Study, Design and Tooling of Industry Standard DevSecOps pipeline

BITS ZG628T: Dissertation

by

Sushrismita Mishra

(2018HT13036)

Dissertation work carried out at

Citrix R &D India Private Limited, Bengaluru

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE PILANI (RAJASTHAN)

December, 2020

Study, Design and Tooling of Industry Standard DevSecOps pipeline

BITS ZG628T: Dissertation

by

Sushrismita Mishra

(2018HT13036)

Dissertation work carried out at

Citrix R & D India Private Limited, Bengaluru

Submitted in partial fulfilment of M.Tech. Software Systems degree programme

Under the Supervision of
Sowrabhi Pannaga S, Staff Data Scientist
Citrix R & D India Private Limited, Bengaluru

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)

December, 2020

CERTIFICATE

This is to certify that the Dissertation entitled *"Study, Design and Tooling of Industry Standard DevSecOps pipeline"* and submitted by "*Sushrismita Mishra*" having ID-No. 2018HT13036 for the partial fulfillment of the requirements of M.Tech. Software Systems degree of BITS, embodies the Bonafide work done by him/her under my supervision.

[illegible]

Signature of the Supervisor

Place : Bengaluru

Date : 04 Oct, 2020

Sowrabhi Pannaga S, Staff Data Scientist
Citrix R & D India Private Ltd , Bengaluru
Name, Designation & Organization &Location

ABSTRACT

Today in industry there is a growing culture of delivering software in faster way by implementing DevOps practices in agile manner. While we keep maturing in the field of delivery, one grey area always remains unaddressed, that is Security. As per *IBM's Cost of Data breach report 2020*, global average total cost of data breach in 2020 till date is $3.86M.Companies mostly invest large amount of effort and resource by broadly implementing network security on their IT perimeters. But with advent of public and hybrid clouds, this perimeter is dramatically dissolving and hence making Software security or application security equally important .When companies test applications for vulnerabilities more often, they prioritize which security defects to fix based on a number of business objectives. While this creates business values and is considered as best practice, it increases the number of vulnerabilities needs to be fixed over the period of time, in turn increasing the security debt. *The 2019 Veracode State of Software Security(SoSS)report* shows that average days to fix flaws is 171days whereas only 56% of flaws eventually get fixed. Many of the companies who have already adopted DevOps practices implementing advanced CI/CD pipelines, automated workflows , still follow the traditional way of delivering security where the application is scanned at the end of initial phase of development and the security issues are addressed reactively by revisiting various aspects of code and with more frequent and large amount releases as part of agile process, there is a greater chance of introducing or reintroducing security bug with each release. So making security part of development lifecycle has become inevitable for secured application delivery. As per IBM's report, saving in average total cost of a data breach in organizations with fully deployed security automation vs those with no automation deployed is $3.58M. At this scenario, DevSecOps "comes to rescue" – the term means adapting modern security practices in the fast and agile world of DevOPs. Though it is there in practice by few of the companies from past few years, it still remains a field to be explored largely. DevSecOps is the future of Security and an industry buzz word. As part of this dissertation work , I am going to study DevSecOps culture mostly focusing on implementing security practices at various points in CI/CD process by doing Static, interactive and dynamic analysis security testing (SAST,IAST and DAST) , designing an industry standard pipeline with various Opensource Tools available in current times and developing custom or wrapper scripts and tools if required to achieve "Continuous Security" .

Broad Area of Work: Software Architecture

Key words : CI, CD, Continuous Security, DevOps, DevSecOps, DAST, IAST, SecDevOps, SAST, Network Security, Security debt, Software Delivery

ACKOWNLEDGEMENT

I would like to thank my supervisor "**Sowrabhi Pannaga S**" for her guidance and whole hearted support. Her valuable comments have been immensely helpful in enhancing the quality of work. Special thanks to my additional examiner "**Tarika Tanushree**" for getting time out of her busy schedule and help me evaluate the project work sincerely.

I am also thankful to "**Citrix R&D India Pvt. Ltd"** and my manager "**Mr. Greg Anth**" for providing me the opportunity and also providing enough resource and time to carry out the research work required for this dissertation. I am also thankful to "**BITS,Pilani"** for arranging the course and making things carryout swiftly.

Last but not the least, I would extend my gratitude to my mentor and guide from BITS **"Mr Avinash Gautam"** for the guidance and evaluation throughout the semester.

TABLE OF CONTENTS

1.INTRODUCTION

Let's begin:
Security at all times had been very important part of any organisations. It is a shared responsibility and in recent times, it is coming more into lime light due to increasing threats and breaches.

Broadly Security can be divided into three types.

1. Network Security:
   Since the inception of days, organizations focus on securing network perimeters from outsiders. This includes network components such as servers, wireless networks, and routers. It's becoming more common to refer to network security as infrastructure security. The rise of IoT, mobile, and cloud computing has created an ever-expanding, increasingly complex network for many organizations. It's harder to secure the boundaries around your network when almost all of the applications and databases your employees use every day are hosted in the cloud, and mobile devices are being used more than ever to communicate and collaborate. So, securing the "edge" points have become more crucial and evident in modern times. It is a broader category to address.

2. Data Security:
   Data Security is a set of standards followed in industry to avoid intentional or accidental loss/destruction/disclosure/modification of data. There are various majors including administrative control, physical security, logical control, organizational standards etc to maintain data security.

3. Application Security:
   Application Security majorly deals with Security of the code, deployment, environment related security risks and its control and mitigation. The ecommerce applications, web applications fall into this category.

In this paper, we will focus on the **Application Security** as the other 2 categories are broad and requires more resource to investigate.

In the recent times Software Industry has seen huge evolution with introduction of public and private cloud in web applications. The software development process has become very swift and agile with use of DevOps practices. The time to market of delivery of software has reduced and software development life cycle accelerated in a great way. But along with all these movement , the thing which always stay in grey area is Security. Security most of the time, comes at the end of development life cycle just before the release of software and always fixing them comes expensive and sometimes they are kept of later for fixing. This is described as Security Debt in technical terms. This practice has been continuing for a quite long time now, but the Cloud Infra is not only empowering developers and host of the Software , it is also making the attackers life easy. Computing power has become cheap and abundant to launch bigger and intense attacks. As per research and information, the tools being used to secure applications is also available at attacker's disposal, thanks to the Opensource Community. The security attacks and incidents has become inevitable and have started surfacing a lot in recent times. In order to keep the platform and application safe from all these threats, attacks and data breach , more careful , proactive approach is required. We will focus on similar approach in this paper.

