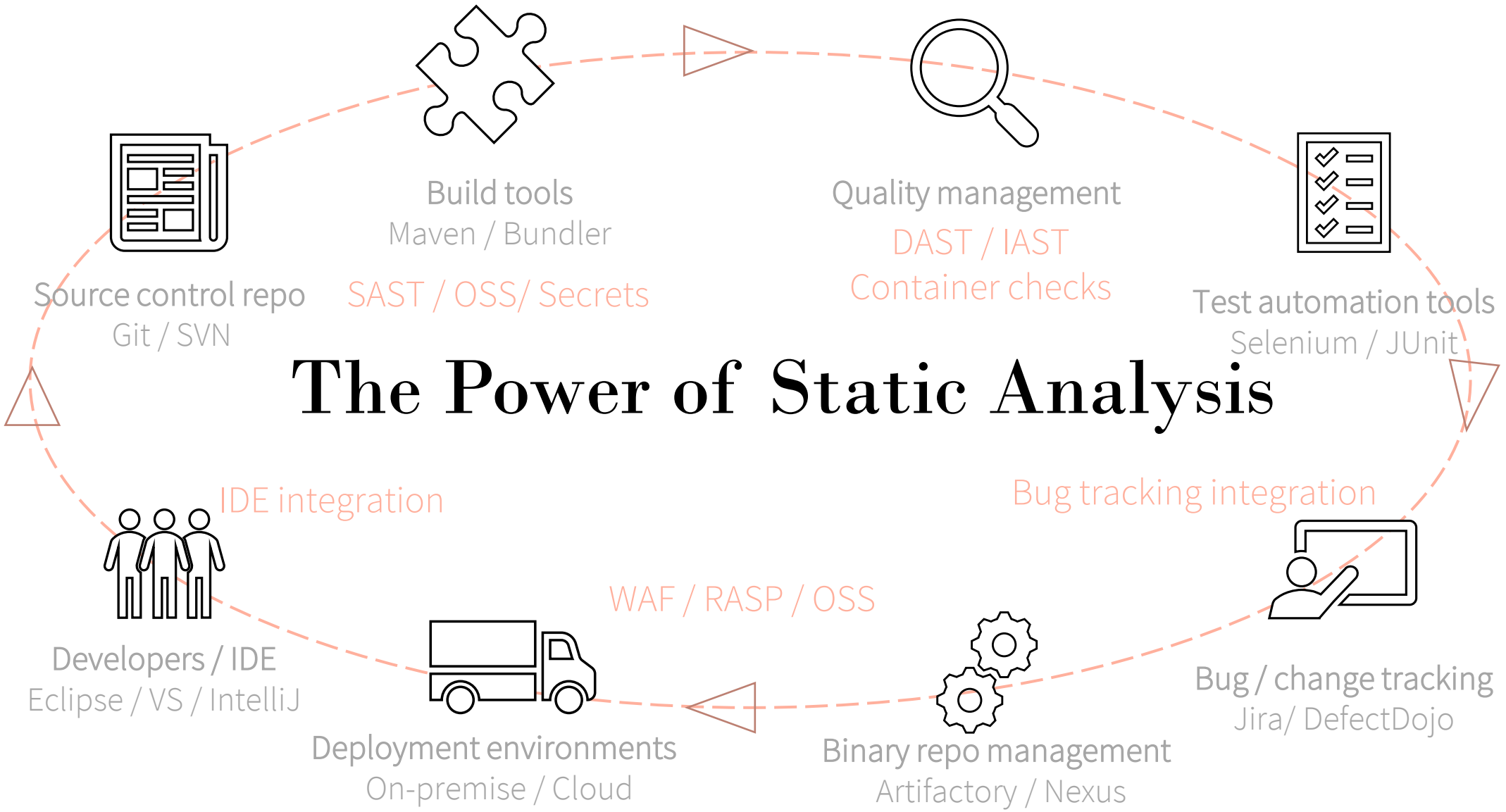

Sergey Egorov

Static analysis types

- SAST/IAST/MAST
- SCA/OSS
- Secrets Scans
- IACS

- Check the code for insecure patterns, build code flow graphs
- Dependencies/ libraries known vulnerabilities
- Regexing for known secret patterns
- Insecure configurations (IAC, API, Cloud, server configs)

