



Veracode Detailed Report  
**Application Security Report**  
**As of 19 Jul 2019**

Prepared for:	Boardwalk Tech
Prepared on:	July 19, 2019
Application:	Boardwalk Application Engine
Industry:	Not Specified
Business Criticality:	BC5 (Very High)
Required Analysis:	Manual Penetration Test, Static
Type(s) of Analysis Conducted:	Static
Scope of Static Scan:	2 of 2 Modules Analyzed

#### Inside This Report

Executive Summary	1
Summary of Flaws by Severity	1
Action Items	1
Flaw Types by Category	3
Policy Summary	4
Findings & Recommendations	5
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 19 Jul 2019

Mitigated Veracode Level: VL4  
Original Veracode Level: VL2  
Rated: Jul 19, 2019

Application: Boardwalk Application Engine Business Criticality: Very High  
Target Level: VL5 Adjusted/Published Rating: A\*/C

### Scans Included in Report

Static Scan	Dynamic Scan	Manual Penetration Test
2019-07-19-08:30:27 Score: 99 Completed: 7/19/19	Not Included in Report	Not Included in Report

## Executive Summary

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

### Application Business Criticality: BC5 (Very High)

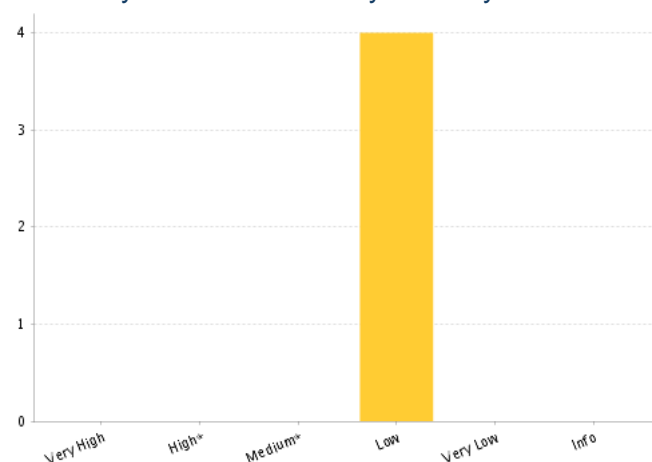
Impacts: Operational Risk (High), Financial Loss (High)

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 Penetration Test
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 Manual Penetration Test but it has not been performed. Please submit your application for Manual Penetration Test and remediate the required detected flaws to conform to your assigned policy.
- ➔ Your policy requires periodic Static Scan. Your next analysis must be completed by 10/19/19. Please submit your application for Static Scan by the deadline and remediate the required detected flaws to conform to your assigned policy.

### Flaw Severities

- Medium 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.

The chart displays the percentage of static code over time. The 'Static' series starts at approximately 50% in mid-May, rises to 100% by mid-May, dips to 50% in late May, then rises to 100% by mid-June, and remains at 100% through late July.

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.

The following modules were included in the application scan:

Module Name	Compiler	Operating Environment	Engine Version
BAE.war	JAVAC_8	Java J2SE 8	20190624162049
BAE.war_htmljscode.veracodegen.htmla.jsa	JAVASCRIPT_5_1	JavaScript	20190624162049

Static Scan Security Quality Score = 99 from prior scan			
Very High	0		
High	0*		
Medium	0*		
Low	4		
API Abuse	1		
Code Quality	1		
Information Leakage	2		
Very Low	0		
Informational	0		
Total	4*		

## Policy Evaluation

Policy Name: Veracode Recommended Very High

Revision: 1

Policy Status: Conditional 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
<b>Minimum Veracode Level</b>	VL5	VL4*	Did not pass
<b>(VL5) Min Analysis Score</b>	90	99*	Passed
<b>(VL5) Max Severity</b>	Medium	Flaws found: 0*	Passed

\* - Reflects violated rules that have mitigated flaws

Scan Requirements

Scan Type	Frequency	Last performed	Status
<b>Static</b>	Quarterly	7/19/19	Passed
<b>Manual</b>	Annually	Never	Passed (until 5/15/20)

Remediation

Flaw Severity	Grace Period	Flaws Exceeding	Status
<b>Very High</b>	0 days	0	Passed
<b>High</b>	0 days	0	Passed
<b>Medium</b>	0 days	0	Passed
<b>Low</b>	0 days	0	Passed
<b>Very Low</b>	0 days	0	Passed
<b>Informational</b>	0 days	0	Passed

Type	Grace Period	Exceeding	Status
<b>Min Analysis Score</b>	0 days	0	Passed

## Findings & Recommendations

### Best Practice Findings

#### Best Practices in use Throughout the Application



This application correctly uses cryptographically secure random number generation.