What is DevOps ? DevOps Culture is bringing a lot more flexibility, agility and speed to the software development lifecycle. And the responsibility of a DevOps engineer different from organization to organization. It can be scaled from OS management to software management. The skills have further expanded in various ways to maintain a uniform, scalable application environment using techs like Container, getting feedback from real-time customers and implement them to improve the current software etc. DevOPs process is involved from code(like Github, bitbucket) ,build (continuous Integrations tools like Jenkins, Bamboo, tools like Docker) , test (Unit tests like Junit , Cucumber , Selenium tests/) , Release (Continuous Deployment) , Deploy (Kubernetes) , Monitor (splunk, Stackdriver etc). Starting from day1 of development to release , the whole process is powered with efficient Tools and processes. In this high velocity of release process, we need to find a way to put in Security. There comes Talk of continuous Security

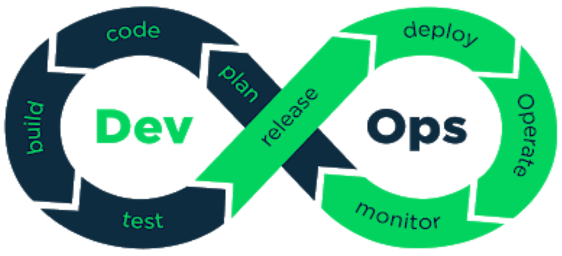


Fig1

What is Continuous Security? Continuous Security is a way to make the Security part of Software Delivery life cycle early on and don't wait till the end of the development to complete. Part of this process the idea is to Predict Security lapses during build phase and Prevent it from appearing in runtime environment and then continuously monitoring the running environment to detect any existing Security bug and Respond reasonably to mitigate them . This whole process goes in a loop and cover both static and dynamic environment. To implement such a rigorous process, a more sophisticated culture or process is required, which is named as DevSecOPs or SecDevOPs by industry leaders.

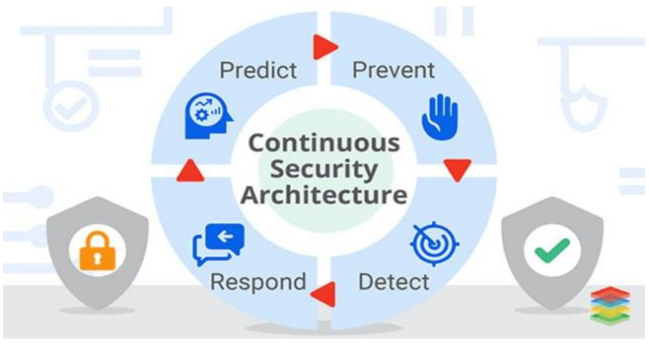


Fig2

What DevSecOPs? Like our previously described DevOps culture, it is also a culture to incorporate security related tools to achieve the continuous security. This field is fairly new and still being adopted by many of already developed and sophisticated development process. We will check more details in this dissertation work .

Background:

In my current work role having adopted to DevOps brings us a lot of agility in terms of building, testing and deploying various web applications and micro services. We have a robust DevOps tools with tight knitted pipeline, but with growing number of microservices and integration between them, security in terms of storing secrets, maintaining lib and package versions, finding CVE etc. has become serious concern. In this scenario, adopting to DevSecOps is the most logical step. There are few sophisticated and costly and many open source tools available who can do different tasks independently. So, studying, designing and tooling a pipeline with various integrations will be very helpful.

Objectives:

- Shifting Security to left in agile process
- Security practice automation
- Reduce cyber-attack surface by being proactive
- Reduce or minimize security debts
- Provide better feedback on the path starting from development to delivery of application
- Connecting the dots with various independent tools to one uniform pipeline

- Minimize the manual scan effort by individuals or teams
- Leverage open-source tools to save cost

Scope of work:

- Find out and explain various Static, Dynamic, Interactive Application Security testing procedures
- Evaluate or develop tools and services in each category for the tasks which can be automated
- For the part which still requires manual testing, evaluate how the existing traditional procedure can be fit into and how better feedback can be of help
- Implement identified tools and services into CI/CD pipeline to achieve "Continuous Security"
- Evaluate if the action can be corrective in nature rather than just informative and what are the odd points of things going haywire
- Identify or develop metric to quantify the work or result which being achieved part of this exercise

2.WELCOME – SECURITY

2.1.SDLC in Nutshell:

In software development life cycle , the basic phases are

- Planning and Requirement Analysis,
- Design,
- Implementation,
- Testing
- Deployment.

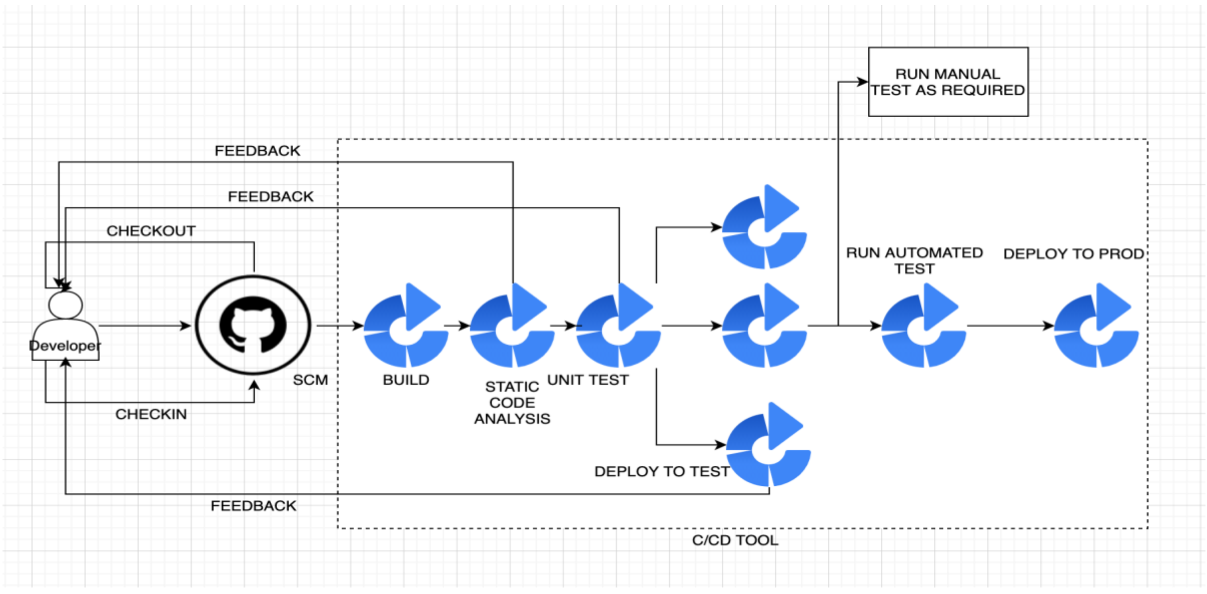
Traditionally, when a feature or application is ready to be released, it goes through different security tests and code is revisited and fixed as per requirement . With advent and wide spread of agile process the overall features are broken into smaller parts and microservices and the same phases are repeated in iteration in one or more sprints. The releases become more frequent and collaboration between different developers become a key aspect. To facilitate the increased numbered and frequent releases, DevOPs processes are immensely helpful.

