GEC_TREAS_MYFUNDING_CheckMarx Scan Report

Project Name	GEC_TREAS_MYFUNDING_CheckMarx
Scan Start	Tuesday, August 31, 2021 2:14:04 PM
Preset	High and Medium
Scan Time	00h:05m:14s
Lines Of Code Scanned	117972
Files Scanned	682
Report Creation Time	Tuesday, August 31, 2021 2:19:54 PM
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258
Team	DTA-Consolidated-Derivatives_1000623096
Checkmarx Version	9.4.0.2076
Scan Type	Incremental
Source Origin	LocalPath
Density	2/10000 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

Included: Confirmed, Not Exploitable, To Verify, Urgent, Proposed Not Exploitable

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.2.1	All
OWASP Top 10 2013	All
FISMA 2014	All
NIST SP 800-53	All
OWASP Top 10 2017	All
OWASP Mobile Top 10 2016	All
OWASP Top 10 API	All
ASD STIG 4.10	All
OWASP Top 10 2010	All

Excluded:

Uncategorized	None
Custom	None
PCI DSS v3.2.1	None
OWASP Top 10 2013	None
FISMA 2014	None
NIST SP 800-53	None

OWASP Top 10 2017	None
OWASP Mobile Top 10 2016	None
OWASP Top 10 API	None
ASD STIG 4.10	None
OWASP Top 10 2010	None

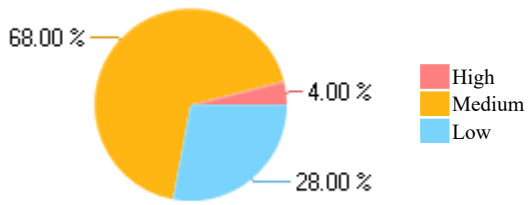
Results Limit

Results limit per query was set to 50

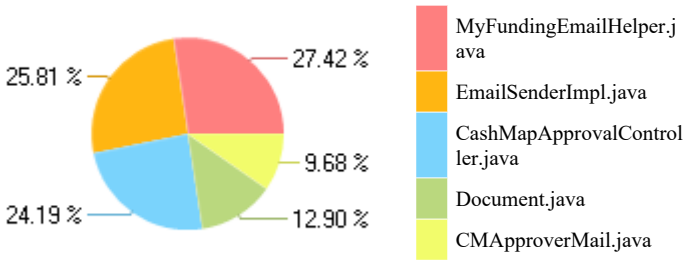
Selected Queries

Selected queries are listed in Result Summary

Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities

Scan Summary - OWASP Top 10 2017

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2017

Category	Threat Agent	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection*	App. Specific	EASY	COMMON	EASY	SEVERE	App. Specific	2	1
A2-Broken Authentication*	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A3-Sensitive Data Exposure*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	6	4
A4-XML External Entities (XXE)*	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	10	3
A6-Security Misconfiguration *	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A7-Cross-Site Scripting (XSS)*	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	1	1
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	0	0
A10-Insufficient Logging & Monitoring*	App. Specific	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	App. Specific	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Top 10 2013

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2013

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection*	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	0	0
A2-Broken Authentication and Session Management*	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)*	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	1	1
A4-Insecure Direct Object References*	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	0	0
A5-Security Misconfiguration *	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure*	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	6	4
A7-Missing Function Level Access Control*	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	0	0
A8-Cross-Site Request Forgery (CSRF)*	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A9-Using Components with Known Vulnerabilities*	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A10-Unvalidated Redirects and Forwards*	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - PCI DSS v3.2.1

Category	Issues Found	Best Fix Locations
PCI DSS (3.2.1) - 6.5.1 - Injection flaws - particularly SQL injection*	4	2
PCI DSS (3.2.1) - 6.5.2 - Buffer overflows	0	0
PCI DSS (3.2.1) - 6.5.3 - Insecure cryptographic storage*	0	0
PCI DSS (3.2.1) - 6.5.4 - Insecure communications*	0	0
PCI DSS (3.2.1) - 6.5.5 - Improper error handling*	0	0
PCI DSS (3.2.1) - 6.5.7 - Cross-site scripting (XSS)*	3	2
PCI DSS (3.2.1) - 6.5.8 - Improper access control*	0	0
PCI DSS (3.2.1) - 6.5.9 - Cross-site request forgery*	0	0
PCI DSS (3.2.1) - 6.5.10 - Broken authentication and session management*	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - FISMA 2014

Category	Description	Issues Found	Best Fix Locations
Access Control*	Organizations must limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems) and to the types of transactions and functions that authorized users are permitted to exercise.	0	0
Audit And Accountability*	Organizations must: (i) create, protect, and retain information system audit records to the extent needed to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity; and (ii) ensure that the actions of individual information system users can be uniquely traced to those users so they can be held accountable for their actions.	0	0
Configuration Management*	Organizations must: (i) establish and maintain baseline configurations and inventories of organizational information systems (including hardware, software, firmware, and documentation) throughout the respective system development life cycles; and (ii) establish and enforce security configuration settings for information technology products employed in organizational information systems.	0	0
Identification And Authentication*	Organizations must identify information system users, processes acting on behalf of users, or devices and authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.	4	2
Media Protection*	Organizations must: (i) protect information system media, both paper and digital; (ii) limit access to information on information system media to authorized users; and (iii) sanitize or destroy information system media before disposal or release for reuse.	2	2
System And Communications Protection*	Organizations must: (i) monitor, control, and protect organizational communications (i.e., information transmitted or received by organizational information systems) at the external boundaries and key internal boundaries of the information systems; and (ii) employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems.	0	0
System And Information Integrity*	Organizations must: (i) identify, report, and correct information and information system flaws in a timely manner; (ii) provide protection from malicious code at appropriate locations within organizational information systems; and (iii) monitor information system security alerts and advisories and take appropriate actions in response.	17	6

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)*	0	0
AC-3 Access Enforcement (P1)*	0	0
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)*	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)*	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)*	0	0
SC-13 Cryptographic Protection (P1)*	1	1
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)*	0	0
SC-23 Session Authenticity (P1)*	0	0
SC-28 Protection of Information at Rest (P1)*	0	0
SC-4 Information in Shared Resources (P1)*	5	3
SC-5 Denial of Service Protection (P1)*	0	0
SC-8 Transmission Confidentiality and Integrity (P1)*	0	0
SI-10 Information Input Validation (P1)*	16	5
SI-11 Error Handling (P2)*	0	0
SI-15 Information Output Filtering (P0)*	1	1
SI-16 Memory Protection (P1)*	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Mobile Top 10 2016

Category	Description	Issues Found	Best Fix Locations
M1-Improper Platform Usage*	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.	0	0
M2-Insecure Data Storage*	This category covers insecure data storage and unintended data leakage.	0	0
M3-Insecure Communication*	This category covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.	0	0
M4-Insecure Authentication*	This category captures notions of authenticating the end user or bad session management. This can include: -Failing to identify the user at all when that should be required -Failure to maintain the user's identity when it is required -Weaknesses in session management	0	0
M5-Insufficient Cryptography*	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasnt done correctly.	0	0
M6-Insecure Authorization*	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.	0	0
M7-Client Code Quality*	This category is the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.	4	1
M8-Code Tampering*	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.	0	0
M9-Reverse Engineering*	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.	0	0
M10-Extraneous Functionality*	Often, developers include hidden backdoor functionality or other internal development security controls that are	0	0

	not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.		

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Top 10 API

Category	Issues Found	Best Fix Locations
API1-Broken Object Level Authorization*	0	0
API2-Broken Authentication*	0	0
API3-Excessive Data Exposure*	0	0
API4-Lack of Resources and Rate Limiting*	2	1
API5-Broken Function Level Authorization*	0	0
API6-Mass Assignment	0	0
API7-Security Misconfiguration*	0	0
API8-Injection*	0	0
API9-Improper Assets Management*	0	0
API10-Insufficient Logging and Monitoring*	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - Custom

Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Scan Summary - ASD STIG 4.10

Category	Issues Found	Best Fix Locations
APSC-DV-000640 - CAT II The application must provide audit record generation capability for the renewal of session IDs.	0	0
APSC-DV-000650 - CAT II The application must not write sensitive data into the application logs.	0	0
APSC-DV-000660 - CAT II The application must provide audit record generation capability for session timeouts.	0	0
APSC-DV-000670 - CAT II The application must record a time stamp indicating when the event occurred.	0	0
APSC-DV-000680 - CAT II The application must provide audit record generation capability for HTTP headers including User-Agent, Referer, GET, and POST.	0	0
APSC-DV-000690 - CAT II The application must provide audit record generation capability for connecting system IP addresses.	0	0
APSC-DV-000700 - CAT II The application must record the username or user ID of the user associated with the event.	0	0
APSC-DV-000710 - CAT II The application must generate audit records when successful/unsuccessful attempts to grant privileges occur.	0	0
APSC-DV-000720 - CAT II The application must generate audit records when successful/unsuccessful attempts to access security objects occur.	0	0
APSC-DV-000730 - CAT II The application must generate audit records when successful/unsuccessful attempts to access security levels occur.	0	0
APSC-DV-000740 - CAT II The application must generate audit records when successful/unsuccessful attempts to access categories of information (e.g., classification levels) occur.	0	0
APSC-DV-000750 - CAT II The application must generate audit records when successful/unsuccessful attempts to modify privileges occur.	0	0
APSC-DV-000760 - CAT II The application must generate audit records when successful/unsuccessful attempts to modify security objects occur.	0	0
APSC-DV-000770 - CAT II The application must generate audit records when successful/unsuccessful attempts to modify security levels occur.	0	0
APSC-DV-000780 - CAT II The application must generate audit records when successful/unsuccessful attempts to modify categories of information (e.g., classification levels) occur.	0	0
APSC-DV-000790 - CAT II The application must generate audit records when successful/unsuccessful attempts to delete privileges occur.	0	0
APSC-DV-000800 - CAT II The application must generate audit records when successful/unsuccessful attempts to delete security levels occur.	0	0
APSC-DV-000810 - CAT II The application must generate audit records when successful/unsuccessful attempts to delete application database security objects occur.	0	0
APSC-DV-000820 - CAT II The application must generate audit records when successful/unsuccessful attempts to delete categories of information (e.g., classification levels) occur.	0	0
APSC-DV-000830 - CAT II The application must generate audit records when successful/unsuccessful logon attempts occur.	0	0
APSC-DV-000840 - CAT II The application must generate audit records for privileged activities or other system-level access.	0	0
APSC-DV-000850 - CAT II The application must generate audit records showing starting and ending time for user access to the system.	0	0
APSC-DV-000860 - CAT II The application must generate audit records when successful/unsuccessful accesses to objects occur.	0	0
APSC-DV-000870 - CAT II The application must generate audit records for all direct access to the information system.	0	0
APSC-DV-000880 - CAT II The application must generate audit records for all account creations, modifications, disabling, and termination events.	0	0
APSC-DV-000910 - CAT II The application must initiate session auditing upon startup.	0	0
APSC-DV-000940 - CAT II The application must log application shutdown events.	0	0

