CANVAS SECURITY ASSESSMENT

INSTRUCTURE

29 November 2011

Version 1.0
Project Reference: SG-1472-11

SecurusGlobal

DRAFT FOR DISCUSSION

Copyright

Engagement Representatives

Drazen Drazic CEO, Securus Global		Phone:+61 (0) 2 9283 0255 email: drazen.drazic@securusglobal.com

Revision History

V0.1		Findings and business analysis included
V0.9		Quality Assurance Review

Table of Contents

1 EXECUTIVE REPORT 3
1.1 Executive Summary 3
1.2 Document Scope 3
1.3 Disclaimer 3
1.4 Summary of Findings 4
1.5 Summary of Recommendations 5
2 TECHNICAL REPORT 6
2.1 Summary of Vulnerabilities 6
2.2 securus-tc20.instructure.com 6
2.3 Detailed Findings 7
3 APPENDICES 19
3.1 Risk Classification 19

1 EXECUTIVE REPORT

1.1 Executive Summary

This report presents the findings of a security assessment of Instructure's Canvas platform conducted between the period 7th November 2011 to 25h November 2011.

It is our impression that CANVAS is generally a secure application and that the issues found can quickly be remediated. CANVAS is built upon a foundation of very widely used programming frameworks that have been subject to extensive security auditing.

During testing several issues were identified, including one critical vulnerability. Due to progressive reporting and status updates with Instructure the critical vulnerability identified was promptly remediated and released to users.

The remaining issues present a moderate risk to the integrity and confidentiality of the stored data which could lead to reputational damages and loss of confidence in the CANVAS system should they risk be realised. None of these issues are associated with major application flaws that are difficult to remediate.

1.2 Document Scope

This report consists of the following components:

- Executive Report: A report targeted at senior executives and business stakeholders that presents a summary of the project, findings, recommendations and key advice along with an analysis of business risk and impact.
- Technical Report: A report targeted at development, security and technical teams required to remediate and address security vulnerabilities that provides a detailed examination of each finding, risk assessment justification and detailed remediation advice necessary for technical teams to assess, reproduce and remediate findings of the security assessment.
- Appendices – Supplementary information supporting the report including our risk assessment methodology and matrix.

1.3 Disclaimer

This security test is a point in time assessment of the state of security in the CANVAS test environment prepared for Securus Global valid at the time that the test was completed. The description of the findings, recommendations and risk will be valid for the time of the test. Any projection of such information to the future is subject to the risk that, because of change, the description may no longer portray the controls in existence.

1.4 Summary of Findings

Following in depth testing of the environment it is our impression that the CANVAS application is robust and has been developed by security aware programmers in line with secure coding practices.

Based on our understanding of your business, we have assessed the level of risk to your organisation based on the nature of the vulnerabilities discovered, their exploitability in your environment and the potential impact should the risk be realised to be low.

There were no major design flaws identified. Vulnerabilities identified were of a nature that allowed remediation without major resign or application re-coding.

During the testing a critical SQL injection vulnerability was identified. We also uncovered an issue with the way the application framework generated session IDs. Both vulnerabilities were fixed quickly in response to our reporting.

The following graph illustrates the levels of risk for the vulnerabilities identified during testing and the residual risk following the vulnerabilities being addressed:

