How-to check your application for security flaws (User Input Security in AIP Console)

Purpose

The purpose of this How-To post is to explain how to check your application for security flaws.
The different actions presented here have been done with **AIP Console 1.9.0**.

Use case

As an **Application Owner** you on-boarded an application in AIP Console and you want to check if contains any security flaw. This can be done by using the User Input Security Analysis feature and, for that, you must have analyzed a version and generated a snapshot.

About User Input Security Analyzer

The **User Input Security Analyzer** is a very powerful feature delivered within the CAST AIP platform. It performs a deep source code scan, to identify all paths from user input methods to specific methods accessing data and finds out the paths that have not been secured. This contributes to a specific subset of security rules supported by the CAST AIP platform. The rest of the security rules (there are a lot) are not based on the dataflow technique.

Since version 8.3.8 of CAST AIP performance has increased significantly:

- In terms of run-time
- In memory consumption
- Accuracy in results, with a very low number of false negatives.

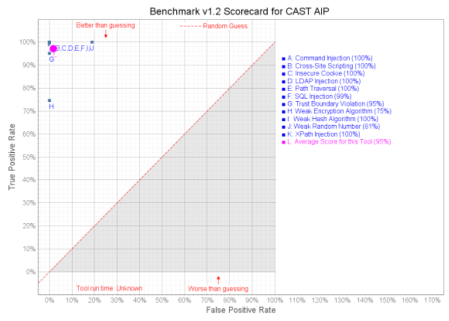
See below the latest CAST OWASP benchmark:

empty
empty

OWASP Benchmark Scorecard for CAST AIP v8.3.18 (SAST)

The OWASP Benchmark is a test suite designed to evaluate the speed, coverage, and accuracy of automated vulnerability detection tools. Without the ability to measure these tools, it is difficult to understand their strengths and weaknesses, and compare them to each other. The Benchmark contains thousands of test cases that are fully runnable and exploitable. The following is the scorecard for the tool CAST AIP against version 1.2 of the Benchmark. It shows how well this tool finds true positives and avoids false positives in the Benchmark test cases.

For more information, please visit the OWASP Benchmark Project Site.



Statistics

Tool elapsed analysis time	Unknown

OWASP Benchmark v1.2 Home Tools ▾ Vulnerabilities ▾ Guide

Download raw results	Actual Results

Detailed Results

Category	CWE #	TP	FN	TN	FP	Total	TPR	FPR	Score
Command Injection	78	126	0	125	0	251	100,00%	0,00%	100,00%
Cross-Site Scripting	79	246	0	209	0	455	100,00%	0,00%	100,00%
Insecure Cookie	614	36	0	31	0	67	100,00%	0,00%	100,00%
LDAP Injection	90	27	0	32	0	59	100,00%	0,00%	100,00%
Path Traversal	22	133	0	135	0	268	100,00%	0,00%	100,00%
SQL Injection	89	269	3	232	0	504	98,90%	0,00%	98,90%
Trust Boundary Violation	501	79	4	43	0	126	95,18%	0,00%	95,18%
Weak Encryption Algorithm	327	97	33	116	0	246	74,62%	0,00%	74,62%
Weak Hash Algorithm	328	129	0	107	0	236	100,00%	0,00%	100,00%
Weak Random Number	330	218	0	223	52	493	100,00%	18,91%	81,09%
XPath Injection	643	15	0	20	0	35	100,00%	0,00%	100,00%
Totals*		**1375**	**40**	**1273**	**52**	**2740**			
Overall Results*							**97,15%**	**1,72%**	**95,43%**