APSC-DV-000950 - CAT II The application must log destination IP addresses.	0	0
APSC-DV-000960 - CAT II The application must log user actions involving access to data.	0	0
APSC-DV-000970 - CAT II The application must log user actions involving changes to data.	0	0
APSC-DV-000980 - CAT II The application must produce audit records containing information to establish when (date and time) the events occurred.	0	0
APSC-DV-000990 - CAT II The application must produce audit records containing enough information to establish which component, feature or function of the application triggered the audit event.	0	0
APSC-DV-001000 - CAT II When using centralized logging; the application must include a unique identifier in order to distinguish itself from other application logs.	0	0
APSC-DV-001010 - CAT II The application must produce audit records that contain information to establish the outcome of the events.	0	0
APSC-DV-001020 - CAT II The application must generate audit records containing information that establishes the identity of any individual or process associated with the event.	0	0
APSC-DV-001030 - CAT II The application must generate audit records containing the full-text recording of privileged commands or the individual identities of group account users.	0	0
APSC-DV-001040 - CAT II The application must implement transaction recovery logs when transaction based.	0	0
APSC-DV-001050 - CAT II The application must provide centralized management and configuration of the content to be captured in audit records generated by all application components.	0	0
APSC-DV-001070 - CAT II The application must off-load audit records onto a different system or media than the system being audited.	0	0
APSC-DV-001080 - CAT II The application must be configured to write application logs to a centralized log repository.	0	0
APSC-DV-001090 - CAT II The application must provide an immediate warning to the SA and ISSO (at a minimum) when allocated audit record storage volume reaches 75% of repository maximum audit record storage capacity.	0	0
APSC-DV-001100 - CAT II Applications categorized as having a moderate or high impact must provide an immediate real-time alert to the SA and ISSO (at a minimum) for all audit failure events.	0	0
APSC-DV-001110 - CAT II The application must alert the ISSO and SA (at a minimum) in the event of an audit processing failure.	0	0
APSC-DV-001120 - CAT II The application must shut down by default upon audit failure (unless availability is an overriding concern).	0	0
APSC-DV-001130 - CAT II The application must provide the capability to centrally review and analyze audit records from multiple components within the system.	0	0
APSC-DV-001140 - CAT II The application must provide the capability to filter audit records for events of interest based upon organization-defined criteria.	0	0
APSC-DV-001150 - CAT II The application must provide an audit reduction capability that supports on-demand reporting requirements.	0	0
APSC-DV-001160 - CAT II The application must provide an audit reduction capability that supports on-demand audit review and analysis.	0	0
APSC-DV-001170 - CAT II The application must provide an audit reduction capability that supports after-the-fact investigations of security incidents.	0	0
APSC-DV-001180 - CAT II The application must provide a report generation capability that supports on-demand audit review and analysis.	0	0
APSC-DV-001190 - CAT II The application must provide a report generation capability that supports on-demand reporting requirements.	0	0
APSC-DV-001200 - CAT II The application must provide a report generation capability that supports after-the-fact investigations of security incidents.	0	0
APSC-DV-001210 - CAT II The application must provide an audit reduction capability that does not alter original content or time ordering of audit records.	0	0
APSC-DV-001220 - CAT II The application must provide a report generation capability that does not alter original content or time ordering of audit records.	0	0
APSC-DV-001250 - CAT II The applications must use internal system clocks to generate time stamps for audit records.	0	0
APSC-DV-001260 - CAT II The application must record time stamps for audit records that can be mapped to Coordinated Universal Time (UTC) or Greenwich Mean Time (GMT).	0	0
APSC-DV-001270 - CAT II The application must record time stamps for audit records that meet a granularity of one	0	0

second for a minimum degree of precision.		
APSC-DV-001280 - CAT II The application must protect audit information from any type of unauthorized read access.	0	0
APSC-DV-001290 - CAT II The application must protect audit information from unauthorized modification.	0	0
APSC-DV-001300 - CAT II The application must protect audit information from unauthorized deletion.	0	0
APSC-DV-001310 - CAT II The application must protect audit tools from unauthorized access.	0	0
APSC-DV-001320 - CAT II The application must protect audit tools from unauthorized modification.	0	0
APSC-DV-001330 - CAT II The application must protect audit tools from unauthorized deletion.	0	0
APSC-DV-001340 - CAT II The application must back up audit records at least every seven days onto a different system or system component than the system or component being audited.	0	0
APSC-DV-001570 - CAT II The application must electronically verify Personal Identity Verification (PIV) credentials.	0	0
APSC-DV-001350 - CAT II The application must use cryptographic mechanisms to protect the integrity of audit information.	0	0
APSC-DV-001360 - CAT II Application audit tools must be cryptographically hashed.	0	0
APSC-DV-001370 - CAT II The integrity of the audit tools must be validated by checking the files for changes in the cryptographic hash value.	0	0
APSC-DV-001390 - CAT II The application must prohibit user installation of software without explicit privileged status.	0	0
APSC-DV-001410 - CAT II The application must enforce access restrictions associated with changes to application configuration.	0	0
APSC-DV-001420 - CAT II The application must audit who makes configuration changes to the application.	0	0
APSC-DV-001430 - CAT II The application must have the capability to prevent the installation of patches, service packs, or application components without verification the software component has been digitally signed using a certificate that is recognized and approved by the orga	0	0
APSC-DV-001440 - CAT II The applications must limit privileges to change the software resident within software libraries.	0	0
APSC-DV-001460 - CAT II An application vulnerability assessment must be conducted.	0	0
APSC-DV-001480 - CAT II The application must prevent program execution in accordance with organization-defined policies regarding software program usage and restrictions, and/or rules authorizing the terms and conditions of software program usage.	0	0
APSC-DV-001490 - CAT II The application must employ a deny-all, permit-by-exception (whitelist) policy to allow the execution of authorized software programs.	0	0
APSC-DV-001500 - CAT II The application must be configured to disable non-essential capabilities.	0	0
APSC-DV-001510 - CAT II The application must be configured to use only functions, ports, and protocols permitted to it in the PPSM CAL.	0	0
APSC-DV-001520 - CAT II The application must require users to reauthenticate when organization-defined circumstances or situations require reauthentication.	0	0
APSC-DV-001530 - CAT II The application must require devices to reauthenticate when organization-defined circumstances or situations requiring reauthentication.	0	0
APSC-DV-001540 - CAT I The application must uniquely identify and authenticate organizational users (or processes acting on behalf of organizational users).	0	0
APSC-DV-001550 - CAT II The application must use multifactor (Alt. Token) authentication for network access to privileged accounts.	0	0
APSC-DV-001560 - CAT II The application must accept Personal Identity Verification (PIV) credentials.	0	0
APSC-DV-001580 - CAT II The application must use multifactor (e.g., CAC, Alt. Token) authentication for network access to non-privileged accounts.	0	0
APSC-DV-001590 - CAT II The application must use multifactor (Alt. Token) authentication for local access to privileged accounts.	0	0
APSC-DV-001600 - CAT II The application must use multifactor (e.g., CAC, Alt. Token) authentication for local access to non-privileged accounts.	0	0
APSC-DV-001610 - CAT II The application must ensure users are authenticated with an individual authenticator prior to using a group authenticator.	0	0
APSC-DV-001620 - CAT II The application must implement replay-resistant authentication mechanisms for network access to privileged accounts.*	0	0

APSC-DV-001630 - CAT II The application must implement replay-resistant authentication mechanisms for network access to non-privileged accounts.	0	0
APSC-DV-001640 - CAT II The application must utilize mutual authentication when endpoint device non-repudiation protections are required by DoD policy or by the data owner.	0	0
APSC-DV-001650 - CAT II The application must authenticate all network connected endpoint devices before establishing any connection.	0	0
APSC-DV-001660 - CAT II Service-Oriented Applications handling non-releasable data must authenticate endpoint devices via mutual SSL/TLS.	0	0
APSC-DV-001670 - CAT II The application must disable device identifiers after 35 days of inactivity unless a cryptographic certificate is used for authentication.	0	0
APSC-DV-001680 - CAT I The application must enforce a minimum 15-character password length.*	0	0
APSC-DV-001690 - CAT II The application must enforce password complexity by requiring that at least one upper-case character be used.	0	0
APSC-DV-001700 - CAT II The application must enforce password complexity by requiring that at least one lower-case character be used.	0	0
APSC-DV-001710 - CAT II The application must enforce password complexity by requiring that at least one numeric character be used.	0	0
APSC-DV-001720 - CAT II The application must enforce password complexity by requiring that at least one special character be used.	0	0
APSC-DV-001730 - CAT II The application must require the change of at least 8 of the total number of characters when passwords are changed.	0	0
APSC-DV-001740 - CAT I The application must only store cryptographic representations of passwords.*	0	0
APSC-DV-001850 - CAT I The application must not display passwords/PINs as clear text.	0	0
APSC-DV-001750 - CAT I The application must transmit only cryptographically-protected passwords.	0	0
APSC-DV-001760 - CAT II The application must enforce 24 hours/1 day as the minimum password lifetime.	0	0
APSC-DV-001770 - CAT II The application must enforce a 60-day maximum password lifetime restriction.	0	0
APSC-DV-001780 - CAT II The application must prohibit password reuse for a minimum of five generations.	0	0
APSC-DV-001790 - CAT II The application must allow the use of a temporary password for system logons with an immediate change to a permanent password.	0	0
APSC-DV-001795 - CAT II The application password must not be changeable by users other than the administrator or the user with which the password is associated.	0	0
APSC-DV-001800 - CAT II The application must terminate existing user sessions upon account deletion.	0	0
APSC-DV-001820 - CAT I The application, when using PKI-based authentication, must enforce authorized access to the corresponding private key.	0	0
APSC-DV-001830 - CAT II The application must map the authenticated identity to the individual user or group account for PKI-based authentication.	0	0
APSC-DV-001870 - CAT II The application must uniquely identify and authenticate non-organizational users (or processes acting on behalf of non-organizational users).	0	0
APSC-DV-001810 - CAT I The application, when utilizing PKI-based authentication, must validate certificates by constructing a certification path (which includes status information) to an accepted trust anchor.	0	0
APSC-DV-001840 - CAT II The application, for PKI-based authentication, must implement a local cache of revocation data to support path discovery and validation in case of the inability to access revocation information via the network.	0	0
APSC-DV-001860 - CAT II The application must use mechanisms meeting the requirements of applicable federal laws, Executive Orders, directives, policies, regulations, standards, and guidance for authentication to a cryptographic module.	0	0
APSC-DV-001880 - CAT II The application must accept Personal Identity Verification (PIV) credentials from other federal agencies.	0	0
APSC-DV-001890 - CAT II The application must electronically verify Personal Identity Verification (PIV) credentials from other federal agencies.	0	0
APSC-DV-002050 - CAT II Applications making SAML assertions must use FIPS-approved random numbers in the generation of SessionIndex in the SAML element AuthnStatement.	0	0
APSC-DV-001900 - CAT II The application must accept FICAM-approved third-party credentials.	0	0
APSC-DV-001910 - CAT II The application must conform to FICAM-issued profiles.	0	0
APSC-DV-001930 - CAT II Applications used for non-local maintenance sessions must audit non-local maintenance	0	0

