unsanitised user input. Cross-site Scripting can be used to perform attacks such as session hijacking, by redirecting a user's cookies allowing them to masquerade as the authenticated user. Stored Cross-site Scripting can affect multiple users, as the dangerous code stored in the database and reflected back on the web page at every visit, so any user that visits the web page will be impacted by the XSS. This kind of XSS, which was classed as high severity, was found 49 times.

2.1.2. Web Application Allows Arbitrary File Uploads

Allowing arbitrary file uploads can grant attackers the ability to upload any malicious content that they choose. Not only could this result in reputational damage if the content is inappropriate, but attackers can also upload files that can grant them further access into the web application and the system on which it is hosted. 26 cases of this were found.

2.1.3. Web Application Vulnerable To SQL Injection

SQL Injection occurs when a back-end database processes malicious user input that has not been sanitised inside a SQL statement. SQL Injection can result in the exposure and modification of sensitive information. In order to prevent this, all user input should be sanitised before processing. High severity SQL Injection was found 12 times.

2.1.4. Web Application Implements Weak Access Control Lists (ACLs)

Access Control Lists are used to control the actions that authorised users can perform. By poorly implementing these ACLs, a user can bypass the expected workflow of the application to perform unauthorised actions. This issue was seen seven times.

2.1.5. Unsupported Version Of .NET Framework Installed

Running an unsupported version of .NET Framework means that security updates and bug fixes are no longer received. This greatly increases the chances of vulnerabilities being introduced into the system as software support are no longer available. Unsupported version of .NET Framework was discovered eight times.

2.1.6. Web Application Uses Insecure Transmission For Sensitive Content

By sending sensitive data over a plain text protocol (HTTP) allows a positioned attacker to intercept traffic between a user's session and the web server and capture sensitive data such as user credentials. Four cases of this issue were discovered.

2.1.7. Out-Of-Date And Vulnerable Version Of PHP

Running an out-of-date version of PHP can expose the system to a number of issues such as Cross-site Scripting, buffer overflow, Denial of Service, and many more security issues. Out-Of-Date PHP was found four times.

2.1.8. Web Application Allows Horizontal And Vertical Privilege Escalation

Privilege Escalation pertains to the ability of an unauthorised user to access, modify and/or delete other user's data including data reserved for user of a higher privilege. Depending on the nature of the functionality accessed, significant impacts to the business can be observed. Four cases of this issue was identified.

2.1.9. Web Application Session ID Passed In URL

Storing sensitive user data such as Session IDs in the URL may lead to theft through the referrer header or unwanted disclosure by being stored in web server logs. This issue was found three times.

2.1.10. Web Application Vulnerable To XML External Entity Attack

An XML External entity Attack or XXE arise due to improper parsing of XML documents allowing an attacker to define ‘entities’ that reference external resources. Successful exploitation of this vulnerability may allow an attacker to traverse the file system, retrieve arbitrary files or cause a Denial-of-Service condition. This issue was found three times.

2.2. Medium Severity Issues

2.2.1. HTTP Strict Transport Security (HSTS) Not Enabled

HSTS was found not to be enabled in 164 cases. By enabling HSTS, users are forced to use the more secure HTTPS and this can prevent users from giving away sensitive information over an unencrypted channel (HTTP).

2.2.2. Web Application Can Be Embedded In Third-Party Websites

By allowing your web application to be embedded into other websites, you leave your users vulnerable to ‘click-jacking’ and ‘frame-jacking’ attacks. They can be lured to another website which has all or part of the legitimate website reproduced within it. Users can unknowingly disclose sensitive data to the falsified site. Cisco found 164 web applications where this was possible.

2.2.3. Web Application Does Not Use ‘secure’ Cookies

Not setting the ‘secure’ flag on cookies means that there is a possibility that they can be transferred over the less secure HTTP protocol. If these cookies are obtained by an attacker and contain session information, an attacker will be able to masquerade as the valid user easily. This was found 122 times which means it was present in over a third of the applications we tested.

2.2.4. Web Application Includes Files From A Third-Party

If the third-party hosting the files is compromised, at attacker can modify the files to deliver malicious content to legitimate users. This issue was discovered 105 times.

2.2.5. Web Application Implements Weak Password Policy

Allowing weak passwords increases the chances of an attacker being able to compromise a user account through brute-forcing, dictionary attacks, or simple guessing. 93 medium severity weak password policies were identified which did not enforce an adequate minimum length or which did not suitably limit the use of passwords based on dictionary words (the latter should be controlled through the enforcement for a requirement for non-alphanumeric characters).

