Flawnter SAST Rules Overview

Static Application Security Testing (SAST) is a crucial process in modern software development, designed to identify vulnerabilities and security flaws within the source code of applications before they are deployed. Flawnter SAST rule sets are predefined sets of rules that Flawnter use to analyze and evaluate source code and even some binaries for potential security issues. These rules play an important role in the SAST process, as they define the criteria for identifying security vulnerabilities, logical errors, and other weaknesses that may be exploited or cause the application to crash or behave abnormally.

These rule sets are crafted to address specific types of security threats, coding standards, and industry best practices from OWASP, SANS/CWE, NIST, SEI CERT, PCI and others. Flawnter uses different type of rule sets to achieve the best results. There are direct rules, generic rules and predictive rules that use AI driven analysis. Additionally Flawnter has deep scan capabilities that help go beyond and potentially find more bugs. Please visit our documentation page https://www.flawnter.com/documentation that will show how organizations can use Flawnter configuration file to tailor these rule sets to their specific needs and achieve a more robust and secure software development lifecycle.

The table provided below does not include quality rules. Instead, it focuses on the most commonly used security test rules. It's important to note that this list is not exhaustive, and there are other security rules that are not included in this table. Furthermore, each primary testing rule may encompass hundreds of sub-test rules. When aggregated, the total number of rules surpasses several thousand.

#	Rule	Description	CWE ID
1	Hard Coded Credentials/Secrets	These rules are to scan source and configuration files for hard coded passwords/tokens/keys/etc.	259, 798
2	Cryptography Issues	Weak algorithms, weak keys, weak hashes, weak randomness, etc.	327, 1346
3	Code Execution/Injection	Any code execution or injection.	78, 94
4	SQL/XML/LDAP/DOM/LOG Injection	Injecting untrusted input into SQL, XML, XPATH, LDAP, DOM, LOG.	89, 90, 91
5	Path Traversal & File Manipulation Attacks	Looks for path/directory traversal attacks that come from untrusted input.	22
6	Cross-site Scripting	Check for any cross-site scripting attacks including DOM based.	79
7	Broken/Improper Authentication	Check for possible authentication bypass and weaknesses.	287
8	Information / Sensitive Data Exposure	Information Disclosure, Sensitive Data Exposure.	200
9	Denial of Service	Check for possible Denial of Service attacks.	400, 770

10	Insecure/Improper Authorization	Insecure/Improper access controls.	284, 285
11	Clickjacking	Check for clickjacking attacks.	1021
12	Response Splitting / Header Injections	Check for untrusted input being sent to functions that may cause response splitting and header injections.	113
13	Buffer Over-read/Over-run/overflow		120, 121, 122
14	Memory Leaks	Check that may lead to memory leaks like forgetting to close objects or freeing memory.	401
15	Security Misconfiguration	Check for incorrect configurations.	16
16	Insecure Communication	Check for cleartext transmission of data. Missing encryption.	319
17	Trust Boundary Violation	Check for trusting unvalidated data.	501
18	Deadlocks	Check for possible deadlocks.	833
19	Insecure Deserialization	Check for possible Insecure Deserialization.	502
20	Xml External Entity	Check for Xml External Entity attacks.	611
21	Incorrect Function Usage	Check for some functions incorrect usage.	684
22	Race Condition	Check for some race conditions.	366
23	Missing Http Only Cookie Attribute	Check for missing HttpOnly flag for sensitive cookies.	1004
24	Missing Secure Cookie Attribute	Check for missing Secure flag for sensitive cookies.	614
25	Parameter Pollution	Improper handling of parameters.	235
26	Debug Enabled	Check to see if debug is turned on.	489
27	Null Pointer Dereference	Check for null pointer dereferences.	476
28	Cross-site Request Forgery	Check for Cross-Site Request Forgery (CSRF) attacks.	352
29	Server Side Request Forgery	Checks for Server Side Request Forgery (SSRF) attacks.	918
30	Resource Injection	Check for resource injections.	99