and diagnostic sessions for organization-defined auditable events.		
APSC-DV-000310 - CAT III The application must have a process, feature or function that prevents removal or disabling of emergency accounts.	0	0
APSC-DV-001940 - CAT II Applications used for non-local maintenance sessions must implement cryptographic mechanisms to protect the integrity of non-local maintenance and diagnostic communications.	0	0
APSC-DV-001950 - CAT II Applications used for non-local maintenance sessions must implement cryptographic mechanisms to protect the confidentiality of non-local maintenance and diagnostic communications.	0	0
APSC-DV-001960 - CAT II Applications used for non-local maintenance sessions must verify remote disconnection at the termination of non-local maintenance and diagnostic sessions.	0	0
APSC-DV-001970 - CAT II The application must employ strong authenticators in the establishment of non-local maintenance and diagnostic sessions.	0	0
APSC-DV-001980 - CAT II The application must terminate all sessions and network connections when non-local maintenance is completed.	0	0
APSC-DV-001995 - CAT II The application must not be vulnerable to race conditions.*	0	0
APSC-DV-002000 - CAT II The application must terminate all network connections associated with a communications session at the end of the session.	0	0
APSC-DV-002010 - CAT II The application must implement NSA-approved cryptography to protect classified information in accordance with applicable federal laws, Executive Orders, directives, policies, regulations, and standards.	0	0
APSC-DV-002020 - CAT II The application must utilize FIPS-validated cryptographic modules when signing application components.	0	0
APSC-DV-002030 - CAT II The application must utilize FIPS-validated cryptographic modules when generating cryptographic hashes.	0	0
APSC-DV-002040 - CAT II The application must utilize FIPS-validated cryptographic modules when protecting unclassified information that requires cryptographic protection.	0	0
APSC-DV-002150 - CAT II The application user interface must be either physically or logically separated from data storage and management interfaces.	0	0
APSC-DV-002210 - CAT II The application must set the HTTPOnly flag on session cookies.	0	0
APSC-DV-002220 - CAT II The application must set the secure flag on session cookies.*	0	0
APSC-DV-002230 - CAT I The application must not expose session IDs.*	0	0
APSC-DV-002240 - CAT I The application must destroy the session ID value and/or cookie on logoff or browser close.*	0	0
APSC-DV-002250 - CAT II Applications must use system-generated session identifiers that protect against session fixation.	0	0
APSC-DV-002260 - CAT II Applications must validate session identifiers.*	0	0
APSC-DV-002270 - CAT II Applications must not use URL embedded session IDs.	0	0
APSC-DV-002280 - CAT II The application must not re-use or recycle session IDs.	0	0
APSC-DV-002290 - CAT II The application must use the Federal Information Processing Standard (FIPS) 140-2-validated cryptographic modules and random number generator if the application implements encryption, key exchange, digital signature, and hash functionality.*	0	0
APSC-DV-002300 - CAT II The application must only allow the use of DoD-approved certificate authorities for verification of the establishment of protected sessions.	0	0
APSC-DV-002310 - CAT I The application must fail to a secure state if system initialization fails, shutdown fails, or aborts fail.	0	0
APSC-DV-002320 - CAT II In the event of a system failure, applications must preserve any information necessary to determine cause of failure and any information necessary to return to operations with least disruption to mission processes.	0	0
APSC-DV-002330 - CAT II The application must protect the confidentiality and integrity of stored information when required by DoD policy or the information owner.*	5	3
APSC-DV-002340 - CAT II The application must implement approved cryptographic mechanisms to prevent unauthorized modification of organization-defined information at rest on organization-defined information system components.	0	0
APSC-DV-002350 - CAT II The application must use appropriate cryptography in order to protect stored DoD information when required by the information owner or DoD policy.*	0	0
APSC-DV-002360 - CAT II The application must isolate security functions from non-security functions.	6	2
APSC-DV-002370 - CAT II The application must maintain a separate execution domain for each executing process.	0	0

APSC-DV-002380 - CAT II Applications must prevent unauthorized and unintended information transfer via shared system resources.	0	0
APSC-DV-002390 - CAT II XML-based applications must mitigate DoS attacks by using XML filters, parser options, or gateways.	0	0
APSC-DV-002400 - CAT II The application must restrict the ability to launch Denial of Service (DoS) attacks against itself or other information systems.*	0	0
APSC-DV-002410 - CAT II The web service design must include redundancy mechanisms when used with high-availability systems.	0	0
APSC-DV-002420 - CAT II An XML firewall function must be deployed to protect web services when exposed to untrusted networks.	0	0
APSC-DV-002610 - CAT II The application must remove organization-defined software components after updated versions have been installed.*	0	0
APSC-DV-002440 - CAT I The application must protect the confidentiality and integrity of transmitted information.	0	0
APSC-DV-002450 - CAT II The application must implement cryptographic mechanisms to prevent unauthorized disclosure of information and/or detect changes to information during transmission unless otherwise protected by alternative physical safeguards, such as, at a minimum, a Prot	0	0
APSC-DV-002460 - CAT II The application must maintain the confidentiality and integrity of information during preparation for transmission.	0	0
APSC-DV-002470 - CAT II The application must maintain the confidentiality and integrity of information during reception.	0	0
APSC-DV-002480 - CAT II The application must not disclose unnecessary information to users.*	0	0
APSC-DV-002485 - CAT I The application must not store sensitive information in hidden fields.	0	0
APSC-DV-002490 - CAT I The application must protect from Cross-Site Scripting (XSS) vulnerabilities.*	1	1
APSC-DV-002500 - CAT II The application must protect from Cross-Site Request Forgery (CSRF) vulnerabilities.*	4	1
APSC-DV-002510 - CAT I The application must protect from command injection.*	0	0
APSC-DV-002520 - CAT II The application must protect from canonical representation vulnerabilities.*	0	0
APSC-DV-002530 - CAT II The application must validate all input.*	4	2
APSC-DV-002540 - CAT I The application must not be vulnerable to SQL Injection.*	0	0
APSC-DV-002550 - CAT I The application must not be vulnerable to XML-oriented attacks.	0	0
APSC-DV-002560 - CAT I The application must not be subject to input handling vulnerabilities.*	4	1
APSC-DV-002570 - CAT II The application must generate error messages that provide information necessary for corrective actions without revealing information that could be exploited by adversaries.*	0	0
APSC-DV-002580 - CAT II The application must reveal error messages only to the ISSO, ISSM, or SA.*	0	0
APSC-DV-002590 - CAT I The application must not be vulnerable to overflow attacks.*	0	0
APSC-DV-002630 - CAT II Security-relevant software updates and patches must be kept up to date.	0	0
APSC-DV-002760 - CAT II The application performing organization-defined security functions must verify correct operation of security functions.	0	0
APSC-DV-002900 - CAT II The ISSO must ensure application audit trails are retained for at least 1 year for applications without SAMI data, and 5 years for applications including SAMI data.	0	0
APSC-DV-002770 - CAT II The application must perform verification of the correct operation of security functions: upon system startup and/or restart; upon command by a user with privileged access; and/or every 30 days.	0	0
APSC-DV-002780 - CAT III The application must notify the ISSO and ISSM of failed security verification tests.	0	0
APSC-DV-002870 - CAT II Unsigned Category 1A mobile code must not be used in the application in accordance with DoD policy.	0	0
APSC-DV-002880 - CAT II The ISSO must ensure an account management process is implemented, verifying only authorized users can gain access to the application, and individual accounts designated as inactive, suspended, or terminated are promptly removed.	0	0
APSC-DV-002890 - CAT I Application web servers must be on a separate network segment from the application and database servers if it is a tiered application operating in the DoD DMZ.	0	0
APSC-DV-002910 - CAT II The ISSO must review audit trails periodically based on system documentation recommendations or immediately upon system security events.	0	0
APSC-DV-002920 - CAT II The ISSO must report all suspected violations of IA policies in accordance with DoD information system IA procedures.	0	0
APSC-DV-002930 - CAT II The ISSO must ensure active vulnerability testing is performed.	0	0

APSC-DV-002980 - CAT II New IP addresses, data services, and associated ports used by the application must be submitted to the appropriate approving authority for the organization, which in turn will be submitted through the DoD Ports, Protocols, and Services Management (DoD PPS	0	0
APSC-DV-002950 - CAT II Execution flow diagrams and design documents must be created to show how deadlock and recursion issues in web services are being mitigated.	0	0
APSC-DV-002960 - CAT II The designer must ensure the application does not store configuration and control files in the same directory as user data.	0	0
APSC-DV-002970 - CAT II The ISSO must ensure if a DoD STIG or NSA guide is not available, a third-party product will be configured by following available guidance.	0	0
APSC-DV-002990 - CAT II The application must be registered with the DoD Ports and Protocols Database.	0	0
APSC-DV-002990 - CAT II The application must be registered with the DoD Ports and Protocols Database.	0	0
APSC-DV-002995 - CAT II The Configuration Management (CM) repository must be properly patched and STIG compliant.	0	0
APSC-DV-003000 - CAT II Access privileges to the Configuration Management (CM) repository must be reviewed every three months.	0	0
APSC-DV-003010 - CAT II A Software Configuration Management (SCM) plan describing the configuration control and change management process of application objects developed by the organization and the roles and responsibilities of the organization must be created and maintained.	0	0
APSC-DV-003020 - CAT II A Configuration Control Board (CCB) that meets at least every release cycle, for managing the Configuration Management (CM) process must be established.	0	0
APSC-DV-003030 - CAT II The application services and interfaces must be compatible with and ready for IPv6 networks.	0	0
APSC-DV-003040 - CAT II The application must not be hosted on a general purpose machine if the application is designated as critical or high availability by the ISSO.	0	0
APSC-DV-003050 - CAT II A disaster recovery/continuity plan must exist in accordance with DoD policy based on the applications availability requirements.	0	0
APSC-DV-003060 - CAT II Recovery procedures and technical system features must exist so recovery is performed in a secure and verifiable manner. The ISSO will document circumstances inhibiting a trusted recovery.	0	0
APSC-DV-003070 - CAT II Data backup must be performed at required intervals in accordance with DoD policy.	0	0
APSC-DV-003080 - CAT II Back-up copies of the application software or source code must be stored in a fire-rated container or stored separately (offsite).	0	0
APSC-DV-003090 - CAT II Procedures must be in place to assure the appropriate physical and technical protection of the backup and restoration of the application.	0	0
APSC-DV-003100 - CAT II The application must use encryption to implement key exchange and authenticate endpoints prior to establishing a communication channel for key exchange.	0	0
APSC-DV-003110 - CAT I The application must not contain embedded authentication data.*	0	0
APSC-DV-003120 - CAT I The application must have the capability to mark sensitive/classified output when required.	0	0
APSC-DV-003130 - CAT III Prior to each release of the application, updates to system, or applying patches; tests plans and procedures must be created and executed.	0	0
APSC-DV-003150 - CAT II At least one tester must be designated to test for security flaws in addition to functional testing.	0	0
APSC-DV-003140 - CAT II Application files must be cryptographically hashed prior to deploying to DoD operational networks.	0	0
APSC-DV-003160 - CAT III Test procedures must be created and at least annually executed to ensure system initialization, shutdown, and aborts are configured to verify the system remains in a secure state.	0	0
APSC-DV-003170 - CAT II An application code review must be performed on the application.	0	0
APSC-DV-003180 - CAT III Code coverage statistics must be maintained for each release of the application.	0	0
APSC-DV-003190 - CAT II Flaws found during a code review must be tracked in a defect tracking system.	0	0
APSC-DV-003200 - CAT II The changes to the application must be assessed for IA and accreditation impact prior to implementation.	0	0
APSC-DV-003210 - CAT II Security flaws must be fixed or addressed in the project plan.	0	0
APSC-DV-003215 - CAT III The application development team must follow a set of coding standards.	0	0
APSC-DV-003220 - CAT III The designer must create and update the Design Document for each release of the application.	0	0