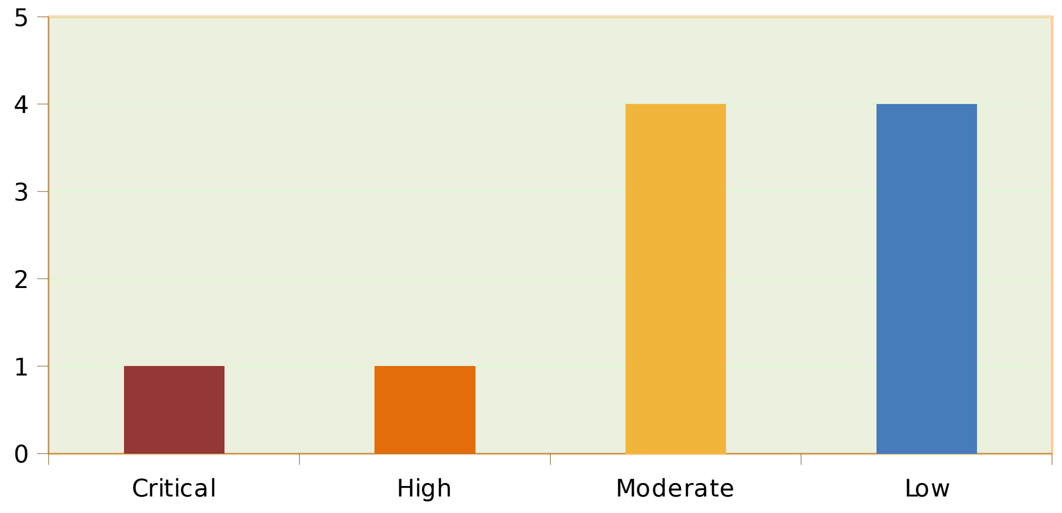

Figure 1 Vulnerabilities identified during testing

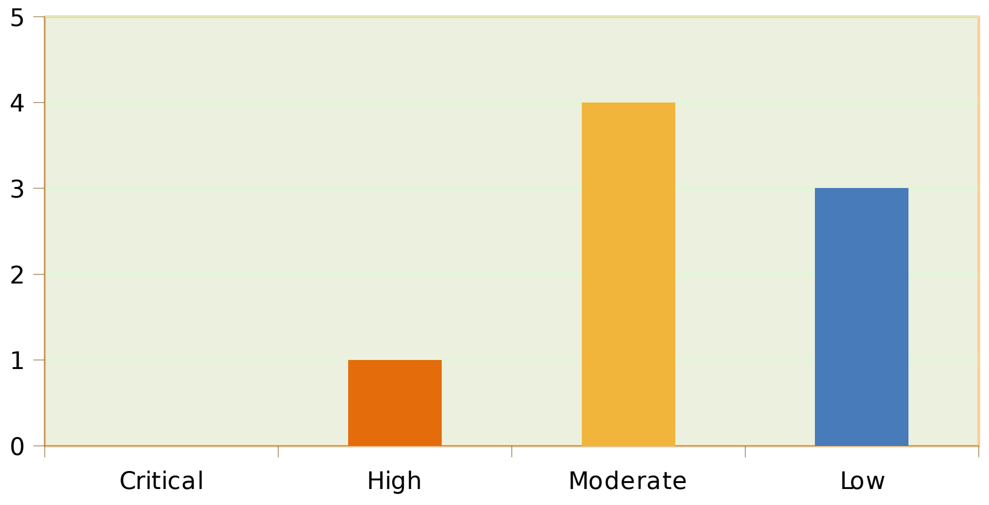

Figure 2 Residual Risk following remediation of vulnerabilities during testing.

1.5 Summary of Recommendations

To address the identified issues it is our recommendation that:

- The JSON interfaces are protected from cross-site reading
- File uploads are limited or handled safely
- Autocompletion of sensitive forms is turned off
- CSRF tokens are deployed wherever state-changing functionality is triggered
- Input to SQL queries is escaped and strict whitelisting is performed
- Allowed markup is restricted further
- Unauthenticated file access is disabled
- Uploaded files are scanned for malware before users are allowed to download them
- IDs are used to identify users when messaging a new user
- Service banners containing version/software information are censored
- A new session ID is created every time a user logs in

For a detailed description of our findings and recommended actions, please refer to the section entitled: ‘Technical Report’.

2 TECHNICAL REPORT

2.1 Summary of Vulnerabilities

The following table is a summary listing of the vulnerabilities discovered through our testing. The detailed technical details for each vulnerability, including remediation advice, is addressed in the following pages.

2.2 securus-tc20.instructure.com

	Finding	Risk
	SQL Injection Vulnerability Identified	**Critical**
	JSON Interfaces Allow Cross-Site Reading	**High**
	Arbitrary File Upload	**Moderate**
	Autocomplete Enabled for Sensitive Data	**Moderate**
	Cross-Site Request Forgery	**Low**
6.	Overly Permissive HTML Sanitisation	**Moderate**
7.	No Access Control for Uploaded Files	**Moderate**
8.	No Anti-Virus Solution in Use	**Low**
9.	Conversation Delivery Name Spoofing	**Low**
10.	Incorrect Session Management	**Low**
11.	Information Disclosure Identified	**Informational**
12.	Partial Regular Expression Matching	**Informational**

2.3 Detailed Findings

2.3.1 SQL Injection Vulnerability Identified — **Critical**

Description

The application does not sanitise user-supplied data prior to it being included within SQL queries. This generally indicates that queries are generated through string concatenation rather than using parameterisation ("parameterised queries"). An attacker may include (or "inject") any content into the final query. This results in an attacker being able to issue arbitrary commands/requests to the database.

