Security Audit Executive Summary

CAST | Software Intelligence for Digital Leaders

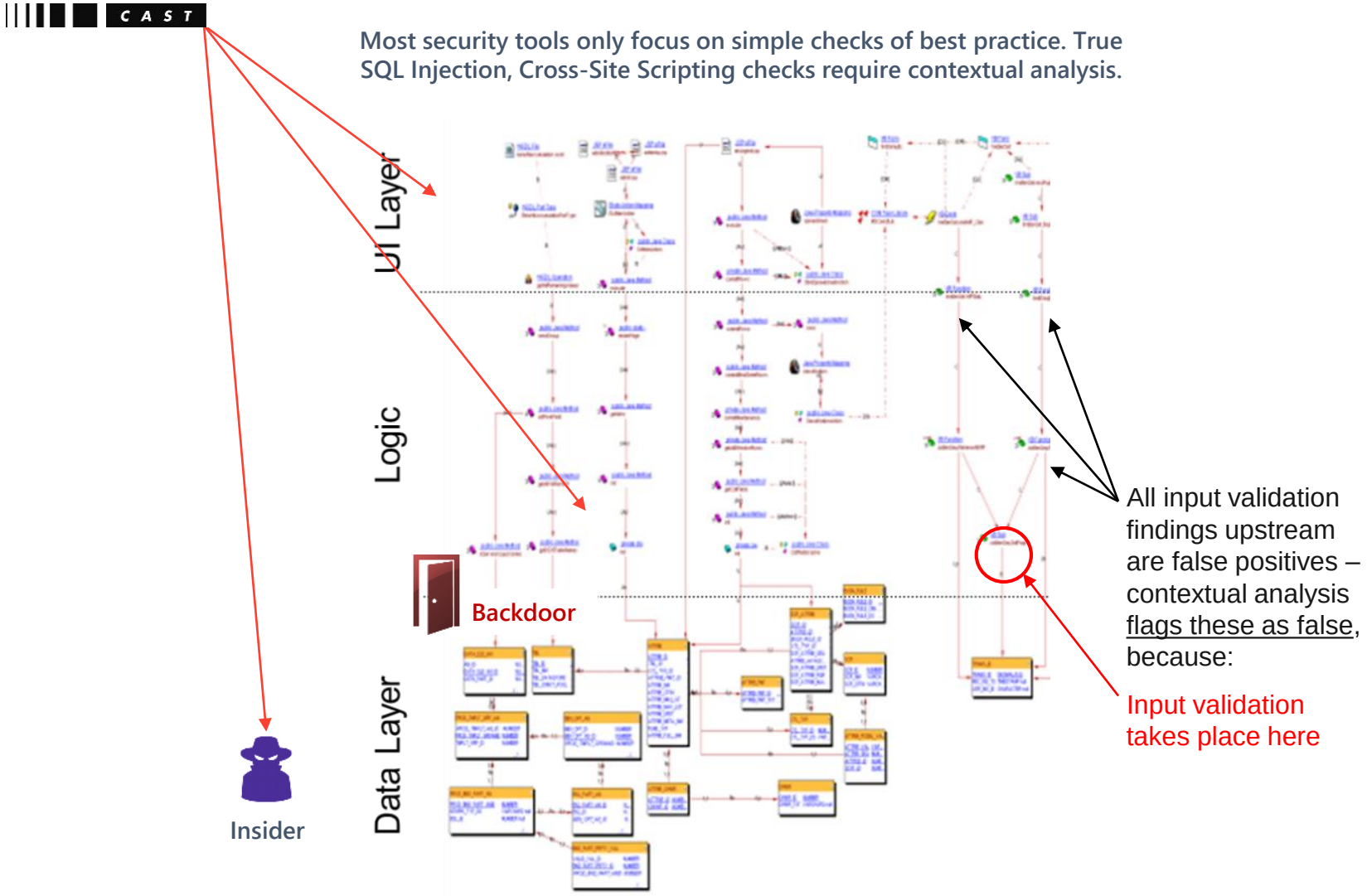
www.castsoftware.com

Context and Objective

Application	Technologies	Size (LOC)
U[redacted]	JEE HTML5 SQL	44,156 225,446 5,210
E[redacted]pt	JEE HTML5 SQL	64,159 227,197 2,050
C[redacted]	JEE PHP HTML5 Python SQL	50,242 1,223 22,545 7,734 3,417