2.2.A typical DevOPs Pipeline:

In a typical DevOps pipeline the developers collaborate through SCM tools like Git,BitBucket and Chekin and Checkout the sharable code. The codes are then used by CI/CD tools and perform various automated steps before releasing them to Production.
There are many popular CI/CD tools like

- Jenkin
- Bamboo
- Travis
- BB pipeline

***more on SCM tools is out of scope, please check appendices for more on SCM tool*

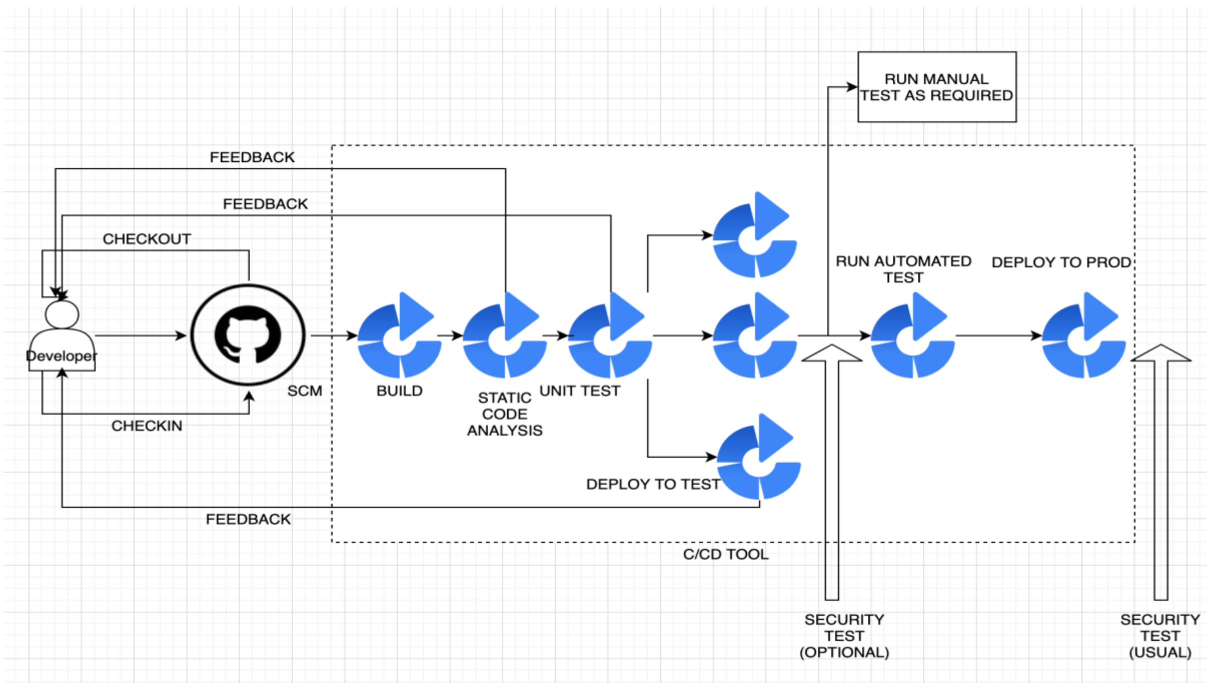


As part of CI process , as the name implies the code is built and tested continuously upon a trigger condition which can be "on each push to remote SCM" or "on each pull request raised" etc. Then the code is built , analysed and unit tested and required feedback is provided to the Developer for awareness or compliance depending on the requirement of the team.

After the Unit Test is successful , the code is deployed to any of the lower environment automated way and then the Automation code is run on them. After a successful run of test suite on the deployed code , the code is now Production ready and then released.

The whole process is done without any manual intervention and there are various break points, check points depending on the results of Static Code Analysis, Unit tests or Regression Tests. So the whole process ensures code quality, easy deployment and reliability of code. With this automated process, teams can make several releases per day or per week without worrying about breaking the current production. Hence bigger features are splitted into smaller components and released frequently rather than waiting for the last Sunday of Quarter to do a big, fat release.

As we can see there is no Security Testing involved in the pipeline and generally Security tasks are done at the phases mentioned for each release.



2.3.Shifting Security to Left:

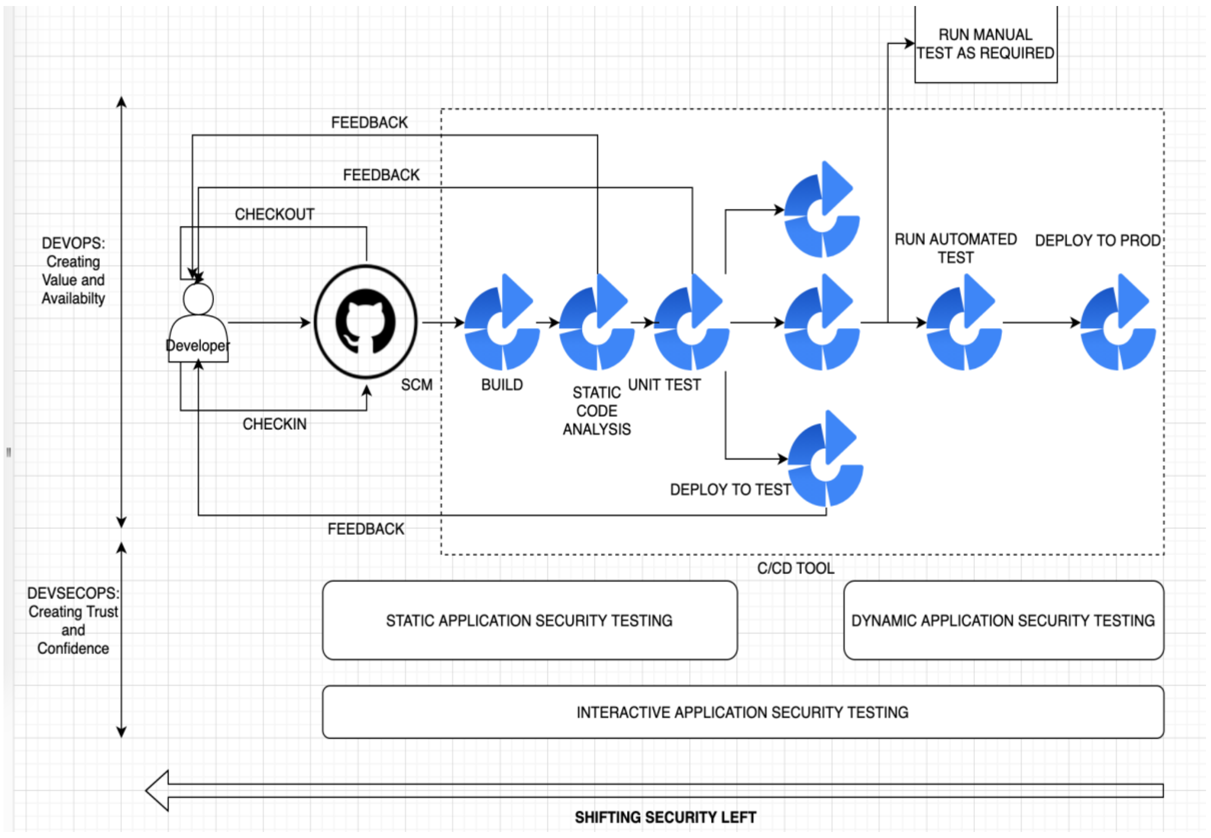
In the above mentioned process , the security tasks are run on the deployed features on Nonprod (Test or UAT) and then necessary action are taken based on Feedback. It comes right at the end of SDLC and just before the release. In some cases or organisations , this step is also skipped and the code is security test is done post deployment to Production, directly on the Production Runtime.