Likelihood

Only a single point of injection was found that would only be identified by a skilled attacker. The exploitation of this vulnerability is non-trivial.

These conditions mean the likelihood of exploitation is **unlikely**.

This vulnerability was addressed during testing and presents no residual risk.

Impact

An attacker would be able to retrieve information from the database used by the application. This information includes database configuration and sensitive user details.
Due to the data that can be compromised, the impact of this vulnerability is considered to be **significant**.

Reproduction Details

This vulnerability can be exploited using a technique known as “boolean blind SQL injection”. The following values for the “order” parameter were sent using a POST request to “https://securus-tc20.instructure.com//users/649291/files/reorder” and triggered two different observable states:

- (SELECT+CASE+WHEN+version()>6+THEN+2182962+ELSE+21829621337+END)%2C2182961
- (SELECT+CASE+WHEN+version()<6+THEN+2182962+ELSE+21829621337+END)%2C2182961

Recommendation

The “order” parameter should be rejected if it is not a comma separated list of valid integers.

2.3.2 JSON Interfaces Allow Cross-Site Reading — High

Description

Multiple JSON interfaces are not sufficiently protected against malicious websites stealing the provided information.

This issue exists because browsers do not enforce the same origin policy for the <script> tag and the format of JSON data is by definition valid JavaScript. This exploitation of this issue is commonly called "JavaScript Hijacking".

The unprotected data can be logged by a malicious webpage by overloading its JavaScript “Object” or “Array” object prototypes with a custom logging function using the __defineSetter__ function.

Likelihood

This issue is easy identify and trivial to exploit. The only requirement for exploitation is that an authenticated user is lured to an attacker’s website. It is **possible** that this issue will be exploited.

Impact

The information that may be stolen is dependent upon which user is lured to the attacker’s website. The information includes but is not limited to the following:

- A list of all the courses the user has access to
- The URLs to directly access the calendar (ICS) files for the courses the user has access to
- Contents of conversations the user has been included in

The impact of this issue is major.

Reproduction Details

Visit these links as an authenticated user and observe that neither the “X-Requested-With” header nor the “api_key” parameter is required:

- https://securus-tc20.instructure.com/courses/173795/files.json
- https://securus-tc20.instructure.com/courses/173795/folders/363886.json
- https://securus-tc20.instructure.com/api/v1/courses.json

Recommendation

It is commonly incorrectly advised that simply checking for the “X-Requested-With” header is sufficient protection. The problem with this check is that already requested and cached data can still be stolen. This is due to the fact the request for the data will never reach the web server and therefore will not be blocked. The caching of JSON data can be prevented by setting the appropriate HTTP response headers to prevent caching.

The most widely used “Best Practice” for preventing this attack is to simply insert “for(;;);” before the JSON data.

2.3.3 Arbitrary File Upload

Moderate

Description

An authenticated attacker can upload files to “/conversations” and have them displayed by the target's browser, including Flash and HTML files.

Likelihood

File upload functionality is commonly targeted by attackers because it is trivial to abuse and assists an attacker in targeting specific victims. Attacks performed using this issue require user interaction and an audit trail would be easy to follow if an attack was detected.

This issue is **likely** to be exploited.

Impact

This issue allows the execution of JavaScript on a potentially trusted Instructure domain.

An attacker could use this issue as part of a Phishing attack with a fake login page or a page containing client-side exploits (e.g Java or Flash vulnerabilities). The impact of this issue is considered **moderate**.

Reproduction Details

Visit the following URL and attach a file to a message:

- https://securus-tc20.instructure.com/conversations

Recommendation

All files should be forced to download instead of displayed inside the browser window. This can be achieved by setting the HTTP response header “Content-Type: application/force-download”.

If certain document types are required to be displayed within the browser window (such as Microsoft Office or PDF files) then a whitelist of safe extensions should be created and only these should be displayed.

2.3.4 Autocomplete Enabled for Sensitive Data

Moderate

Description

Forms which request sensitive data within the application have the “Autocomplete” feature enabled. This is an opt-out feature.

The feature automatically stores a user’s input to assist the user with completion of the form at a later date. In the case of the user submitting credentials, the user will be presented with the option to save their credentials. The storage of this information is typically not conducted in a secure manner.

Likelihood

This feature is commonly used by the end-user and exploitation does not require any technical skills for an attacker. Users are likely to use the application in a shared environment. It is **possible** this issue will be exploited.

Impact

This issue raises the risk of host compromise when used in conjunction with an attack to gain access to the end-user’s host. The impact of this issue is **moderate**.

