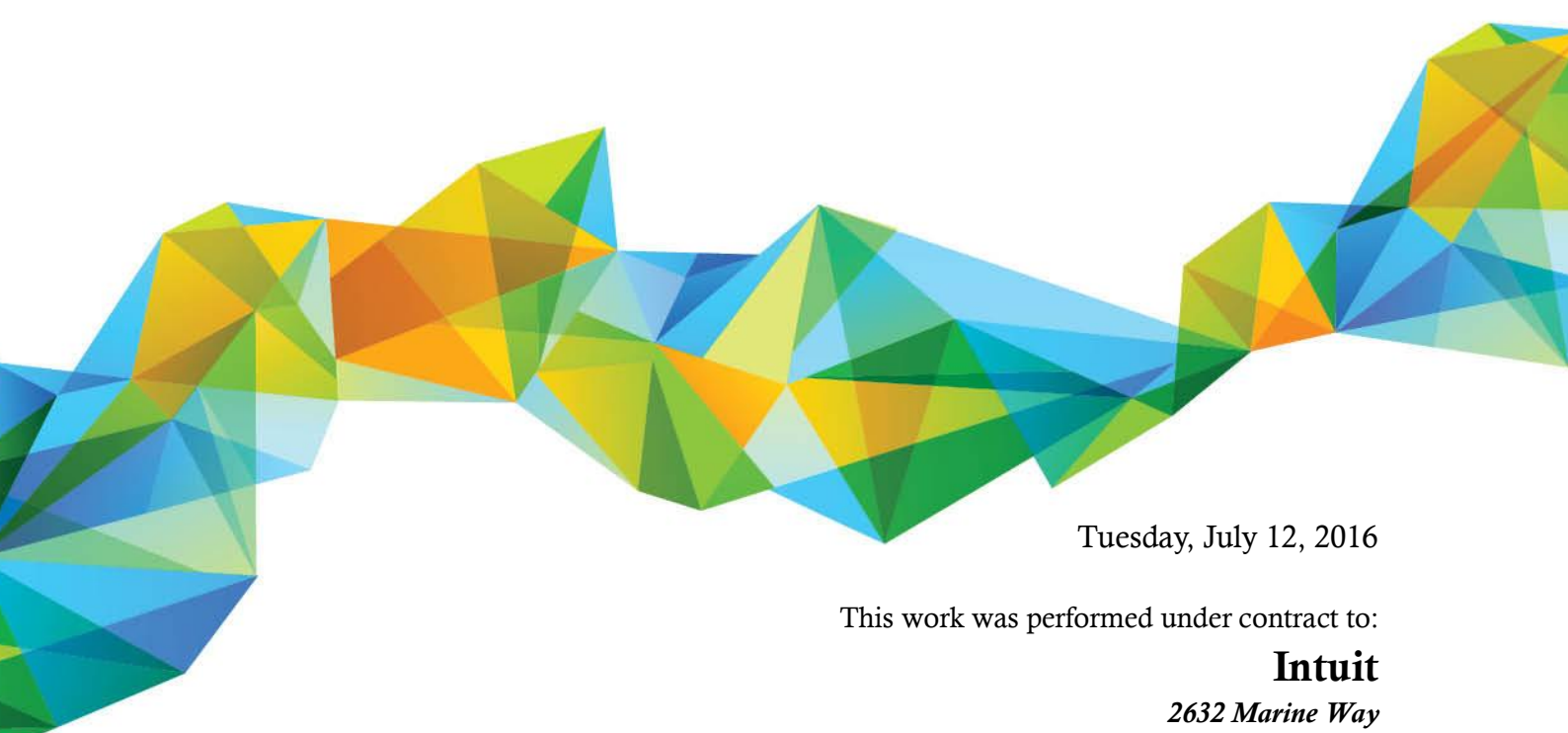




Intuit

StratPad

QuickBooks API Scan



Tuesday, July 12, 2016

This work was performed under contract to:

Intuit

2632 Marine Way

Mountain View, CA 94043

For more information contact:

Chandrani Dey

Security Consultant

Dzung Pham

Security Consultant

Pooja Garg

Managing Consultant

Copyright © 2016 by Cigital, Inc.® All rights reserved. No part or parts of this documentation may be reproduced, translated, stored in any electronic retrieval system, transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior written permission of the copyright owner. Cigital, Inc. retains the exclusive title to all intellectual property rights relating to this documentation.

The information in this documentation is subject to change without notice and should not be construed as a commitment by Cigital, Inc. Cigital, Inc. makes no representations or warranties, express or implied, with respect to the documentation and shall not be liable for any damages, including any indirect, incidental, consequential damages (such as loss of profit, loss of use of assets, loss of business opportunity, loss of data or claims for or on behalf of user's customers), that may be suffered by the user.

Cigital and the Cigital logo are trademarks of Cigital, Inc. Other brands and products are trademarks of their respective owner(s).

