

C O N T E N T S

1	About Author	11
2	Introduction	12
3	Devsecops- Pipelines	14
3.1	DevSecOps	14
3.1.1	Phase of DevSecOps	15
3.2	Consistent cybersecurity	17
3.3	CI/CD process safety	18
3.4	Testing, security and metrics:	19
3.4.1	SAST	19
3.4.2	DAST	19
3.4.3	IAST	19
3.4.4	MAST	20
3.5	DevSecOps Team	20
3.6	DevSecOps' fundamental components	20
3.7	Summary	22
4	Static Application Security Testing-Sast	23
4.1	SAST	23
4.2	SAST vs DAST	24
4.2.1	Top SAST Tools	24
4.3	Continuous Integration Tools	25
4.3.1	Enkins	25
4.3.2	Travis CI	25
4.3.3	GitLab CI	26
4.4	Integrate SonarQube with Jenkins	26
4.4.1	Pre-requisites	27
4.4.2	Jenkins Installation	27
4.4.3	SonarQube Installation	27
4.5	Integration of SonarQube in Jenkins	28
4.6	GitLab integration with Jenkins	31
4.7	Example Jenkins Pipeline script:	34
4.8	GitLab and SonarScanner	35

4.9	Travis CI and SonarCloud -----	35
4.10	Code Quality-SonarQube -:-----	37
4.10.1	Key Aspects to Measure -----	37
4.10.2	How to Improve Code Quality -----	38
4.11	SonarQube -----	40
4.12	Key concepts in Static Code Analysis -----	41
4.12.1	Write Clean Code -----	42
4.13	Summary -----	44
5	Java-springboot project- with travis ci-docker- jenk ins-sonarqube code scan -----	46
5.1	System description -----	46
5.2	Applied Techniques - Technology and Framework -----	47
5.3	System Functionality Description -----	49
5.3.1	SignUp-----	49
5.3.2	Save User Profile-----	50
5.3.3	SignIn -----	50
5.3.4	Access Denied -----	50
5.4	System Architecture Diagram -----	52
5.4.1	VIEW-----	52
5.4.2	Controller -----	52
5.4.3	Service -----	52
5.4.4	Repository -----	52
5.4.5	Model -----	53
5.5	Testing and Development-----	53
5.5.1	Unit Testing -----	53
5.5.2	Integration Testing -----	54
5.5.3	E2E Testing -----	54
5.5.4	E2E test of the GUI -----	55
5.6	Tools for Test Driven Development -----	55
5.6.1	Docker -----	55
5.6.2	Local Build and on Server CI (Maven and Travis CI) -----	56
5.6.3	GitHub -----	56
5.6.4	MongoDB-----	57
5.6.5	Code Quality with Sonarcloud -----	57

5.6.6	Spring Boot	57
5.6.7	Coveralls	58
5.7	Summary	58
6	Security in cloud platform	59
6.1	Amazon Web Services (AWS)	59
6.2	Microsoft Azure	60
6.3	Google Cloud	61
6.4	Cloud Computing Security Risks	62
6.5	Cloud Vulnerability and Penetration Test	62
6.6	Firewall	63
6.7	Cloud Security Best Practices	63
6.8	Implement a Strong Password Security Policy	67
6.9	Security Checklist for Cloud	67
6.10	Azure security technical features	70
6.10.1	Azure Active Directory	71
6.10.2	Single Sign-On	72
6.10.3	Multi-factor authentication	72
6.10.4	Security monitoring, alarms, and reporting based On machine learning	73
6.10.5	Device registration	74
6.10.6	Privileged Identity Management	74
6.10.7	Identity Protection	75
6.10.8	Web application firewall	75
6.10.9	Network Watcher	76
6.11	AWS- Security, Identity and Compliance	76
6.11.1	Amazon Cloud Directory	76
6.11.2	AWS Identity and Access Management	77
6.11.3	Amazon Inspector	77
6.11.4	AWS Certificate Manager	78
6.11.5	AWS CloudHSM	78
6.11.6	AWS Directory Service	79
6.11.7	AWS Key Management Service	79
6.11.8	AWS Organizations	79
6.11.9	AWS Shield	80

