MANAGED SECURITY SERVICES
PENETRATION TEST (Sample Report)

Report on the results and associated recommendations
arising from a Black Box Penetration Test of support.clientcompany.com.

Date Issued:

Report on the results and associated recommendations arising from a security test against the Customer Interface/Dashboard Application

/Document Control

VERSION CONTROL

Version	Date	Editor	Comments
0.1	5th November		DRAFT

DISTRIBUTION

Individual	Date	Method
Jack Black	5th November	Tresorit

/Executive Summary

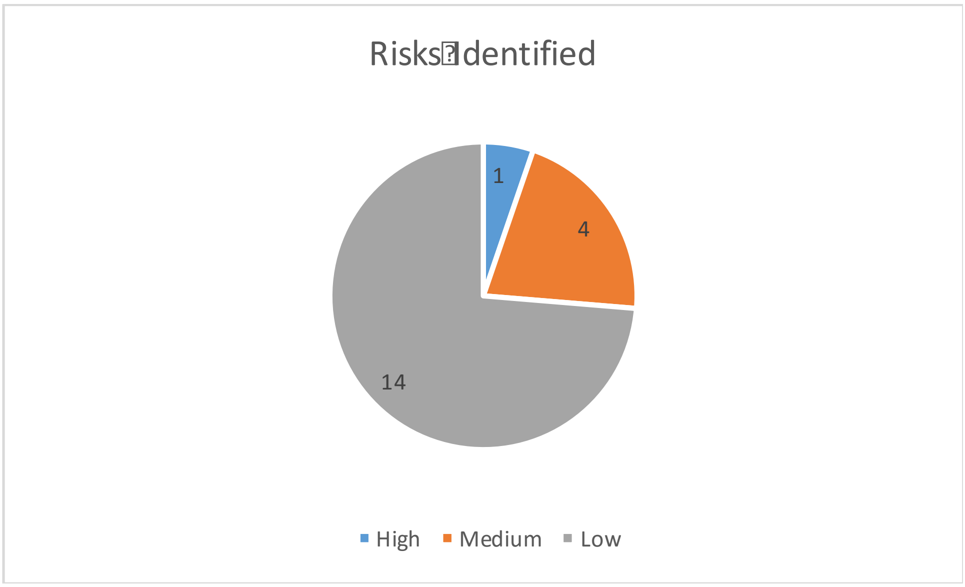
Pulsar has been engaged by ClientCompany to undertake security testing against the support.clientcompany.com web application. The testing took place over the period from 21st October to 30th October 2015. During this period the application was analysed and assessed using a combination of standard tools and utilities and the knowledge and experience of our technical team. Although at the time of this engagement, the application was not in production, we nonetheless stopped short of undertaking specific tests that would either a) evidently risk the integrity and stability of the systems, or b) actively exploit potential vulnerabilities.

Overall we believe that a reasonable level of security has been attained by the applications that were the target of this test, but due to there being a high and some medium and low risk issues, remedial action needs to be carried out prior to official launch of the product. Testing revealed elements that are well-protected against several well-known vulnerabilities.

The high risk vulnerability is called Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext (BREACH) which exploits the compression in the underlying HTTP protocol.

We also found that information has been disclosed by the application that could be of use to an attacker, including the underlying web server version.

Regarding low risk issues, some misconfigurations have been detected, and some information has been revealed. Remediation requires minimal effort. For example, Clickjacking seems not particularly risky however it also should be attended in the near future as a real attack could superimpose transparent boxes over the actual content, causing the user to enter data into the attacker’s site where it could be recorded and used for more serious attacks.



Our Penetration Test has identified 1x High, 4x Medium and 14x Low risks

HIGH RISKS

The identified high risk vulnerability/recommendation is:

- BREACH attack

MEDIUM RISKS

The four identified medium risk vulnerabilities/recommendations are:

- Application error message
- No anti-virus on file uploads

- TLS1/SSLv3 Renegotiation Vulnerability
- Web Application Potentially Vulnerable to Clickjacking

LOW RISKS

The fourteen identified low risk vulnerabilities/recommendations are:

- Cacheable HTTPS response
- Http Server Type And Version Disclosure
- Missing XSS protection HTTP header for IE
- No Content-Disposition header in API responses
- No forward secrecy ciphers
- No session timeout
- OPTIONS Method Enabled
- OSCP stapling not enabled
- Password field with autocomplete enabled
- Portal name information disclosure
- Simultaneous logins allowed
- Strict transport security (HSTS) not enforced
- User enumeration
- Weak password policy

We strongly recommend that ClientCompany does not disregard the findings encountered in this report. If these vulnerabilities/recommendations are dealt with and fixed, the organisation will find that the defence-in-depth posture of the system will improve substantially. We also recommend that in line with good security practice, ClientCompany conducts periodic re-testing to ensure that neither intentional nor inadvertent changes have compromised their systems, and that new vulnerabilities have not become a threat to them.

/System Overview:

DESCRIPTION

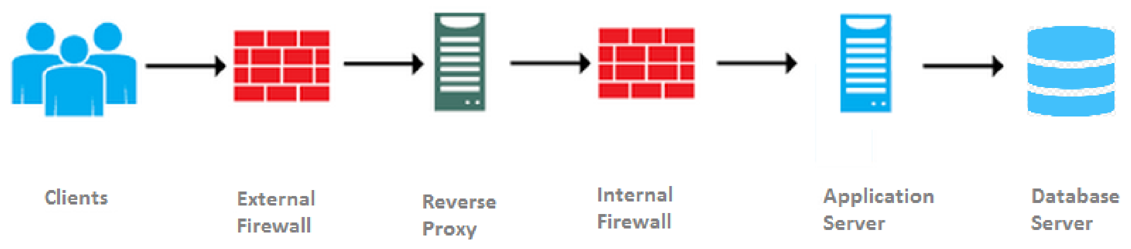
An externally facing Web Application hosted on ClientCompany Infrastructure in the Chandler McLeod Datacentre. Application is known as the “ClientCompany Customer Support Centre”. System is a customised 3rd party product developed by Atlassian (https://www.atlassian.com) (JiRA). User AAA is provided by integration with Active Directory and Crowd using LDAP.

Web Application is externally (Internet) accessible to any form of client and any external source via https://support.clientcompany.com

DNS / IP

- support.clientcompany.com. 000.000.000.000

Support.clientcompany.com technical architecture / overview



Environment

Clients

- Clients should be able to connect to support.clientcompany.com from ANY IP address (no white listing)
- Clients will only be ClientCompany customers
- Clients will be browsing to support.clientcompany.com from potentially ANY web enabled device (e.g. phone, tablet, desktop)

External Firewall

- This is the CMG NetScaler Firewall
- support.clientcompany.com will resolve this this server
- SSL will terminate here
- support.clientcompany.com communication will then forward to reverse proxy server on HTTP protocol (TCP: 80)

Reverse Proxy

- Reverse proxy will host the following ClientCompany web portals

- help.clientcompany.com
- support.clientcompany.com
- setup.clientcompany.com
- training.clientcompany.com (future)
- my.clientcompany.com (future)
- Reverse Proxy is Apache 2.4 listening on TCP port 80

Application Server

- Application Server is the Atlassian JIRA product, where customers will access the Service Desk module which is serviced on the following URL:
- xxxxxx.xxx.xxxxx:8080

Application server is a Java application and the Java Servlet engine is Tomcat

/Goals

Primary goal of this penetration test is to validate that the appropriate security measures have been implemented by ClientCompany at the various layers of the portal to mitigate malicious activity occurring.1

Secondary Goals

[1] Taken from ClientCompany Provided requirements document “Vulnerability Assessment requirements for support-clientcompany-com.pdf”

- To provide assurance to ClientCompany and their customers, that the web application adequately protects against unauthorised access to information.
- To achieve a high standard security posture and to identify all potential risks with the web application.
- To achieve confidence in their customers that their secure information has been proactively tested by a 3rd party, and to attain an executive report stating that all risks have been mitigated.

/Threat Assessment

ATTACK SOURCES

Through our discussions with Jack Black, and our understanding of the system, Pulsar has assessed the most likely Attack Source to be customers (ClientCompany's Customers and users of the application). Secondary attack sources may be ClientCompany's competitors. Finally, unauthenticated actors using scanning techniques to locate vulnerable servers online for various reasons is possible.

MOTIVATION

These users would be motivated by gaining access to unauthorised data primarily, whether it is information about other customers or competitors looking to acquire ClientCompany's Intellectual Property.

ATTACK TARGET

In the case of this project, the attack target is defined as support.clientcompany.com and potentially its hosting infrastructure and network would be affected by an attack.

RISK PROFILE | LIKLIHOOD

The information/data within the system potentially contains customer sensitive information such as employee data (bank details, dob etc.) and has been rated as <u>medium – high</u> value (commercially) and therefore likelihood of attack is rated as <u>medium</u>.

1. Web Application Penetration Test Report

This Penetration Test was undertaken using Pulsar's own methodology using methodology and the ASVS Version 3 (9th October 2015) framework from OWASP.

The Application is Java based JIRA, which is developed using the Struts Framework and runs on Apache/Coyote.

The scope of this report can be summarised as follows:

- 000.000.000.000
- https://support.clientcompany.com

All product names referenced herein are trademarks of their respective companies.