Cigital, Inc.
21351 Ridgetop Circle
Suite 400
Dulles, VA 20166
United States

+ 1 703.404.9293

www.cigital.com

Table of Contents

TABLE OF CONTENTS	3
DOCUMENT REVISION HISTORY	4
CONTACTS	4
EXECUTIVE SUMMARY	5
1 INTRODUCTION.....	6
1.1 OBJECTIVE	6
1.2 SCOPE.....	6
2 METHODOLOGY	7
2.1 ASSESSMENT TYPE	7
2.2 RISK ASSESSMENT METHODOLOGY	7
3 FINDINGS	9
3.1 SUMMARY OF FINDINGS.....	9
3.2 FINDING DETAILS	9
3.2.1 <i>High Priority Findings</i>	9
3.2.2 <i>Medium Priority Findings</i>	12
3.2.3 <i>Low Priority Findings</i>	15
3.2.4 <i>Minimal Priority Findings</i>	20
ABOUT CIGITAL, INC.	23

Document Revision History

Version	Modification	Date	Author
1.0	Security report	07/11/2016	Chandrani Dey
1.1	Security Report Review	07/11/2016	Dzung Pham

Contacts

Contact	Title	Organization	Phone #	Email Address
Pooja Garg	Managing Consultant	Cigital, Inc.	703-404-9293 x2215	pgarg@cigital.com
Matt Hale	Project manager	Cigital, Inc.	703-404-9293 x5228	mhale@cigital.com
Dzung Pham	Security Consultant	Cigital, Inc.	703-404-9293 x5289	dpham@cigital.com
Chandrani Dey	Security Consultant	Cigital, Inc.	703-404-9293	cdey@cigital.com

Executive Summary

Intuit has engaged Cigital, Inc. to perform a QuickBooks API Scan on StratPad Web Application. StratPad is a business planning web application that makes business plans easy. The purpose of this assessment was to assess the overall security posture of the application from a black-box perspective. This includes determining the application's ability to resist common attack patterns and identifying vulnerable areas in the internal or external interfaces that may be exploited by a malicious user.

During the assessment, Cigital identified the following:

- 1 *High priority* finding
- 2 *Medium priority* findings
- 3 *Low priority* findings
- 2 *Minimal priority* findings

While performing the assessment of StratPad Web Application, Cigital identified that security controls were effective in resisting common attack patterns like:

- Input Validation Attacks

1 Introduction

Intuit engaged Cigital, Inc. to perform a QuickBooks API Scan on StratPad Web Application beginning on July 07, 2016 and ending on July 11, 2016. StratPad is a business planning web application that makes business plans easy.

1.1 Objective

The objective of this assessment was to assess the overall security posture of the application from a black-box perspective. This includes determining the application's ability to resist common attack patterns and identifying vulnerable areas in the internal or external interfaces that may be exploited by a malicious user.

1.2 Scope

The scope of this engagement was limited to components and interfaces specific to StratPad Web Application. The following URL(s) were considered in-scope:

- <https://staging.stratpad.com>
- <https://jstratpad.appspot.com>

2 Methodology

2.1 Assessment Type

Cigital was engaged to perform a time-boxed manual security assessment against the target application. This assessment involved a deep automated scan using automated scanning tools to discover common vulnerabilities, as well as manual testing. Manual testing includes validation of all issue types covered under the automated scan as well as checks for problems not typically found by automated scanners such as authentication, authorization and business logic flaws.

2.2 Risk Assessment Methodology

The severity assigned to each vulnerability was calculated using the NIST 800-30 r1 standard. This standard determines the risk posed by application based on the likelihood an attacker exploits the vulnerability and the impact that it would have on the business.

Likelihood

The difficulty of exploiting the described security vulnerability includes required skill level and the amount of access necessary to visit the element susceptible to the vulnerability. The difficulty is rated with the following values:

- **Critical:** An attacker is almost certain to initiate the threat event.
- **High:** An untrained user could exploit the vulnerability or the vulnerability is very obvious and easily accessible.
- **Medium:** The vulnerability requires some hacking knowledge or access is restricted in some way.
- **Low:** Exploiting the vulnerability requires application access, significant time, resource or a specialized skillset.
- **Minimal:** Adversaries are highly unlikely to leverage the vulnerability.

Impact

The impact the vulnerability would have on the organization if it were successfully exploited is rated with the following values:

- **Critical:** The issue causes multiple severe or catastrophic effects on organizational operations, organizational assets or other organizations.
- **High:** Exploitation produces severe degradation in mission capability to the point that the organization is not able to perform primary functions or results in damage to organizational assets.
- **Medium:** Threat events trigger degradation in mission capability to an extent the application is able to perform its primary functions, but their effectiveness is reduced and there may be damage to organizational assets.
- **Low:** Successful exploitation has limited degradation in mission capability; the organization is able to perform its primary functions, but their effectiveness is noticeably reduced and may result in minor damage to organizational assets.

- **Minimal:** The threat could have a negligible adverse effect on organizational operations or organizational assets.

Severity

The vulnerability severity is determined using the likelihood and impact weights in the following table:

		Impact				
		<i>Minimal</i>	<i>Low</i>	<i>Medium</i>	<i>High</i>	<i>Critical</i>
Likelihood	<i>Critical</i>	Minimal	Low	Medium	High	Critical
	<i>High</i>	Minimal	Low	Medium	High	Critical
	<i>Medium</i>	Minimal	Low	Medium	Medium	High
	<i>Low</i>	Minimal	Low	Low	Low	Medium
	<i>Minimal</i>	Minimal	Minimal	Minimal	Low	Low

3 Findings

3.1 Summary of Findings

Finding	Likelihood	Impact	Severity	Status
Clickjacking (aka UI Redressing)	Medium	Critical	High	Open
Excessive Session Timeout Duration	Medium	High	Medium	Open
Weak SSL Ciphers	Medium	High	Medium	Open
Login Page Username Enumeration	High	Low	Low	Open
Cacheable SSL Pages	Low	Low	Low	Open
Vulnerable Server Version	Low	Low	Low	Open
Hidden Directory Detected	High	Minimal	Minimal	Open
Verbose Server Banner	High	Minimal	Minimal	Open

3.2 Finding Details

3.2.1 High Priority Findings

3.2.1.1 Clickjacking (aka UI Redressing)

Description:

The application is vulnerable to clickjacking, which occurs when the application fails to ensure it is the top-level page being rendered in a web browser. Clickjacking occurs when a malicious website presents a page containing an iframe that incorporates legitimate content beneath an invisible malicious page. The attacker-controlled page contains UI elements aligned in such a way that when the user thinks they are clicking on a UI element in the framed page, they are actually clicking on the invisible malicious element floating above it. By employing this technique the victim's click has been "hijacked".