[redacted]T Inc. is in need of make invest decisions for core applications. To help gain insight into application health and safety, [redacted] Inc. engaged CAST to conduct a security assessment of X applications. This assessment is based on the automated application analysis provided by CAST Application Intelligence Platform (AIP)

Approach: Software Intelligence



Contextual Software Analysis finds flaws that traditional application security tools can't catch:

Forbidden access to data, lack of input validation, backdoors and insider threats.

Current security analysis tools review code at the unit level to ensure programming best practices are followed. Without contextual analysis current tools:

- Miss important problems
- Provide way too many findings that are irrelevant

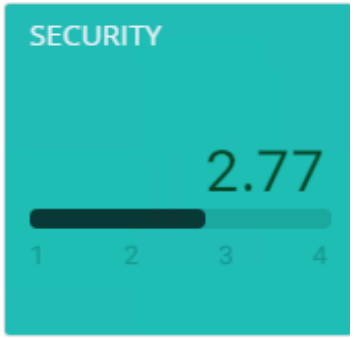
Summary of C[redacted]E results

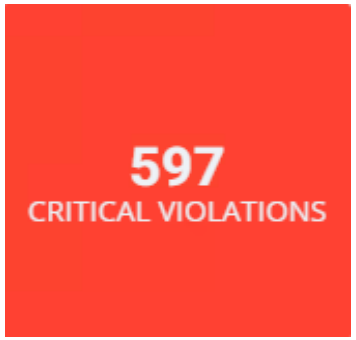
C[redacted]E has 80,000 lines of code with 13,232 possible test cases to cover the whole application.

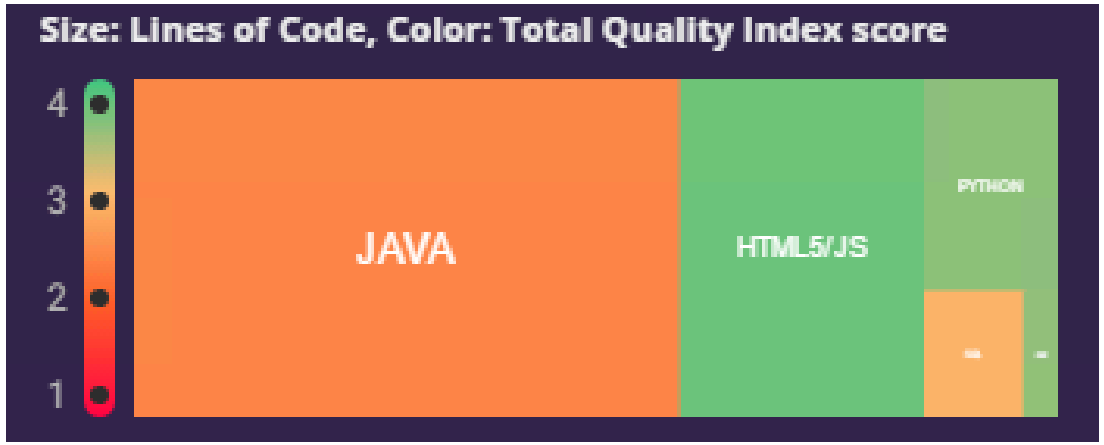
TQI (Total Quality Index) is an aggregate of all 5 measures calculated by CAST AIP.

The **application shows a high risk in Robustness and Security.**

Transferability is at medium risk but with some room for improvement.