Reproduction Details

Submit any form which accepts sensitive user-submitted data, such as the login form and observe that a freshly installed browser prompts you to save the credentials.

Recommendation

Disable the autocomplete functionality. This can be achieved by adding the following to the HTML <form> tags:

```
autocomplete=”off”
```

2.3.5 Cross-Site Request Forgery

Low

Description

Various functions throughout the application that trigger changes on the server are not protected by a CSRF token.

This can be exploited when an attacker can make the victim's browser perform cross-site requests, for example by using a malicious webpage or an image reference in the wiki. The two examples identified are:
- A quiz can be started without the student's knowledge
- When the victim is logged in as a privileged user, the attacker can force the site to start "exporting" the courses that user administrates

Likelihood

This issue is trivial to find and easy to exploit. The specific flaws identified would be of minimal use to an attacker to exploit making it **unlikely** that these two specific flaws will be exploited.

Impact

If a time-limited quiz is started without the student's knowledge that will cause the student to fail the quiz.

Repeatedly exporting courses can consume considerable amounts of CPU and disk space, potentially destabilizing the application during times of high load. Such an attack would be easy to detect and stop, so the threat posed is limited.

The impact of the two identified vulnerabilities is **minor** although it should be noted that additional CSRF vulnerabilities may be more serious.

Reproduction Details

To reproduce the quiz issue, make a GET request to the following URL with appropriate IDs while authenticated as a student:
https://securus-tc20.instructure.com/courses/**COURSE_ID**/quizzes/**QUIZ_ID**/take

To reproduce the course export issue, make a GET request to the following URL with the appropriate ID while authenticated as a user with privileges that allows exporting courses:
https://securus-tc20.instructure.com/courses/**COURSE_ID**/content_exports

Recommendation

It is recommended that any functionality that modifies data or application state on the server is protected using a CSRF token.

2.3.6 Overly Permissive HTML Sanitisation **Moderate**

Description

The wiki correctly uses whitelisting to filter HTML but the whitelist is too permissive and allows potentially dangerous tags and attributes.

Likelihood

The flaw is trivial to exploit and is easy to identify by examining the source code making it a **likely** target for exploitation.

Impact

This issue could be used by attackers as part of a phishing attack by displaying a fake login page on the trusted domain. Attackers could also use this issue to perform secondary attacks on the application or to assist in client-side exploitation of the victim. The impact of this issue is **moderate**.

Reproduction Details

Include the markup within any Wiki submission form and observe that the insecure tags are not filtered.

Recommendation

It is recommended that the following tags are removed from the whitelist or alternatively modify the policy to only source files from a strictly trusted directory or subdomain (ensuring user uploaded content is not trusted):

- `<object>`
- `<embed>`

Furthermore it is recommended that these tags are removed from the whitelist:

- `<iframe>`
- `<style>`

2.3.7 No Access Control for Uploaded Files

Moderate

Description

Uploaded files are only protected by a token and do not require an authenticated session. This means that an unauthenticated user that has discovered a URL either through accidental exposure or a secondary vulnerability can download user uploaded files.

Likelihood

This issue is dependent upon a secondary vulnerability. It would be more common for authenticated user's to download and redistribute the content. This means it is **unlikely** that this issue will be exploited.

Impact

The impact of this issue is would result in course content or other user uploaded content being available to people who are not authenticated. This also has the impact of not being able to not being able to accurately track who has downloaded which files. This issue has **moderate** impact.

Reproduction Details

Visit the following URL without any authenticated session cookies and observe that you can access the user uploaded file:
https://safefiles-tc20.instructure.com/files/2182307/download?verifier=YZoXmH9osXRPXteTqLGa2XXCKCY0G3ZdGV29EnXs

Recommendation

Sensitive files should only be accessible with an authenticated session.

2.3.8 No Anti-Virus Solution in Use

Low

Description

Files that are uploaded onto the web server by users are not automatically scanned to detect if they contain malware.

Likelihood

No technical skills are required but if malware distribution was detected a clear audit trail would exist. These conditions mean it would be **rare** to be exploited.

Impact

An attacker who takes advantage of this lack of security control could use the website to distribute malware. This could incur reputational damage if such an incident was made public. The impact of this issue is considered **moderate**.

Reproduction Details

Upload the EICAR anti-virus test file and verify that the file can be downloaded from the website.

Recommendation

All files that are uploaded by users should be scanned for malware before being available for download.

2.3.9 Conversation Delivery Name Spoofing — Low

Description

The user_name is displayed as the intended recipient but the user_id supplied is what decides which user will receive the message.