Clickjacking is used to trick a user into thinking they are clicking on a UI element on a visible page, when the click is actually captured by an entirely different element on an invisible layer. This results in the victim unintentionally carrying out an action on behalf of the attacker. Examples include allowing access to the client computer's hardware features (e.g., enabling the webcam) or submitting a form to initiate an unintended transaction (such as "liking" a Facebook page, initiating a money transfer, etc.).

Real-world examples of clickjacking include exploits on embedded Twitter "Follow" buttons and Facebook "Like" buttons. In these attacks, the malicious website contains valid "Follow" or "Like" buttons, renders them transparent, and adds a decoy element underneath the button. The user believes they are clicking on the visible element, while their click is actually performing a "like" or "follow" action on Facebook or Twitter, respectively.

Instances:

1. <https://staging.stratpad.com>
2. <https://staging.stratpad.com/stratweb.html#nav/3/5/0>
3. <https://staging.stratpad.com/stratweb.html#nav/6/5/0>

Note: This finding is systemic throughout the application.

Steps To Reproduce:

1. Create an iframe by copying the following code in a HTML file

```
<!DOCTYPE html>
<html>
<body>
<p>Site is Vulnerable to Click-Jacking.</p>
<iframe src="https://staging.stratpad.com" width = "900" height = "600">
</iframe>
</body>
</html>
```

2. Open the HTML file in a browser.
3. Observe that the basic authentication pop up allow us to authenticate to the application within an iframe.
4. Observe that the pre-authenticated page of the application gets loaded into iframe.
5. Login to the application.
6. Observe that post-authenticated page of the application successfully gets loaded into iframe.

Evidence:

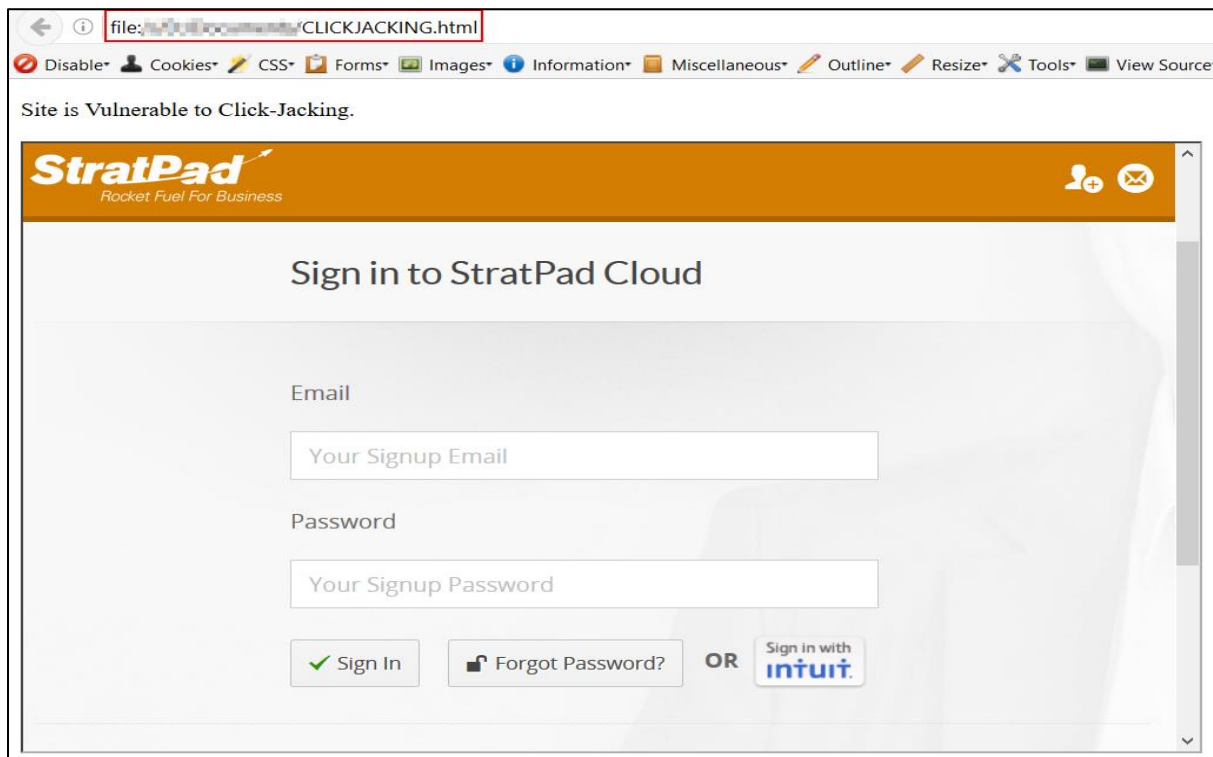


Figure 1: Application successfully loads pre-authenticated application page into iframe