C[redacted]E Critical Violations

CAST

SECURITY

2.77

RULES	WEIGHT	% COMPLIANCE	# VIOLATIONS
Avoid testing floating point numbers for equality		96%	2
Check usage of '==' and '!=' on objects		99%	18
The exception Exception should never been thrown. Always Subclass Exception and throw the subclassed Classes.		99%	9
Avoid empty catch blocks		99%	7
Avoid cyclical calls and inheritances between packages		78%	16
Close the outermost stream ASAP		28%	20
Avoid to use this within Constructor in multi-thread environment		99%	3
Avoid copying needless the variables (PHP)		97%	2

ROBUSTNESS

2.78

RULES	WEIGHT	% COMPLIANCE	# VIOLATIONS
Avoid testing floating point numbers for equality		96%	2
Check usage of '==' and '!=' on objects		99%	18
The exception Exception should never been thrown. Always Subclass Exception and throw the subclassed Classes.		99%	9
Avoid to use this within Constructor in multi-thread environment		99%	3
Avoid empty catch blocks		99%	7
Avoid cyclical calls and inheritances between packages		78%	16
Suspicious similar method names or signatures in an inheritance tree		95%	13
Avoid classes overriding only equals() or only hashCode()		83%	2
Proper overriding of 'clone()'		99%	2

TRANSFERABILITY

3.01

RULES	WEIGHT	% COMPLIANCE	# VIOLATIONS
Avoid classes overriding only equals() or only hashCode()		83%	2
Suspicious similar method names or signatures in an inheritance tree		95%	13
Proper overriding of 'clone()'		99%	2

- The majority of critical violations were found in the Java code.
- Overall compliance level is high with relatively few violations to investigate and fix.
- 7 critical violations per 1k LOC reported (including Changeability and Efficiency). Average found during CAST assessments are 4 critical violations per 1k LOC.

CR[redacted]E Complexity

- List of highly complex objects that represent potential opportunities for engineering mistakes.

Top Cyclomatic Complexity x High Fan-Out

Object Name	Cyclomatic Complexity	Fan-Out
net.sf.jlinkgrammar.Linkage.merge_constituents	93	6
net.sf.jlinkgrammar.Linkage.linkage_print_diagram	83	12
net.sf.jlinkgrammar.Linkage.gen_comp	68	8
net.sf.jlinkgrammar.Linkage.read_constituents_from_domains	68	14
net.sf.jlinkgrammar.Sentence.separate_word	60	24
net.sf.jlinkgrammar.ParseInfo.parse_set	59	11
net.sf.jlinkgrammar.Sentence.mark_region	56	6
net.sf.jlinkgrammar.Sentence.count	51	7
net.sf.jlinkgrammar.Parser.InitializeVars	47	14
net.sf.jlinkgrammar.Linkage.last_minute_fixes	46	6
net.sf.jlinkgrammar.Sentence.build_AND_disjunct_list	41	7
net.sf.jlinkgrammar.Linkage.build_linkage_postscript_string	37	6
com.pacificmetrics.automatedscoring.modules.ps.ParserScorer.process	37	20
org.json.XML.parse	36	16
org.json.XML.parse	36	16
com.pacificmetrics.automatedscoring.service.scoring.ScorerImpl.executeModules	30	46
net.sf.jlinkgrammar.Dictionary.restricted_expression	30	12
net.sf.jlinkgrammar.Sentence.power_prune	30	9
net.sf.jlinkgrammar.Sentence.analyze_fat_linkage	29	16
com.pacificmetrics.automatedscoring.modules.ps.CopyProportionCalculator.compute	29	9
com.pacificmetrics.automatedscoring.webservice.ScorerBean_ScorerBeanPort_Client.main	28	30
com.pacificmetrics.automatedscoring.webservice.ScorerBean_ScorerBeanPort_Client.main	28	30
net.sf.jlinkgrammar.Sentence.pp_prune	27	13
com.pacificmetrics.automatedscoring.common.ParseResponse.parseXmlSingleTable	27	49
net.sf.jlinkgrammar.Sentence.is_canonical_linkage	26	7
test.com.pacificmetrics.automatedscoring.service.TestUtil.compareScoredOutputWithExer	26	13
com.pacificmetrics.automatedscoring.modules.ps.NonEnglishCheck.getNonEnglishProportic	25	6
com.pacificmetrics.automatedscoring.modules.util.TextPreprocessingUtil.processFractions	25	20

- Highly complex artefacts that call many other artefacts are difficult to understand and test.