As per the recent research and analysis , it takes at least a month to patch a security bug in production and about 85% of the bugs remain unpatched for more than a year of time. Usually people wait for the quite time of the year, to such security maintenance. But with the larger internet footprint of recent time, it is rare for a well-established website to get that quite time of the year.

Cons of this process:

- As we discussed before larger features are broken into small pieces and being worked upon indecently by different developers, with this approach there is a greater chance of reintroducing Security Bugs with each release
- Security Debt grows with growing time
- Sometime little delay in patching the system can cause compromise
- Developers are not aware of any security loophole until the code is almost in production or already deployed to Production
- Developers might have completed the development of feature A and move to feature B by the time the code is ready to be deployed to Production (manual testing process is involved sometime) , so when the security bug is encountered the developers have to checkout and revisit the code once more which takes more time than fixing the issue then and there , when it is encountered. So the human resource effort is wasted and again the new code needs to be retested

Keeping all these cons into mind , Shifting security to Left in the Pipeline can be very beneficial to Developers


RUN MANUAL TEST AS REQUIRED
FEEDBACK
FEEDBACK
CHECKOUT
DEVOPS: Creating Value and Availabilty
RUN AUTOMATED TEST
DEPLOY TO PROD
Developer
SCM
BUILD
STATIC CODE ANALYSIS
UNIT TEST
CHECKIN
DEPLOY TO TEST
FEEDBACK
C/CD TOOL
DEVSECOPS: Creating Trust and Confidence
STATIC APPLICATION SECURITY TESTING
DYNAMIC APPLICATION SECURITY TESTING
INTERACTIVE APPLICATION SECURITY TESTING
SHIFTING SECURITY LEFT

As mentioned in the above diagram we are being proactive with respect to Security Test and beginning tests right at the moment the code is checked into the SCM .The process is called Static Application Security Testing. As the name suggests, this kind of testing focuses on scanning the code at rest, we don’t need a working application to run **SAST**. The continuous security testing begins with feedback from that point in time and the process proceeds along with the DevOps Pipeline as we move the feature step by step. *Few features of SAST*:

- White box Testing
- Requires Source Code
- Finds Vulnerability early in SDLC, so less expensive
- Does not cover run time environments, so configurations and runtime dependencies are not tested
- Developer Centric as it provides early feedback to Developer to fix any Security bug in code

Then right at the end of cycle, the production environment can be tested and monitored as before (DAST) to avoid any accidental leakage or new threats.
Few features of DAST:

- Black Box Testing
- Runs on a running application
- Finds vulnerability towards end of SDLC, so more tedious and expensive
- Can find Environment related Vulnerabilities
- Infrastructure Centric and requires more effort to fix them
- Can vary from nonprod to prod , depending on the symmetry of the deployed enviornments. So usually run on Production envs as well

There is another kind of approach where the small portion of application is Security tested from within the application . It is called Interactive Application Security Testing. It can belong anywhere in the pipeline as it can be done at any moment of time in any production ready code
Few features of IAST:

- Reports vulnerability in real time
- This type of testing also doesn’t test the entire application or codebase, but only whatever is exercised by the functional test.
- IAST works inside the application, which makes it different from both SAST and DAST

The above pipeline easier said than implemented. To implement all the kind of testing in pipeline we need many tools which can perform the extend of operation, many integrations between CI/CD tools and security tools , then the automation code to facilitate the pipeline. That’s brings us to our next topic Automated Security Testing vs Manual Security Testing.

3.MANUAL VS AUTOMATIC SECURITY TESTING

Security testing is being practiced in industry since a long time by a dedicated group of people who have extended knowledge in Security aspects of applications and network and are trained to perform various security related proactive and reactive actions. This group exploit various weak points in application and try to mitigate the risks before they get hacked by outsiders. This process requires weeks of manual deliberation, collaboration with application teams and infrastructure teams , threat modelling, review and penetration testing. Usually a team of testers and security engineers work on a given environment to identify and report the issues . The fix is verified by the same team. While the process is very effective in identifying security loop holes, sometimes due to large number of release the cross over between security testings beats the whole purpose of scanning as the bugs may or may not be reintroduced at some other release. All market leading tools with respect to Security such as Qualsys, CheckMark, Veracode provides this feature and security engineers use them to maximum extent

However as we discussed previously in order to make the DevSecOps pipeline a real thing, we need to automate few of the testing activities and which can be easily integrated to other CI/CD tools or any other service. The process is called Automated Security Testing. There are various tools in market and open-source which can perform such actions . For example: snyk , gitorb, github, dependabot , veracode, zap etc .These tools can be integrated into pipeline or tests can be scheduled at specific day or time. As an example1: while we deploy softwares we can write automations to check

1. What all ports are opened?
2. Is the destination pingable?
3. Is there any default credentials pertaining to platform?
4. Do the HTTP request validate cookies in response?
5. HTTP Strict transport protocol, XSS attack etc

Example2: We can check static details in the code without deployment
1. When we check in code to SCM, are we checking any password in plaintext?
2. Is there any production configuration accessible to general audience?

There are list of tools which can be mixed and matched to archive the required outcome. As per IBM’s Digital Asset report, savings in average total cost of a data breach in an organizations with fully developed Security Automation vs those with no automation deployed is $3.58 million. Security automation solutions including incident response(IR) team formation, testing the IR plans etc saves a lot of cost for an organization. Hence Security Automation is on rise.

Due to its vast scope in nature , we will skip to describe the Automated Testing in detail and we hereby introduce the concept as this is the building block for the DevSecOps pipeline.

4.DEVELOPER CENTRIC APPROACH

In this chapter we will discuss the security approaches part of automated pipeline at the static code level . This is mainly helpful for developers as it provides early feedback about the code's security issue and can be addressed at early phase in SDLC.

Why Developer Centric Approach Matters?

- Improved productivity
- Reduced Risk
- Improved security outlook
- Efficiency from a single platform

4.1.Secret Scanning in SCM:

Source code is made to be duplicated and distributed, hence it is considered as leaky asset. It can be cloned to a compromised server or desktop , intentionally or accidentally published in whole or in part, uploaded to the website, released to any customer, shared via in Slack, or end up in the package manager or mobile application. Hence keeping the secrets (app passwords, database passwords, api keys etc) is not safe. If a single developer's account gets compromised , the attacker can get access to all the secrets. Hardcoded credentials are difficult to keep track and rotate as well.

Secret detection and handling can be confused with SAST. However SAST only deals with current state of a project , whereas the secret Management involves all historical commits in the repository. If a secret is ever checked in to the remote SCM is considered to be compromised and has to be rotated. This can archived using Git Hooks.