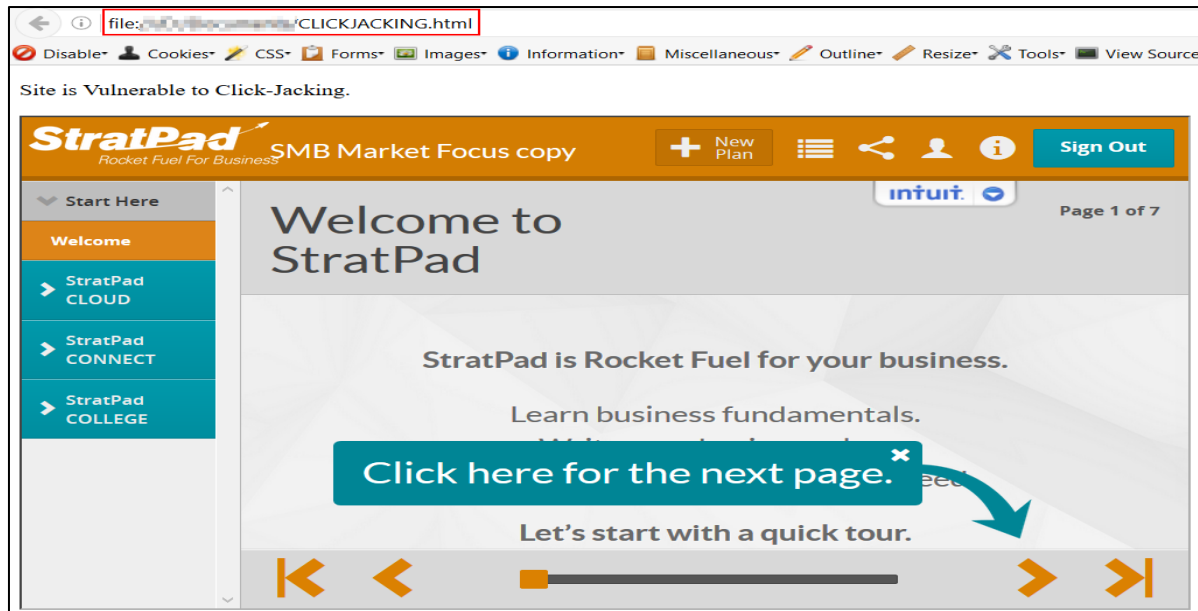


Figure 2: Application successfully loads post-authenticated application page into iframe

Likelihood: Medium

Impact: Critical

Remediation:

A combination of frame-busting JavaScript code and the X-FRAME-OPTIONS and Content-Security-Policy headers provide adequate protection against clickjacking. Frame-busting JavaScript provides significant protection against non-expert attackers; it uses CSS to show the web page only if the top object is equal to self (i.e. the page is not being loaded into a frame).

```
<style>
html { display:none; }
</style>
<script>
if (self == top) {
document.documentElement.style.display = 'block';
} else {
top.location = self.location;
}
</script>
```

In addition to frame-busting code, two HTTP headers should be employed to protect against clickjacking. The X-FRAME-OPTIONS header supports two options that help protect against clickjacking. The "DENY" option prevents supporting browsers from rendering the page if it resides inside any iframe. The "SAMEORIGIN" option prevents browsers from rendering the page in an iframe on all pages hosted outside the framed page's domain.

The Content-Security-Policy header, which specifies whitelists of trusted content that the browser may load along with page content, can be set to prevent a page from being framed using the "frame-ancestors" option:

```
Content-Security-Policy: frame-ancestors 'none'
```

These headers are supported to varying degrees by modern browsers, and should be employed along with frame-busting code to provide protection across as many browsers and browser versions as possible.

3.2.2 Medium Priority Findings

3.2.2.1 Excessive Session Timeout Duration

Description:

The application does not log the user out after a reasonable period of inactivity. Inactivity timeout periods vary depending on the sensitivity of the data and functionality the application contains, but idle sessions longer than 15-30 minutes are typically considered vulnerable.

An attacker's ability to successfully hijack a victim's session increases the longer an idle user's session remains valid. Longer session timeouts also prevent server memory from being released, resulting in potential denial of service conditions when an attacker initiates a large number of sessions in a sufficiently short period of time.

Instances:

1. <https://jstratpad.appspot.com/logIn>
 - a. Cookie: JSESSIONID

Steps To Reproduce:

1. Configure your browser to use a proxy tool such as Burp Suite.
2. Navigate to the following URL <https://staging.stratpad.com>
3. Authenticate to the application using basic authentication.
4. Login to the application.
5. Make a note of the time in which the request was made.
6. After 30 minutes of inactivity, send a new request to the application.
7. Observe that the application does not log us out and responds successfully to requests.

Evidence:

#	Host	Method	URL	Status	Length	Time	Comment
5143	https://js.appcenter.intuit.com	GET	/Content/IA/intuit.pp.anywhere-1.3.0.js	304	197	07:53:34 11 J...	
5144	https://staging.stratpad.com	GET	/stratweb.html	200	13131	07:54:19 11 J...	
5145	https://staging.stratpad.com	GET	/css/style.1606271231.css	200	555782	07:54:24 11 J...	
5146	https://staging.stratpad.com	GET	/js/1606271231/lb/require/require.js	200	15858	07:54:24 11 J...	
5147	https://staging.stratpad.com	GET	/images/logo@2x.png	304	200	07:54:33 11 J...	
5148	https://staging.stratpad.com	GET	/js/1606271231/main.js	200	1685962	07:54:33 11 J...	
5149	https://js.appcenter.intuit.com	GET	/Content/IA/intuit.pp.anywhere-1.3.0.js	304	197	07:54:41 11 J...	
5150	https://rest.stratpad.com	GET	/_ah/channel/jsapi.js			07:54:41 11 J...	
5151	https://maxcdn.bootstrapcdn.c...	GET	/bootstrap/3.3.4/js/bootstrap.min.js	304	426	07:54:41 11 J...	
5209	https://staging.stratpad.com	GET	/stratweb.html	200	13131	08:46:32 11 J...	