Rollback

During the testing the following accounts were used:

- Pen1
- Pen2

We recommend that these accounts are deleted along with any data associated with them (where applicable).

1. AUTHENTICATION

The purpose of these tests is to validate the methods employed to authenticate a user to the system.

The authentication process was first examined for potential weaknesses and then subjected to attacks in order to assess susceptibility to those weaknesses.

1.1.1 Enumeration

During enumeration, we examined the logon process of the application for any sensitive information available during authentication.

The login credentials (i.e. username and password) are submitted along with relevant session details in a HTTP POST.

1.1.2 Attack

Information from the enumeration phase was used to devise possible attacks on the logon process. We explored different attacks on the logon process:

- Brute-force valid password
- Subvert logon
- Valid user enumeration
- Obtain credentials from network traffic
- Obtain credentials from web browser

1.1.2.1 Brute-force valid authentication

In a brute force attack, an attacker will use a program which systematically attempts to force logon to a system by attempting all possible combinations of an authenticator (e.g. “AAAAAAAAAA”, “AAAAAAAAAB”, “AAAAAAAAAC” – “ZZZZZZZZZZ”).

The account attacked locked after a reasonable number of attempts, and for this reason, brute force attacks against it are not feasible. This is in line with good security practice.

1.1.2.2 Subvert Logon – forceful browsing

Attackers may guess or otherwise obtain URL references to internal functions and resubmit them to the web server in an attempt to bypass logon functions. We performed forceful browsing attacks to attempt to obtain URL references only available after authentication.

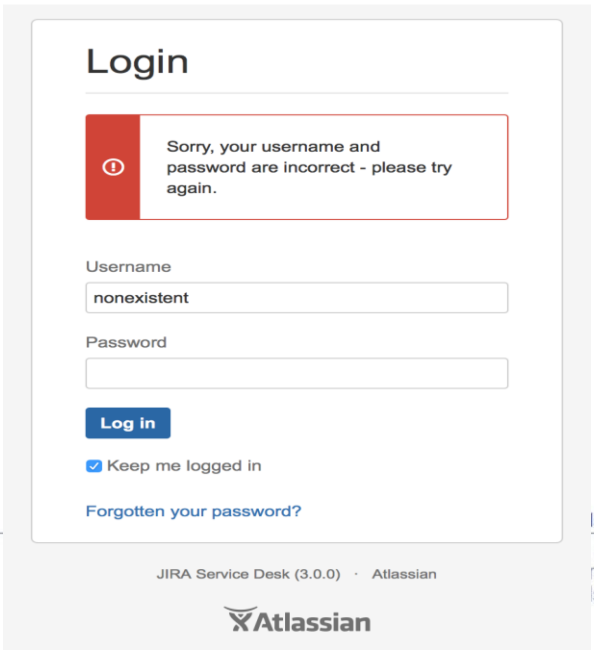
No protected content was retrieved using these methods. Attempts to view unauthorised content resulted in the support.clientcompany.com login screen. This is in line with good security practice.

1.1.2.3 Subvert logon - Injection

No susceptibility was found in the login to SQL injection, or other injection type attacks.

1.1.2.4 Valid user enumeration through error messages

The user logon does not allow user names to be determined through differential error messages (i.e. it is not possible to determine from the message received whether the login has been attempted with an invalid username, or invalid password or both) as illustrated in the below screenshots. This is in line with good security practice.



Login

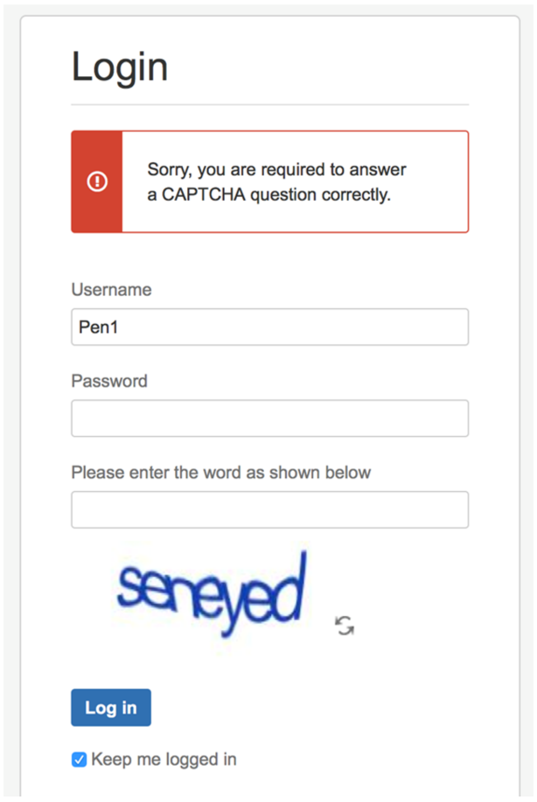
Sorry, your username and password are incorrect - please try again.

Username: nonexistent

Password:

Figure 1: Invalid Login Response

There is a way however, to find out whether or not a User Name is valid. After a number of attempts, if a user exists, a Captcha screen is displayed. There is no such restriction in case of invalid users.



Login

Sorry, you are required to answer a CAPTCHA question correctly.

Username: Pen1

Password:

Please enter the word as shown below:

Figure 2: Captcha on unsuccessful login of a valid user

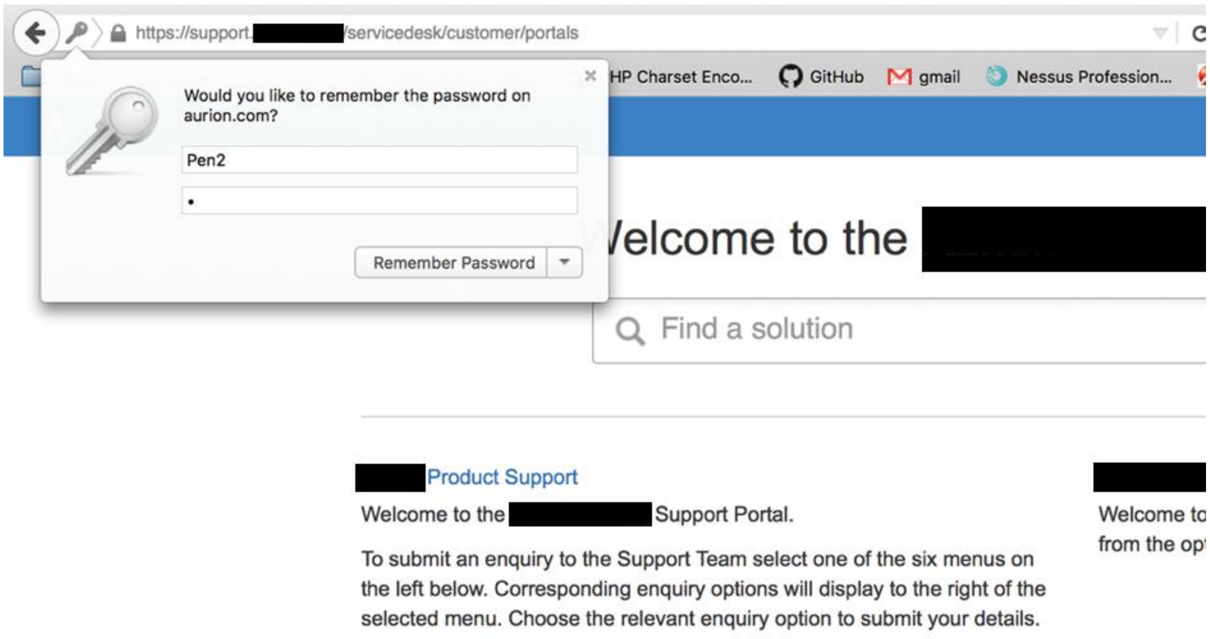
We recommend using Captcha in case of invalid users as well, so the application does not disclose whether a given username is valid or invalid.

1.1.2.5 Obtain Logon credentials from web browser

Most modern browsers offer to store usernames and passwords when entered for the convenience of the user.

If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application. This may present a risk if a user stores the password on an untrusted or shared machine. However correct use of browser password managers on individually owned machines will reduce this risk.

During testing, leading browsers offered to store the username and password to logon to the support.clientcompany.com application:



https://support.[redacted]/servicedesk/customer/portals

Would you like to remember the password on aurion.com?

Pen2

•

Remember Password

/elcome to the [redacted]

Find a solution

[redacted] Product Support

Welcome to the [redacted] Support Portal.

To submit an enquiry to the Support Team select one of the six menus on the left below. Corresponding enquiry options will display to the right of the selected menu. Choose the relevant enquiry option to submit your details.

[redacted]

Welcome to
from the op

Figure 3: Autocomplete enabled on login form

If the application is likely to be used from shared systems, we recommend that browsers are prevented from storing credentials entered into HTML forms.

Pulsar recommends that the 'autocomplete=off' directive is embedded in the HTML source code in order to avoid this.

1.1.2.6 Obtain Logon credentials from network traffic

Logons to the application are provided over the encrypted HTTPS protocol. This means that it is not possible for attackers to eavesdrop on unencrypted plain-text data, or to intercept login credentials. This is in line with good security practice.

2. SESSION HANDLING

We enumerated the mechanisms typically used to control session and tested them for suitability.