Top Cyclomatic Complexity x Low Documentation

Object Name	Cyclomatic Complexity	Documentation Ratio
net.sf.jlinkgrammar.Linkage.linkage_print_diagram	83	4
net.sf.jlinkgrammar.Linkage.build_linkage_postscript_string	37	2
com.pacificmetrics.automatedscoring.modules.ps.ParserScorer.process	37	4
com.pacificmetrics.automatedscoring.service.scoring.ScorerImpl.executeModules	30	2
net.sf.jlinkgrammar.Dictionary.restricted_expression	30	0
com.pacificmetrics.automatedscoring.webservice.ScorerBean_ScorerBeanPort_Client.mair	28	0
com.pacificmetrics.automatedscoring.webservice.ScorerBean_ScorerBeanPort_Client.mair	28	0
test.com.pacificmetrics.automatedscoring.service.TestUtil.compareScoredOutputWithExer	26	3
com.pacificmetrics.automatedscoring.modules.util.TextPreprocessingUtil.processFractions	25	2
net.sf.jlinkgrammar.ParseOptions.issue_special_command	23	3
net.sf.jlinkgrammar.Linkage.cons_of_domain	21	0
com.pacificmetrics.automatedscoring.modules.ps.ImproperFormattingCheck.process	20	2
net.sf.jlinkgrammar.GlobalBean.process_some_linkages	19	0
[S:\Source\ACT\CRASE\Source\src\crase-java\application\RELEASES\release-2.4\server	18	4
[S:\Source\ACT\CRASE\Source\src\crase-java\application\RELEASES\release-2.3\server	18	4
net.sf.jlinkgrammar.Dictionary.advance	15	1
net.sf.jlinkgrammar.Linkage.print_tree	14	0
net.sf.jlinkgrammar.ParseOptions.print_expression	13	0
net.sf.jlinkgrammar.Dictionary.check_connector	13	3
com.pacificmetrics.pathos.bean.PathosScoringResponse.equals	13	0
com.pacificmetrics.automatedscoring.service.configuration.load.modules.EssayScorerLoad	11	0
net.sf.jlinkgrammar.Disjunct.disjuncts_equal	11	4
com.pacificmetrics.pathos.bean.ScoringStatus.equals	10	0
com.pacificmetrics.pathos.bean.Essay.equals	10	0
com.pacificmetrics.pathos.bean.ConstructedResponse.equals	10	0
S:\Source\ACT\CRASE\Source\src\crase-java\webservice\src\test\php-json-scenarios\(	10	0
com.pacificmetrics.automatedscoring.modules.comps.RulesUtil.rounded	10	0
com.pacificmetrics.automatedscoring.modules.es.EssayScorer.logReqScore	10	0

- Highly complex artefacts that have few comments are difficult to understand and are more likely to be copy/pasted to avoid mistakes.

Finding 1: Multiple artifacts inserting, updating, deleting

- Too many artefacts access to SQL tables :
  - Stored procedures
  - Stored functions
  - Triggers
  - Java Fields
  - Java Methods
- Not secure for data
- **RECOMMENDATION:** Redesign the Table access – Make the access unified