What are git hooks? Git hooks are scripts that are triggered by certain conditions satisfied in the software development process, like committing or pushing. It strengthen the review process by enforcing few rules automatically and reduce time and mistake by reviewers

Hooks can be configured at

- Client Side – locally on developer's workstation
- Server Side – On the remote VCS server

Examples of client-side hooks:
Pre-commit:

The pre-commit hook is run when committing, before you even type in a commit message. It's used to inspect the change that's about to be committed.Non-zero exit status from this hook aborts the commit

Post-commit:

Similar post-commit runs certain task after a change in committed in developer's workstation.

Pre-push:

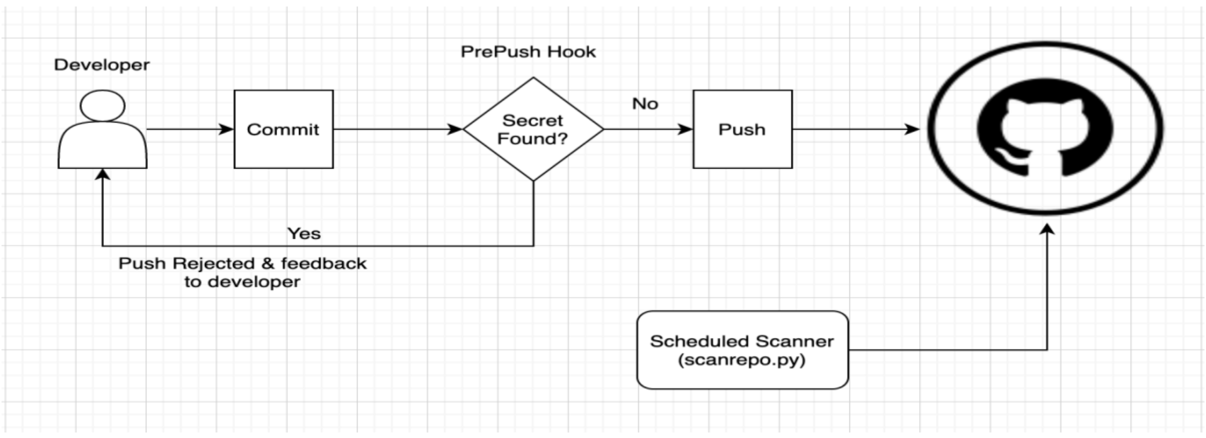
This script is run after a change in commited in Developer's local workstation and before pushing into Origin,

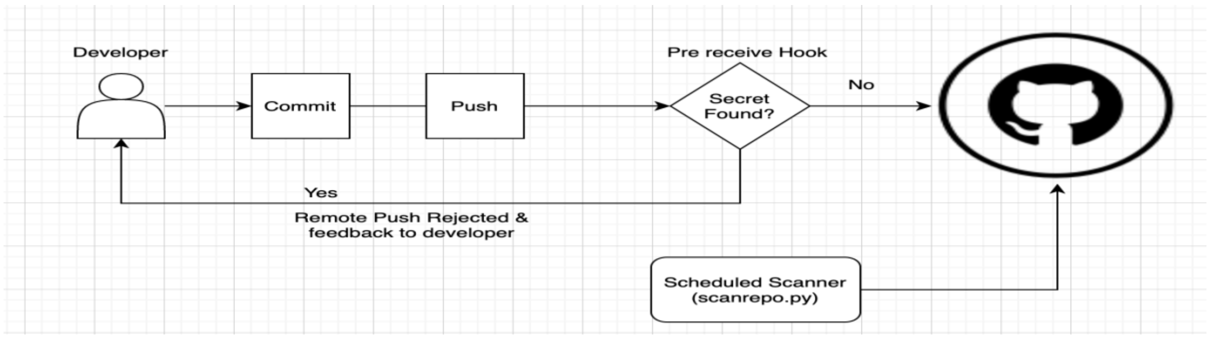
Examples of server-side hooks:
Pre-receive:

A pre-receive hook is a script that executes on the server. It performs checks on the content of the push; if it exits non-zero, the push is rejected. This is blocking in nature, so lots of false positive can make the experience bad for developers and needs to be used very carefully.

Post-receive:

similarly Post-receive scripts can be used *after the entire process of pushing code to the server is completed and can be used to update other services or notify users.This is nonblocking in nature.*


Developer
PrePush Hook
Commit
Secret Found?
No
Push
Yes
Push Rejected & feedback to developer
Scheduled Scanner (scanrepo.py)


Developer
Commit
Push
Pre receive Hook
Secret Found?
No
Yes
Remote Push Rejected & feedback to developer
Scheduled Scanner (scanrepo.py)

Where in the DevSecOps pipeline to implement automated secrets scanning? **Client-side or server-side?**

As mentioned before if the secret is discovered after the secret reaches remote SCM server, it must be considered compromised, which requires rotating the exposed credential. So the earlier the best. However implementing and maintaining Client Side hooks in large organization can be very difficult and tedious. From the server, the code can uncontrollably spread in a lot of different places, with the hardcoded secrets in it. So Server-side secrets detection is more preferable and is a must have.

Blocking or nonblocking Secret Detection in the SDLC? From the feedback I got during the process, when trying to impose rules that are too tight, people tend bend them, often in an effort to collaborate better and do their job quickly and effortlessly, as it becomes a blocker. Eventually it does not matter anymore. Security must not be a blocker. As automation can fail in some scenarios, it is a clear tradeoff and culture rather than a fixed process. So, the developers and owner of repositories should be responsible enough to maintain and respect the secret management rules.

**scanrepo.py is a custom script created by me to scan codes on Bitbucket cloud. There are many pre written scripts are available in github.

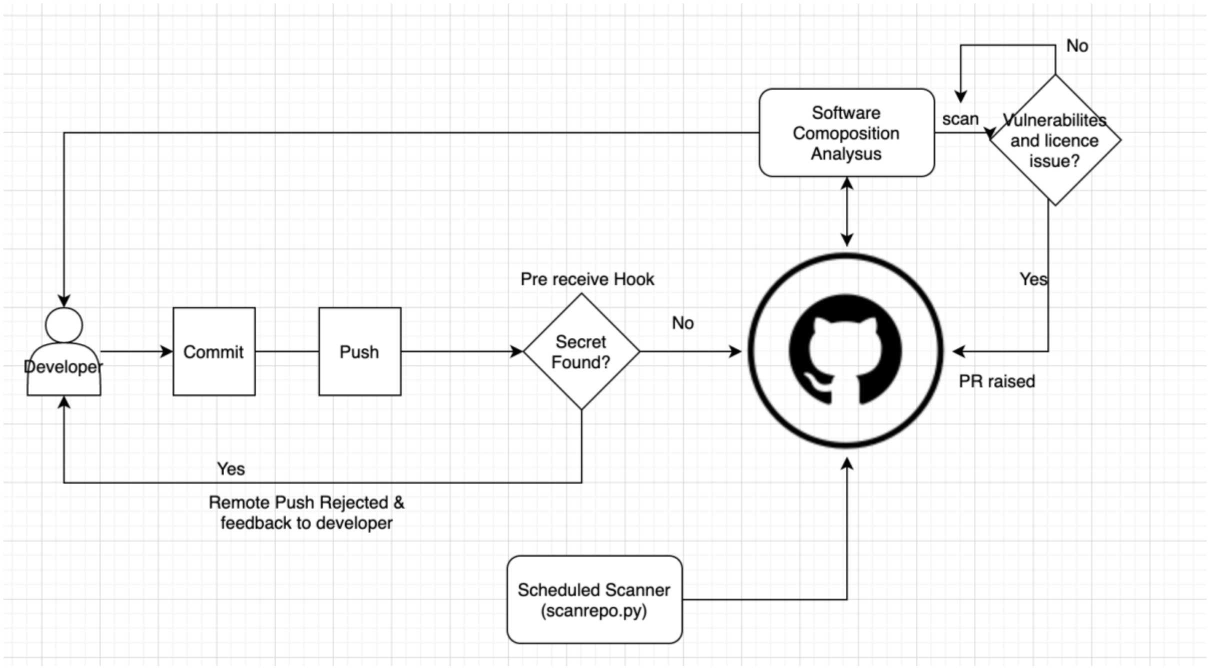
4.2. Open-Source Security & SCA:

Now a days opensource packages and tools are on rise and are present in almost all software and projects. These packages introduce many other dependency packages knowingly or unknowingly both from private and public repositories. Software Composition Analysis tools let you scan your repository to find the dependencies, discover vulnerabilities in those dependencies and patch them. With this implemented when a new vulnerability is discovered for third party or opensource dependencies, the pipeline will be able to detect and fix the vulnerabilities. These tools can directly work on repositories or part of CI Pipeline. Example of such tools

- Snyk - (This is demoed)
- Dependancy Graph
- Dependabot
- Dependancy Insights
- Whitesource

Another important aspect of Opensource Code is its license management. These tools help analyzing the license and report if there is any problem.

Adding to the previous pipeline:


No
Software Comoposition Analysus
scan
Vulnerabilites and licence issue?
Yes
Pre receive Hook
Developer
Commit
Push
Secret Found?
No
PR raised
Yes
Remote Push Rejected & feedback to developer
Scheduled Scanner (scanrepo.py)

Here the SCA tool is directly integrated to Repository providers such as Github, Bitbucket and scans the code of the respective branches and raise PRs If the code is automatically fixable or it send notification to organization owner/repository owner or committer as per selected notification preference.

If every single commit needs to be verified or analyzed the same procedure can be part of CI pipeline before building the code and gateways can be defined whether to break the pipeline or pass it based on some allowed threshold.

Whether SCA should be blocking or nonblocking? Initially there can be lots of false positives and failing the builds at first can create erosion between security practices and developers whose productivity will be blocked by lots of false positive. So as future the reports from the scan needs to be scrutinized, deduplicated and provide timely feedback to developers to fix the problems rather than breaking the level. As the pipeline matures and rules are finalized, breaking the build blocking the development can be consider for stringent security

These tools can also be integrated to Developer’s IDE as plugins to early detect any vulnerabilities in any package version.

4.3. Static Depth - Security Hotspots and Security Vulnerabilities at code:

Security Hotspots highlight suspicious code blocks that developers should review and triage as they may hide a vulnerability. As developer code and discover hotspots, they learn how to evaluate the security risk while becoming more acquainted with secure coding practices.

Security Vulnerabilities are those which require immediate action and already important to fix them.

Different SAST tools detect both security hotspots and Vulnerabilities at code level in all majorly used application language such as java, nodejs etc. SAST tools scale well. They can be run repeatedly, too, such as during overnight builds. They can be easily integrated into IDE. They can also identify common errors like buffer overflows, XSS problems and SQL injection flaws. As mentioned above, after they find an error, they can make life easier for developers by identifying source files, line numbers, and even subsections of lines containing errors. SAST tools are not all in all ,they can’t find authentication problems, configuration flaws, bad encryption etc. For which DAST tools are needed.

Few of the popular tools in this category are:

- Sonarqube
- CheckMarx
- Veracode
- AppScan
- Brakeman
- FindBugs
- JsHint

Usually this tools are integrated to CI pipeline to run scans at required frequency. For larger codes it is run overnight or for smaller code bases it can be run on each commit or PR created. The tools provide rich UI showing the results categoriesed based on Date/time, type, severity and Developers can follow through the hints and details provided to fix the anomalies.

Like all other tools, these tools also produce lots of false positives. As the pipeline matures the rules also mature and at level1 braking builds based on failures at this level is not recommended.

4.4. Dynamic Depth – Running Dynamic Scans in CI pipeline:

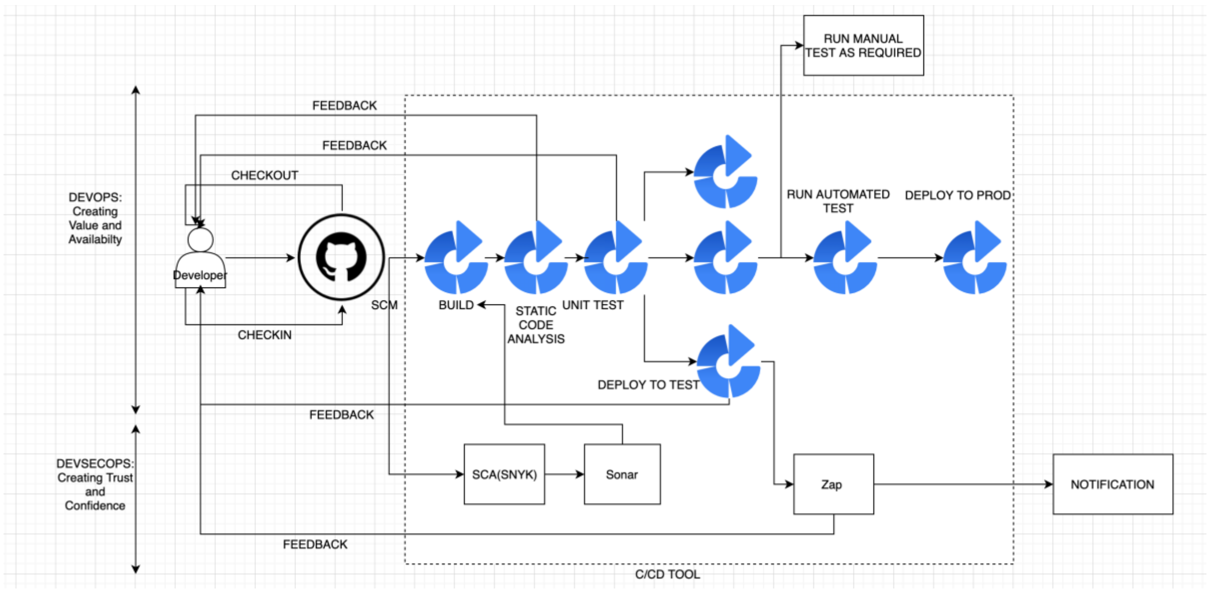
Dynamic scans offers scanning application content and code in a running environment apart from infrastructure setup. We can perform Dynamic scanning in various ways

- Scanning public attach surface (without authentication)
  In this mode we can spider or crawl through the UI , testing various components. As there is no need to authenticate the scanner with the site, it is easier to achieve
- Scanning authenticated parts (post login)
  It includes additional challenges of properly maintaining sessions, logout detection and relogin, selection of different test users with different roles etc
- Scanning different application layers or backends
  This includes scanning webservices like SOAP, REST
- Targeted Scan of specific forms or UI or service layers
  Testing custom code, complex business logic , Compliance etc

We can perform dynamic scans using various tools few of them are :

- ZAP
- Arachni

A BDD (Behaviour Driven Development) based Framework "Gauntlt" provides integration to various security scanning tools including Zap , Archini, sqlmap, nmap, curl and any custom security framework and makes it easier to integrated to CI pipeline.