We identified to following cookies used by ClientCompany application:

- JSESSIONID

Testing indicated that cookies are used as tokens responsible for establishing and maintaining user sessions.

1.1.3 Session approximation attack

Samples of these session identifiers were collected to assess any pattern in their generation.

The samples were analysed for obvious patterns in generation. Analysis indicates that session identifiers appear to provide adequate entropy (i.e. high quality of randomness). This is illustrated in the below figure:

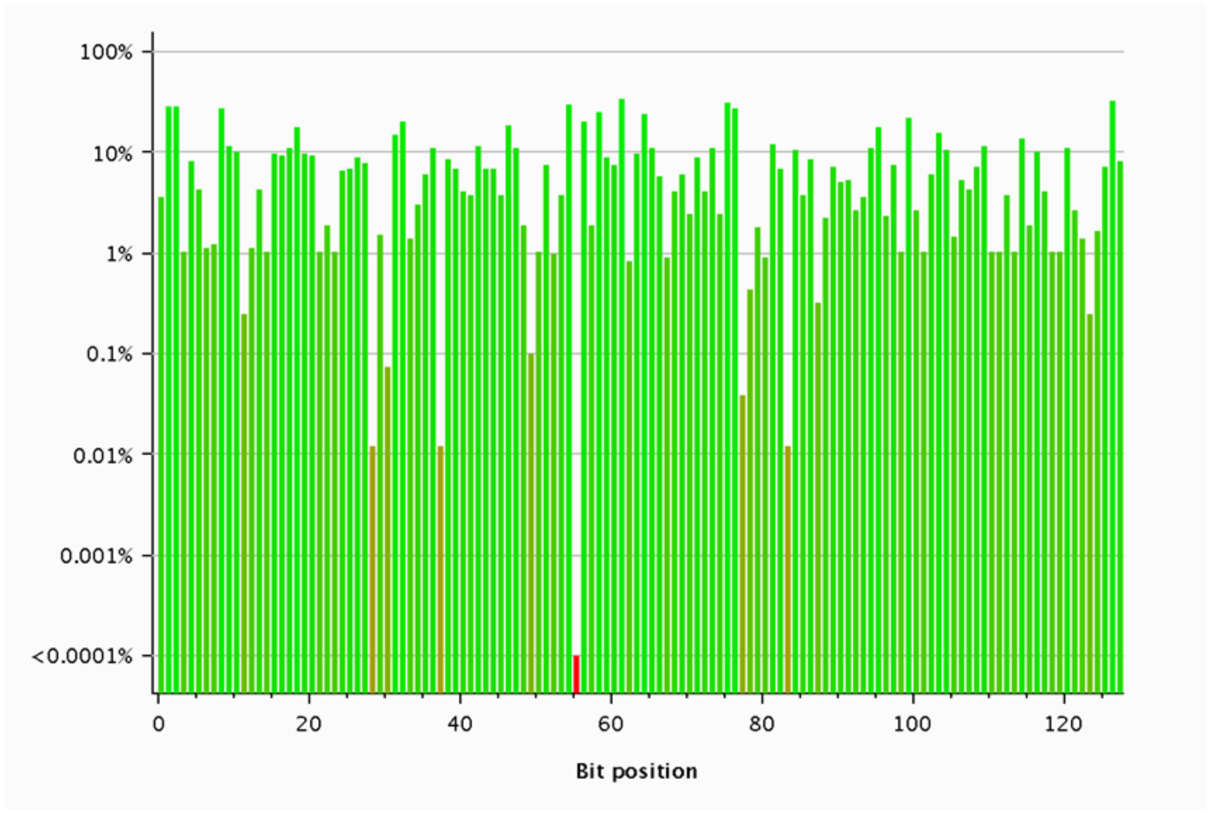


Figure 4: Chart indicating the degree of confidence in the randomness of the sample at each bit position from the cookie obtained from the ClientCompany application

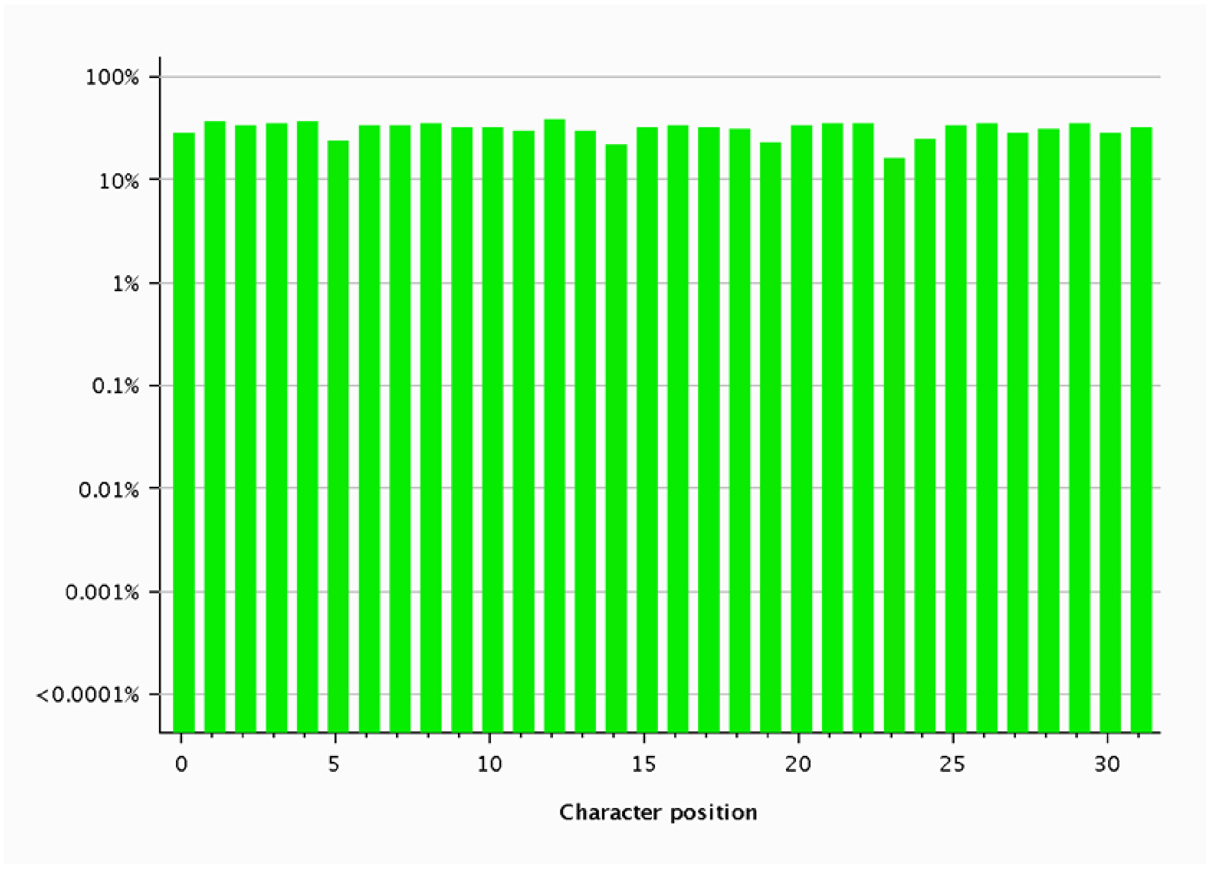


Figure 5: Chart indicating the degree of confidence in the randomness of the sample at each character position from the cookie obtained from the ClientCompany application.

It is unlikely that an attacker can ascertain the value of a valid session identifier within a reasonable timeframe and thus hijack a valid session.

The session expiration can be extended to two weeks using the 'Remember me' option which can pose a security risk. We recommend reviewing this function.

In case the user ignores the 'Remember me' function the session identifiers appear to expire after a reasonable period of time, which is in line with good security practice.

1.1.4 Session replay

Applications which do not properly expire valid sessions may be susceptible to input from the session being replayed in order to gain access to the session.

After logout, we attempted to replay requests made during the session. These requests were processed successfully by the server, and on this basis, we conclude that the session is not properly destroyed server-side after session expiry, and it is therefore possible to replay the logon/session.

The logout function on the application also appears to operate incorrectly, as it is possible to browse back to pages previously accessed in the same browser session after logout.

We recommend a thorough review of the way sessions are handled within the application.

1.1.5 Session fixation

Any session identifier which is set prior to the user authenticating and does not change throughout the session could expose the application to a session fixation attack.

With regard to the logon in this case, no cookies used in session handling were detected to be set prior to logon, and therefore an unauthenticated attacker would not be able to retrieve an unbound cookie prior to authentication. This is in line with good security practice.

1.1.6 Interception of session identifier

All traffic is sent over the encrypted HTTPS protocol. This means that both before and after successful authentication, an attacker with access to the network traffic or to devices such as a proxy server would be unable to use the session identifier to hijack active user sessions. This is in line with good security practice.

In addition to using encrypted connections, it is recommended that:

- The ‘secure’ option is set on all session cookies to ensure that they are only ever sent over encrypted (HTTPS) connections.

In this case the secure flag is set which is in line with good security practice.

- The HttpOnly flag is set on all session cookies. If the HttpOnly attribute is set on a cookie, then the cookie cannot be accessed by client-side JavaScript. This measure can prevent certain client-side attacks, such as Cross-Site Scripting, from trivially capturing the cookie's value via an injected script. We recommend that [ORG] sets the HttpOnly flag by including this attribute within the relevant Set-cookie directive.