Additionally, you are doing a good job at protecting against these flaw types:



#### Cross-Site Scripting

##### CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)

- \* This application has 217 opportunities for this flaw, and 215 were successfully defended against using security best practices.
- \* The remaining 2 flaws should be addressed and are described in the following section, “Detailed Flaws by Severity.”

### Detailed Flaws by Severity

#### Very High (0 flaws)

No flaws of this type were found

#### High (0 flaws\*)

**Fix Required by Policy**

No flaws of this type were found

#### Medium (0 flaws\*)

**Fix Required by Policy**

No flaws of this type were found

#### Low (4 flaws)

##### → API Abuse(1 flaw)

#### Description

An API is a contract between a caller and a callee. Incorrect usage of certain API functions can result in exploitable security vulnerabilities.

The most common forms of API abuse are caused by the caller failing to honor its end of this contract. For example, if a program fails to call `chdir()` after calling `chroot()`, it violates the contract that specifies how to change the active root directory in a secure fashion. Providing too few arguments to a varargs function such as `printf()` also violates the API contract and will cause the missing parameters to be populated with unexpected data from the stack.

Another common mishap is when the caller makes false assumptions about the callee's behavior. One example of this is when a caller expects a DNS-related function to return trustworthy information that can be used for authentication purposes. This is a bad assumption because DNS responses can be easily spoofed.

#### Recommendations

When calling API functions, be sure to fully understand and adhere to the specifications to avoid introducing security vulnerabilities. Do not make assumptions about trustworthiness of the data returned from API calls or use the data in a context that was unintended by that API.

## Associated Flaws by CWE ID:

## → J2EE Bad Practices: Direct Management of Connections (CWE ID 245)(1 flaw)

## Description

The J2EE application directly manages connections rather than using the container's resource management facilities to obtain connections as specified in the J2EE standard. Every major web application container provides pooled database connection management as part of its resource management framework. Duplicating this functionality in an application is difficult and error prone, which is part of the reason it is forbidden under the J2EE standard.

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

## Recommendations

Request the connection from the container rather than attempting to access it directly.

## Instances found via Static Scan

Flaw Id	Module #	Class #	Module	Location	Fix By
869	4	-	BAE.war	com/.../DatabaseLoader.java 209	

## → Code Quality(1 flaw)

## Description

Code quality issues stem from failure to follow good coding practices and can lead to unpredictable behavior. These may include but are not limited to:

- \* Neglecting to remove debug code or dead code
- \* Improper resource management, such as using a pointer after it has been freed
- \* Using the incorrect operator to compare objects
- \* Failing to follow an API or framework specification
- \* Using a language feature or API in an unintended manner

While code quality flaws are generally less severe than other categories and usually are not directly exploitable, they may serve as indicators that developers are not following practices that increase the reliability and security of an application. For an attacker, code quality issues may provide an opportunity to stress the application in unexpected ways.

## Recommendations

The wide variance of code quality issues makes it impractical to generalize how these issues should be addressed. Refer to individual categories for specific recommendations.

## Associated Flaws by CWE ID:

## → Improper Resource Shutdown or Release (CWE ID 404)(1 flaw)

## Description

The application fails to release (or incorrectly releases) a system resource before it is made available for re-use. This condition often occurs with resources such as database connections or file handles. Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, it may be possible to launch a denial of service attack by depleting the resource pool.

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

## Recommendations

When a resource is created or allocated, the developer is responsible for properly releasing the resource as well as accounting for all potential paths of expiration or invalidation. Ensure that all code paths properly release resources.

## Instances found via Static Scan

Flaw Id	Module #	Class #	Module	Location	Fix By
879	2	-	BAE.war	.../BoardwalkRequestReader.java 59	

## → 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:

## → Information Exposure Through an Error Message (CWE ID 209)(2 flaws)

### Description



The software generates an error message that includes sensitive information about its environment, users, or associated data. The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more deadly attacks. If an attack fails, an attacker may use error information provided by the server to launch another more focused attack. For example, file locations disclosed by an exception stack trace may be leveraged by an attacker to exploit a path traversal issue elsewhere in the application.

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

## Recommendations

Ensure that only generic error messages are returned to the end user that do not reveal any additional details.