6.11.10	AWS WAF -----	80
6.11.11	Amazon Athena -----	81
6.11.12	Amazon EMR -----	81
6.11.13	Amazon CloudSearch -----	81
6.11.14	Amazon Elasticsearch Service -----	82
6.11.15	Amazon QuickSight -----	82
6.11.16	AWS Data Pipeline -----	82
6.11.17	AWS Glue -----	83
6.12	Google Cloud security -----	83
6.12.1	Transparency -----	84
6.12.2	Hierarchy of Resources -----	84
6.12.3	Privilege -----	85
6.12.4	Credential management -----	85
6.12.5	Access and Firewall -----	85
6.12.6	Generating and analyzing activity logs -----	86
6.12.7	Life Cycles of VM Images -----	87
6.13	Summary -----	87
7	Owasp top 10 web app vulnerabilities -----	88
7.1	Injection: -----	88
7.2	Broken Authentication: -----	88
7.3	Sensitive Sensitive Data: -----	89
7.4	External Entities in XML: -----	89
7.5	Broken Access Control: -----	89
7.6	Misconfiguration of security: -----	89
7.7	XSS (Cross-Site Scripting): -----	89
7.8	Using Components with Vulnerabilities that are well known -----	90
7.9	Insecure Deserialization: -----	90
7.10	Inadequate logging and monitoring: -----	90
7.11	Server-Side Request Forgery -----	91
7.12	Attacks on low levels of the ISO / OSI stack -----	92
7.12.1	Physical attacks -----	92
7.12.2	Connection level connections -----	93
7.12.3	Network-level attacks -----	95
7.13	Attacks on high levels of the ISO / OSI stack -----	100

7.13.1	Transport level attacks -----	100
7.13.2	Attacks on middleware -----	102
7.13.3	SQL Injection -----	103
7.13.4	Cross-Site-Scripting (XSS) -----	105
7.13.5	Attacks on higher protocols -----	107
7.14	Buffer overflow -----	109
7.15	Summary -----	111
8	Deep analysis of client-side cross-site request forgery attack -----	112
8.1	Attack -----	112
8.2	Practical example of Attack -----	113
8.3	Code representation -----	116
8.4	Detection of Client-side CSRF -----	117
8.5	How to Protect Against Cross-Site Request Forgery -----	118
8.5.1	Implement an Anti-CSRF Token -----	118
8.5.2	In cookies, set the SameSite flag -----	119
8.6	IT Security Standards -----	119
8.6.1	Popular cybersecurity frameworks -----	120
8.7	Cybersecurity Regulations -----	121
8.8	Summary -----	121
9	Kubernetes architecture-security -----	123
9.1	Kubernetes Architecture -----	124
9.1.1	ETCD -----	125
9.1.2	Kube-API Server -----	126
9.1.3	Kube Controller Manager -----	126
9.1.4	Kube- Scheduler -----	128
9.1.5	Kubelet -----	129
9.1.6	Kube Proxy -----	130
9.1.7	POD -----	131
9.1.8	Pod with YAML -----	132
9.1.9	Pod templates -----	135
9.1.10	ReplicationController -----	137
9.1.11	Replica set -----	137
9.1.12	How to create Replication Controller -----	137