In this case the HttpOnly flag is set which is in line with good security practice.

1.1.7 Session hijack

Session is based upon cookies returned from the browser with requests. Testing indicates that it is possible to have concurrent sessions under the same user account. If an attack such as Cross-Site Scripting were to succeed, it may be possible for an attacker to intercept the session undetected. We recommend disallowing concurrent logons to help prevent session hijacking in the event of a cross-site scripting or other session stealing attack.

3. INTER-SYSTEM COMMUNICATION

An attacker may abuse the web application’s communications with other systems, in order to have it act as a proxy to other systems to which the application has privileged access. In the context of the ClientCompany Application the following scenarios were explored:

- SQL Injection – Attacks on database server

- HTML script injection (XSS) – Attacks on other users

We examined the recorded HTTP sessions for inputs likely to be relayed to other systems by the web server.

1.1.8 SQL Injection

Web based applications that allow user input into an SQL statement and do not sanitise this can be susceptible to SQL injection vulnerabilities. This may allow an attacker to:

- View data they are unauthorised to view
- Edit data they are unauthorised to edit
- Execute stored procedures they are unauthorised to execute

All user input should be parsed and sanitised to prevent the inadvertent or deliberate entry of data that could be interpreted as SQL. This means rejecting data which is not of the correct type, particularly in numeric fields, and removing any characters that have a special purpose within SQL queries such as the single quote character.

To provide confidence that the method of parsing user input adopted provides adequate protection from SQL injection attacks, we selected a sample of inputs and subjected them to common SQL inputs e.g. “0 or 1=1”(in a numeric clause) or “’or ‘’=’”(in an alpha-numeric clause).

Canonical versions of the meta-characters were also attempted where appropriate.

These attacks are designed to return errors from backend systems if the input is relayed to them without first being sanitised. Responses from the server were examined for indications of errors such as:

- Code Exceptions
- SQL statements
- Operating System errors
- Response delays
- Differential Responses

The application is not vulnerable to SQL injection at this time.

1.1.9 Cross-Site Scripting (stored)

Cross-Site scripting occurs when data passed to the Web Application can be echoed back to the user's browser without appropriate sanitization. This can allow an attacker to take control of the browser session when a user clicks on an affected link, and can also result in unauthorised access to the application via the theft of session tokens. The most common use of Cross-Site Scripting is to relay another user's cookie to the attacker through a browser script. The typical method of executing XSS attacks relies on un-sanitised input from URL strings being reflected in HTML responses to the user.

No evidence of vulnerability to XSS was discovered in the application. The application encodes all potentially dangerous characters.'

1.1.10 BREACH Attack

The login service of the application is potentially vulnerable to the BREACH (Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext) attack.

This alert was issued because the following conditions were met:

- The page content is served via HTTPS
- The server is using HTTP-level compression
- URL encoded POST input os_destination was reflected into the HTTP response body.
- HTTP response body contains a secret named atl_token

An attacker with the ability to inject partial chosen plaintext into a victim's requests and measure the size of encrypted traffic can leverage information leaked by compression to recover targeted parts of the plaintext.

BREACH is a category of vulnerabilities and not a specific instance affecting a specific piece of software. To be vulnerable, a web application must:

- Be served from a server that uses HTTP-level compression
- Reflect user-input in HTTP response bodies
- Reflect a secret (such as a CSRF token) in HTTP response bodies

We recommend the following mitigations (ordered by effectiveness):

- Disabling HTTP compression
- Separating secrets from user input
- Randomizing secrets per request
- Masking secrets (effectively randomizing by XORing with a random secret per request)
- Protecting vulnerable pages with CSRF

- Length hiding (by adding random number of bytes to the responses)
- Rate-limiting the requests

Also see here for more information: https://www.youtube.com/watch?v=xvVIZ4OyGnQ

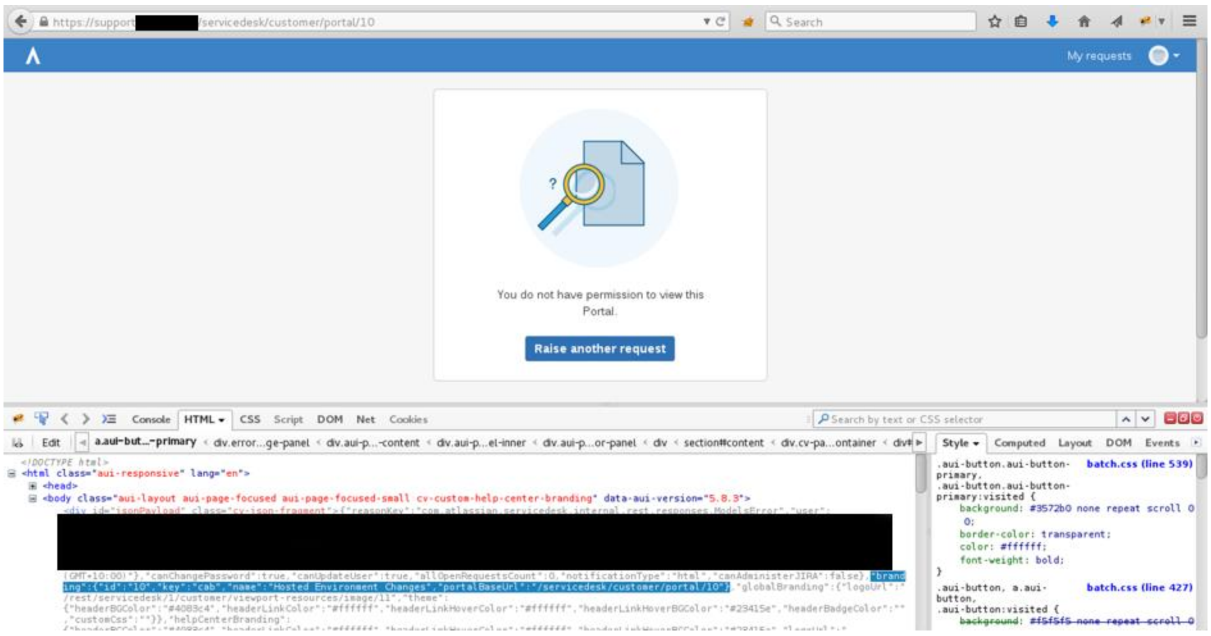
4. AUTHORISATION

It is important that user data is segregated so that users cannot view or modify other users' data.

During testing we accessed functionality which should not have been accessed directly by our test user roles:

- http://support.clientcompany.com/servicedesk/customer/portal/10

Unauthorized users can reveal the name of the portal from the source code of the site.


https://support[redacted]/servicedesk/customer/portal/10

You do not have permission to view this Portal.

```
<!DOCTYPE html>
<html class="aui-responsive" lang="en">
  <head>
  <body class="aui-layout aui-page-focused aui-page-focused-small cv-custom-help-center-branding" data-aui-version="5.8.3">
    <div id="jsonPayload" class="cv-json-fragment">{"reasonKey":"com.atlassian.servicedesk.internal.rest.responses.ModelsError","user":
[redacted]
    (GMT+10:00)"},"canChangePassword":true,"canUpdateUser":true,"allOpenRequestsCount":0,"notificationType":"html","canAdministerJIRA":false},"brand
    ing":{"id":"10","key":"cab","name":"Hosted Environment Changes","portalBaseUrl":"/servicedesk/customer/portal/10"},"globalBranding":{"logoUrl":"
    /rest/servicedesk/1/customer/viewport-resources/image/11","theme":
    {"headerBGColor":"#4083c4","headerLinkColor":"#ffffff","headerLinkHoverColor":"#ffffff","headerLinkHoverBGColor":"#23415e","headerBadgeColor":""
    ,"customCss":""}},"helpCenterBranding":
```

```
.aui-button.aui-button-          batch.css (line 539)
primary,
.aui-button.aui-button-
primary:visited {
    background: #3572b0 none repeat scroll 0
     0;
    border-color: transparent;
    color: #ffffff;
    font-weight: bold;
}
.aui-button, a.aui-              batch.css (line 427)
button,
.aui-button:visited {
    ~~background: #f5f5f5 none repeat scroll 0~~
```

Figure 6: Portal name revealed in source code

We recommend that the application code is reviewed in order to fix this unnecessary information disclosure.

5. CROSS-SITE REQUEST FORGERY

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he is currently authenticated. An attacker may force the users of a web application to execute actions of the attacker's choosing by sending a link to an action, which must be able to be accomplished with a single click. A successful CSRF exploit can compromise end user data and operations.