APSC-DV-003230 - CAT II Threat models must be documented and reviewed for each application release and updated as required by design and functionality changes or when new threats are discovered.	0	0
APSC-DV-003235 - CAT II The application must not be subject to error handling vulnerabilities.*	0	0
APSC-DV-003250 - CAT I The application must be decommissioned when maintenance or support is no longer available.	0	0
APSC-DV-003236 - CAT II The application development team must provide an application incident response plan.	0	0
APSC-DV-003240 - CAT I All products must be supported by the vendor or the development team.	0	0
APSC-DV-003260 - CAT III Procedures must be in place to notify users when an application is decommissioned.	0	0
APSC-DV-003270 - CAT II Unnecessary built-in application accounts must be disabled.	0	0
APSC-DV-003280 - CAT I Default passwords must be changed.	0	0
APSC-DV-003330 - CAT II The system must alert an administrator when low resource conditions are encountered.	0	0
APSC-DV-003285 - CAT II An Application Configuration Guide must be created and included with the application.	0	0
APSC-DV-003290 - CAT II If the application contains classified data, a Security Classification Guide must exist containing data elements and their classification.	0	0
APSC-DV-003300 - CAT II The designer must ensure uncategorized or emerging mobile code is not used in applications.*	0	0
APSC-DV-003310 - CAT II Production database exports must have database administration credentials and sensitive data removed before releasing the export.	0	0
APSC-DV-003320 - CAT II Protections against DoS attacks must be implemented.	0	0
APSC-DV-003340 - CAT III At least one application administrator must be registered to receive update notifications, or security alerts, when automated alerts are available.	0	0
APSC-DV-003360 - CAT III The application must generate audit records when concurrent logons from different workstations occur.	0	0
APSC-DV-003345 - CAT III The application must provide notifications or alerts when product update and security related patches are available.	0	0
APSC-DV-003350 - CAT II Connections between the DoD enclave and the Internet or other public or commercial wide area networks must require a DMZ.	0	0
APSC-DV-003400 - CAT II The Program Manager must verify all levels of program management, designers, developers, and testers receive annual security training pertaining to their job function.	0	0
APSC-DV-000010 - CAT II The application must provide a capability to limit the number of logon sessions per user.	0	0
APSC-DV-000060 - CAT II The application must clear temporary storage and cookies when the session is terminated.	0	0
APSC-DV-000070 - CAT II The application must automatically terminate the non-privileged user session and log off non-privileged users after a 15 minute idle time period has elapsed.*	0	0
APSC-DV-000080 - CAT II The application must automatically terminate the admin user session and log off admin users after a 10 minute idle time period is exceeded.	0	0
APSC-DV-000090 - CAT II Applications requiring user access authentication must provide a logoff capability for user initiated communication session.	0	0
APSC-DV-000100 - CAT III The application must display an explicit logoff message to users indicating the reliable termination of authenticated communications sessions.	0	0
APSC-DV-000110 - CAT II The application must associate organization-defined types of security attributes having organization-defined security attribute values with information in storage.	0	0
APSC-DV-000120 - CAT II The application must associate organization-defined types of security attributes having organization-defined security attribute values with information in process.	0	0
APSC-DV-000130 - CAT II The application must associate organization-defined types of security attributes having organization-defined security attribute values with information in transmission.	0	0
APSC-DV-000160 - CAT II The application must implement DoD-approved encryption to protect the confidentiality of remote access sessions.*	0	0
APSC-DV-000170 - CAT II The application must implement cryptographic mechanisms to protect the integrity of remote access sessions.	0	0
APSC-DV-000190 - CAT I Messages protected with WS_Security must use time stamps with creation and expiration times.	0	0
APSC-DV-000180 - CAT II Applications with SOAP messages requiring integrity must include the following message elements:-Message ID-Service Request-Timestamp-SAML Assertion (optionally included in messages) and	0	0

all elements of the message must be digitally signed.		
APSC-DV-000200 - CAT I Validity periods must be verified on all application messages using WS-Security or SAML assertions.	0	0
APSC-DV-000210 - CAT II The application must ensure each unique asserting party provides unique assertion ID references for each SAML assertion.	0	0
APSC-DV-000220 - CAT II The application must ensure encrypted assertions, or equivalent confidentiality protections are used when assertion data is passed through an intermediary, and confidentiality of the assertion data is required when passing through the intermediary.	0	0
APSC-DV-000230 - CAT I The application must use the NotOnOrAfter condition when using the SubjectConfirmation element in a SAML assertion.	0	0
APSC-DV-000240 - CAT I The application must use both the NotBefore and NotOnOrAfter elements or OneTimeUse element when using the Conditions element in a SAML assertion.	0	0
APSC-DV-000250 - CAT II The application must ensure if a OneTimeUse element is used in an assertion, there is only one of the same used in the Conditions element portion of an assertion.	0	0
APSC-DV-000260 - CAT II The application must ensure messages are encrypted when the SessionIndex is tied to privacy data.	0	0
APSC-DV-000290 - CAT II Shared/group account credentials must be terminated when members leave the group.	0	0
APSC-DV-000280 - CAT II The application must provide automated mechanisms for supporting account management functions.	0	0
APSC-DV-000300 - CAT II The application must automatically remove or disable temporary user accounts 72 hours after account creation.	0	0
APSC-DV-000320 - CAT III The application must automatically disable accounts after a 35 day period of account inactivity.	0	0
APSC-DV-000330 - CAT II Unnecessary application accounts must be disabled, or deleted.	0	0
APSC-DV-000420 - CAT II The application must automatically audit account enabling actions.	0	0
APSC-DV-000340 - CAT II The application must automatically audit account creation.	0	0
APSC-DV-000350 - CAT II The application must automatically audit account modification.	0	0
APSC-DV-000360 - CAT II The application must automatically audit account disabling actions.	0	0
APSC-DV-000370 - CAT II The application must automatically audit account removal actions.	0	0
APSC-DV-000380 - CAT III The application must notify System Administrators and Information System Security Officers when accounts are created.	0	0
APSC-DV-000390 - CAT III The application must notify System Administrators and Information System Security Officers when accounts are modified.	0	0
APSC-DV-000400 - CAT III The application must notify System Administrators and Information System Security Officers of account disabling actions.	0	0
APSC-DV-000410 - CAT III The application must notify System Administrators and Information System Security Officers of account removal actions.	0	0
APSC-DV-000430 - CAT III The application must notify System Administrators and Information System Security Officers of account enabling actions.	0	0
APSC-DV-000440 - CAT II Application data protection requirements must be identified and documented.	0	0
APSC-DV-000520 - CAT II The application must audit the execution of privileged functions.	0	0
APSC-DV-000450 - CAT II The application must utilize organization-defined data mining detection techniques for organization-defined data storage objects to adequately detect data mining attempts.	0	0
APSC-DV-000460 - CAT I The application must enforce approved authorizations for logical access to information and system resources in accordance with applicable access control policies.*	0	0
APSC-DV-000470 - CAT II The application must enforce organization-defined discretionary access control policies over defined subjects and objects.*	0	0
APSC-DV-000480 - CAT II The application must enforce approved authorizations for controlling the flow of information within the system based on organization-defined information flow control policies.*	0	0
APSC-DV-000490 - CAT II The application must enforce approved authorizations for controlling the flow of information between interconnected systems based on organization-defined information flow control policies.	0	0
APSC-DV-000500 - CAT II The application must prevent non-privileged users from executing privileged functions to include disabling, circumventing, or altering implemented security safeguards/countermeasures.	0	0
APSC-DV-000510 - CAT I The application must execute without excessive account permissions.	0	0

APSC-DV-000530 - CAT I The application must enforce the limit of three consecutive invalid logon attempts by a user during a 15 minute time period.	0	0
APSC-DV-000560 - CAT III The application must retain the Standard Mandatory DoD Notice and Consent Banner on the screen until users acknowledge the usage conditions and take explicit actions to log on for further access.	0	0
APSC-DV-000540 - CAT II The application administrator must follow an approved process to unlock locked user accounts.	0	0
APSC-DV-000550 - CAT III The application must display the Standard Mandatory DoD Notice and Consent Banner before granting access to the application.	0	0
APSC-DV-000570 - CAT III The publicly accessible application must display the Standard Mandatory DoD Notice and Consent Banner before granting access to the application.	0	0
APSC-DV-000580 - CAT III The application must display the time and date of the users last successful logon.	0	0
APSC-DV-000630 - CAT II The application must provide audit record generation capability for the destruction of session IDs.	0	0
APSC-DV-000590 - CAT II The application must protect against an individual (or process acting on behalf of an individual) falsely denying having performed organization-defined actions to be covered by non-repudiation.	0	0
APSC-DV-000600 - CAT II For applications providing audit record aggregation, the application must compile audit records from organization-defined information system components into a system-wide audit trail that is time-correlated with an organization-defined level of tolerance	0	0
APSC-DV-000610 - CAT II The application must provide the capability for organization-identified individuals or roles to change the auditing to be performed on all application components, based on all selectable event criteria within organization-defined time thresholds.	0	0
APSC-DV-000620 - CAT II The application must provide audit record generation capability for the creation of session IDs.	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Top 10 2010