9.1.13	How to create replicset ? -----	138
9.1.14	Why we need to Label in Pods and Controller? -----	140
9.1.15	How to update our replica set ? -----	140
9.1.16	Deployments -----	141
9.1.17	Namespaces -----	143
9.1.18	Kube system, kube public -----	143
9.1.19	NameNamespace- default -----	144
9.1.20	Daemon Sets -----	145
9.2	Kubernetes Security -----	146
9.2.1	Securing a Cluster -----	146
9.2.2	Access to the Kubernetes API can be restricted. -----	147
9.2.3	Authentication for APIs -----	147
9.2.4	Authorization of APIs -----	147
9.2.5	Keeping track of who has access to the Kubelet -----	148
9.2.6	Managing which nodes pods have access -----	148
9.2.7	How to monitor Kubernetes cluster -----	149
9.3	Best Practices for Kubernetes Architecture -----	150
9.3.1	High level of availability -----	150
9.3.2	Durability -----	151
9.3.3	Flexibility -----	151
9.3.4	Protection -----	152
9.4	Kubernetes Architecture Security Configuration -----	152
9.5	Summary -----	153
10	Infrastructure as code -terraform - security -----	154
10.1	Ways to make Infrastructure as Code more secure -----	154
10.2	Security Policies and Configuration -----	155
10.3	Security Enablers in Infrastructure as Code -----	156
10.3.1	IaC Governance that is Automated -----	156
10.3.2	Code-governed and code-secured -----	156
10.3.3	Workflow that never stops -----	156
10.4	The advantages of establishing Infrastructure-as-Code (IaC) security ----	156
10.4.1	Consistent compliance -----	156
10.4.2	Continuous Threat Assessment and Risk Evaluation -----	157
10.4.3	Encryption of Data as a Prerequisite -----	157

<u>10.4.4</u>	Tracking and Alerts that are Automatically -----	157
10.5	Example - IaC -----	158
10.6	The CDLC includes IaC -----	158
10.7	The importance of IaC security -----	159
10.8	Vulnerability Scanning Tools for Infrastructure as Code -----	160
<u>10.8.1</u>	Checkov -----	160
<u>10.8.2</u>	TFLint -----	160
<u>10.8.3</u>	Terrafirma -----	160
<u>10.8.4</u>	Accurics -----	161
<u>10.8.5</u>	CloudSploit -----	161
10.9	Tools available to Secure Cloud Platform -----	161
10.10	Penetration Testing in Cloud Computing -----	162
10.11	AWS and Azure Cloud Penetration Testing -----	163
<u>10.11.1</u>	Best Practices for Penetration Testing in the Cloud -----	164
10.12	Example- Penetration Testing -----	165
10.13	Summary -----	166
11	Project - infrastructure as code (terraform) - security -----	167
11.1	Project Description -----	167
11.2	Code Description -----	167
11.3	Finding Problem in the Code -----	169
11.4	Tools Helps to Resolve Bugs in the Terraform -----	172
11.5	Terrascan vs Checkov -----	178
11.6	Vulnerability Fixing in Terraform using Terrascan -----	178
11.7	Vulnerability Fixing in Terraform using Checkov -----	181
11.8	Integration of Tools in CI/CD Pipeline -----	181
<u>11.8.1</u>	Integration of Jenkins or GitLab -----	181
<u>11.8.2</u>	Integration of Jenkins-SonarQube -----	182
<u>11.8.3</u>	The Importance of Infrastructure as Code -----	183
11.9	Summary -----	183
12	Security standards and tools in it industry-conclusion -----	184
12.1	Implementation Tools for Infrastructure as Code -----	184
<u>12.1.1</u>	Ansible -----	185
<u>12.1.2</u>	AWS CloudFormation -----	186

<u>12.1.3</u>	Chef -----	188
<u>12.1.4</u>	Puppet -----	188
12.2	How do you pick the finest IaC tools? -----	190
12.3	AWS's Security Features in General -----	190
12.4	AWS-Security-Features -----	197
12.5	Vulnerability Management -----	199
<u>12.5.1</u>	Vulnerability Management -Phase -----	199
12.6	Security Content Automation protocol (SCAP) -----	200
<u>12.6.1</u>	Component of SCAP -----	201
<u>12.6.2</u>	Common Vulnerability Scoring System (CVSS) -----	201
12.7	Incident Handling Process- NIST -----	202
12.8	Security Standards -----	203
12.9	Conclusion -----	204