Request

Response

Raw

Headers

Hex

HTML

Render

```

<body>
<header id="pageHeader">
  <div id="fullLoader"></div>
  <div id="stratFileTitle"></div>
  <div class="l-grid l-col-50 group">
    <div>
      <a href="/" id="pageLogo">StratPad</a>
      <i class="icon-misc-cloud"></i>
    </div>
    <div>
      <ul id="navMenubar"></ul>
      <a href="#" id="stratPadSignout" class="blue-btn"></a>
    </div>
  </div>
</header>

```

Figure 3: The session remains active even after half an hour of inactivity

Likelihood: Medium

Impact: High

Remediation:

Configure the application server to log out after a sufficiently short idle period and redirect the user to a splash page or the login page after a certain period of inactivity has passed. No prior authenticated user data or functionality should continue to be displayed after the timeout occurs. Determine session timeout settings which sufficiently protect end users and the application while not leaving the system unusable due to frequent login requirements. Session timeouts of 15-30 minutes are common for most web applications and vary depending on the sensitivity of the information available during each session.

3.2.2.2 Weak SSL Ciphers

Description:

The server-side SSL/TLS endpoint is configured to allow weak SSL/TLS cipher suites. These cipher suites have proven cryptographic flaws that can allow an attacker to decrypt or modify traffic. These weak cipher suites include the following:

- Cipher suites that use RC4 for encryption; RC4 contains several known weaknesses. In particular, an attacker that can gather a large number of ciphertexts that contain the same plaintext encrypted using different keys may be able to recover the plaintext (the typical target of this attack would be a cookie that is sent in multiple SSL/TLS sessions).
- Cipher suites that use block ciphers (e.g. AES, 3DES) in CBC mode; these are vulnerable to the BEAST attack if SSL 3.0 or TLS 1.0 are supported. Even if newer versions of TLS are also supported by the server, older client software might establish SSL 3.0 or TLS 1.0 connections. Additionally, an attacker may be able to use the POODLE attack to downgrade the connection to SSL 3.0 or TLS 1.0 even if both the client and the server support newer versions of TLS. BEAST allows an attacker on the same network as an end user, who can inject code into any site open in the user's browser, to decrypt cookies (or other sensitive data that is part of each request) for the vulnerable site.

A server-side SSL/TLS endpoint that supports weak ciphers could allow an attacker to read or modify traffic sent in SSL/TLS connections with that endpoint.

Instances:

1. <https://staging.stratpad.com>

Steps To Reproduce:

1. Download the SSL scanning tool sslyze from the below mentioned URL:
<https://github.com/nabla-c0d3/sslyze>
2. Navigate to the directory where sslyze is present, open command prompt and run the following command:

```
sslyze.exe --sslv2 --sslv3 --tlsv1 --tlsv1_1 --tlsv1_2 --  
hide_rejected_ciphers staging.stratpad.com
```

3. Observe the output and note that the application domain supports deprecated protocols such as TLSV1 and SSLV3 and weak ciphers such as RC4-SHA.

Note: Application is also vulnerable to POODLE vulnerabilities as it supports SSLv3.

Evidence:

SCAN RESULTS FOR STAGING.STRATPAD.COM:443 - 192.241.237.47:443			

* TLSV1_2 Cipher Suites:			
Accepted:			
	SEED-SHA	-	128 bits
	RC4-SHA	-	128 bits
	CAMELLIA128-SHA	-	128 bits
	AES128-SHA256	-	128 bits
	AES128-SHA	-	128 bits
	AES128-GCM-SHA256	-	128 bits
	ECDHE-RSA-DES-CBC3-SHA	ECDH-256 bits	112 bits
	DES-CBC3-SHA	-	112 bits
* TLSV1_1 Cipher Suites:			
Accepted:			
	SEED-SHA	-	128 bits
	RC4-SHA	-	128 bits
	CAMELLIA128-SHA	-	128 bits
	AES128-SHA	-	128 bits
	ECDHE-RSA-DES-CBC3-SHA	ECDH-256 bits	112 bits
	DES-CBC3-SHA	-	112 bits
* TLSV1 Cipher Suites:			
Accepted:			
	SEED-SHA	-	128 bits
	RC4-SHA	-	128 bits
	CAMELLIA128-SHA	-	128 bits
	AES128-SHA	-	128 bits
	ECDHE-RSA-DES-CBC3-SHA	ECDH-256 bits	112 bits
	DES-CBC3-SHA	-	112 bits
* SSLV3 Cipher Suites:			
Accepted:			
	SEED-SHA	-	128 bits
	RC4-SHA	-	128 bits
	CAMELLIA128-SHA	-	128 bits
	AES128-SHA	-	128 bits
	ECDHE-RSA-DES-CBC3-SHA	ECDH-256 bits	112 bits
	DES-CBC3-SHA	-	112 bits

Figure 4: Application domain supports deprecated protocols such as TLSV1 and SSLV3 and weak ciphers such as RC4-SHA

Likelihood: Medium

Impact: High

Remediation:

The server-side TLS endpoint's configuration should be updated to allow only TLSv1.2 connections with cipher suites that use:

- Ephemeral Diffie-Hellman for key exchange (optionally, allow RSA for key exchange if necessary for supporting some clients)
- Block ciphers with key lengths of at least 128 bits (AES-128 and AES-256; optionally allow 3DES with 112-bit keys if necessary for supporting some clients)
- Block ciphers in GCM mode. Note: If CBC mode must be allowed for supporting some clients, use only CBC mode cipher suites that use the SHA2 family of hash functions (SHA256, SHA384, SHA512)

Note that all modern browsers support TLSv1.2.