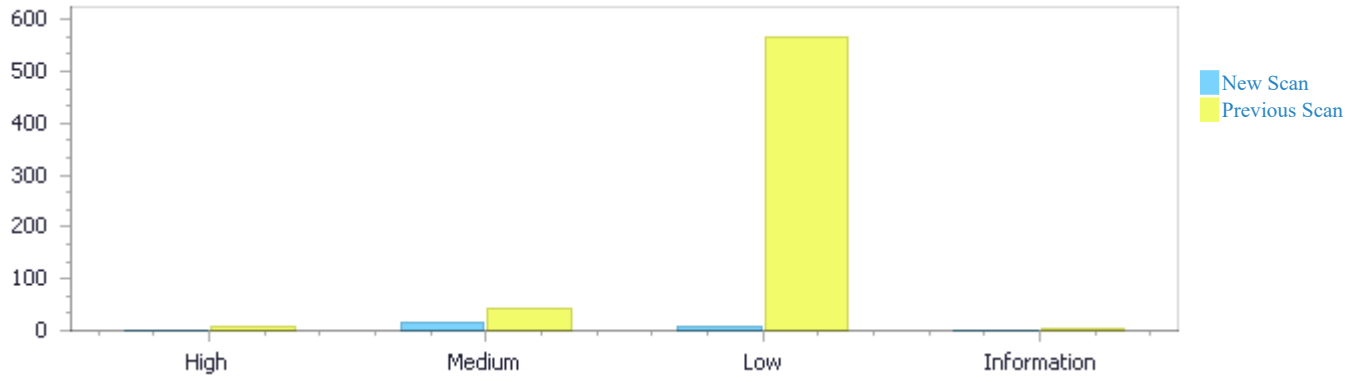
Category	Issues Found	Best Fix Locations
A1-Injection*	0	0
A2-Cross-Site Scripting (XSS)*	0	0
A3-Broken Authentication and Session Management*	0	0
A4-Insecure Direct Object References*	0	0
A5-Cross-Site Request Forgery (CSRF)	0	0
A6-Security Misconfiguration*	0	0
A7-Insecure Cryptographic Storage*	0	0
A8-Failure to Restrict URL Access	0	0
A9-Insufficient Transport Layer Protection*	0	0
A10-Unvalidated Redirects and Forwards*	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Results Distribution By Status

Compared to project scan from 8/31/2021 1:11 PM

	High	Medium	Low	Information	Total
New Issues	0	0	0	0	0
Recurrent Issues	1	17	7	0	25
Total	1	17	7	0	25
Fixed Issues	8	24	560	3	595



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	0	0	0	0
To Verify	1	17	7	0	25
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	1	17	7	0	25

Result Summary

Vulnerability Type	Occurrences	Severity
Reflected XSS All Clients	1	High
Input Path Not Canonicalized	4	Medium
Privacy Violation	4	Medium
SSRF	4	Medium
HTTP Response Splitting	2	Medium
Unchecked Input for Loop Condition	2	Medium
Use of a One Way Hash without a Salt	1	Medium
Trust Boundary Violation in Session Variables	6	Low
Heap Inspection	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	11
src/main/java/com/ge/treasury/myfunding/utils/MyFundingEmailHelper.java	10
src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java	10
src/main/java/com/ge/treasury/myfunding/domain/myfunding/Document.java	8
src/main/java/com/ge/treasury/myfunding/domain/CMApproverMail.java	4
src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java	4
src/main/java/com/ge/treasury/myfunding/domain/SltApproverMail.java	4
src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java	4
src/main/java/com/ge/treasury/myfunding/service/MyFundingRequestServiceManagerImpl.java	2
src/main/java/com/ge/treasury/myfunding/service/MyFundingAutomationServiceImpl.java	2

Scan Results Details

Reflected XSS All Clients

Query Path:
Java\Cx\Java High Risk\Reflected XSS All Clients Version:7

Categories

PCI DSS v3.2.1: PCI DSS (3.2.1) - 6.5.7 - Cross-site scripting (XSS)
OWASP Top 10 2013: A3-Cross-Site Scripting (XSS)
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-15 Information Output Filtering (P0)
OWASP Top 10 2017: A7-Cross-Site Scripting (XSS)
ASD STIG 4.10: APSC-DV-002490 - CAT I The application must protect from Cross-Site Scripting (XSS) vulnerabilities.

Description

Reflected XSS All Clients\Path 1:

Severity	High
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=3
Status	Recurrent
Detection Date	8/24/2021 2:12:58 PM

The application's getMDMBank embeds untrusted data in the generated output with getMDMBank, at line 607 of src/main/java/com/ge/treasury/myfunding/controllers/MDMController.java. This untrusted data is embedded straight into the output without proper sanitization or encoding, enabling an attacker to inject malicious code into the output.
The attacker would be able to alter the returned web page by simply providing modified data in the user input countryCode, which is read by the getMDMBank method at line 607 of src/main/java/com/ge/treasury/myfunding/controllers/MDMController.java. This input then flows through the code straight to the output web page, without sanitization.
This can enable a Reflected Cross-Site Scripting (XSS) attack.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/MDMController.java	src/main/java/com/ge/treasury/myfunding/controllers/MDMController.java
Line	607	616
Object	countryCode	getMDMBank

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/MDMController.java

Method public List<MDMBank> getMDMBank(final HttpServletRequest request,@RequestParam(value = "countryCode", required = true) String countryCode) throws JSONException {

```
....
607.        public List<MDMBank> getMDMBank(final HttpServletRequest
request,@RequestParam(value = "countryCode", required = true) String
countryCode) throws JSONException {
....
616.                    return MDMBank.getMDMBank(jsonString);
```

Input Path Not Canonicalized

Query Path:
Java\Cx\Java Medium Threat\Input Path Not Canonicalized Version:4

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Mobile Top 10 2016: M7-Client Code Quality
ASD STIG 4.10: APSC-DV-002560 - CAT I The application must not be subject to input handling vulnerabilities.

Description

Input Path Not Canonicalized\Path 1:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=4
Status	Recurrent
Detection Date	8/24/2021 2:13:00 PM

Method sendCMAEmail at line 87 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets dynamic data from the cmapproverMail element. This element's value then flows through the code and is eventually used in a file path for local disk access in sendMailWithAttachment at line 255 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	87	348
Object	cmapproverMail	source

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody CMApproverMail cmapproverMail)

```
....
87.   public String sendCMAEmail(HttpServletRequest request,
@RequestBody CMApproverMail cmapproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java

Method public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
348.                                  source = new
FileDataSource(boxTemp+documentLocal.getFileName());
```

Input Path Not Canonicalized\Path 2:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=5
Status	Recurrent
Detection Date	8/24/2021 2:13:00 PM

Method sendCMAEmail at line 87 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets dynamic data from the cmapproverMail element. This element's value then flows through the code and is eventually used in a file path for local disk access in downloadFile at line 280 of src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java
Line	87	285
Object	cmapproverMail	st

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody CMApproverMail cmapproverMail)

```
....
87.   public String sendCMAEmail(HttpServletRequest request,
@RequestBody CMApproverMail cmapproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java

Method private void downloadFile(HttpServletResponse response, BoxFile file, Boolean isDocuSign, Document doc) {

```
....
285.                    st = new FileOutputStream(new
File(boxTemp+doc.getFileName()));
```

Input Path Not Canonicalized\Path 3:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=6
Status	Recurrent
Detection Date	8/24/2021 2:13:00 PM

Method sendCMAEmail at line 106 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets dynamic data from the sltApproverMail element. This element's value then flows through the code and is eventually used in a file path for local disk access in sendMailWithAttachment at line 255 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	106	348
Object	sltApproverMail	source

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method	public String sendCMAEmail(HttpServletRequest request, @RequestBody SltApproverMail sltApproverMail)

```
....
106.        public String sendCMAEmail(HttpServletRequest request,
@RequestBody SltApproverMail sltApproverMail)
```

File Name	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Method	public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
348.                                source = new
FileDataSource(boxTemp+documentLocal.getFileName());
```

Input Path Not Canonicalized\Path 4:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=7
Status	Recurrent
Detection Date	8/24/2021 2:13:00 PM

Method sendCMAEmail at line 106 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets dynamic data from the sltApproverMail element. This element's value then flows through the code and is eventually used in a file path for local disk access in downloadFile at line 280 of src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java. This may cause a Path Traversal vulnerability.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java
Line	106	285
Object	sltApproverMail	st

Code Snippet

File Name	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java
Method	public String sendCMAEmail(HttpServletRequest request, @RequestBody SltApproverMail sltApproverMail)

```
....
106.        public String sendCMAEmail(HttpServletRequest request,
@RequestBody SltApproverMail sltApproverMail)
```

File Name	src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java
Method	private void downloadFile(HttpServletResponse response, BoxFile file, Boolean isDocuSign, Document doc) {

```
....
285.                    st = new FileOutputStream(new
File(boxTemp+doc.getFileName()));
```

Privacy Violation

Query Path:
Java\Cx\Java Medium Threat\Privacy Violation Version:8

Categories

PCI DSS v3.2.1: PCI DSS (3.2.1) - 6.5.1 - Injection flaws - particularly SQL injection
OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Identification And Authentication
NIST SP 800-53: SC-4 Information in Shared Resources (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure
ASD STIG 4.10: APSC-DV-002330 - CAT II The application must protect the confidentiality and integrity of stored information when required by DoD policy or the information owner.

Description

Privacy Violation\Path 1:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=10
Status	Recurrent
Detection Date	8/24/2021 2:13:20 PM

Method setCreditAmount at line 32 of src/main/java/com/ge/treasury/myfunding/domain/myfunding/WssServiceAmount.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/domain/myfunding/WssServiceAmount.java	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java
Line	32	84
Object	creditAmount	wssServiceAmount

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/domain/myfunding/WssServiceAmount.java

Method public void setCreditAmount(String creditAmount) {

```
....
32.   public void setCreditAmount(String creditAmount) {
```

File Name src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java

Method public Object getWssServiceAmount(HttpServletRequest request,

```
....
84.         return wssServiceAmount;
```

Privacy Violation\Path 2:

Severity	Medium

Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=11
Status	Recurrent
Detection Date	8/24/2021 2:13:20 PM

Method getWssServiceAmountDetails at line 428 of src/main/java/com/ge/treasury/myfunding/service/WssServiceImpl.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/service/WssServiceImpl.java	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java
Line	444	84
Object	creditAmout	wssServiceAmount

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/service/WssServiceImpl.java

Method public WssServiceAmount getWssServiceAmountDetails(ResponseEntity<String> response, String event, String subType) {

```
....
444.                                utilizationAmout =
String.valueOf((Double.parseDouble(creditAmout) -
Double.parseDouble(utilizationAmout)));
```

File Name src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java

Method public Object getWssServiceAmount(HttpServletRequest request,

```
....
84.         return wssServiceAmount;
```

Privacy Violation\Path 3:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=12
Status	Recurrent
Detection Date	8/24/2021 2:13:20 PM

Method getTMSAmountInfo at line 627 of src/main/java/com/ge/treasury/myfunding/tms/TMSServiceImpl.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/tms/TMSServiceImpl.java	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java
Line	652	75
Object	creditAmout	amount

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/tms/TMSServiceImpl.java

Method public TMSAmount getTMSAmountInfo(ResponseEntity<String> response, String event) {

```
....
652.                        tmsAmount.setCreditAmount(creditAmout);
```

File Name src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java

Method public Object getWssServiceAmount(HttpServletRequest request,

```
....
75.                     return amount;
```

Privacy Violation\Path 4:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=13
Status	Recurrent
Detection Date	8/24/2021 2:13:20 PM

Method getTMSAmountInfo at line 627 of src/main/java/com/ge/treasury/myfunding/tms/TMSServiceImpl.java sends user information outside the application. This may constitute a Privacy Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/tms/TMSServiceImpl.java	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java
Line	645	75
Object	creditAmout	amount

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/tms/TMSServiceImpl.java

Method public TMSAmount getTMSAmountInfo(ResponseEntity<String> response, String event) {

```
....
645.                                utilizationAmout =
String.valueOf((Double.parseDouble(creditAmout) -
Double.parseDouble(utilizationAmout)));
```

File Name src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java

Method public Object getWssServiceAmount(HttpServletRequest request,

```
....
75.                     return amount;
```

SSRF

Query Path:
Java\Cx\Java Medium Threat\SSRF Version:3

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A5-Broken Access Control

ASD STIG 4.10: APSC-DV-002500 - CAT II The application must protect from Cross-Site Request Forgery (CSRF) vulnerabilities.