Likelihood

This flaw is considered **rare** to be exploited as it requires a strong element of social engineering.

Impact

If an attacker can trick a user to write a confidential message after clicking specially crafted a link then an attacker can receive the message intended for a different recipient. The impact of this issue is **moderate**.

Reproduction Details

Send a message using the form located at the following URL and observe that the message is delivered to the user specified by the “user_id” parameter:
https://securus-c20.instructure.com/conversations#/conversations?user_id=1234&user_name=TheRealAdmin

Recommendation

The username displayed in the input box should be resolved using this specified “user_id” parameter.

2.3.10 Incorrect Session Management — Low

Description

The underlying framework used by CANVAS to generate a static session IDs for each user based on a salt and the user ID. This makes it infeasible to predict or brute force, but also impossible to invalidate without the user changing their password

NOTE: This issue was remediated during the testing period.

Likelihood

This issue requires a high level of technical understanding to identify and requires an attacker to compromise the session cookie. This issue would only **rarely** be exploited.

This vulnerability was addressed during testing and presents no residual risk.

Impact

If a session cookie is compromised once, it can be reused forever. The impact is **moderate**.

Reproduction Details

Log in, capture the cookie, log out. Using the cookie in another browser you will still be able to use CANVAS functionality as an authenticated user.

Recommendation

Logging out should invalidate the session on the server-side as well as the client-side. A new and unique session ID should be generated upon each login.

2.3.11 Information Disclosure Identified	**Informational**

Description

The CANVAS servers expose detailed information about the software running on them in HTTP headers and error messages. This includes the web server software (Apache) and some of the Ruby frameworks (Rack).

Likelihood

An attacker will almost certainly read this information as one of the first steps of the recon phase of an attack.

Impact

With access to freely available common web proxy or browser debugging tools, an attacker can request the URLs directly and analyse the HTTP response header returned. Additionally, this issue would be identified automatically using any automated, standard web application testing tool. As such, the likelihood of being identified is **likely**.

Reproduction Details

Access the homepage and intercept the response with a HTTP proxy. Observe that the header contains “Server” and “X-Powered-By” fields which contain version numbers.

Recommendation

The display of software and version names can easily and quickly be turned off in the configuration files of the software. Usually it will still be possible to fingerprint the application stack through undocumented behaviour known to the attacker to happen in specific versions, but this requires much more effort and is less likely to yield accurate results.

2.3.12 Partial Regular Expression Matching **Informational**

Description

It is recommended that regular expressions are surrounded by \A and \z to match against the full string when doing string matching instead of only looking for substrings.

CANVAS does this in most of the code, but there are some places where it does not. No vulnerabilities caused by lack of strict matching were identified, but partial matching on variable input often leads to unexpected behaviour, especially when the application is modified or new features are added.

3 APPENDICES

3.1 Risk Classification

Securus Global follows the International Standards ISO 31000 and ISO 31010 for risk identification, classification and assessment. The following classification matrixes have been used to derive likelihood and impact.

3.1.1 Risk Matrix

Likelihood	**Consequence**				
	Insignificant	**Minor**	**Moderate**	**Major**	**Significant**
Almost Certain	Low	Moderate	High	Critical	Critical
Likely	Low	Moderate	Moderate	High	Critical
Possible	Low	Low	Moderate	High	Critical
Unlikely	Low	Low	Moderate	Moderate	High
Rare	Low	Low	Low	Moderate	High

3.1.2 Risk Classification

Rating	**Description**
Critical	Immediate action required
High	Senior management attention needed.
Moderate	Management responsibility should be specified
Low	Manage by routine procedures

3.1.3 Likelihood Description

Consequence	**Description**
Almost certain	It is almost certain expected to occur in most circumstances
Likely	Will probably occur in most circumstances
Possible	Might occur at some time
Unlikely	Could occur at some time
Rare	May occur only in exceptional circumstances

3.1.4 Consequence Descriptions

Consequence	**Description**
Significant	May cause extended system outage or may result in complete compromise of information or services.
Major	May cause considerable system outage, and/or loss of connected customers or business confidence. May result in compromise of large amount of information or services.
Moderate	May cause damage to the reputation of system management, and/or notable loss of confidence in the system's resources or services. It will require expenditure of significant resources to repair.
Minor	Will result in some tangible harm, albeit negligible and perhaps only noted by a few individuals. Will require some expenditure of resources to repair.
Insignificant	Will have little or no impact if threat is realised and vulnerability is exploited.