3.2.3 Low Priority Findings

3.2.3.1 Login Page Username Enumeration

Description:

The application's login functionality returns different responses depending on whether the entered username is valid or not. The difference in responses may be as straightforward as "Invalid password" versus "User [username] does not exist". Responses may also be more subtle. The application may return a generic failed login message that differs slightly depending on whether a valid username and invalid password is supplied versus an invalid username, or the application may display a single error message, but some other (potentially non-visible) aspect of the response's content is different.

An attacker can abuse this design to compile a list of valid users through automated brute force guessing attempts. Simple scripts can be found or written that automatically guess usernames by submitting them to the login function with a dummy password and observing the server's responses.

Username enumeration provides an attacker with one of two pieces of information required to authenticate to the application. By automating guesses, an attacker is able to retrieve a large list of valid usernames for an application. Once the attacker has a list of valid usernames, they can begin guessing passwords in an attempt to steal credentials and impersonate other users. Password guessing attempts may be done manually, or via automated means depending on what login anti-automation mechanisms (if any) the application has in place. Valid usernames may also be used in phishing exercises as well as large-scale account lockout denial of service attacks.

Instances:

1. <https://jstratpad.appspot.com/login>

Steps To Reproduce:

1. Navigate to the following URL <https://staging.stratpad.com>
2. Authenticate to the application using basic authentication.
3. Try to login with an existing email address and incorrect password.
4. Now try to login again with a non-existing email address and any password.
5. Note the response returned by application for both the actions.
6. Observe that the application returns different responses depending on whether the specified email address exists or not.

Evidence:

The screenshot shows a web browser at the URL <https://staging.stratpad.com>. The page features the StratPad logo with the tagline "Rocket Fuel For Business". A red error message box at the top states: "Sorry, that email and password combination does not match any in our system. Please try again, or sign up." Below this, the heading "Sign in to StratPad Cloud" is displayed. The login form includes an "Email" field containing "yarrjvcr@sharklasers.com" and a "Password" field with the placeholder text "Your Signup Password". At the bottom of the form, there are three buttons: "Sign In" (with a green checkmark icon), "Forgot Password?", and "Sign in with intuit" (with the Intuit logo). The "Sign In" button is highlighted with a green checkmark.

Figure 5: Application response for an existing email address

The screenshot shows the same web browser at the URL <https://staging.stratpad.com>. The page features the StratPad logo with the tagline "Rocket Fuel For Business". A red error message box at the top states: "That email is not in our system. Please try again, or sign up." Below this, the heading "Sign in to StratPad Cloud" is displayed. The login form includes an "Email" field containing "abc@sharklasers.com" and a "Password" field with masked characters (dots). At the bottom of the form, there are three buttons: "Sign In" (with a green checkmark icon), "Forgot Password?", and "Sign in with intuit" (with the Intuit logo). The "Sign In" button is highlighted with a green checkmark.

Figure 6: Application response for a non-existing address

Likelihood: High

Impact: Low

Remediation:

The application should return the same response whether or not the supplied username is associated with a valid account. An example response for all combinations of invalid credentials could be, "The username and password entered do not match."

3.2.3.2 Cacheable SSL Pages

Description:

Application pages served over SSL are cached locally on disk. This occurs as a result of missing HTML cache control tags and missing or misconfigured cache control headers in HTTP responses from the server.

An attacker with local access to a user's web browser may be able to retrieve cached copies of the pages that the user previously accessed, exposing any stored sensitive data.

Instances:

1. <https://staging.stratpad.com/stratweb.html>
 - a. Header: Cache-Control
2. <https://jstratpad.appspot.com/stratfiles/5708998892322816/serviceProviders?category=Consultant>
 - a. Header: Cache-Control
3. <https://jstratpad.appspot.com/stratfiles/5708998892322816/serviceProviders?category=Bookkeeper>
 - a. Header: Cache-Control
4. <https://jstratpad.appspot.com/stratfiles/5708998892322816/serviceProviders?category=Accountant>
 - a. Header: Cache-Control

Note: This finding is systemic throughout the application.

Steps To Reproduce:

1. Configure your browser to use a proxy tool such as Burp Suite.
2. Navigate to the following URL <https://staging.stratpad.com>
3. Authenticate to the application using basic authentication.
4. Login to the application.
5. Navigate to one of the URLs mentioned in the "Instances" sectioned above.
6. Intercept the response in Burp Suite.
7. Observe that Cache-Control header is not set to "no-store" and "no-cache" by the application.