Description

SSRF\Path 1:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=22
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

The application sends a request to a remote server, for some resource, using message in src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java:255. However, an attacker can control the target of the request, by sending a URL or other data in cmapproverMail at src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java:87.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	87	373
Object	cmapproverMail	message

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody CMApproverMail cmapproverMail)

```
....
87.   public String sendCMAEmail(HttpServletRequest request,
@RequestBody CMApproverMail cmapproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java

Method public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
373.              Transport.send(message);
```

SSRF\Path 2:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=23
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

The application sends a request to a remote server, for some resource, using message in src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java:255. However, an attacker can control the target of the request, by sending a URL or other data in request at src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java:106.

	Source	Destination

File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	106	373
Object	request	message

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody SltApproverMail sltApproverMail)

```
....
106.        public String sendCMAEmail(HttpServletRequest request,
@RequestBody SltApproverMail sltApproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java

Method public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
373.            Transport.send(message);
```

SSRF\Path 3:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=24
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

The application sends a request to a remote server, for some resource, using message in src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java:255. However, an attacker can control the target of the request, by sending a URL or other data in sltApproverMail at src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java:106.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	106	373
Object	sltApproverMail	message

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody SltApproverMail sltApproverMail)

```
....
106.        public String sendCMAEmail(HttpServletRequest request,
@RequestBody SltApproverMail sltApproverMail)
```

File Name	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Method	public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
373.                Transport.send(message);
```

SSRF\Path 4:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=25
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

The application sends a request to a remote server, for some resource, using message in src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java:255. However, an attacker can control the target of the request, by sending a URL or other data in request at src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java:87.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	87	373
Object	request	message

Code Snippet

File Name	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java
Method	public String sendCMAEmail(HttpServletRequest request, @RequestBody CMApproverMail cmapproverMail)

```
....
87.   public String sendCMAEmail(HttpServletRequest request,
@RequestBody CMApproverMail cmapproverMail)
```

File Name	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Method	public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
373.                Transport.send(message);
```

HTTP Response Splitting

Query Path:
Java\Cx\Java Medium Threat\HTTP Response Splitting Version:1

Categories

PCI DSS v3.2.1: PCI DSS (3.2.1) - 6.5.7 - Cross-site scripting (XSS)
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)

OWASP Top 10 2017: A1-Injection
ASD STIG 4.10: APSC-DV-002530 - CAT II The application must validate all input.

Description

HTTP Response Splitting\Path 1:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=1
Status	Recurrent
Detection Date	8/24/2021 2:12:58 PM

Method sendCMAEmail at line 87 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets user input from the cmapproverMail element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in downloadFileByVersionId at line 236 of src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java
Line	87	265
Object	cmapproverMail	setHeader

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody CMApproverMail cmapproverMail)

```
....
87.   public String sendCMAEmail(HttpServletRequest request,
@RequestBody CMApproverMail cmapproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java

Method public void downloadFileByVersionId(String fileId, HttpServletRequest request,

```
....
265.                    response.setHeader(headerKey, headerValue);
```

HTTP Response Splitting\Path 2:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=2
Status	Recurrent
Detection Date	8/24/2021 2:12:58 PM

Method sendCMAEmail at line 106 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets user input from the sltApproverMail element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in an HTTP response header in downloadFileByVersionId at line 236 of src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java. This may enable an HTTP Response Splitting attack, in certain older versions that do not mitigate this attack.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java
Line	106	265
Object	sltApproverMail	setHeader

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody SltApproverMail sltApproverMail)

```
....
106.        public String sendCMAEmail(HttpServletRequest request,
@RequestBody SltApproverMail sltApproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/service/BoxServiceImpl.java

Method public void downloadFileByVersionId(String fileId, HttpServletRequest request,

```
....
265.                    response.setHeader(headerKey, headerValue);
```

Unchecked Input for Loop Condition

Query Path:
Java\Cx\Java Medium Threat\Unchecked Input for Loop Condition Version:7

Categories

ASD STIG 4.10: APSC-DV-002530 - CAT II The application must validate all input.
OWASP Top 10 API: API4-Lack of Resources and Rate Limiting

Description

Unchecked Input for Loop Condition\Path 1:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=8
Status	Recurrent
Detection Date	8/24/2021 2:13:19 PM

Method approveRejectCashMap at line 122 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets user input from element deal . This element's value flows through the code without being validated, and is eventually used in a loop condition in statusInsert at line 1189 of src/main/java/com/ge/treasury/myfunding/service/MyFundingAutomationServiceImpl.java. This constitutes an Unchecked Input for Loop Condition.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/service/MyFundingAutomationServiceImpl.java
Line	122	1198
Object	deal	it

Code Snippet

File Name	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java
Method	public ResponseEntity<Deal> approveRejectCashMap(HttpServletRequest request, @RequestBody Deal deal)

```
....
122.        public ResponseEntity<Deal>
approveRejectCashMap(HttpServletRequest request, @RequestBody Deal deal)
```

File Name	src/main/java/com/ge/treasury/myfunding/service/MyFundingAutomationServiceImpl.java
Method	private void statusInsert(Deal deal, HashMap<String, List<HashMap<String, String>>> snuActorDetailsListMap) {

```
....
1198.             while (it.hasNext()) {
```

Unchecked Input for Loop Condition\Path 2:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=9
Status	Recurrent
Detection Date	8/24/2021 2:13:19 PM

Method editDeal at line 271 of src/main/java/com/ge/treasury/myfunding/controllers/MyFundingRequestController.java gets user input from element deal . This element's value flows through the code without being validated, and is eventually used in a loop condition in statusInsert at line 1189 of src/main/java/com/ge/treasury/myfunding/service/MyFundingAutomationServiceImpl.java. This constitutes an Unchecked Input for Loop Condition.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingRequestController.java	src/main/java/com/ge/treasury/myfunding/service/MyFundingAutomationServiceImpl.java
Line	271	1198
Object	deal	it

Code Snippet

File Name	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingRequestController.java
Method	public ResponseEntity<Deal> editDeal(HttpServletRequest request, @RequestBody Deal deal)

```
....
271.        public ResponseEntity<Deal> editDeal(HttpServletRequest
request, @RequestBody Deal deal)
```

File Name	src/main/java/com/ge/treasury/myfunding/service/MyFundingAutomationServiceImpl.java
Method	private void statusInsert(Deal deal, HashMap<String, List<HashMap<String, String>>> snuActorDetailsListMap) {

```
....
1198.             while (it.hasNext()) {
```

Use of a One Way Hash without a Salt

Query Path:
Java\Cx\Java Medium Threat\Use of a One Way Hash without a Salt Version:2

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Media Protection
NIST SP 800-53: SC-13 Cryptographic Protection (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Use of a One Way Hash without a Salt\Path 1:

Severity	Medium
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=14
Status	Recurrent
Detection Date	5/4/2021 2:51:20 PM

The application protects passwords with digest in setKey, of src/main/java/com/ge/treasury/myfunding/utils/MyFundingHelper.java at line 2251, using a cryptographic hash ""SHA-1"". However, the code does not salt the hash with an unpredictable, random value, allowing an attacker to reverse the hash value.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/utils/MyFundingHelper.java	src/main/java/com/ge/treasury/myfunding/utils/MyFundingHelper.java
Line	2255	2256
Object	""SHA-1""	digest

Code Snippet
File Name src/main/java/com/ge/treasury/myfunding/utils/MyFundingHelper.java
Method public static void setKey(String myKey) {

```
....
2255.                                    sha =
MessageDigest.getInstance("SHA-1");
2256.                                    key = sha.digest(key);
```

Trust Boundary Violation in Session Variables

Query Path:
Java\Cx\Java Low Visibility\Trust Boundary Violation in Session Variables Version:4

Categories

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A5-Broken Access Control
ASD STIG 4.10: APSC-DV-002360 - CAT II The application must isolate security functions from non-security functions.

Description

Trust Boundary Violation in Session Variables\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=16
Status	Recurrent

Detection Date	7/8/2021 1:34:03 PM

Method getWssServiceAmount at line 66 of src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java gets user input from element facilityID. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in createBodyUsingVMTemplate at line 164 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	67	193
Object	facilityID	getValue

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java

Method public Object getWssServiceAmount(HttpServletRequest request,

```
....
67.                 @RequestParam(value = "facilityID") String facilityID,
@RequestParam(value = "eventType") String eventType, @RequestParam(value
= "subType") String subType) {
```

File Name src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java

Method private String createBodyUsingVMTemplate(String templateFile,

```
....
193.                    context.put(tokenEntry.getKey(),
tokenEntry.getValue());
```

Trust Boundary Violation in Session Variables\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=17
Status	Recurrent
Detection Date	7/8/2021 1:34:03 PM

Method getWssServiceAmount at line 66 of src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java gets user input from element facilityID. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in createBodyUsingVMTemplate at line 164 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	67	193
Object	facilityID	getKey

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/MyFundingTMSController.java

Method	public Object getWssServiceAmount(HttpServletRequest request,

```
....
67.                 @RequestParam(value = "facilityID") String facilityID,
@RequestParam(value = "eventType") String eventType, @RequestParam(value
= "subType") String subType) {
```

File Name	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Method	private String createBodyUsingVMTemplate(String templateFile,

```
....
193.                    context.put(tokenEntry.getKey(),
tokenEntry.getValue());
```

Trust Boundary Violation in Session Variables\Path 3:

Severity	Low
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=18
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

Method sendCMAEmail at line 106 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets user input from element sltApproverMail. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in sendMailWithAttachment at line 255 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	106	321
Object	sltApproverMail	getValue

Code Snippet	
File Name	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java
Method	public String sendCMAEmail(HttpServletRequest request, @RequestBody SltApproverMail sltApproverMail)

```
....
106.        public String sendCMAEmail(HttpServletRequest request,
@RequestBody SltApproverMail sltApproverMail)
```

File Name	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Method	public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
321.                context.put(tokenEntry.getKey(),
tokenEntry.getValue());
```

Trust Boundary Violation in Session Variables\Path 4:

Severity	Low
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=19
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

Method sendCMAEmail at line 87 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets user input from element cmapproverMail. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in sendMailWithAttachment at line 255 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	87	321
Object	cmapproverMail	getValue

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody CMApproverMail cmapproverMail)