## Instances found via Static Scan

Flaw Id	Module #	Class #	Module	Location	Fix By
807	9	-	BAE.war	.../xlServiceLogic.java 299	
788	9	-	BAE.war	.../xlServiceLogic.java 310	

## Very Low (0 flaws)

No flaws of this type were found

## Info (0 flaws)

No flaws of this type were found

## 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	
<b>Static Scan</b>			
Scan Name:	2019-07-19-08:30:27	Scan Name:	2019-07-18-08:30:23
Completed:	7/19/19	Completed:	7/18/19
Score:	99	Score:	99

## Appendix B: Approved Mitigated Flaws (by Boardwalk Tech)

NOTE: Except in circumstances where the Customer has purchased Veracode's Mitigation Proposal Review Solution, Veracode does not review the mitigation strategy described below and is not responsible for its contents or the accuracy of any statements provided.

High (1 flaw)


Fix Required by Policy:  Flaw no longer impacts results.  
 Flaw continues to impact results.

### → SQL Injection(1 flaw)

Associated Flaws by CWE ID:

#### → Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') (CWE ID 89)(1 flaw)

##### Instances found via Static Scan

Flaw Id	Exploitability	Module #	Class #	Module	Location
 814	Neutral	6	-	BAE.war	.../Get_Boardwalk_Template_PropLogic.java 243
↳ <i>Mitigate by Design (Boardwalk Tech):</i> This is a known functionality(External Queries) of Boardwalk					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					
↳ <i>Reject Mitigation (Boardwalk Tech):</i> Please provide detailed description and Mitigate again.					
↳ <i>Mitigate by Design (Boardwalk Tech):</i> This functionality is being used to call custom stored procedures from the Boardwalk excel client. The custom stored procedures are required to be registered on server before that can be called. The registered stored procedures are stored in a SQL file along with sample signature. While the real call servlet verifies the call along with parameters passed. This approach mitigates the risk of SQL Injection.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					

Medium (58 flaws)


Fix Required by Policy:  Flaw no longer impacts results.  
 Flaw continues to impact results.


### → Cross-Site Scripting(2 flaws)

Associated Flaws by CWE ID:

#### → Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (CWE ID 80)(2 flaws)

##### Instances found via Static Scan

Flaw Id	Exploitability	Module #	Class #	Module	Location
 799	V.Likely	9	-	BAE.war	.../xlServiceLogic.java 299
↳ <i>Mitigate by Network Environment (Boardwalk Tech):</i> This servlet is invoked by Excel and data is passed in a custom protocol which will break the VBA parsing code if the string is escaped. Hence it must not be handled.					



Flaw Id	Exploitability	Module #	Class #	Module	Location
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					
 801	Likely	9	-	BAE.war	.../xlServiceLogic.java 310
↳ <i>Mitigate by Network Environment (Boardwalk Tech):</i> This servlet is invoked by Excel and data is passed in a custom protocol which will break the VBA parsing code if the string is escaped. Hence it must not be handled.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approve					



## → CRLF Injection(31 flaws)

Associated Flaws by CWE ID:

## → Improper Neutralization of CRLF Sequences ('CRLF Injection') (CWE ID 93)(4 flaws)


### Instances found via Static Scan




Flaw Id	Exploitability	Module #	Class #	Module	Location
 370	Likely	3	-	BAE.war	.../BoardwalkUserWizards.java 458
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					
 326	Likely	3	-	BAE.war	.../BoardwalkUserWizards.java 458
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					





Flaw Id	Exploitability	Module #	Class #	Module	Location
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					
 790	Likely	5	-	BAE.war	.../forgotPasswordLogic.java 124
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					
 786	Likely	5	-	BAE.war	.../forgotPasswordLogic.java 124
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofixcrlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					

## → Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting') (CWE ID 113)(27 flaws)




### Instances found via Static Scan





Flaw Id	Exploitability	Module #	Class #	Module	Location
 177	Neutral	1	-	BAE.war	com/.../table/BlobManager.java 117
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofix-">https://community.veracode.com/s/question/0D53400004DJusECAT/how-tofix-</a>					




Flaw Id	Exploitability	Module #	Class #	Module	Location
<p>crlf-http-response-splitting-in-java. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					
 10	Neutral	1	-	BAE.war	com/.../table/BlobManager.java 124
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved					
 133	Likely	7	-	BAE.war	servlets/LoginServlet.java 186
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 165	Likely	7	-	BAE.war	servlets/LoginServlet.java 257
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					




Flaw Id	Exploitability	Module #	Class #	Module	Location
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 509	Likely	7	-	BAE.war	servlets/LoginServlet.java 316
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 821	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1194
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 812	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1261
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 824	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1330
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the					