Evidence:

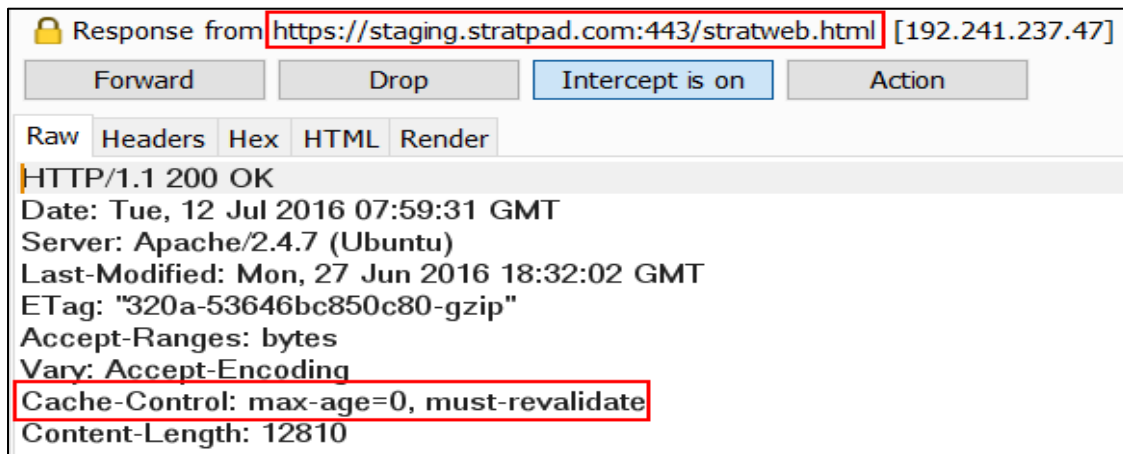


Figure 7: The Cache-Control header is not set to 'no-store' and 'no-cache'

Likelihood: Low

Impact: Low

Remediation:

The web server should be configured to set cache behavior on all pages. To prevent a page from being cached, the Cache-Control directive must be set to no-store. This is the most secure of the cache-control directives. It instructs the browser not to cache the page and not store the page in its cache folder. This directive should be used for all sensitive pages. With this set, the application will have the greatest control possible over how its pages will be cached. Other directives, such as Pragma: no-cache and Expires HTTP headers should be set as well (Note: these headers do not guarantee that a browser will not store the data in its cache folder, but are honored in certain browsers).

Note: The (misleadingly named) "no-cache" directive instructs the browser to revalidate with the server before serving the page from the cache. The browser may still store the page in its cache. In addition, some modern browsers have been modified to implement the "no-cache" directive like the "no-store" directive. To be on the safer side, developers can use both "no-cache" and "no-store" when serving sensitive pages.

3.2.3.3 Vulnerable Server Version

Description:

The web or application server has known published vulnerabilities. Published vulnerabilities have a greater likelihood of exploitation by attackers due to readily available proof-of-concept code that exploits the issue, or integrates the exploits into freely available testing tools. Depending on the nature of these vulnerabilities, this could allow an attacker to compromise the server and any data stored within.

Usage of a web or application server platform with known vulnerabilities opens up any data stored or application hosted on the server to exploitation.

Instances:

1. <https://staging.stratpad.com>

Steps To Reproduce:

1. Navigate to the URL mentioned in the "Instances" section above.
2. Click on 'cancel' on the basic authentication pop-up page.
3. Observe in response that remote host is using vulnerable "Apache 2.4.7" server version which is vulnerable to DOS attack.
4. For more information about the public CVE entry for this server version refer to the below mentioned link:
https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-161847/year-2014/Apache-Http-Server-2.4.7.html

Evidence:



Figure 8: Application returns vulnerable "Apache 2.4.7" version which is vulnerable to DOS attack.

Likelihood: Low

Impact: Low

Remediation:

Upgrade or patch the web or application server to a version which does not currently have any known vulnerabilities.

Furthermore, as soon as it is identified that the server version is vulnerable to an attack, preventative measures should be taken to mitigate the vulnerability until an upgrade or patch is released.

3.2.4 Minimal Priority Findings

3.2.4.1 Hidden Directory Detected

Description:

The hidden directory enumeration issue exists when the server responds with a '403 Forbidden' error while trying to access a valid application directory. During the assessment Cigital was able to detect multiple hidden directories by viewing the '403 Forbidden' response from the server. An attacker will try to access multiple directories within the application by guessing their names or launching a brute force attack. The server will typically respond with a '404 Not Found' error if a directory does not exist, however if a valid directory exists, the server responds with '403 Forbidden' error. The attacker can use this difference in the response to enumerate the application directories and file structure.

The presence of hidden directories allows an attacker to gather information regarding the file and directory structure of the application by viewing the '403 Forbidden' server response. An attacker can list the server directories by studying the different error responses that are thrown by the application server. This information can result in mapping of the subdirectories, files, and subsequently the entire application directory structure. It may also help the attacker identify the technology stack used in the application by studying the presence or absence of technology-specific server directories.

Instances:

1. <https://staging.stratpad.com/fonts/>
2. <https://staging.stratpad.com/css/>
3. <https://staging.stratpad.com/javascript/>
4. <https://staging.stratpad.com/cgi-bin/>
5. <https://staging.stratpad.com/icons/small/>
6. <https://staging.stratpad.com/icons/>
7. <https://staging.stratpad.com/reference/>
8. <https://staging.stratpad.com/js/>
9. <https://staging.stratpad.com/images/>

Steps To Reproduce:

1. Configure your browser to use a proxy tool such as Burp Suite.
2. Navigate to one of the URL mentioned in the "Instances" section above.
3. Intercept the response in Burp Suite and observe that application responds with '403' Forbidden error which confirms existence of the directory.