```
....
87.   public String sendCMAEmail(HttpServletRequest request,
@RequestBody CMApproverMail cmapproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java

Method public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
321.                context.put(tokenEntry.getKey(),
tokenEntry.getValue());
```

Trust Boundary Violation in Session Variables\Path 5:

Severity	Low
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=20
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

Method sendCMAEmail at line 106 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets user input from element

sltApproverMail. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in sendMailWithAttachment at line 255 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	106	321
Object	sltApproverMail	getKey

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody SltApproverMail sltApproverMail)

```
....
106.        public String sendCMAEmail(HttpServletRequest request,
@RequestBody SltApproverMail sltApproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java

Method public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
321.                context.put(tokenEntry.getKey(),
tokenEntry.getValue());
```

Trust Boundary Violation in Session Variables\Path 6:

Severity	Low
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=21
Status	Recurrent
Detection Date	8/24/2021 2:13:23 PM

Method sendCMAEmail at line 87 of src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java gets user input from element cmapproverMail. This element's value flows through the code without being properly sanitized or validated and is eventually stored in the server-side Session object, in sendMailWithAttachment at line 255 of src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java. This constitutes a Trust Boundary Violation.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java	src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java
Line	87	321
Object	cmapproverMail	getKey

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/controllers/CashMapApprovalController.java

Method public String sendCMAEmail(HttpServletRequest request, @RequestBody CMApproverMail cmapproverMail)

```
....
87.   public String sendCMAEmail(HttpServletRequest request,
@RequestBody CMApproverMail cmapproverMail)
```

File Name src/main/java/com/ge/treasury/myfunding/email/EmailSenderImpl.java

Method public void sendMailWithAttachment(List<String> sltList, List<String> ccMailList, String emailSubject, String template, Map<String, Object> emailTokenValues, List<Document> docList, boolean b) throws IOException {

```
....
321.              context.put(tokenEntry.getKey(),
tokenEntry.getValue());
```

Heap Inspection

Query Path:
Java\Cx\Java Low Visibility\Heap Inspection Version:7

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Media Protection
NIST SP 800-53: SC-4 Information in Shared Resources (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure
ASD STIG 4.10: APSC-DV-002330 - CAT II The application must protect the confidentiality and integrity of stored information when required by DoD policy or the information owner.

Description

Heap Inspection\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://checkmarx.digital.ge.com/CxWebClient/ViewerMain.aspx?scanid=911212&projectid=30258&pathid=15
Status	Recurrent
Detection Date	5/4/2021 2:51:12 PM

Method password; at line 100 of src/main/java/com/ge/treasury/myfunding/utils/MyFundingEmailHelper.java defines password, which is designated to contain user passwords. However, while plaintext passwords are later assigned to password, this variable is never cleared from memory.

	Source	Destination
File	src/main/java/com/ge/treasury/myfunding/utils/MyFundingEmailHelper.java	src/main/java/com/ge/treasury/myfunding/utils/MyFundingEmailHelper.java
Line	100	100
Object	password	password

Code Snippet

File Name src/main/java/com/ge/treasury/myfunding/utils/MyFundingEmailHelper.java

Method private String password;

```
....
100.      private String password;
```

Reflected XSS All Clients

Risk

What might happen

A successful XSS exploit would allow an attacker to rewrite web pages and insert malicious scripts which would alter the intended output. This could include HTML fragments, CSS styling rules, arbitrary JavaScript, or references to third party code. An attacker could use this to steal users' passwords, collect personal data such as credit card details, provide false information, or run malware. From the victim's point of view, this is performed by the genuine website, and the victim would blame the site for incurred damage. The attacker could use social engineering to cause the user to send the website modified input, which will be returned in the requested web page.

Cause

How does it happen

The application creates web pages that include untrusted data, whether from user input, the application's database, or from other external sources. The untrusted data is embedded directly in the page's HTML, causing the browser to display it as part of the web page. If the input includes HTML fragments or JavaScript, these are displayed too, and the user cannot tell that this is not the intended page. The vulnerability is the result of directly embedding arbitrary data without first encoding it in a format that would prevent the browser from treating it like HTML or code instead of plain text.
Note that an attacker can exploit this vulnerability either by modifying the URL, or by submitting malicious data in the user input or other request fields.

General Recommendations

How to avoid it

- Fully encode all dynamic data, regardless of source, before embedding it in output.
- Encoding should be context-sensitive. For example:
  - HTML encoding for HTML content
  - HTML Attribute encoding for data output to attribute values
  - JavaScript encoding for server-generated JavaScript
- It is recommended to use the platform-provided encoding functionality, or known security libraries for encoding output.
- Implement a Content Security Policy (CSP) with explicit whitelists for the application's resources only.
- As an extra layer of protection, validate all untrusted data, regardless of source (note this is not a replacement for encoding). Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
  - Data type
  - Size
  - Range
  - Format
  - Expected values
- In the `Content-Type` HTTP response header, explicitly define character encoding (charset) for the entire page.
- Set the `HTTPOnly` flag on the session cookie for "Defense in Depth", to prevent any successful XSS exploits from stealing the cookie.

Source Code Examples

Java

Returning Data To Clients Without Encoding

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

      PrintWriter out = response.getWriter();
      String loc = request.getParameter("location");

      out.println("<h1> Location: " + loc + "<h1>");
```

```
}
```

Returning Data to Clients After Encoding The User Input

```
// Using HtmlEscapers by Google Guava

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

      PrintWriter out = response.getWriter();
      String loc = request.getParameter("location");
      String escapedLocation = HtmlEscapers.htmlEscaper().escape(loc);

      out.println("<h1> Location: " + escapedLocation + "<h1>");
}
```

HTTP Response Splitting

Risk

What might happen

If the header setting code is of a vulnerable version, an attacker could:

- Arbitrarily change the application server's response header to a victim's HTTP request by manipulating headers
- Arbitrarily change the application server's response body by injecting two consecutive line breaks, which may result in Cross-Site Scripting (XSS) attacks
- Cause cache poisoning, potentially controlling any site's HTTP responses going through the same proxy as this application.

Cause

How does it happen

Since user input is being used in an HTTP response header, an attacker could include NewLine characters to make the header look like multiple headers with engineered content, potentially making the response look like multiple responses (for example, by engineering duplicate content-length headers). This can cause an organizational proxy server to provide the second, engineered response to a victim's subsequent request; or, if the proxy server also performs response caching, the attacker can send an immediate subsequent request to another site, causing the proxy server to cache the engineered response as a response from this second site and to later serve the response to other users.

Many modern web frameworks mitigate this issue, by offering sanitization for new line characters in strings inserted into headers by default. However, since many older versions of web frameworks fail to automatically mitigate this issue, manual sanitization of input may be required.

General Recommendations

How to avoid it

1. Validate all input, regardless of source (including cookies). Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
2. Additionally, remove or URL-encode all special (non-alphanumeric) user input before including it in the response header.
3. Make sure to use an up-to-date framework.

Source Code Examples

Java

User Input Affects a Response Header

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      // Vulnerable in legacy versions of Java
    String username = request.getParameter("username");
    response.setContentType("text/html");
    Cookie cookie = new Cookie("user", username);
    cookie.setMaxAge(3600);
    response.addCookie(cookie);
}
```

User Input URL Encoded Prior Setting A Response Header

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
```

```
ServletException, IOException {
      // Defense In Depth - URLEncode user input before setting a response header
    String username = request.getParameter("username");
    response.setContentType("text/html");
    Cookie cookie = new Cookie("user", URLEncoder.encode(username, "UTF-8"));
    cookie.setMaxAge(3600);
    response.addCookie(cookie);
}
```

Input Path Not Canonicalized

Risk

What might happen

An attacker could define arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing denial of service (DoS).

Cause

How does it happen

The application uses user input in the file path for accessing files on the application server's local disk.

General Recommendations

How to avoid it

1. Ideally, avoid depending on dynamic data for file selection.
2. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
3. Accept dynamic data only for the filename, not for the path and folders.
4. Ensure that file path is fully canonicalized.
5. Explicitly limit the application to use a designated folder that is separate from the applications binary folder.
6. Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

Source Code Examples

Java

Absolute Path Traversal in "filename" Parameter

```
private String getFileContents(HttpServletRequest request) throws ServletException,
FileNotFoundException, IOException {
      String filename = request.getParameter("filename");
      Path path = Paths.get(filename);
      byte[] fileContentBytes = Files.readAllBytes(path);
      String fileContents = new String(fileContentBytes, FILE_CONTENT_ENCODING_STRING);
      return fileContents;
}
```

Relative Path Traversal in "filename" Parameter

```
private String getFileContents(HttpServletRequest request) throws ServletException,
FileNotFoundException, IOException {
      String filename = request.getParameter("filename");
      Path path = Paths.get(SERVED_FILES_DIR + filename);
      byte[] fileContentBytes = Files.readAllBytes(path);
      String fileContents = new String(fileContentBytes, FILE_CONTENT_ENCODING_STRING);
      return fileContents;
```

```
}
```

Path Traversal Mitigated via Sanitization of Path Variable

```
private static String sanitizePathTraversal(String filename) {
      Path p = Paths.get(filename);
      return p.getFileName().toString();
}

private String getFileContents_fixed(HttpServletRequest request) throws ServletException,
FileNotFoundException, IOException {
      String filename = sanitizePathTraversal(request.getParameter("filename")); // Ensures
access only to files in a given folder, no traversal
      Path path = Paths.get(SERVED_FILES_DIR + filename);
      byte[] fileContentBytes = Files.readAllBytes(path);
      String fileContents = new String(fileContentBytes, FILE_CONTENT_ENCODING_STRING);
      return fileContents;
}
```

Unchecked Input for Loop Condition

Risk

What might happen

An attacker could input a very high value, potentially causing a denial of service (DoS).

Cause

How does it happen

The application performs some repetitive task in a loop, and defines the number of times to perform the loop according to user input. A very high value could cause the application to get stuck in the loop and to be unable to continue to other operations.

General Recommendations

How to avoid it

Ideally, don't base a loop on user-provided data. If it is necessary to do so, the user input must be first validated and its range should be limited.

Source Code Examples

Java

Loop Condition Is Not Bounded By Any Value

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    int loopCount = 0;
    try{
        loopCount = Integer.parseInt(request.getParameter("loopCount"));
    } catch(NumberFormatException e){
        return DEFAULT_VAL;
    }
    for(int i=0; i < loopCount; i++){
        //Do Something
    }
}
```

Loop Condition is Bounded With MAX_LOOPS

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    int loopCount = 0;
    try{
        loopCount = Integer.parseInt(request.getParameter("loopCount"));
    } catch(NumberFormatException e){
        return DEFAULT_VAL;
    }
    if(loopCount > MAX_LOOPS){
        loopCount = MAX_LOOPS;
    }
    for(int i=0; i < loopCount; i++){
        //Do Something
    }
}
```

Privacy Violation

Risk

What might happen

A user's personal information could be stolen by a malicious programmer, or an attacker that intercepts the data.