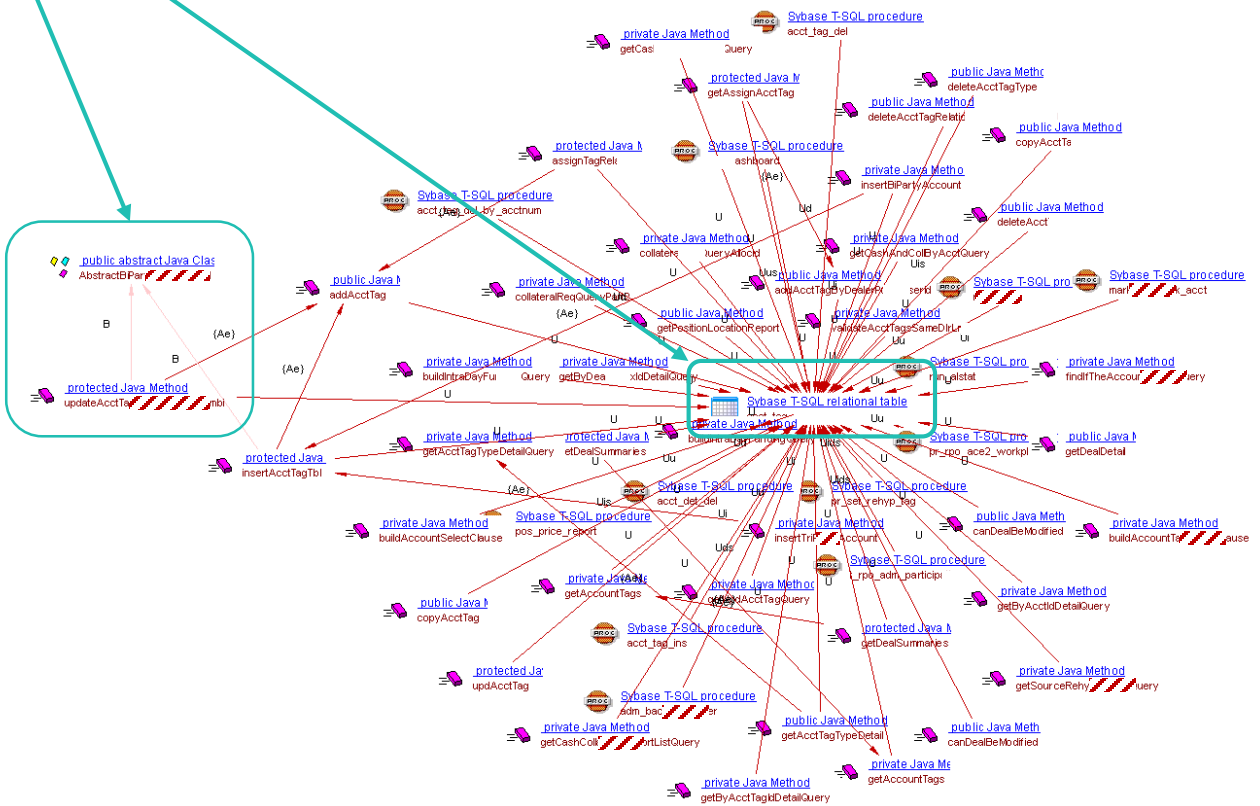
System blueprint generated by CAST

Finding 2: Data inconsistency risk linked to non unified accesses

Table Access Violations found

Quality Rules, Distributions and Measures

Grade	Var.	Rule Name	Contr.	Critical
1.01	-0.00%	Avoid cyclical calls and inheritances between packages	7x104%	Yes
1.63	0.00%	Avoid Functions and Procedures doing an Insert, Update or Delete without managing	1x0%	No
1.84	0.00%	Avoid having multiple Artifacts inserting data on the same SQL Table	7x0%	No
2.37	0.00%	Avoid having multiple artifacts deleting data on the same SQL table	7x0%	No
2.78	0.00%	Avoid having multiple Artifacts updating data on the same SQL Table	7x0%	No
3.28	-0.00%	Avoid calls between JSP Pages	3x99%	No
3.40	0.00%	Avoid direct access to Database Tables	6x99%	No
4.00	0.00%	Avoid use of standard SQL API	8x99%	No
4.00	0.00%	Avoid direct use of Database objects (JSP/ASP)	6x99%	No
4.00	0.00%	Avoid too many packages referencing Mainframe	5x104%	No