Evidence:

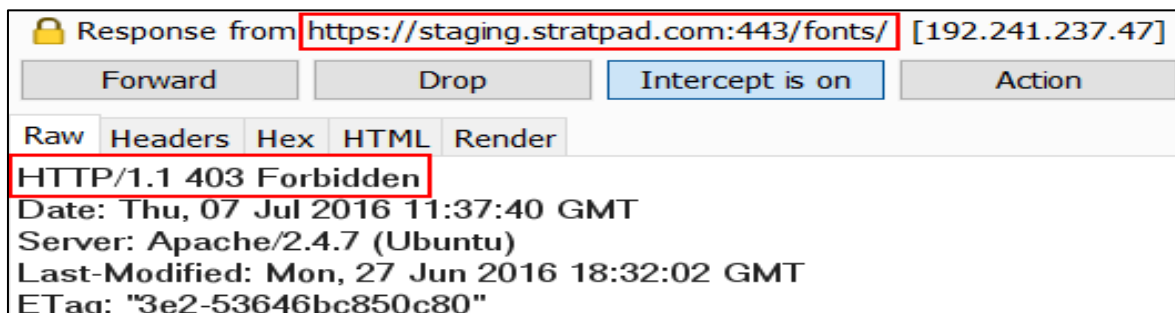


Figure 9: 403 Forbidden response is received from the application confirming the existence of the hidden "/fonts" directory

Likelihood: High

Impact: Minimal

Remediation:

The application server should return a '404 Not Found' error response instead of a '403 Forbidden' error response and remove any directories which are not required. The application should respond with a '404 Not Found' error instead of a '403 Forbidden' error when a request is made for existing or a non-existing directory. This will help in obfuscating the valid directories which exist on the server. Additionally, unused files and directories should be removed from the application server.

3.2.4.2 Verbose Server Banner

Description:

Verbose server information is sent in the HTTP responses from the server. The information included in the response contains the server name, type, and version number.

Below is an example of a HTTP response that contains verbose server banners:

```
HTTP/1.1 200 OK
Server: Apache 2.0
Cache-control: private
X-Powered-By: JSP/2.2
Content-Type: text/html; charset=utf-8
Content-Language: en-US
Content-Length: 3347
```

Verbose server banners provide additional information that allows an attacker to perform targeted attacks to the specific technology stack in use by the application and underlying infrastructure.

Instances:

1. <https://staging.stratpad.com/stratweb.html>
 - a. Header: Server
2. <https://staging.stratpad.com/reference/en.lproj/onStrategy06.htm>
 - a. Header: Server

Note: This finding is systemic for staging.stratpad.com domain.

Steps To Reproduce:

1. Configure your browser to use a proxy tool such as Burp Suite.
2. Navigate to the following URL <https://staging.stratpad.com>
3. Authenticate to the application using basic authentication.
4. Login to the application.
5. Navigate to the URL mentioned in the "Instances" section above.
6. Intercept the response in Burp Suite and Observe that the "Server" header Contains sensitive server information in response.

Evidence:

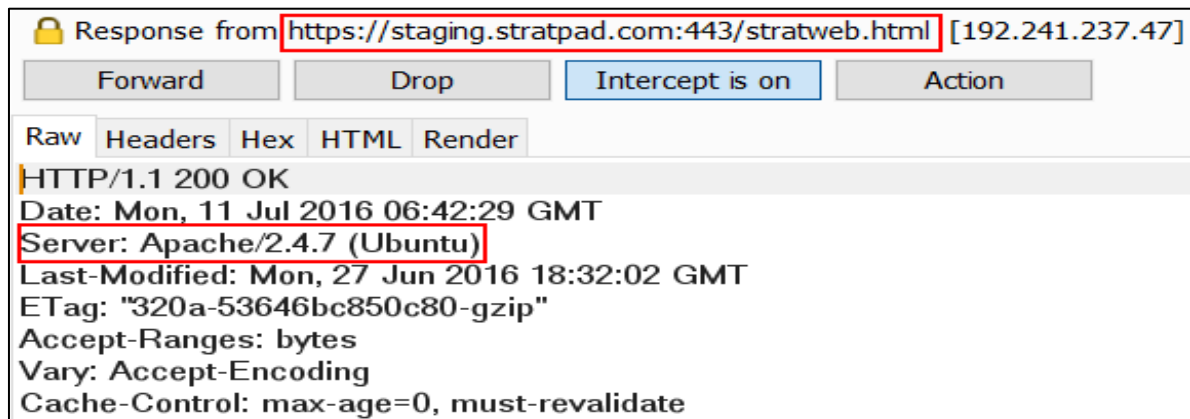


Figure 10: Applications response header "Server" is disclosing sensitive server information

Likelihood: High

Impact: Minimal

Remediation:

Verbose server information should be removed from all HTTP responses. This can be performed by modifying the server's configuration files or through the use and configuration of a web application firewall.

About Cigital, Inc.

Cigital, Inc. is the leading software security and quality consulting firm. Established in 1992, Cigital plans and implements initiatives that help organizations ensure their applications are secure and reliable while also improving how they build and deploy software. Our recognized experts apply a combination of proven methodologies, tools, and best practices to meet each client's unique requirements, providing resources and knowledge to deliver value to their business.

Cigital has enabled some of the most well-known organizations world-wide in financial services, communications, insurance, online gaming, hospitality, e-commerce and government to reduce their mission-critical software business risks. Our offices in the UK and Northern Europe broaden our reach and allow us to support trans-Atlantic clients more completely with European consulting, assessment and training operations.

Our expert advisors are recognized thought leaders in software security and have written the books on software security and quality with such publications as the Web Security Testing Cookbook, Exploiting Online Games, Software Security and Building Secure Software.

Cigital is headquartered near Washington, D.C. with regional offices in the U.S., London and India.