Language specific SAST

- python - Bandit
- java - FindSecBugs
- csharp - SecurityCodeScan
- cpp - FlawFinder
- php - PHP_CodeSniffer
- javascript - NodeJS Scan
- ruby - Brakeman
- go - GoSec
- kotlin - MobSF
- swift - MobSF

- docker run --rm -v "${PWD}:/src" ghcr.io/pycqa/bandit/**bandit** -r /src -f json -o /src/bandit.json
- docker run --rm -v "${PWD}:/src" -v "${PWD}:/report" registry.gitlab.com/gitlab-org/security-products/analyzers/**brakeman** /analyzer r --target-dir /src --artifact-dir /report
- docker run --rm -v $(pwd):/src -w /src securego/**gosec** -fmt json -out gosec-report.json ./...
- **mobsfscan** --sarif . -o mobsf_sast-report.sarif

https://gitlab.com/gitlab-org/security-products/analyzers

Multilanguage SAST

- Semgrep / Semgrep Pro
- GitLab SAST (Semgrep)
- GitHub SAST (CodeQL)
- Snyk

- docker run --rm -v "${PWD}:/src" **returntocorp/semgrep** semgrep --json -o semgrep.json
- docker run --rm -v "${PWD}:/src" -v "${PWD}:/report" **registry.gitlab.com/gitlab-org/security-products/analyzers/semgrep** /analyzer r --target-dir /src --artifact-dir /report
- https://docs.github.com/en/code-security/codeql-cli/getting-started-with-the-codeql-cli/setting-up-the-codeql-cli
- **https://app.snyk.io/login**

LLM SAST

```
model="gpt-4o",
messages=[
{"role": "system", "content": f"Find security vulnerabilities is the provided code. \
    The file name is {file} \
    Output found vulnerabilities strictly in html table format with the next fields: \
    vulnerability name, description, file name, vulnerable parameter, code snippet and remediation advice. \
    If you can not find any vulnerabilities, output only the next phrase 'No vulnerabilities found'. \
    "},
{"role": "user", "content": code}
],
stop=["\nstop", "Privacy: no-training"]
)
```

Secrets Scans

- docker run --rm -v $(pwd):/src trufflesecurity/**trufflehog** git file://../../src -j
- docker run --rm -v $(pwd):/src zricethezav/**gitleaks** detect -s="/src" -r="/src/gitleaks-report.json"
- docker run --rm -v $(pwd):/scan ghcr.io/praetorian-inc/**noseyparker**:v0.16.0 report --format json -o /scan/parker-report.json (use **:latest** tag if upload to Dojo is not required)
- https://github.com/Yelp/**detect-secrets**

- GitLeaks
- TruffleHog
- Nosey Parker
- Detect Secrets

Dependency Checks

- trivy fs -f json -o trivy-dep-report.json --scanners vuln .
- docker run --rm -v $(pwd):/src --rm ghcr.io/google/osv-scanner -r --format sarif --output /src/osv-report.sarif /src
- docker run --rm -v $(pwd):/src -v $(pwd):/usr/share/dependency-check/data -v $(pwd):/reports owasp/dependency-check --scan /src --format "JSON" --project "Test" --out /reports

- Trivy
- OSV Scanner
- OWASP Dependency Check
- Snyk

Configuration checks (IACS)

- Checkmarx KICS
- Chekhov
- TFSec
- Terrascan

- docker run --rm -v $(pwd):/src checkmarx/**kics** scan -s -p "/src" -o "/src"
- docker run --rm -w /src -v $(pwd):/src bridgecrew/**checkov** --directory /src -o json | tee checkov-output.json
- docker run --rm -v $(pwd):/src aquasec/**tfsec** /src -f json | tee tfsec-output.json

Reducing false positives

- Exclude paths via ignore config files
- Exclude checks or rules using scanners features
- Manually remove test/local/mock paths from code folder before scan
- Customize rule sets
- Use taint etc scanners which can identify code execution paths
- For regular checks prefer incremental scans

Custom rules

Good old grep / regex

https://semgrep.dev/playground

https://github.com/semgrep/semgrep-rules

https://github.com/projectdiscovery/nuclei-templates

If you're not writing custom Nuclei templates, you're missing out

Defect Dojo

```
git clone https://github.com/DefectDojo/django-DefectDojo

cd django-DefectDojo && ./dc-up-d.sh postgres-redis

docker compose logs initializer | grep "Admin password:"
```

Demo Time

https://scansuite.gitbook.io/scansuite