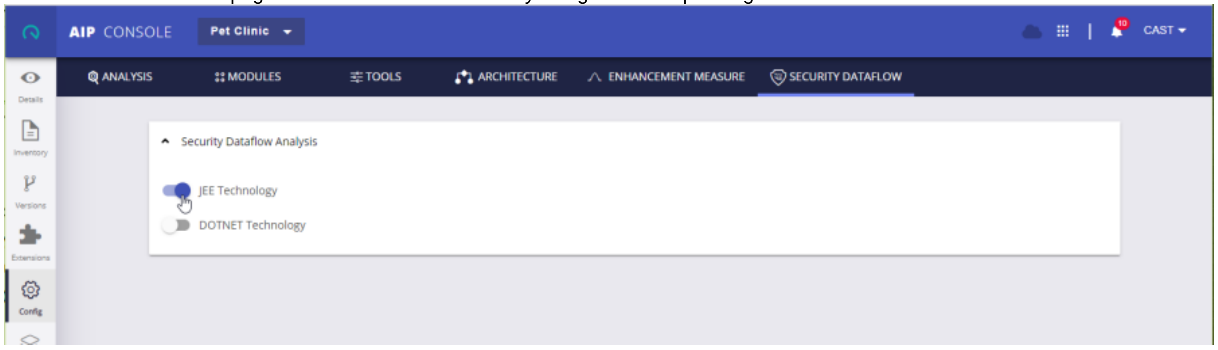
*-The Overall Results are averages across all the vulnerability categories. You can't compute these averages by simply calculating the TPR and FPR rates using the values in the Totals row. If you did that, categories with larger number of tests would carry more weight than categories with less tests. The proper calculation of the Overall Results is to add up all the TPR, FPR, and Score values, and then divide by the number of vulnerability categories, which is how they are calculated.

Key

Common Weakness Enumeration (CWE)	The primary MITRE CWE number for this vulnerability category.
True Positive (TP)	Tests with real vulnerabilities that were correctly reported as vulnerable by the tool.
False Negative (FN)	Tests with real vulnerabilities that were not correctly reported as vulnerable by the tool.
True Negative (TN)	Tests with fake vulnerabilities that were correctly not reported as vulnerable by the tool.
False Positive (FP)	Tests with fake vulnerabilities that were incorrectly reported as vulnerable by the tool.
True Positive Rate (TPR) = TP / (TP + FN)	The rate at which the tool correctly reports real vulnerabilities. Also referred to as Recall, as defined at Wikipedia.
False Positive Rate (FPR) = FP / (FP + TN)	The rate at which the tool incorrectly reports fake vulnerabilities as real.
Score = TPR - FPR	Normalized distance from the random guess line.

Enabling the feature

The User Input Security Analysis is available for only JEE and DotNet technologies. If your application contains these technologies, then go to the **Config** / **SECURITY DATAFLOW** page and activate the detection by using the corresponding slider:



Security Dataflow Analysis

JEE Technology [✓ on]
DOTNET Technology

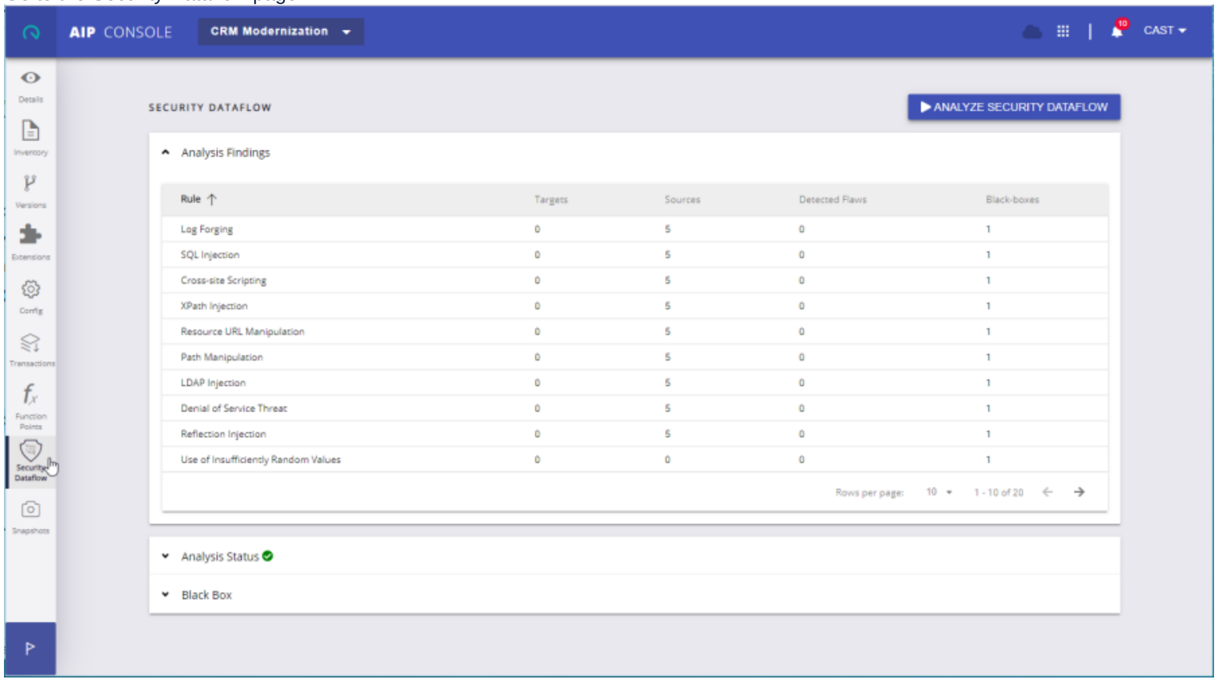
Until you deactivate the detection, each time you will run a scan to produce a snapshot, the technology analyzers will generate intermediate code (aka Cast-IL) for all source files (of the considered technology) they will parse. This intermediate code will be then processed by the Security Dataflow Analyzer to search for unsecure paths between user input sources and user input targets.