Cause

How does it happen

The application sends user information, such as passwords, account information, or credit card numbers, outside the application, such as writing it to a local text or log file or sending it to an external web service.

General Recommendations

How to avoid it

1. Personal data should be removed before writing to logs or other files.
2. Review the need and justification of sending personal data to remote web services.

Source Code Examples

Java

Leaking a Password Back to the User Constitutes a Privacy Violation

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      PrintWriter out = response.getWriter();
      HttpSession session = request.getSession();
      boolean isAuthenticated = session.getAttribute("isAuthenticated");
      if (isAuthenticated) {
            byte[] password = request.getParameter("password").getBytes();
            updatePassword(session, password);
      out.println("New password is " + (new String(password));
      } else {
            out.println("Authentication Failed");
      }
}
```

Use of a One Way Hash without a Salt

Risk

What might happen

If an attacker gains access to the hashed passwords, she would likely be able to reverse the hash due to this weakness, and retrieve the original password. Once the passwords are discovered, the attacker can impersonate the users, and take full advantage of their privileges and access their personal data. Furthermore, this would likely not be discovered, as the attacker is being identified solely by the victims' credentials.

Cause

How does it happen

Typical cryptographic hashes, such as SHA-1 and MD5, are incredibly fast. Combined with attack techniques such as precomputed Rainbow Tables, it is relatively easy for attackers to reverse the hashes, and discover the original passwords. Lack of a unique, random salt added to the password makes brute force attacks even simpler.

General Recommendations

How to avoid it

Generic Guidance: - Always use strong, modern algorithms for encryption, hashing, and so on. - Do not use weak, outdated, or obsolete algorithms. - Ensure you select the correct cryptographic mechanism according to the specific requirements.
Specific Recommendations: - Passwords should be protected using a password hashing algorithm, instead of a general cryptographic hash. This includes adaptive hashes such as bcrypt, scrypt, PBKDF2 and Argon2. - Tune the work factor, or cost, of the adaptive hash function according to the designated environment and risk profile. - Do not use a regular cryptographic hash, such as SHA-1 or MD5, to protect passwords, as these are too fast. - If it is necessary to use a common hash to protect passwords, add several bytes of unique, random data ("salt") to the password before hashing it. Store the salt with the hashed password, and do not reuse the same salt for multiple passwords.

Source Code Examples

Java

Unsalted Hashed Password

```
private String protectPassword(String password) {
      byte[] data = password.getBytes();
      byte[] hash = null;

      MessageDigest md = MessageDigest.getInstance("MD5");
      hash = md.digest(data);

      return Base64.getEncoder().encodeToString(hash);
}
```

Fast Hash with Salt

```
private String protectPassword(String password) {
      byte[] data = password.getBytes("UTF-8");
      byte[] hash = null;

      try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");

            SecureRandom rand = new SecureRandom();
            byte[] salt = new byte[32];
            rand.nextBytes(salt);
```

```
            md.update(salt);
            md.update(data);

            hash = md.digest();
    }
     catch (GeneralSecurityException gse) {
            handleCryptoErrors(gse);
    }
     finally {
            Arrays.fill(data, 0);
    }

     return Base64.getEncoder().encodeToString(hash);
}
```

Slow, Adaptive Password Hash

```
private String protectPassword(String password) {
     byte[] data = password.getBytes("UTF-8");
     byte[] hash = null;

     try {
            SecureRandom rand = new SecureRandom();
            byte[] salt = new byte[32];
            rand.nextBytes(salt);

            SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
            PBEKeySpec spec = new PBEKeySpec(data, salt, ITERATION_COUNT, KEY_LENGTH);
            // ITERATION_COUNT should be configured by environment, KEY_LENGTH should be 256
            SecretKey key = skf.generateSecret(spec);

            hash = key.getEncoded();
    }
     catch (GeneralSecurityException gse) {
            handleCryptoErrors(gse);
    }
     finally {
            Arrays.fill(data, 0);
    }

     return Base64.getEncoder().encodeToString(hash);
}
```

SSRF

Risk

What might happen

An attacker can abuse this flaw to make arbitrary requests, originating from the application server. This can be exploited to scan internal services; proxy attacks into a protected network; bypass network controls; download unauthorized files; access internal services and management interfaces; and possibly control the contents of requests and even steal server credentials.

Cause

How does it happen

The application accepts a URL (or other data) from the user, and uses this to make a request to another remote server. However, the attacker can inject an arbitrary URL into the request, causing the application to connect to any server the attacker wants. Thus, the attacker can abuse the application to gain access to services that would not otherwise be accessable, and cause the request to ostensibly originate from the application server.

General Recommendations

How to avoid it

- Do not connect to arbitrary services based on user input.
- If possible, the application should have the user's browser retrieve the desired information directly.
- If it is necessary for the application to proxy the request on the server, explicitly whitelist the allowed target URLs, and do not include any sensitive server information.

Source Code Examples

Java

Retrieve and Display Contents of URL

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
  if (request.getParameterMap().containsKey("url")) {
    String url = request.getParameter("url");
    PrintWriter out = response.getWriter();
    URL u = new URL(url);
    InputStreamReader sr = new InputStreamReader(u.openConnection().getInputStream());
    BufferedReader reader = new BufferedReader(sr);
    String line = reader.readLine();
    while (line != null) {
      out.write(line);
      line = reader.readLine();
    }
  }
}
```

Validate and Redirect User's Browser

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
  if (request.getParameterMap().containsKey("url")) {
    String url = request.getParameter("url");
    if (url.startsWith("/") && !url.startsWith("//")) {
      response.sendRedirect(url);
    } else {
      response.sendRedirect("/");
    }
  }
```

```
}
```

Heap Inspection

Risk

What might happen

All variables stored by the application in unencrypted memory can potentially be retrieved by an unauthorized user, with privileged access to the machine. For example, a privileged attacker could attach a debugger to the running process, or retrieve the process's memory from the swapfile or crash dump file. Once the attacker finds the user passwords in memory, these can be reused to easily impersonate the user to the system.

Cause

How does it happen

String variables are immutable - in other words, once a string variable is assigned, its value cannot be changed or removed. Thus, these strings may remain around in memory, possibly in multiple locations, for an indefinite period of time until the garbage collector happens to remove it. Sensitive data, such as passwords, will remain exposed in memory as plaintext with no control over their lifetime.

While it may still be possible to retrieve data from memory, even if it uses a mutable container that is cleared, or retrieve a decryption key and decrypt sensitive data from memory - layering sensitive data with these types of protection would significantly increase the required effort to do so. By setting a high bar for retrieving sensitive data from memory, and reducing the amount and exposure of sensitive data in memory, an adversary is significantly less likely to succeed in obtaining valuable data.

General Recommendations

How to avoid it

When it comes to avoiding Heap Inspection, it is important to note that, given any read access to memory or a memory dump of an application, it is always likely to disclose some sensitive data to an adversary - these suggestions are part of defense-in-depth principles for protection of sensitive data in cases where such memory read access is successfully obtained. These recommendations will enable significant reduction in the lifespan and exposure of sensitive data in memory; however - given enough time, effort and unlimited access to memory, they will only go so far in protecting sensitive data being used by the application. The only way to handle Heap Inspection issues is to minimize and reduce data exposure, and obscure it in memory wherever possible.

- Do not store sensitive data, such as passwords or encryption keys, in memory in plain-text, even for a short period of time.
- Prefer to use specialized classes that store encrypted data in memory to ensure it cannot be trivially retrieved from memory.
- When required to use sensitive data in its raw form, temporarily store it in mutable data types, such as byte arrays, to reduce readability from memory, and then promptly zeroize the memory locations, to reduce exposure duration of this data while in memory.
- Ensure that memory dumps are not exchanged with untrusted parties, as even by ensuring all of the above - it may still be possible to reverse-engineer encrypted containers, or retrieve bytes of sensitive data from memory and rebuild it.
- In Java, do not store passwords in immutable strings - prefer using an encrypted memory object, such as SealedObject.

Source Code Examples

Java

Plaintext Password in Immutable String

```
class Heap_Inspection
{
  private String password;

  public void setPassword(String password)
 {
      this.password = password;
  }
}
```

Password Protected in Memory

```
class Heap_Inspection_Fixed
{
  private SealedObject password;

  public void setPassword(Character[] input)
 {
      Key key = getKeyFromConfiguration();
             Cipher c = Cipher.getInstance(CIPHER_NAME);
             c.init(Cipher.ENCRYPT_MODE, key);
             List<Character> characterList = Arrays.asList(input);
             password = new SealedObject((Serializable) characterList, c);
             Arrays.fill(input, '\0'); // Zero out input. Will also overwrite the values in
characterList by reference.
  }
}
```

Trust Boundary Violation in Session Variables

Risk

What might happen

Code that reads from Session variables may trust them as server-side variables, but they may have been tainted by user inputs. This can lead to tampering with parameters used to authenticate or authorize users. Further, tainted Session variables offer an additional attack surface against the application - if untrusted data taints a Session variable, and that Session variable is then used elsewhere without sanitization as if it were trusted, it could lead to further attacks such as Cross-Site Scripting, SQL Injection and more.

Cause

How does it happen

Server-side Session variables, or objects, are values assigned to a specific session, which is associated with a specific user. Often, they hold data relevant to that user's session, such as specific identifiers, user-type, authorization, authentication information and more. As such, the paradigm often associated to the Session object is that its contents can be trusted, as users cannot generally set these values themselves.

The application places user input, which is untrusted data, in the server-side Session object, which is considered a trusted location. This could lead developers to treat untrusted data as trusted.

General Recommendations

How to avoid it

1. Validate and sanitize all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size
    - Range
    - Format
    - Expected values
2. Don't mix untrusted user input with trusted data.

Source Code Examples

Java

Setting User Role by Relying on User Input, Allowing Users to Tamper Its Value and Elevate Their Privilege

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      HttpSession session = request.getSession();
      String username = request.getParameter("username");
      byte[] password = request.getParameter("password").getBytes();
      if (isAuthenticated(username, password)) {
            String role = request.getParameter("role"); // Role can be tampered by user
            session.setAttribute("isAuthenticated",true);
            session.setAttribute("role", role);
            // Render page //
      } else {
            // Authentication error //
      }
}
```

Derive User Role from An Internal Mechanism Based On The Current User

```
public void doPost (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
```

```
      HttpSession session = request.getSession();
      String username = request.getParameter("username");
      byte[] password = request.getParameter("password").getBytes();
      if (isAuthenticated(username, password)) {
            String role = getUserRole(username); // Role is not derived from user input, but
some post authentication mechanism
            session.setAttribute("isAuthenticated",true);
            session.setAttribute("role", role);
            // Render page //
      } else {
            // Authentication error //
      }
}
```

Scanned Languages

Language	Hash Number	Change Date
Java	3775534314482291	8/15/2021
JavaScript	0662353247298292	8/15/2021
VbScript	2310601648786149	8/15/2021
Python	6957663651711215	8/15/2021
Common	0574372965153514	8/15/2021