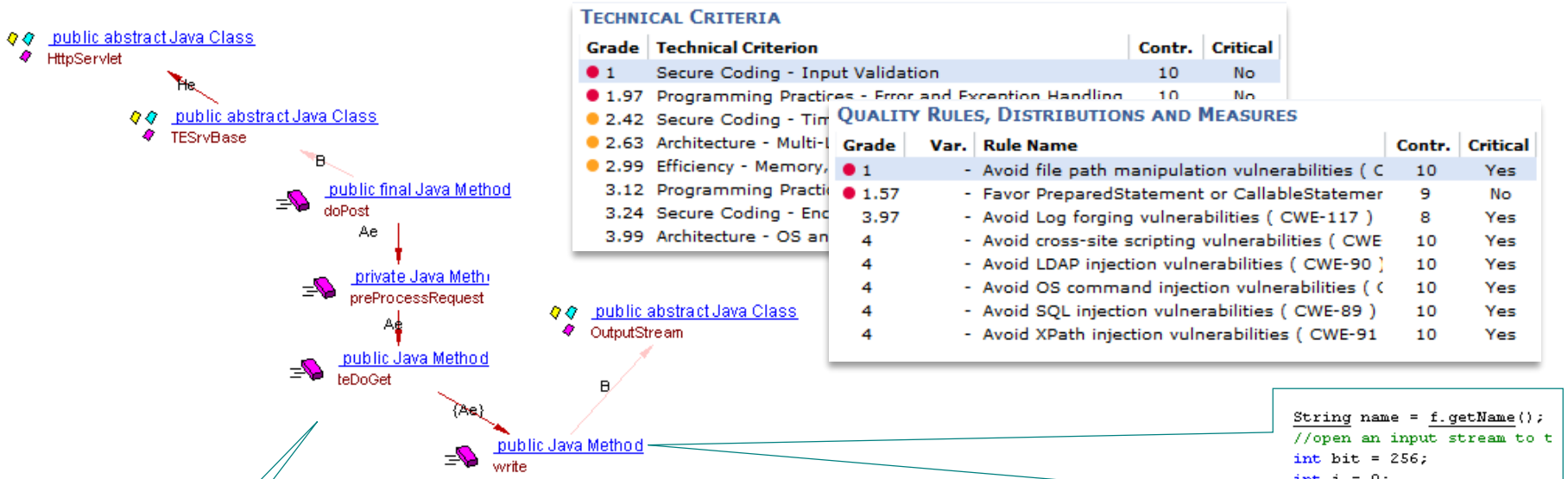


RECOMMENDATION:

Redesign the data access flow.

Finding 3: Input Validation Vulnerability

- In the teDoGet method below, the input parameter is not validated before the InputStream is created and the file is processed with output written. Input validation vulnerability may result into SQL Injection.
- RECOMMENDATION: To avoid the creation of Injection flaws, follow (OWASP) recommendation to validate all user input.



TECHNICAL CRITERIA

Grade	Technical Criterion	Contr.	Critical
1	Secure Coding - Input Validation	10	No
1.97	Programming Practices - Error and Exception Handling	10	No
2.42	Secure Coding - Tim		
2.63	Architecture - Multi-L		
2.99	Efficiency - Memory,		
3.12	Programming Practi		
3.24	Secure Coding - Enc		
3.99	Architecture - OS an		

QUALITY RULES, DISTRIBUTIONS AND MEASURES

Grade	Var.	Rule Name	Contr.	Critical
1	-	Avoid file path manipulation vulnerabilities (C	10	Yes
1.57	-	Favor PreparedStatement or CallableStatemer	9	No
3.97	-	Avoid Log forging vulnerabilities (CWE-117)	8	Yes
4	-	Avoid cross-site scripting vulnerabilities (CWE	10	Yes
4	-	Avoid LDAP injection vulnerabilities (CWE-90)	10	Yes
4	-	Avoid OS command injection vulnerabilities ((	10	Yes
4	-	Avoid SQL injection vulnerabilities (CWE-89)	10	Yes
4	-	Avoid XPath injection vulnerabilities (CWE-91	10	Yes

```
String sFileDoc = req.getParameter("file");
String msg = "";
ServletOutputStream outs = res.getOutputStream();

if (sFileDoc != null) {
    try {
        File f = new File(sPDFpath +"/web/" +sessUser.getLogonID().getKNum().toStringWithK().toLowerCase()+ "/"+sFileDoc);
        InputStream in = new FileInputStream(f);
```

```
String name = f.getName();
//open an input stream to t
int bit = 256;
int i = 0;
while ((bit >= 0)) {
    bit = in.read();
    outs.write(bit);
}
outs.flush();
outs.close();
in.close();
```