Flaw Id	Exploitability	Module #	Class #	Module	Location
<p>internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 779	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1431
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 773	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2686
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 785	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2824
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will</p>					

Flaw Id	Exploitability	Module #	Class #	Module	Location
address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 774	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2861
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 822	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2948
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 776	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3081
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 780	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3200

Flaw Id	Exploitability	Module #	Class #	Module	Location
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 850	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3312
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 853	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3324
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 825	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3385
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the</p>					

Flaw Id	Exploitability	Module #	Class #	Module	Location
<a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 798	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3444
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 763	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3527
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					
 781	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3999
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113					

Flaw Id	Exploitability	Module #	Class #	Module	Location
 787	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4025
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 818	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4177
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 764	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4260
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
 778	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4496
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at </p>					

Flaw Id	Exploitability	Module #	Class #	Module	Location
<p>fix-crlf-http-response-splitting-in-java. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p> <p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
✓ 768	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4725
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p> <p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					
✓ 793	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4942
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p> <p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 113</p>					

## → Insufficient Input Validation(25 flaws)




Associated Flaws by CWE ID:




## → URL Redirection to Untrusted Site ('Open Redirect') (CWE ID 601)(25 flaws)

### Instances found via Static Scan





Flaw Id	Exploitability	Module #	Class #	Module	Location
✓ 466	Likely	7	-	BAE.war	servlets/LoginServlet.java 186










Flaw Id	Exploitability	Module #	Class #	Module	Location
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 7	Likely	7	-	BAE.war	servlets/LoginServlet.java 257
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 446	Likely	7	-	BAE.war	servlets/LoginServlet.java 316
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 770	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1194
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the					




Flaw Id	Exploitability	Module #	Class #	Module	Location
<a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 828	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1261
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 805	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1330
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 791	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 1431
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					







Flaw Id	Exploitability	Module #	Class #	Module	Location
 808	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2686
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601</p>					
 829	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2824
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601</p>					
 827	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2861
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601</p>					
 789	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 2948
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at </p>					

Flaw Id	Exploitability	Module #	Class #	Module	Location
fix-crlf-http-response-splitting-in-java. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 775	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3081
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 806	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3200
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 832	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3312
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					

Flaw Id	Exploitability	Module #	Class #	Module	Location
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 833	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3324
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 771	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3385
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 782	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3444
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 777	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3527
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the					

Flaw Id	Exploitability	Module #	Class #	Module	Location
<p>internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 817	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 3999
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 795	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4025
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 803	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4177
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will					

Flaw Id	Exploitability	Module #	Class #	Module	Location
address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 792	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4260
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 796	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4496
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 830	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4725
↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a> . Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a> , it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.					
↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601					
 802	Likely	8	-	BAE.war	servlets/MyTablesLogic.java 4942

Flaw Id	Exploitability	Module #	Class #	Module	Location
<p>↳ <i>Mitigate by Design (Boardwalk Tech):</i> The CRLF injection issues and Insufficient Input Validation issues flagged for this particular line cannot be fixed using the ESAPI JAR file as ESAPI JAR file itself(the internal jar files used by ESAPI JAR) has errors which prevent it from getting applied. More information on these errors is available at <a href="https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java">https://community.veracode.com/s/question/0D53400004DJusECAT/how-to-fix-crlf-http-response-splitting-in-java</a>. Instead, as per the <a href="https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/">https://webspidertech.wordpress.com/2018/07/31/veracode-flaws-crlf-http-response-splitting-cwe-113-java/</a>, it can be understood that replaceAll() call to replace the CRLF instances within the strings will address the potential threat. We have implemented the replaceAll() fix and mitigated issues like these by design.</p>					
<p>↳ <i>Approve Mitigation (Boardwalk Tech):</i> Approved-CWE ID 601</p>					

Appendix C: Referenced Source Files

Id	Filename	Path
1	BlobManager.java	com/boardwalk/table/
2	BoardwalkRequestReader.java	servlets/
3	BoardwalkUserWizards.java	com/boardwalk/wizard/
4	DatabaseLoader.java	com/boardwalk/database/
5	forgotPasswordLogic.java	servlets/
6	Get_Boardwalk_Template_Pro pLogic.java	servlets/
7	LoginServlet.java	servlets/
8	MyTablesLogic.java	servlets/
9	xlServiceLogic.java	servlets/

Appendix D: Dynamic Flaw Inventory

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