User Input Security analysis findings

Once the scan completes, you can review the intermediate results produced by the Security Dataflow Analyzer.

ⓘ Intermediate results, produced by the Security Dataflow Analyzer, are used as input for the security rules.

Go to the Security Dataflow page:



SECURITY DATAFLOW

Analysis Findings

Rule	Targets	Sources	Detected Flaws	Black-boxes
Log Forging	0	5	0	1
SQL Injection	0	5	0	1
Cross-site Scripting	0	5	0	1
XPath Injection	0	5	0	1
Resource URL Manipulation	0	5	0	1
Path Manipulation	0	5	0	1
LDAP Injection	0	5	0	1
Denial of Service Threat	0	5	0	1
Reflection Injection	0	5	0	1
Use of Insufficiently Random Values	0	0	0	1

Analysis Status

Black Box

The page shows 3 sections:

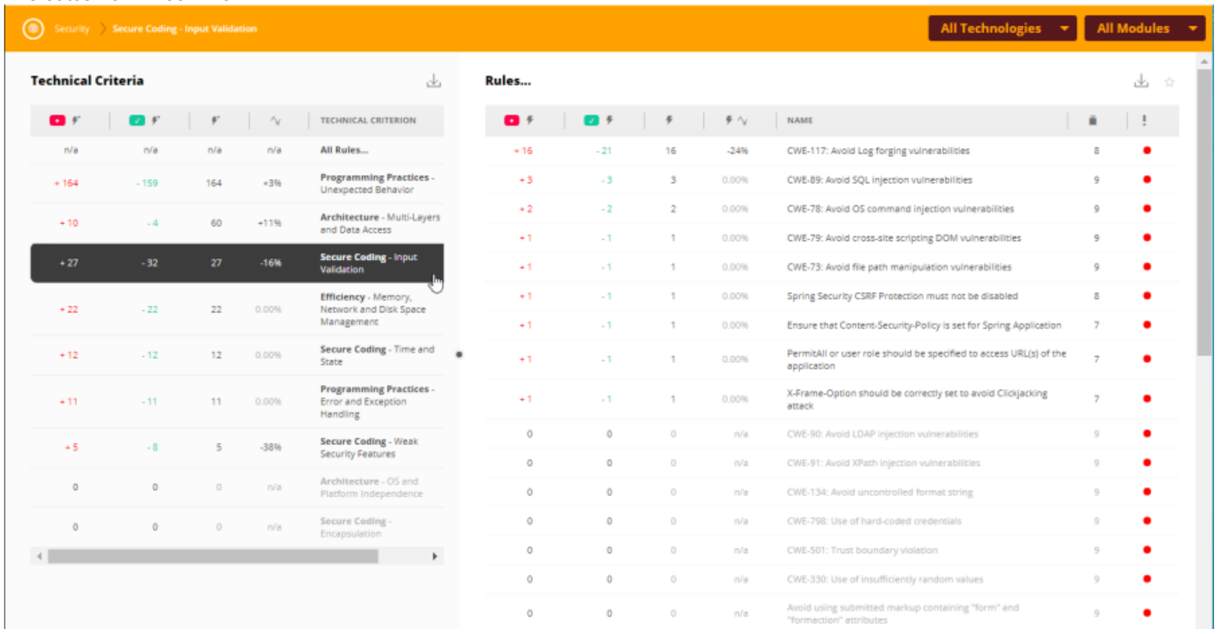
- Analysis Findings
- Analysis Status and
- Block Box