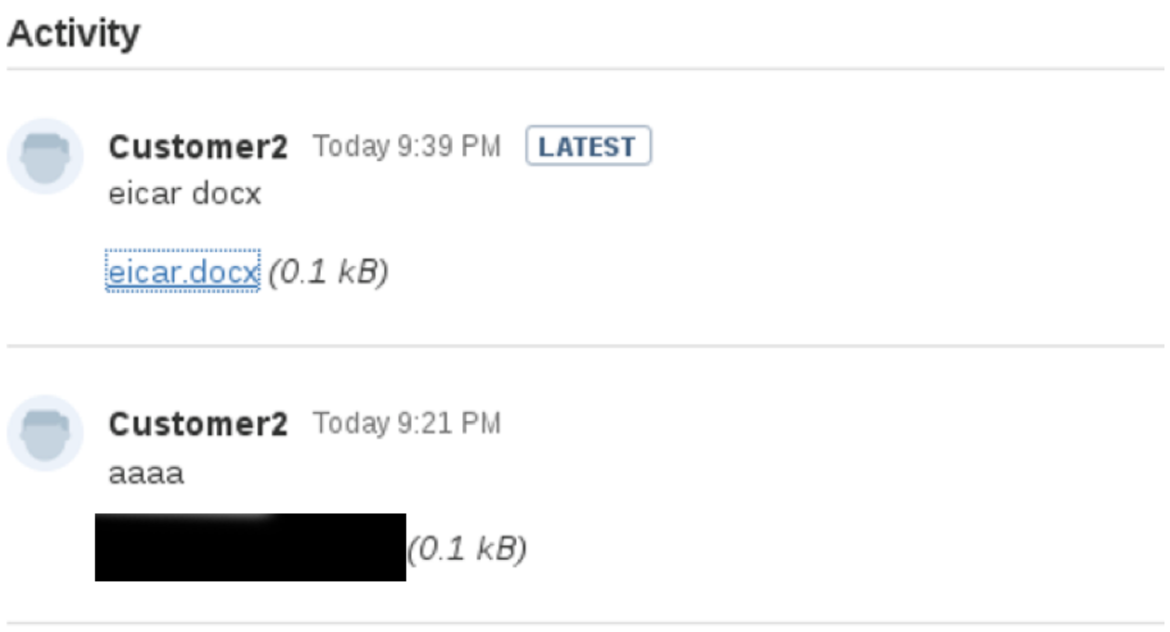
The application does not provide any protection against CSRF attacks.

The application uses cookies and headers to protect against CSRF. This is in line with good security practice.

6. APPLICATION FUNCTIONALITY

1.1.11 Insecure File Upload mechanism

The application provides functionality for file uploads. During testing we attempted to upload instances of the eicar virus test file to establish permitted file types and whether the uploaded files were scanned by an anti-virus application. The application file upload controls were found to accept all file types including executables and viruses.



Activity

Customer2 Today 9:39 PM LATEST
eicar docx

eicar.docx (0.1 kB)

Customer2 Today 9:21 PM
aaaa

[redacted] (0.1 kB)

Figure 7: Insecure File Upload mechanism

There is a risk that files uploaded to the site by an attacker could be accessed by other users, resulting in them being infected with malware. Whilst this was not exploitable within the timescale of the test, it is possible that allowing this might make the entire web server vulnerable to compromise.

We recommend that this feature be restricted to certain file types (for example Microsoft Office documents, PDFs and image files) and that these are checked by file type rather than by extension, as the latter control is trivially bypassed by renaming the file. In addition, anti-virus software should be installed on the web server to protect against malware.

1.1.12 Click-Jacking Vulnerability

The application does not set a number of HTTP headers which can be used to increase the overall security of the application. Whilst these headers will not work with all older browsers, they can be used in conjunction with most modern browsers to improve the overall security of the application. Specifically, the X-Frame-Options header is not set, which means that the application can be 'framed' with the attacker's content.

A real attacker would superimpose transparent boxes over the actual content, causing the user to enter data into the attacker's site where it could be recorded.

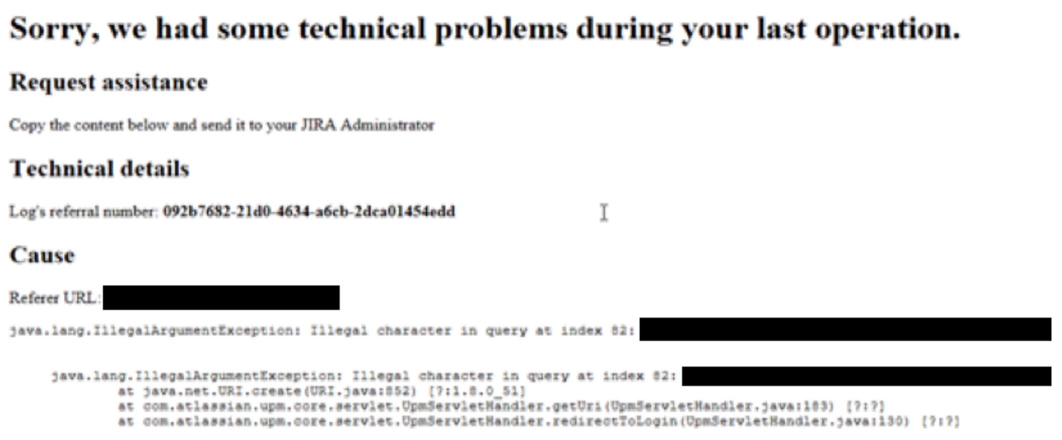
We recommend that ClientCompany review the configuration of their production web server and/or the coding of the site and make sure that it is not possible for the site to be enclosed in a frame. This can be done either by using a "frame-busting" script, or by setting appropriate HTTP headers. Setting the X-Frame-Options header to either 'DENY' or 'SAMEORIGIN' should mitigate this attack.

1.1.13 Information Disclosure

In general, the application disclosed far too much information about its contents and structure to a potential hacker.

1.1.13.1 Application error message

In places unnecessarily verbose error messages disclosed technologies in use.



Sorry, we had some technical problems during your last operation.

Request assistance

Copy the content below and send it to your JIRA Administrator

Technical details

Log's referral number: **092b7682-21d0-4634-a6cb-2dca01454edd**

Cause

Referer URL: [redacted]

```
java.lang.IllegalArgumentException: Illegal character in query at index 82: [redacted]

    java.lang.IllegalArgumentException: Illegal character in query at index 82: [redacted]
            at java.net.URI.create(URI.java:852) [?:1.8.0_51]
            at com.atlassian.upm.core.servlet.UpmServletHandler.getUri(UpmServletHandler.java:183) [?:?]
            at com.atlassian.upm.core.servlet.UpmServletHandler.redirectToLogin(UpmServletHandler.java:130) [?:?]
```

Figure 8: Error message

Error messages can provide attackers with information about application structure and also information about the success or failure of their attacks.

We recommend that only generic error messages are provided to the end user, and any technical information required for troubleshooting is logged to the server.

1.1.13.2 Http Server Type & Version Disclosure

The HTTP server in use discloses information about the versions of software in use in returned HTTP headers ‣(Apache-Coyote/1.1). This information could be of use to an attacker in attempting to target specific vulnerable web server versions.

This information could be of use to an attacker in attempting to target specific vulnerable web server versions.

We recommend removing all banner information from the web server. For Apache this can be done by setting ServerSignature Off and ServerTokens Prod in the Apache configuration file.

7. APPLICATION CONFIGURATION

1.1.14 TLS1/SSLv3 Renegotiation Vulnerability

The tested service encrypts traffic using TLS / SSL but allows a client to insecurely renegotiate the connection after the initial handshake.

A remote attacker could leverage this issue to inject an arbitrary amount of plaintext into the beginning of the application protocol stream, which could facilitate man-in-the-middle attacks if the service assumes that the sessions before and after renegotiation are from the same 'client' and merges them at the application layer.

We recommend installing a patch to mitigate this risk.

1.1.15 Cacheable HTTPS response

The application returns data from HTTPS pages which does not have appropriate cache headers set to prevent the content from being stored by client browser caches. As a result of this, sensitive information may be stored on client PCs and be vulnerable to unauthorised access.

Affected services:

- https://support.clientcompany.com/rest/servicedesk/1/servicedesk/customer/avatar/10122
- https://support.clientcompany.com/s/en_AU-sioy4b/70107/1803fe89f63e8cdc34cd0c4ef555051e/3.0.0/_/download/resources/com.atlassian.servicedesk:cv-sd-shared/img/avatarpicker/icon-downarrow.svg
- https://support.clientcompany.com/s/en_AU-sioy4b/70107/1803fe89f63e8cdc34cd0c4ef555051e/3.0.0/_/download/resources/com.atlassian.servicedesk:cv-sd-shared/img/avatarpicker/icon-image- large.svg

- https://support.clientcompany.com/s/en_AU-sioy4b/70107/1803fe89f63e8cdc34cd0c4ef555051e/3.0.0/_/download/resources/com.atlassian.servicedesk:cv-sd-shared/img/avatarpicker/icon-image.svg
- https://support.clientcompany.com/secure/useravatar

We recommend that appropriate Cache Control headers should be set on all HTTPS pages. Setting Cache-control: no-cache, no-store, must-revalidate and Pragma: no-cache should intruct browsers appropriately.

1.1.16 Missing XSS protection HTTP header for IE

The web server does not set a HTTP header which can be used to increase the overall security of the application.

Whilst this header will not work with all older browsers, it can be used in conjunction with most modern browsers to improve the overall security of the application.

We recommend reviewing the following header and consider implementing the options described: X-XSS-Protection: Setting this header to '1;mode=block' will ensure that Internet Explorer 8 and above activate their built-in anti-XSS filters.

1.1.17 No Content-Disposition header in API responses

The web server does not set a HTTP header which can be used to increase the overall security of the application.

Under certain circumstances, the improper use of this header can lead to Reflected File Download (RFD) attack.