2.2.6. Web Application Allows User Enumeration

Some web applications respond differently to user input based on whether or not the input is valid, such as usernames or email addresses. This can allow an attacker to determine, through trial and error, what users exist on the system. This information can then be used in further attacks. 70 cases of this issue were discovered.

2.2.7. Web Application Vulnerable To Reflective Cross-site Scripting

See the High Severity Issues section of this paper. 68 cases of medium severity Reflective Cross-site Scripting were found by Cisco.

2.2.8. Out-Of-Date And Vulnerable Software Installed

Running out-of-date software on your web application can expose the application to well known or not yet public vulnerabilities if they have been discovered in the software you are using. Later versions of the software may fix these security holes so it is important to stay up to date.

2.2.9. Web Application Available Over Unencrypted Channel

Offering a web application over an unencrypted channel (like HTTP) can result in an attacker utilising a "Man-In-The-Middle" style attack to intercept authentication credentials or alter end user web content. 52 instances of this issue was identified.

2.2.10. Web Application Vulnerable to Cross-site Request Forgery Attacks

Cross-site Request Forgery can be used to force a logged-on user's browser to send unauthorised HTTP requests, which can trick the web application into handling what the application thinks are legitimate requests. 96 cases of medium severity CSRF issues were detected. Cisco identified 49 cases of CSRF.

2.3. Low Severity Issues

2.3.1. 'X-Content-Type-Options' HTTP Header Not Specified

This header can prevent dangerous drive-by downloads from malicious user content uploaded to a site. 261 cases were found where this header was not enabled. This is the most common issue we have reported and is present in approximately 81% of the applications tested.

2.3.2. Web Application Does Not Specify Content Security Policy (CSP)

The Content-Security-Policy header allows you to control what web assets can be loaded in the browser. Without this header, there is an increased risk of Cross-site Scripting attacks. This security flag not to be enabled a total of 252 times (78.3% of the 322 applications tested). That means that approximately three quarters of web applications we tested had this header missing.

2.3.3. 'X-XSS-Protection' HTTP Header Not Specified

The X-XSS-Protection HTTP Header makes it much more difficult for attackers to exploit occurrences of Reflective Cross-site Scripting, but this was found to be lacking on 252 occasions.

2.3.4. 'X-Permitted-Cross-Domain-Policies' HTTP Header Not Specified

The lack of the X-Permitted-Cross-Domain-Policies header allows an attacker to leverage issues within a web application in order to forge a cross-domain policy. An attacker can then use a Flash application hosted on a web server under their own domain to access the website via the user's browser. This issue was identified 184 times.

2.3.5. Gratuitous Information Disclosure Within HTTP Response Headers

Although not a direct vulnerability, attackers will often use as much information as they can gather to help leverage a successful attack. By suitably configuring the web server to not display this kind of information, you can reduce the total amount of information that an attacker can retrieve. This issue was discovered 144 times.

2.3.6. HTTP Public Key Pinning (HPKP) Not Enabled

The HTTP Public Key Pinning (HTKP) header has a key role in achieving strength-in-depth within web applications. HPKP can be used to prevent impersonation attacks by 'pinning' the expected certificate or public key to a host. Cisco observed 184 instances of this issue.

2.3.7. Web Application Does Not Use 'HTTPOnly' Cookies

Enabling the 'HTTPOnly' flag when setting a new cookie makes it more difficult for an attacker to compromise cookies. If this is not enabled and the cookies in question are used to keep track of a user's session, compromise of these cookies can lead to whole user account compromise. This option was found not to be enabled in 132 cases.

2.3.8. Web Application Allows Concurrent Logins

Allowing concurrent logins can make it more difficult for a user or a system administrator to identify account compromise. It also means that, if there is an account compromise, it can be difficult to identify illegitimate user actions. This was discovered 137 times.

2.3.9. SSL Service Does Not Mandate Use Of Forward Secrecy

Forward secrecy is an important part of cryptography, and by not mandating the use of forward secrecy, you risk an attacker being able to decrypt previous intercepted communications if the server's private key is compromised. This was found 108 times, approximately once in every three applications.

2.3.10. Web Application Form AutoComplete Enabled

AutoComplete was found to be enabled 106 times in 2016. By having this feature enabled (by default, if not manually specified), an attacker could easily view sensitive information saved on an unlocked machine.

2.4. Informational Severity Vulnerabilities

2.4.1. Web Application Does Not Display Last Successful And Unsuccessful Login

By displaying the last successful and unsuccessful login, a user is able to identify whether a different individual has been accessing their account, and can take appropriate action. 70 cases of this were found.