The first one presents the searches with the number of user input targets and sources, and the number of flaws that have been detected.
The second section indicates if the security dataflow analysis went well or not. In case the Dataflow analysis failed, it means that the values presented in above section may be not relevant.
The third section is dedicated to customizing the User Input Security Analysis with additional black-boxes. This requires the version 8.3.11 of CAST AIP to be installed on the Analysis Node. You will find more details on how to create a black-box in User Input Security - manually configuring blackbox methods.
You can restart the detection by clicking the below button:

▶ ANALYZE SECURITY DATAFLOW

This will run only the Security Dataflow Analysis as the intermediate code is already available. Once completed, Analysis Findings table will be refreshed. However, it is only for tuning purpose. To propagate these results to the dashboard, you will have to produce a new analysis.

The outcome will look like:



Technical Criteria

				TECHNICAL CRITERION
n/a	n/a	n/a	n/a	**All Rules...**
+ 164	- 159	164	+3%	**Programming Practices** - Unexpected Behavior
+ 10	- 4	60	+11%	**Architecture** - Multi-Layers and Data Access
+ 27	- 32	27	-16%	**Secure Coding** - Input Validation
+ 22	- 22	22	0.00%	**Efficiency** - Memory, Network and Disk Space Management
+ 12	- 12	12	0.00%	**Secure Coding** - Time and State
+ 11	- 11	11	0.00%	**Programming Practices** - Error and Exception Handling
+ 5	- 8	5	-38%	**Secure Coding** - Weak Security Features
0	0	0	n/a	**Architecture** - OS and Platform Independence
0	0	0	n/a	**Secure Coding** - Encapsulation

Rules...

				NAME		!
+ 16	- 21	16	-24%	CWE-117: Avoid Log forging vulnerabilities	8	●
+ 3	- 3	3	0.00%	CWE-89: Avoid SQL injection vulnerabilities	9	●
+ 2	- 2	2	0.00%	CWE-78: Avoid OS command injection vulnerabilities	9	●
+ 1	- 1	1	0.00%	CWE-79: Avoid cross-site scripting DOM vulnerabilities	9	●
+ 1	- 1	1	0.00%	CWE-73: Avoid file path manipulation vulnerabilities	9	●
+ 1	- 1	1	0.00%	Spring Security CSRF Protection must not be disabled	8	●
+ 1	- 1	1	0.00%	Ensure that Content-Security-Policy is set for Spring Application	7	●
+ 1	- 1	1	0.00%	PermitAll or user role should be specified to access URL(s) of the application	7	●
+ 1	- 1	1	0.00%	X-Frame-Option should be correctly set to avoid Clickjacking attack	7	●
0	0	0	n/a	CWE-90: Avoid LDAP injection vulnerabilities	9	●
0	0	0	n/a	CWE-91: Avoid XPath injection vulnerabilities	9	●
0	0	0	n/a	CWE-134: Avoid uncontrolled format string	9	●
0	0	0	n/a	CWE-798: Use of hard-coded credentials	9	●
0	0	0	n/a	CWE-501: Trust boundary violation	9	●
0	0	0	n/a	CWE-330: Use of insufficiently random values	9	●
0	0	0	n/a	Avoid using submitted markup containing "form" and "formaction" attributes	9	●