We recommend reviewing the following header and consider implementing the options described:

Setting Content-Disposition: header to attachment; filename="api.json" (or other appropriate filename for the content type) to avoid having the browser parse the filename from the URL.

1.1.18 No forward secrecy ciphers

No forward secrecy ciphers were found on the server. Forward Secrecy offers substantial privacy and confidentiality benefits for encrypted channels accessing the Internet. Unfortunately, it's benefits are often not fully realized due to configuration errors, misconfiguring services that negatively affect the effectiveness of Forward Secrecy, or avoiding the use of it.

If forward secrecy is utilized, encrypted communications recorded in the past cannot be retrieved and decrypted should long-term secret keys or passwords be compromised in the future.

Use cryptographic parameters (like DH-parameter) that use a secure length that match to the supported keylength of your certificate (>=2048 bits or equivalent Elliptic Curves).

1.1.19 OPTIONS Method Enabled

The HTTP protocol provides a number of functions as standard. Many of these are not required for typical usage, and the availability of some may pose a risk to the web server. The detected web server was challenged with different verbs to confirm their existence.

The webserver responded and listed functions which may not be necessary in terms of functionality, such as the HTTP method OPTIONS, which details the communication options available.

We recommend that the verb lists are reviewed and unless other verbs are required, the client disables everything apart from GET and POST.

1.1.20 OSCP stapling not enabled

OSCP stapling is not set on the server. The advantage to OCSP Stapling is the improvement in speed and availability of the OCSP certificate status check.

If OCSP Stapling is used, the CA will see OCSP requests only from the web site, not the web site’s end users. Without OSCP stapling outdated CERT can be used.

We recommend checking your web server software, and change or upgrade if necessary to get OCSP Stapling support.

1.1.21 Strict transport security (HSTS) not enforced

The remote HTTPS server is not enforcing HTTP Strict Transport Security (HSTS).

HSTS is a recently developed mechanism which aims to protect HTTPS websites against downgrade attacks and simplifies protection against cookie hijacking. It allows web servers to declare that browsers should only interact with it using encrypted (HTTPS) connections, and never via clear text (HTTP) connections.

The lack of HSTS allows downgrade attacks, SSL-stripping man-in-the-middle attacks, and weakens cookie-hijacking protections.

We recommend that the server enforces this by setting the Strict-Transport-Security header.

2. Security Policy

This section covers issues which are not specifically technical, but which nonetheless could cause the application to be vulnerable to an attacker.

2.1.1 Password Quality

The application's password complexity requirements are too low. It may be possible to either guess passwords or to carry out a brute force attack on them.

Passwords should be at least 8 characters long and ideally should be required to contain a mixture of upper case letters, lower case letters, numeric and special characters.

2.1.2 Account Lockout

Accounts lock after a reasonable number of incorrect password attempts. This is in line with good security practice.

3. Tests not carried out

Certain risky or potentially disruptive tests were not carried out, specifically:

- Denial of service, flooding or “bombing”-type attacks, for obvious reasons.
- TCP, RIP and ARP protocol tampering type attacks (including fragmentation scanning). These types of attacks are highly likely to cause application disruption.
- Brute force login attempts (except on designated accounts) and account compromise attacks. These can lead to user accounts being locked, and therefore can effectively function as a Denial of Service attack.

4. Vulnerabilities and Recommendations

Information gained from the enumeration phase was also used in a manual search for known vulnerabilities.

8. Definition of risk categories

🔓	Vulnerability A flaw inherent in the security mechanism itself, or which can be reached through security safeguards, that allows for privileged access to the location, people, business processes or infrastructure, and/or allows corruption or deletion of data.
⛓	Weakness A flaw inherent in the platform or environment within which a security mechanism resides, a misconfiguration, survivability fault, usability fault, or failure to meet the requirements of the security posture.
❗	Concern The issue or vulnerability presents a low risk to the business. The threat posed by the risk should be reassessed on a regular basis as it may change. Action should still be taken to address concerns, as failure to do so could, for example, leave the ClientCompany vulnerable to a determined attacker with sufficient time and resources to exploit the vulnerability.
🚰	Information Leak A flaw inherent in the security mechanism itself, or which can be reached through

	security safeguards, which allows for privileged access to privileged or potentially sensitive information concerning data, business processes, people, or infrastructure.

Pulsar utilise OWASP framework to report risk assessment levels of Low, Medium or High which are assigned based on the following definitions (an assessment of the probability of the risk actually existing is made where it cannot be positively verified through testing and included in this assessment).

High

An issue which, if exploited, has the potential for severe impact on the confidentiality, availability and/or integrity of your information assets; the issue may be relatively straightforward to uncover or technical exploitation of this may be relatively trivial.

Medium

An issue which, if exploited, has the potential for a moderate level of impact on the confidentiality, availability and/or integrity of your information assets; discovery of the issue may require a reasonable level of technical capability and it may also be technically quite challenging to exploit or require a reasonable level of resource/time.

Low

An issue which, if exploited, has a potentially low level of impact on the confidentiality, availability and/or integrity of your information assets; it may also be technically difficult to exploit in reality or require significant resource/time allocation.

This can also be summarised by the following table as the broad application of the likelihood and impact to deriving the three levels above.

		Likelihood of Discovery/Exploitation		
		Likely	**Possible**	**Unlikely**
Impact	**Severe**	**High**	**High**	**Medium**
	Moderate	**High**	**Medium**	**Low**
	Minor	**Medium**	**Low**	**Low**

9. Observations and recommendations

Where evidence, investigation and the experience of the Pulsar technical team suggests a detected vulnerability is a false positive, it is not included in the main report.

HIGH	**BREACH Attack** This web application is potentially vulnerable to the BREACH attack. BREACH (Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext) is a category of vulnerabilities and not a specific instance affecting a specific piece of software. An attacker can leverage information leaked by compression to recover targeted parts of the plaintext. We recommend the following mitigations: • Disabling HTTP compression • Separating secrets from user input • Randomizing secrets per request • Masking secrets (effectively randomizing by XORing with a random secret per request) • Protecting vulnerable pages with CSRF • Length hiding (by adding random number of bytes to the responses) • Rate-limiting the requests See Section: BREACH Attack

MED	**Application Error Message** The application provides technical information in its error messages which could be of use to an attacker. On improperly secured web servers, error messages are left at their default settings, which may disclose technology and software versions in use Error messages can provide attackers with information about application structure and also information about the success or failure of their attacks. We recommend that only generic error messages are provided to the end-user and any technical information required for troubleshooting is logged to the server. See Section Application error message
MED	**No anti-virus on file uploads** The application contains file upload controls. These were found to accept all file types including executables and viruses. We recommend that this feature be restricted to certain file types (for example Microsoft Office documents, PDFs and image files) and that these are checked by file type rather than by extension, as the latter control is trivially by-passable by renaming the file. In addition, anti-virus software should be installed on the web server to protect against malware. See Section: Insecure File Upload mechanism

MED	**Click-Jacking Vulnerability** During testing we noted that the application did not have any defences against click-jacking attacks. This would allow attackers to superimpose transparent boxes over the actual content and capture traffic intended for the site. We recommend implementing the X-Frame-Options header and setting the value to either 'DENY' or 'SAMEORIGIN' to address this issue. See Section Click-Jacking Vulnerability
MED	**TLS1/SSLv3 Renegotiation Vulnerability** The remote service encrypts traffic using TLS / SSL but allows a client to insecurely renegotiate the connection after the initial handshake. An unauthenticated, remote attacker may be able to leverage this issue to inject an arbitrary amount of plaintext into the beginning of the application protocol stream, which could facilitate man-in-the-middle attacks if the service assumes that the sessions before and after renegotiation are from the same 'client' and merges them at the application layer. We recommend that the vendor of the software be contacted to provide a patch. See Section: TLS1/SSLv3 Renegotiation Vulnerability

LOW	**No Content-Disposition header in API responses** The web server does not set a HTTP header which can be used to increase the overall security of the application. Under certain circumstances, the improper use of this header can lead to Reflected File Download (RFD) attack. We recommend reviewing the following header and consider implementing the options described: See Section No Content-Disposition header in API responses
LOW	**Cacheable Https Response** The application returns data from HTTPS pages which does not have appropriate cache headers set to prevent the content from being stored by client browser caches. As a result sensitive information could be stored on client PCs which could be vulnerable to unauthorized access. We recommend that appropriate Cache Control headers should be set on all HTTPS pages. Setting Cache-control: no-cache, no-store, must-revalidate and Pragma: no-cache should instruct browsers appropriately. See Section: Cacheable HTTPS response

LOW	**Strict Transport Security (HSTS) Not Enforced** None of the web servers appear to enforce HTTP Strict Transport Security (HSTS). HSTS is a recently developed mechanism which aims to protect HTTPS websites against downgrade attacks and simplifies protection against cookie hijacking. It allows web servers to declare that browsers should only interact with it using encrypted (HTTPS) connections, and never via clear text (HTTP) connections. The lack of HSTS allows downgrade attacks, SSL-stripping man-in-the-middle attacks, and weakens cookie-hijacking protections. We recommend that the server enforces this by setting the "Strict-Transport-Security" header. See Section: Strict transport security (HSTS) not enforced
LOW	**Missing XSS protection HTTP header for IE** The web server does not set a HTTP header which can be used to increase the overall security of the application. Whilst this header will not work with all older browsers, it can be used in conjunction with most modern browsers to improve the overall security of the application. We recommend reviewing the following header and consider implementing the options described: X-XSS-Protection: Setting this header to '1;mode=block' will ensure that Internet Explorer 8 and above activate their built-in anti-XSS filter. See Section: Missing XSS protection HTTP header for IE