2.4.2. Application Username Not Case-Sensitive

If an application does not support case-sensitive usernames, it decreases the number of possible usernames that can be iterated through a brute-force attack,

decreasing the time it would take for an attacker to compromise an account. 62 instances of this issue were found.

2.4.3. HTTP GET And POST Request Methods Interchangeable

By not mandating the use of POST requests where necessary, it can present an attacker with an opportunity to carry out several types of exploit when combined with other vulnerabilities. 25 cases of this issue were discovered.

2.4.4. Web Application Does Not Display Secure Use Policy

Web applications that display the secure use policy allows users to make themselves aware about using the site securely with the idea of making them less likely to succumb to attacks. Cisco identified this issue on 14 occasions.

2.4.5. Web Application Allows Client Browser To Cache Sensitive Information

If an attacker is able to gain physical access to a computer, it would be relatively easy to recover sensitive information that was stored by the web application, by clicking the 'Back' button or by accessing cache the browser's cache. 12 cases of this were found and marked as informational severity.

2.4.6. Web Server Does Not Use Custom Error Pages

Default error pages may lead to the disclosure of system information to attackers giving them a better understanding of the underlying system. Eight instances of this issue was found.

2.4.7. Default Apache Directories And Files Present

Similarly to the above, further information about the web server may be obtained by an attacker. Cisco reported this issue eight times.

2.4.8. Web Application Does Not Specify Character Set

In the character set is not specified in every web response, the browser may interpret the response content using a different character set which may lead to unexpected results such as Cross-site scripting. This issue was identified on eight occasions.

2.4.9. Users Unable To Log Out

Users who are unable to log out of their current session may lead to malicious users with access to their computer to use their session. Cisco identified seven application to be affected by this issue.

2.4.10. Test Files Identified

Test files which remain on the server may disclose sensitive information and are often subjected to the same security measures as fully developed code. This particular issue has been identified seven times.

2.5. Top Ten Most Common Issues

All of the Top Ten Most Common Issues have been described in the in previous sections:
Medium Severity:

- HTTP Strict Transport Security (HSTS) Not Enabled (found 164 times)
- Web Application Can Be Embedded In Third-Party Websites (found 164 times)

Low Severity:

- 'X-Content-Type-Options' HTTP Header Not Specified (found 261 times)
- Web Application Does Not Specify Content Security Policy (CSP) (found 252 times)
- 'X-XSS-Protection' HTTP Header Not Specified (found 252 times)
- 'X-Permitted-Cross-Domain-Policies' HTTP Header Not Specified (found 184 times)
- HTTP Public Key Pinning (HPKP) Not Enabled (found 150 times)
- Gratuitous Information Disclosure Within HTTP Response Headers (found 144 times)
- Web Application Allows Concurrent Logins (found 137 times)
- Web Application Does Not Use 'HTTPOnly' Cookies (found 132 times)

3. OWASP Top Ten

3.1. Overview

This section will detail the issues found by Cisco in all engagements, and how they fit into the categories detailed in the OWASP Top Ten - 2017.
In 2016, Cisco tested a total of 322 web applications where 6798 issues were found in total. 4057 of these issues could be categorised as being part of the OWASP Top Ten - 2017. This paper will classify the issues found in these applications into the following categories:

- Injection
- Broken Authentication and Session Management
- Cross-Site Scripting (XSS)
- Broken Access Control
- Security Misconfiguration
- Sensitive Data Exposure
- Insufficient Attack Protection
- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Underprotected APIs

For more detail on these categories, including recommendations for prevention, please see the OWASP Top Ten - 2017 document.