RUN MANUAL TEST AS REQUIRED
FEEDBACK
FEEDBACK
CHECKOUT
DEVOPS: Creating Value and Availabilty
Developer
SCM
BUILD
STATIC CODE ANALYSIS
UNIT TEST
RUN AUTOMATED TEST
DEPLOY TO PROD
CHECKIN
DEPLOY TO TEST
FEEDBACK
DEVSECOPS: Creating Trust and Confidence
SCA(SNYK)
Sonar
Zap
NOTIFICATION
FEEDBACK
C/CD TOOL

5.Metric

OWASP defines a Maturity Model with respect to DevSecOPs Pipeline maturity named " DevSecOPs Maturity Model" . In this model below 4 components are defined and configured as per its implementation level.

- Static Depth: How deep the static code of the application is tested part of the pipeline?
- Dynamic Depth: How much features or functionalities can be covered by running Dynamic Scanning on the environment?
- Intensity: At what intensity the pipeline is incorporated in day to day CI/CD pipeline
- Consolidation: How well all the tools run and can produce reports to be studied/understood by developers?

Let's evaluate the pipeline we just designed.

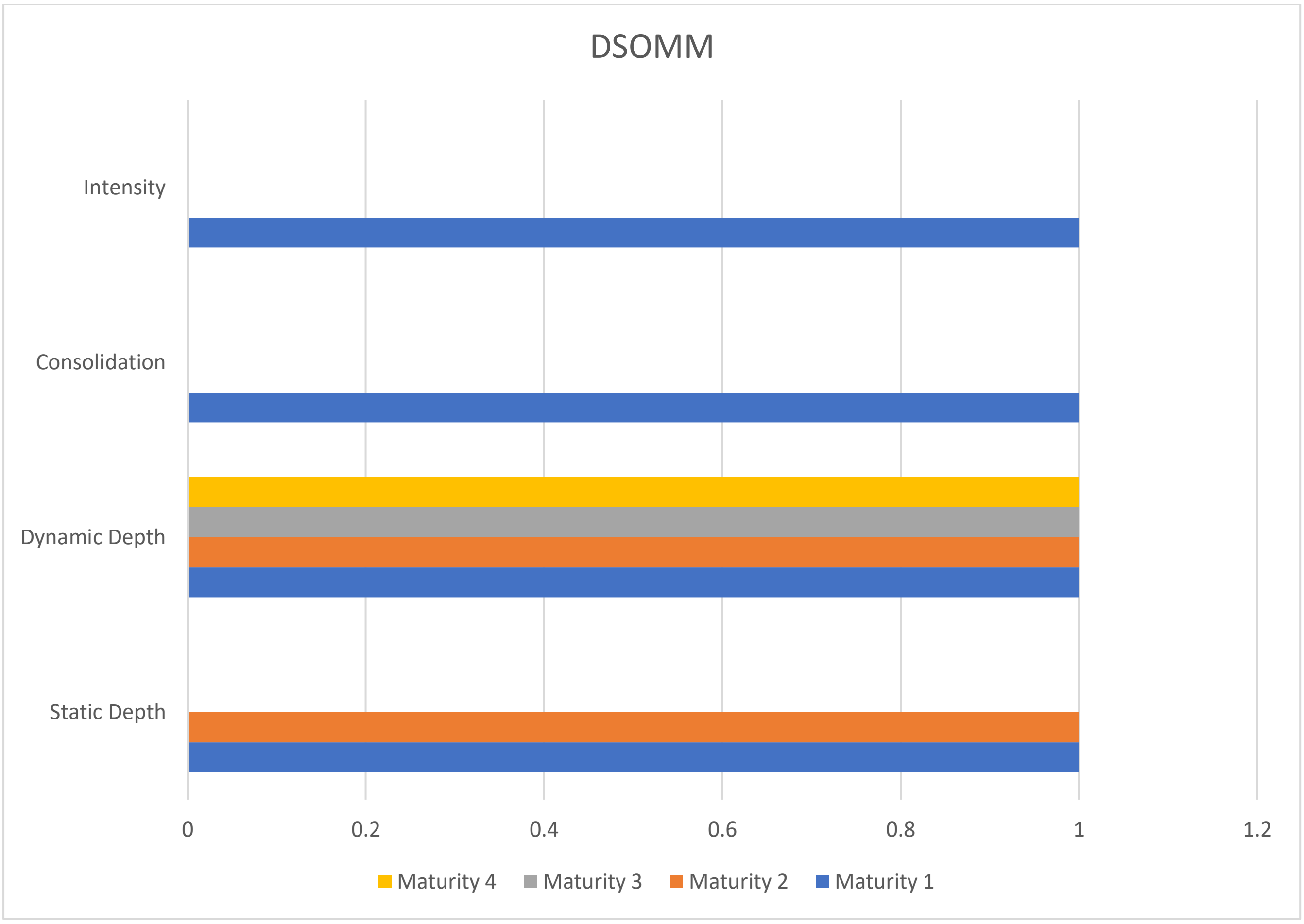
1. For static depth of the code base we are able to design the Secret Scanning, SCA, SAST , however these tools produce a lot of false positives and needs careful deliberation to come up with a proper ruleset to rule out false positives. Hence matching with the guideline the maturity Level is at 2

2. For Dynamic depth we are able to be penetrated deeper with 4 different case studies and we are able to test custom & complex business logic, so the maturity level is at 4

3. For intensity, at the beginning of the implementation everything is very new and raw. So, enforcing these stringent rules can hurt developers' productivity and interest towards the security practices can be reduced or lost. So, at first, these tests can be run as Nightly or weekly jobs. As there are lots of scope in this regard to improve, the maturity level us at 1

4. Consolidation is the most important thing in the process to produce results and feedbacks. In this pipeline we are not breaking any builds based on certain criteria , rather we are producing Compiled reports in html or xml format and provide feedback to developers on the same. So the maturity level is at 1

All the metric is provided in below chart:


DSOMM
Intensity
Consolidation
Dynamic Depth
Static Depth
0
0.2
0.4
0.6
0.8
1
1.2
Maturity 4
Maturity 3
Maturity 2
Maturity 1

6.Infrastructure Centric Approach

As we focused mainly on application security in the above document approach , but we cannot ignore infrastructure security. It has various components which requires attentions and there are various tools in market which implements different security majors. We will not discuss them at large in this paper. Few of them are highlighted below:

- Container Security -
  There are software like snyk , GCR scanner which scan respects docker images in real time and lists out any CVE errors and report them for action
- Infrastructure as code Security –
  With cloud native application , IaaS has become a common thing for eg: Kubernetes yaml files , CloudFormation Template, Terraform config files. Tools like snyk and many like can scan these configuration files and detect any outdated API or version used and report them back