LOW	**No forward secrecy ciphers** No forward sercrecy ciphers were found on the server. Forward Secrecy offers substantial privacy and confidentiality benefits for encrypted channels accessing the Internet. Unfortunately, it's benefits are often not fully realized due to configuration errors, misconfiguring services that negatively affect the effectiveness of Forward Secrecy, or avoiding the use of it. If forward secrecy is utilized, encrypted communications recorded in the past cannot be retrieved and decrypted should long-term secret keys or passwords be compromised in the future. Use cryptographic parameters (like DH-parameter) that use a secure length that match to the supported keylength of your certificate (>=2048 bits or equivalent Elliptic Curves). See Section: No forward secrecy ciphers

LOW	**OSCP stapling not enabled** OSCP stapling is not set on the server. The advantage to OCSP Stapling is the improvement in speed and availability of the OCSP certificate status check. If OCSP Stapling is used, the CA will see OCSP requests only from the web site, not the web site's end users. Without OSCP stapling outdated CERT can be used. We recommend checking your web server software, and change or upgrade if necessary to get OCSP Stapling support. See Section: OSCP stapling not enabled
LOW	**Http Server Type And Version Disclosure** • Apache-Coyote/1.1 The HTTP server in use discloses information about the versions of software in use in returned HTTP headers. This information could be of use to an attacker in attempting to target specific vulnerable web server versions. We would recommend removing all banner information from the web server. For Apache this can be done by setting "ServerSignature Off" and "ServerTokens Prod" in the Apache configuration file. For PHP this can be done by setting "expose_php = off" in the php.ini file. For IIS http://learn.iis.net/page.aspx/938/urlscan-3-reference/ See Section: Http Server Type & Version Disclosure

LOW	**OPTIONS Method Enabled** The HTTP protocol provides a number of functions as standard. Many of these are not required for typical usage, and the availability of some may pose a risk to the web server. The detected web server was challenged with different verbs to confirm their existence. The webserver responded and listed functions which may not be necessary in terms of functionality, such as the HTTP method OPTIONS, which details the communication options available. We recommend that the verb lists are reviewed and unless other verbs are required, the client disables everything apart from GET and POST. See Section: OPTIONS Method Enabled
LOW	**Simultaneous logins allowed** The application allows for the same user to have multiple connections to it simultaneously. Whilst not a serious security issue, restricting a user to a single active session can help to mitigate some of the risks of credential sharing and stealing. We recommend ensuring that users can only have a single active session with the application. See Section: Session hijack

LOW	**Portal name information disclosure** Unauthorized users can reveal the name of the portal from the source code of following site: • http://support.clientcompany.com/servicedesk/customer/portal/10 We recommend that the application code is reviewed in order to fix this unnecessary information disclosure See Section: AUTHORISATION
LOW	**Password field with autocomplete enabled** Autocomplete is not disabled on forms within the application which process sensitive user information or user credentials. Autocomplete allows for browsers to cache information from HTML forms locally and should not be enabled on any forms which process sensitive information as this will then be cached on local PCs We recommend setting autocomplete=off in all forms which process sensitive information. See Section: Obtain Logon credentials from web browser

LOW	**User enumeration** There is a way to find out whether a user name id valid. After a number of attempts if a user exists a Captcha screen is displayed. There is no such restriction in case of invalid users. We recommend using Captcha in case of invalid users as well so the application does not disclose whether a given username is valid or invalid. See Section: Valid user enumeration through error messages
LOW	**Weak password policy** The application allows users to set weak passwords. This could allow an attacker to successfully gain unauthorised access to the application via a password guessing attack. We recommend that password length and complexity are enforced for all users of the application. These settings should be in-line with the type of information stored and processed in the application. See Section: Password Quality
LOW	**No Session Timeout** The session expiration can be extended to two weeks using the 'Remember me' option which can pose a security risk. We recommend reviewing this function. In case the user ignores the 'Remember me' function the session identifiers appear to expire after a reasonable period of time, which is in line with good security practice. See Section: Subvert logon - Injection

4.1 Issues by OWASP ASVS 3.0

The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls.

The primary aim of the OWASP Application Security Verification Standard (ASVS) Project is to normalize the range in the coverage and level of rigor available in the market when it comes to performing Web application security verification using a commercially-workable open standard. The standard provides a basis for testing application technical security controls, as well as any technical security controls in the environment, that are relied on to protect against vulnerabilities such as Cross-Site Scripting (XSS) and SQL injection. This standard can be used to establish a level of confidence in the security of Web applications[2]

The following table shows help.clientcompany.com issues by OWASP ASVS Category.

[2] Source: https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

FAIL	Critical level
FAIL	High level
FAIL	Medium level
FAIL	Low level
PASS	

High level	BREACH attack
Medium level	Application error message
Medium level	No anti-virus on file uploads
Medium level	TLS1/SSLv3 renegotiation vulnerability
Medium level	Web application potentially vulnerable to clickjacking
Low level	Cacheable HTTPS response
Low level	HTTP server type and version disclosure
Low level	Missing XSS protection HTTP header for IE
Low level	No Content-Disposition header in API responses
Low level	No forward secrecy ciphers
Low level	No session timeout
Low level	Options method enabled
Low level	OSCP stapling not enabled
Low level	Password field with autocomplete set
Low level	Portal name information disclosure
Low level	Simultaneous logins allowed
Low level	Strict transport security (HSTS) not enforced
Low level	User enumeration
Low level	Weak password policy

#	Chapter	#	Description	Result	Note
V01	Architecture, design and threat modelling	01.01	Verify that all application components are identified and are known to be needed.	**PASS**	See Scope
V02	Authentication	02.01	Verify all pages and resources by default require authentication except those specifically intended to be public (Principle of complete mediation).	**PASS**	
V02	Authentication	02.02	Verify that all password fields do not echo the user's password when it is entered.	**PASS**	
V02	Authentication	02.04	Verify all authentication controls are enforced on the server side.	**PASS**	
V02	Authentication	02.06	Verify all authentication controls fail securely to ensure attackers cannot log in.	**PASS**	
V02	Authentication	02.07	Verify password entry fields allow, or encourage, the use of passphrases, and do not prevent long passphrases/highly complex passwords being entered.	**FAIL**	See "Weak password policy"
V02	Authentication	02.08	Verify all account identity authentication functions (such as update profile, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism.	**PASS**	
V02	Authentication	02.09	Verify that the changing password functionality includes the old password, the new password, and a password confirmation.	**PASS**	

V02	Authentication	02.16	Verify that credentials are transported using a suitable encrypted link and that all pages/functions that require a user to enter credentials are done so using an encrypted link.	PASS	
V02	Authentication	02.17	Verify that the forgotten password function and other recovery paths do not reveal the current password and that the new password is not sent in clear text to the user.	PASS	
V02	Authentication	02.18	Verify that information enumeration is not possible via login, password reset, or forgot account functionality.	FAIL	See "User enumeration"
V02	Authentication	02.19	Verify there are no default passwords in use for the application framework or any components used by the application (such as 'admin/password').	PASS	
V02	Authentication	02.20	Verify that request throttling is in place to prevent automated attacks against common authentication attacks such as brute force attacks or denial of service attacks.	PASS	
V02	Authentication	02.22	Verify that forgotten password and other recovery paths use a soft token, mobile push, or an offline recovery mechanism.	PASS	
V02	Authentication	02.24	Verify that if knowledge based questions (also known as "secret questions") are required, the questions should be strong enough to protect the application.	N/A	

V02	Authentication	02.27	Verify that measures are in place to block the use of commonly chosen passwords and weak passphrases.	FAIL	See "Weak password policy"
V02	Authentication	02.30	Verify that if an application allows users to authenticate, they use a proven secure authentication mechanism.	PASS	
V02	Authentication	02.32	Verify that administrative interfaces are not accessible to untrusted parties	PASS	
V03	Session management	03.01	Verify that there is no custom session manager, or that the custom session manager is resistant against all common session management attacks.	PASS	
V03	Session management	03.02	Verify that sessions are invalidated when the user logs out.	PASS	
V03	Session management	03.03	Verify that sessions timeout after a specified period of inactivity.	FAIL	See "No session timeout"
V03	Session management	03.05	Verify that all pages that require authentication have easy and visible access to logout functionality.	PASS	
V03	Session management	03.06	Verify that the session id is never disclosed in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.	PASS	
V03	Session management	03.07	Verify that all successful authentication and re-authentication generates a new session and session id.	PASS	