Benchmark against JEE applications

Health Factors Benchmark Results

TQI

90.42%

Rank: 60/939

Security

95.31%

Rank: 35/939

Transferability

86.82%

Rank: 186/939

Robustness

92.44%

Rank: 162/939

938 applications

268 organizations

360.18M lines of code

Appmarq

CAST

Software Intelligence for Digital Leaders

About CAST

Industry's Choice for Software Intelligence

Hundreds of Enterprise Customers

at&t · BMW · BNY MELLON · Allianz · BOEING · KAISER PERMANENTE

ISVs and Global SI's Customers

Infosys · IBM · Capgemini · First Data · Atos · CGI · Broadridge · accenture · sopra steria · HCL · DXC.technology · Cognizant · Tech Mahindra · wipro

CISQ Consortium for IT Software Quality

Software Engineering Institute

Go to Market Partners

BCG The Boston Consulting Group · Microsoft · McKinsey&Company · ISG · IBM · Software Engineering Institute · Cognizant · ca technologies

Software Intelligence Pioneer & Leader

- $155 million R&D investments
- 25 years and counting
- Global presence
  US–EU–INDIA–CHINA

Business Relevant, Accurate and Actionable

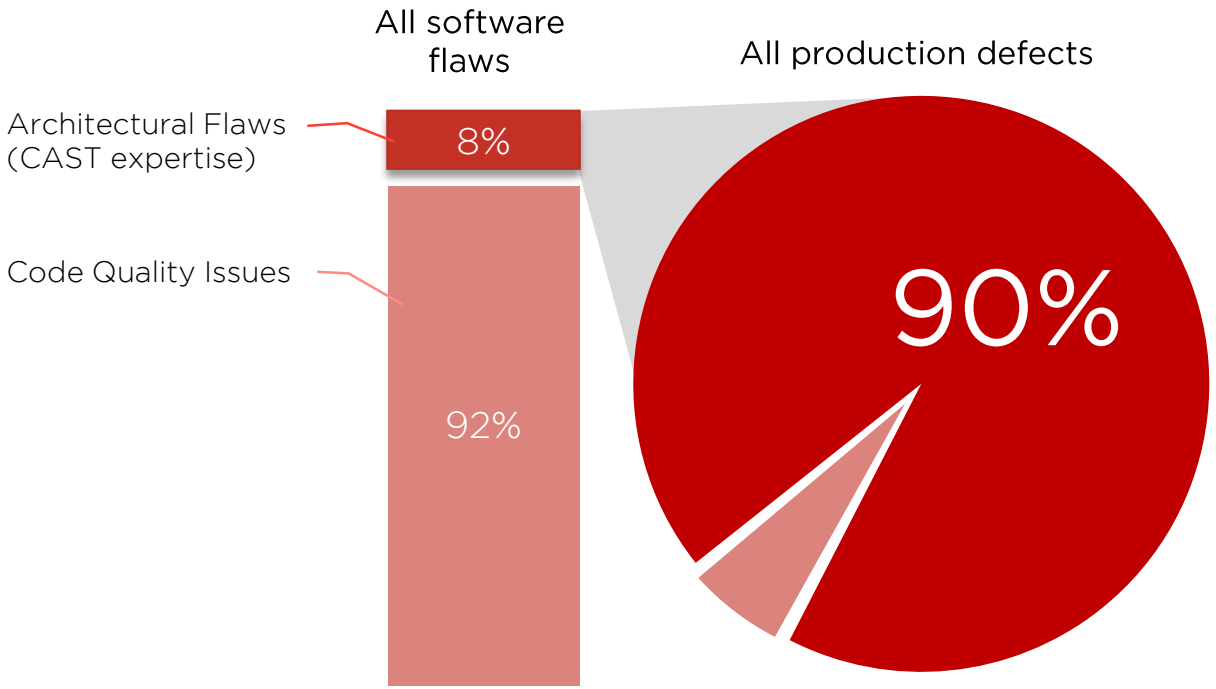
"CAST excels at architectural assessment."
Melinda Ballou, Research Director

"Most accurate for application security."
Amy Demartine, Principal Analyst

"The leader in its space."
Chandranshu Singh, Research Director

"The industry reference for software risk and size."
Dr. B. Curtis, OMG/SEI/CISQ

"Sound, thoroughly vetted technology."
Jim Duggan, VP Research



"8% of programming mistakes lead to 90% of production issues"
Dr. Richard Soley, PhD MIT