- Logging
  In security logging is very important as it provides first point of insight about any suspicious activity. Tools like Splunk is best suited to monitor logs from CI servers and even traditionally they are used to log the firewall logs in Network Security
- Monitoring
  Logging and monitoring go hand in hand and very very important from Security point of view. There are few market leaders in this category which has been providing service in the industry since a long time . Few of them are : Metaspoilt, evident.io, splunk, FireEye Madian

- Infrastructure Hardening
  Infrastructure Hardening reduces the risk of exploit and had been in practice since a long time.In this approach the base images (be it docker image or virtual machine image) , a certain set of rules and required settings are only permitted. With the advent of various Configuration Mangement tools , it can be better controlled . Market leader is such tools is Chef.
- Patch Management
  Security patches are released frequently all vendors or on SoS basis as applicable. To manage such Patches tools like Chef-Inspec is very useful which maintains a certain Patch level across environments and apply changes as required and scheduled.

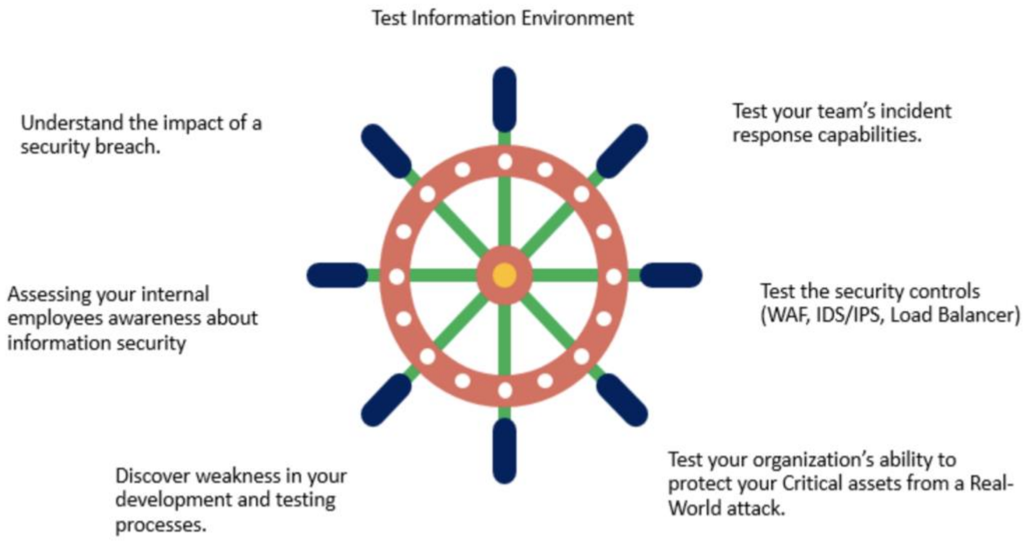
7.What’s more?

In spite of all these automations and pipelines we cannot rule out the need for manual testing with respect to Security Testing. Different organizations do the testing in different ways, but mostly Red/Blue team testing is a popular choice in this regard.

7.1.Red Team Exercise:

Red Team Exercise is an way of performing of multi-layered cyber-attack targeting already known and agreed upon objectives that include networks, technical and physical assets, storage devices and many more. The exercise and assessment performed helps in improving security defences by letting the developers or ops person to experience a real-world data breach and thereby giving a bigger picture of your organization’s risk posture, security architecture, and your team’s readiness in detecting and mitigating the threat proactively.

Red Team Exercise detects security vulnerabilities by penetrating networks, assessing processes, and testing the defensive capabilities of security teams in all possible ways.


Test Information Environment
Test your team’s incident response capabilities.
Test the security controls (WAF, IDS/IPS, Load Balancer)
Test your organization’s ability to protect your Critical assets from a Real-World attack.
Discover weakness in your development and testing processes.
Assessing your internal employees awareness about information security
Understand the impact of a security breach.

Benefits of RedTeam Exercise:

- Quantify Risk Factors
- Upgrade Security Postures
- Be prepared
- Know about the gaps
- Test the IR (Incident Response) efficiency

6.2. Blue Team Exercise:

A blue team consists of security professionals who have an inside out view of the organization. They have access to the Code Base, Infrastructures details, Hosting platform etc details. Their task is to protect the organization’s critical assets against any kind of threat. They are well aware of the business objectives and the organization’s security strategy. Therefore, their task is to strengthen the castle walls so no intruder can compromise the defences. These are proactive actions taken by organisations to avoid any sort of intrusion.

8.Conclusion:

Security is the grey area which no one likes to talk about because of the extra work and effort it induces on each individual in any roles or responsibilities. But as the time comes, Security is becoming primary concern for major of the organisations as everyone is moving to Private/public Cloud. In this paper we studied various security practices with respect to web applications and checked out many of the tools which helps in doing Security review and testing.

The primary objective of us was to create a DevSecOps pipeline keeping Security Automation testing in mind and using a number of tools to achieve the same. However as we progressed we mentioned even if we have the most roubust and perfect pipeline , we can not rule out the Pen Testing or manual Security testing. But by implementing pipelines we can free up Security Engineer’s time and mind power to focus on more important and serious issues rather than repeating the same procedure for each and every release or environment. The pipeline also helps in keeping up the speed with the DevOPs practices. *While DevOps practice provides us Availability and agility, implementing DevSecOps will provide us Confidence and Trust.*

9.Future Work:

This paper is very basic in nature as Security is a very vast area and covering each component in detail is nearly impossible. While the paper focuses on the DevSecOps pipeline of Application testing , the similar research can be done in Infrastructure Security Testing as well.

Below are few topics which can be further researched:

- Implementing Zap code with java or similar
- Infrastructure Security Testing Pipeline – Including it in DevSecOPs

REFERENCES

[1] https://www.redhat.com/en/topics/devops/what-is-devsecops

[2] https://www.csoonline.com/article/3398485/28-devsecops-tools-for-baking-security-into-the-development-process.html

[3]https://www.veracode.com/state-of-software-security-report

[4]http://www.rcolomo.com/papers/314.pdf

[5]https://www.devsecops.org/blog/tag/DevSecOps+Explained

[6]https://www.ibm.com/security/digital-assets/cost-data-breach-report/#/

[7]https://www.atlassian.com/continuous-delivery/principles/devsecops

[8] https://github.com/devsecops/awesome-devsecops

[9] https://christian-schneider.net/slides/OWASP-AppSecEU-2015_SecDevOps.pdf

GLOSSARY

LIST OF ACRONYMS AND ABBREVIATIONS

ABBREVIATIONS	**MEANING**
CI	Continuous Integration
CD	Continuous Deployment
SAST	Static Application Security Testing
DAST	Dynamic Application Security Testing
IAST	Interactive Application Security Testing
SDLC	Software Development LifeCycle
PR	Pull Requests
SCM	Source Control Management
VCS	Version Control System
IDE	Integrated Development Enviornment
SCA	Software Composition Analysis