V03	Session management	03.11	Verify that session ids are sufficiently long, random and unique across the correct active session base.	**PASS**	
V03	Session management	03.12	Verify that session ids stored in cookies have their path set to an appropriately restrictive value for the application, and authentication session tokens additionally set the 'HttpOnly' and 'secure' attributes	**PASS**	
V03	Session management	03.16	Verify that the application limits the number of active concurrent sessions.	**FAIL**	See "Simultaneous logins allowed"
V03	Session management	03.17	Verify that an active session list is displayed in the account profile or similar of each user. The user should be able to terminate any active session.	**FAIL**	See "Simultaneous logins allowed"
V03	Session management	03.18	Verify the user is prompted with the option to terminate all other active sessions after a successful change password process.	**FAIL**	See "Simultaneous logins allowed"
V04	Access control	04.01	Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege.	**PASS**	
V04	Access control	04.04	Verify that access to sensitive records is protected, such that only authorized objects or data is accessible to each user (for example,	**PASS**	

			protect against users tampering with a parameter to see or alter another user's account).		
V04	Access control	04.05	Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.	**PASS**	
V04	Access control	04.08	Verify that access controls fail securely.	**PASS**	
V04	Access control	04.09	Verify that the same access control rules implied by the presentation layer are enforced on the server side.	**PASS**	
V04	Access control	04.13	Verify that the application or framework uses strong random anti-CSRF tokens or has another transaction protection mechanism.	**PASS**	
V04	Access control	04.16	Verify that the application correctly enforces context-sensitive authorisation so as to not allow unauthorised manipulation by means of parameter tampering.	**FAIL**	See "Portal name information disclosure"
V05	Malicious input handling	05.01	Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.	**PASS**	
V05	Malicious input handling	05.03	Verify that server side input validation failures result in request rejection and are logged.	**PASS**	
V05	Malicious input handling	05.05	Verify that input validation routines are	**PASS**	

			enforced on the server side.		
V05	Malicious input handling	05.10	Verify that all SQL queries, HQL, OSQL, NOSQL and stored procedures, calling of stored procedures are protected by the use of prepared statements or query parameterization, and thus not susceptible to SQL injection	PASS	
V05	Malicious input handling	05.11	Verify that the application is not susceptible to LDAP Injection, or that security controls prevent LDAP Injection.	N/A	
V05	Malicious input handling	05.12	Verify that the application is not susceptible to OS Command Injection, or that security controls prevent OS Command Injection.	PASS	
V05	Malicious input handling	05.13	Verify that the application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI) when content is used that is a path to a file.	PASS	
V05	Malicious input handling	05.14	Verify that the application is not susceptible to common XML attacks, such as XPath query tampering, XML External Entity attacks, and XML injection attacks.	N/A	
V05	Malicious input handling	05.15	Ensure that all string variables placed into HTML or other web client code is either properly contextually encoded manually, or utilize templates that automatically encode contextually to ensure the application is not	PASS	

			susceptible to reflected, stored and DOM Cross-Site Scripting (XSS) attacks.	🟩	
V05	Malicious input handling	05.22	Make sure untrusted HTML from WYSIWYG editors or similar are properly sanitized with an HTML sanitizer and handle it appropriately according to the input validation task and encoding task.	N/A	
V07	Cryptography at rest	07.02	Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable oracle padding.	N/A	
V07	Cryptography at rest	07.07	Verify that cryptographic algorithms used by the application have been validated against FIPS 140-2 or an equivalent standard.	N/A	
V08	Error handling and logging	08.01	Verify that the application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id, software/framework versions and personal information	**FAIL**	See "Application error message"
V09	Data protection	09.01	Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.	**FAIL**	See "Password field with autocomplete set"
V09	Data protection	09.03	Verify that all sensitive data is sent to the server in the HTTP message body or headers	**PASS**	

			(i.e., URL parameters are never used to send sensitive data).	🟩	
V09	Data protection	09.04	Verify that the application sets appropriate anti-caching headers as per the risk of the application, such as the following: Expires: Tue, 03 Jul 2001 06:00:00 GMT Last-Modified: {now} GMT Cache-Control: no-store, no-cache, must-revalidate, max-age=0 Cache-Control: post-check=0, pre-check=0 Pragma: no-cache	**FAIL**	See "Cacheable HTTPS response"
V09	Data protection	09.09	Verify that data stored in client side storage - such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies - does not contain sensitive or PII).	N/A	
V10	Communications	10.01	Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid.	**PASS**	
V10	Communications	10.03	Verify that TLS is used for all connections (including both external and backend connections) that are authenticated or that involve sensitive data or functions, and does not fall back to insecure or unencrypted protocols. Ensure the strongest alternative is the preferred algorithm.	**FAIL**	See "BREACH attack"

V10	Communications	10.03	Verify that TLS is used for all connections (including both external and backend connections) that are authenticated or that involve sensitive data or functions, and does not fall back to insecure or unencrypted protocols. Ensure the strongest alternative is the preferred algorithm.	FAIL	See "TLS1/SSLv3 renegotiation vulnerability"
V10	Communications	10.11	Verify that HTTP Strict Transport Security headers are included on all requests and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains	FAIL	See "Strict transport security (HSTS) not enforced"
V10	Communications	10.13	Ensure forward secrecy ciphers are in use to mitigate passive attackers recording traffic.	FAIL	See "No forward secrecy ciphers"
V10	Communications	10.14	Verify that proper certification revocation, such as Online Certificate Status Protocol (OSCP) Stapling, is enabled and configured.	FAIL	See "OSCP stapling not enabled"
V10	Communications	10.15	Verify that only strong algorithms, ciphers, and protocols are used, through all the certificate hierarchy, including root and intermediary certificates of your selected certifying authority.	PASS	
V11	HTTP security configuration	11.01	Verify that the application accepts only a defined set of required HTTP request methods, such as GET and POST are accepted, and unused methods (e.g. TRACE, PUT, and DELETE) are explicitly blocked.	FAIL	See "Options method enabled"

V11	HTTP security configuration	11.02	Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1).	PASS	
V11	HTTP security configuration	11.05	Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.	FAIL	See "HTTP server type and version disclosure"
V11	HTTP security configuration	11.06	Verify that all API responses contain X-Content-Type-Options: nosniff and Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).	FAIL	See "No Content-Disposition header in API responses"
V11	HTTP security configuration	11.07	Verify that the Content Security Policy V2 (CSP) is in use in a way that either disables inline JavaScript or provides an integrity check on inline JavaScript with CSP noncing or hashing.	FAIL	See "Web application potentially vulnerable to clickjacking"
V11	HTTP security configuration	11.08	Verify that the X-XSS-Protection: 1; mode=block header is in place.	FAIL	See "Missing XSS protection HTTP header for IE"
V16	File and resources	16.01	Verify that URL redirects and forwards only allow whitelisted destinations, or show a warning when redirecting to potentially untrusted content.	PASS	
V16	File and resources	16.02	Verify that untrusted file data submitted to the application is not used directly with file I/O commands,_particularly to protect against path traversal, local file include, file mime type, and OS command injection vulnerabilities.	PASS	

V16	File and resources	16.03	Verify that files obtained from untrusted sources are validated to be of expected type and scanned by antivirus scanners to prevent upload of known malicious content.	**FAIL**	See "No anti-virus on file uploads"
V16	File and resources	16.04	Verify that untrusted data is not used within inclusion, class loader, or reflection capabilities to prevent remote/local file inclusion vulnerabilities.	**PASS**	
V16	File and resources	16.05	Verify that untrusted data is not used within cross-domain resource sharing (CORS) to protect against arbitrary remote content.	**PASS**	
V16	File and resources	16.08	Verify the application code does not execute uploaded data obtained from untrusted sources.	**PASS**	
V16	File and resources	16.09	Do not use Flash, Active-X, Silverlight, NACL, client-side Java or other client side technologies not supported natively via W3C browser standards.	**PASS**	
V17	Mobile	17.01	Verify that ID values stored on the device and retrievable by other applications, such as the UDID or IMEI number are not used as authentication tokens.	N/A	
V17	Mobile	17.02	Verify that the mobile app does not store sensitive data onto potentially unencrypted shared resources on the device (e.g. SD card or shared folders).	N/A	

V17	Mobile	17.03	Verify that sensitive data is not stored unprotected on the device, even in system protected areas such as key chains.	N/A	
V17	Mobile	17.07	Verify that the application sensitive code is laid out unpredictably in memory (For example ASLR).	N/A	
V17	Mobile	17.09	Verify that the app does not export sensitive activities, intents, content providers etc., for other mobile apps on the same device to exploit.	N/A	
V17	Mobile	17.11	Verify that the appÕs exposed activities, intents, content providers etc. validate all inputs.	N/A	
V18	Web services	18.01	Verify that the same encoding style is used between the client and the server.	**PASS**	
V18	Web services	18.02	Verify that access to administration and management functions within the Web Service Application is limited to web service administrators.	N/A	
V18	Web services	18.03	Verify that XML or JSON schema is in place and verified before accepting input.	N/A	
V18	Web services	18.04	Verify that all input is limited to an appropriate size limit.	N/A	
V18	Web services	18.05	Verify that SOAP based web services are compliant with Web Services-Interoperability (WS-I) Basic Profile at minimum.	N/A	

V18	Web services	18.06	Verify the use of session-based authentication and authorization. Please refer to sections 2, 3 and 4 for further guidance. Avoid the use of static "API keys" and similar.	PASS	
V18	Web services	18.07	Verify that the REST service is protected from Cross-Site Request Forgery.	PASS	
V19	Configuration	19.01	All components should be up to date with proper security configuration(s) and version(s). This should include removal of unneeded configurations and folders such as sample applications, platform documentation, and default or example users.	PASS	