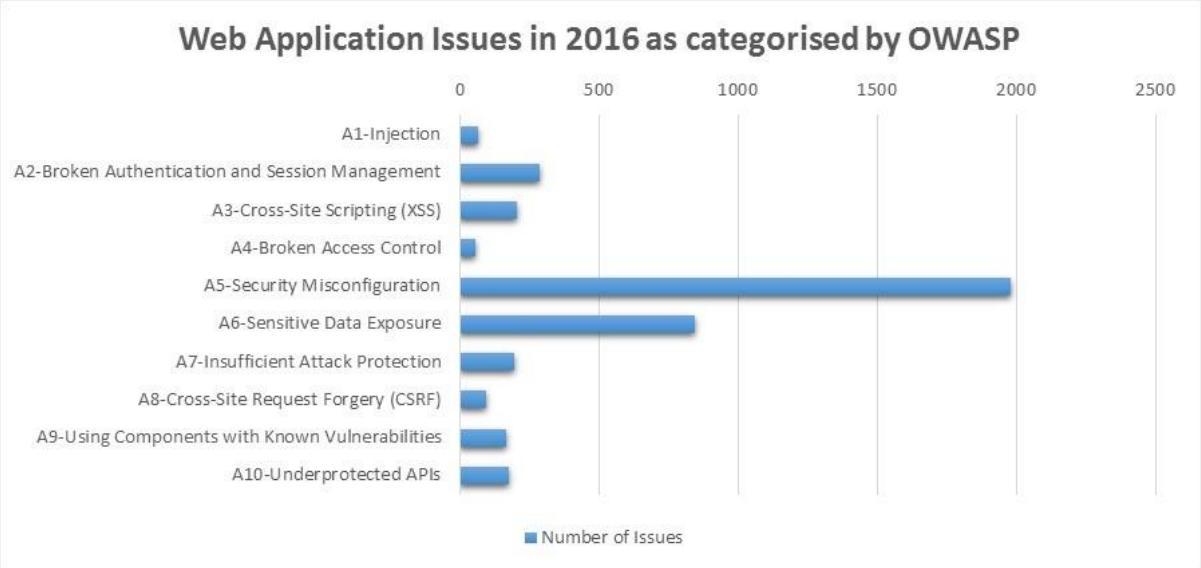

Figure 6: A Graph Showing The Number Of Issues Found In 2016 As Categorised By OWASP

3.2. The Categories

3.2.1. Injection

Injection vulnerabilities occur when data is not validated before being sent to an interpreter. If data is not validated, an attacker can manipulate the query or command to return unexpected data. Instances of injection were found 64 times, with the most common kind of injection being SQL injection which was found 12 times.

3.2.2. Broken Authentication and Session Management

Instances of broken authentication and session management were found 284 times, making this category the second highest. The most common issue discovered was "No Session Timeout Configured", occurring 49 times. Not having a session timeout can mean that, if a user forgets to log out, an attacker would be able to navigate back to the application and the original user's session would continue as normal.

3.2.3. Cross-Site Scripting (XSS)

Cross-site Scripting was found to have occurred 203 times. According to our data, the issue that was discovered the most was "Web Application Vulnerable To Reflective Cross-site Scripting", occurring 140 times - nearly half of all web applications tested in 2015. Cross-site Scripting occurs when a browser processes user input that is not sufficiently sanitised.

3.2.4. Broken Access Control

There were 56 issues related to broken or missing access control access control according to our data, and all 56 issues were cases of a web application implementing weak Access Control Lists (ACLs). By not having strong ACLs, it may be possible for users to perform unauthorised actions, which could lead to further issues, such as the disclosure of sensitive information.

3.2.5. Security Misconfiguration

This category was the highest, with 1977 issues found to relate to security misconfiguration. The most common issue in this category was "X-Content-Type-Options HTTP Header Not Specified", being identified 261 times! More information can be found in the Low Severity Issues section of this paper.

3.2.6. Sensitive Data Exposure

Instances of data exposure were detected 845 times. The most common issue in this category was "Gratuitous Information Disclosure Within HTTP Response Headers", being discovered 144 times. More detail about this specific issue can be found in the Low Severity Issues section of this paper.

3.2.7. Insufficient Attack Protection

Web application with instances of insufficient attack protection was detected 193 times. The most prevalent issue being "Web Application Contains No Anti-Automation Features" which was reported on 132 instances. Without anti-automation features, e.g. CAPTCHA, an attacker who successfully automates his attacks may lead to a number of issues including a denial-of-service condition.

3.2.8. Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery enables an attacker to carry out legitimate, unintended requests using the session of a valid user. 94 cases of CSRF were identified.

3.2.9. Using Components with Known Vulnerabilities

By using a component that has known vulnerabilities, you risk compromising the whole security of the web application if the vulnerability is serious. 167 cases of these vulnerable components were found. A sensible patching system can help to mitigate the use of these vulnerable components.

3.2.10. Underprotected APIs

Instances of substandard protection for APIs was detected on 174 occasions, the most common issue being "Web Service Available Over Unencrypted Channel" was reported on 12 times.

4. Comparing Development Languages

There is a broad spectrum of languages that web applications are developed in, and it is often the case that certain languages are found more often to be written in such a way as to increase the application's exposure to vulnerabilities. This increased exposure can be result from poor coding standards and insufficient security awareness, therefore this section aims to highlight the average number of issues found in applications where the following technologies and development languages were identified:

- asp - Microsoft Active Server Pages
- aspx - Microsoft ASP.NET
- cfm - The Cold Fusion Inguage

- jsp - Java Server Pages
- php - The PHP language
- pl - The Perl language
- py - The Python language

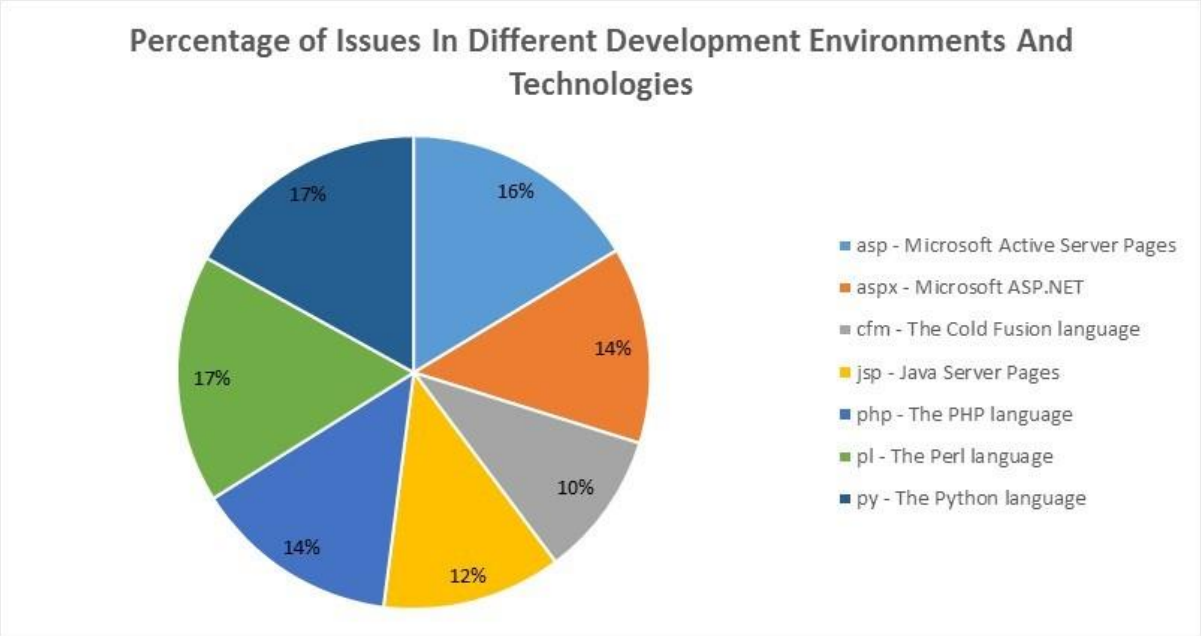

Figure 7: A Graph Showing the Percentage of Issues per Development Language

As the chart shows, there is not a significant difference in the number of issues for different languages and technologies. One reason for this may be that it was not possible to gather information for all of the web applications tested this year, so the dataset used for this section was significantly smaller that the sections above, leading to less reliability in the results. What this does show, however, is that developers need to be aware of common security pitfalls (irrespective of the language they choose to use), to enable them to reduce the number of easily identifiable vulnerabilities to a minimum.

5. Conclusions

Cisco considers that the specific issues resulting from the inappropriate deployment of applications should be mitigated. Whilst doing so will pose challenges, Cisco does not believe that these are insurmountable, however it is noted that considerable risk can be avoided by considering security early on in the project.
Cisco recommends that existing processes and procedures are reviewed and updated regularly, however we also believe that new policies and procedures may be required in the following areas:

- Baselining your shared (OS, databases, web and application server) components
- Understanding the inherent risks exposed by your choice of application stack
- Defining verification activities that can be performed internally by your application support teams prior to any external testing being performed

- Creating metrics that allow for identification of root causes so that these can be fed back into your development teams

Of course, Cisco does not deal in business risk. Each individual project and business must evaluate its own risk. It is perfectly acceptable for businesses with a mature risk handling process to simply "write off" risk, accepting it as part of the project. Of course, this does not address the underlying security issues, which will likely increase over time as new exploits and techniques are developed. For this reason, it is important to ensure that projects are well maintained, and regularly security tested, to ensure that their security posture has not degraded in any of the above areas.

6. References

- **https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project**- OWASP Top Ten Project
- **https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010%20-%202017%20RC1-English.pdf**- OWASP Top 10 2017