LS^2

Shift Left Application Security
SeaGL Open Source Conf 2019

Imagery - the bridge on the left is the future. The one on the right is the past.

AGENDA

- Intro, Problems, Challenges
- A Solution
- Strategy
- Tactics
- Summary
- Q & A

QUESTIONS FOR YOU

- What's your current role?
  - Dev, QA, Security, Infrastructure, Management, BA, UX
- Does your company or organization have a formal application security program?
- Are you responsible for delivering software or packaged software to end users? (custom or COTS…doesn't matter)

If so, you are in the right place!

ABOUT ME

- FullStack software developer at ThoughtWorks Inc. 13 years of experience in software/IT.
- 8 years as a contract software developer While working on a contract at NIKE I learned the concept of "shift left testing" for *performance testing.* This led to me being the first engineer to figure out how to *fail* the automated build if performance goals were not met.
- Certified Ethical Hacker. Developer who is enthusiastic about security and privacy

As a longtime motorcycle rider, “Left Shift” came more naturally than “Shift Left”.

For purposes of this talk, they mean the same thing.

SHIFT LEFT @NIKE

- Performance issues were resolved almost immediately after discovery
- Close collaboration between application developers and performance engineers
- First code commit: **1**-JUL-2014 > First commit of perf test code: **6**-JUL-2014 > Go Live: JAN-2015.

Traditionally performance testing was done near the end of a product's development cycle.

NIKE committed to changing their people and process to "shift left" with performance testing. This paid off in good ways.

IF THIS COULD WORK FOR PERFORMANCE, WHY NOT APPLICATION SECURITY?

Lots of similarities exist between Performance testing and Security testing.

—

1. Security testing is usually done late in the product's dev cycle
2. Security testing uses tools that require experts to run them or to interpret the results
3. Devs tend to not know much about it

WHAT IS APPLICATION SECURITY?

- **Formal Definition (wikipedia)**: Application security (aka AppSEC) encompasses measures taken to improve the security of an application often by finding, fixing and preventing security vulnerabilities.

- **My Definition** (inspired by Tanya Janca, one of my security heroes): Anything a team does that enhances the security of an *application* or its *data*.

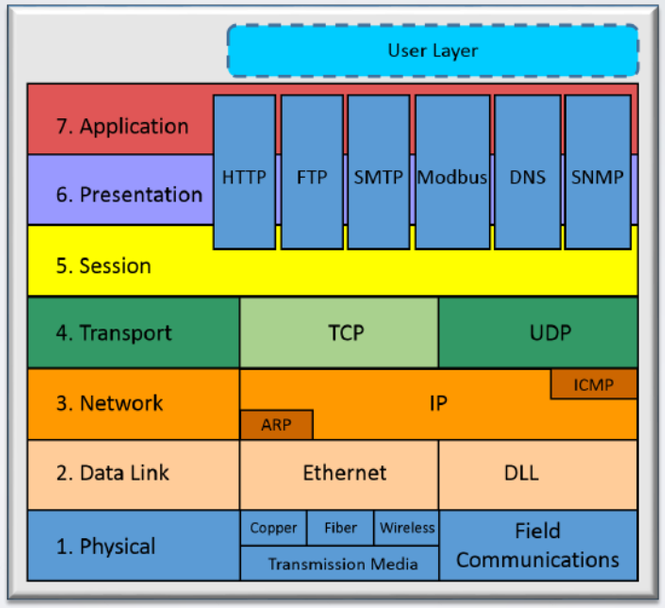
If you have not seen her presentation on “Shift Left like a Boss”, I would highly recommend it. Also the DevSlop series is excellent for getting started with application security

https://www.devseccon.com/seattle-2019/speaker/tanya-janca/
https://github.com/shehackspurple/TTT-Pushing-Left

PROBLEMS | CHALLENGES

“Problems are only opportunities wearing work clothes.” – Henry J. Kaiser

LARGE ATTACK SURFACE. MULTIPLE PROTOCOLS.



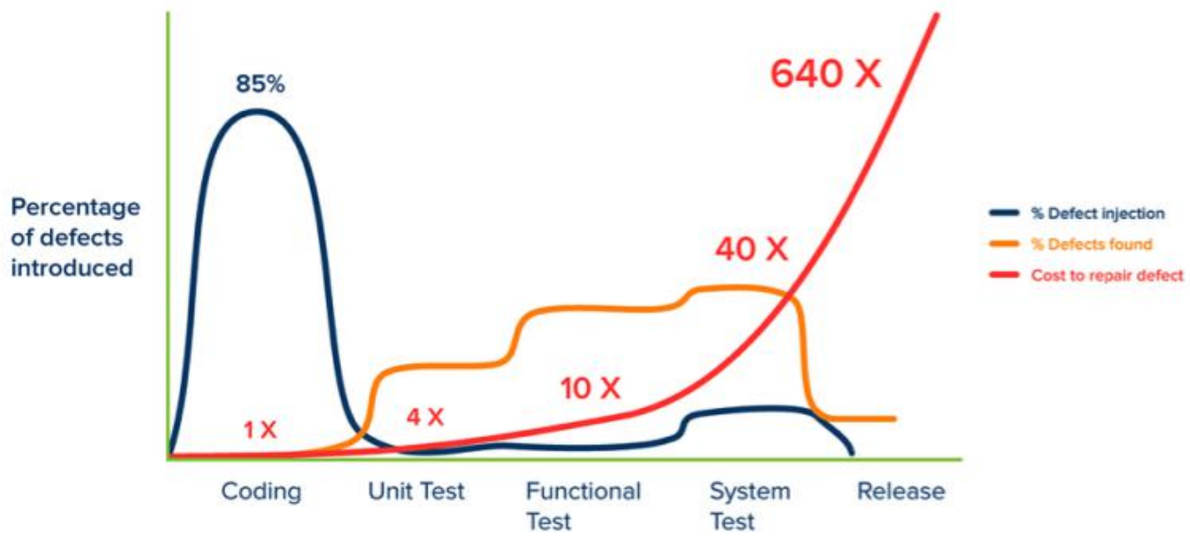
OSI model builds a protocol in layers

Application layer is the hardest to defend.

- Vulnerabilities often rely on complex user input scenarios
- Layer is most accessible and exposed to the outside world

In 2018, 20.4% of web traffic had bad or malicious bots designed to create automated attacks on web applications, APIs and mobile applications -> Source: 2019 Bad Bot Report (Distil Networks)

WAITING IS EXPENSIVE



Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality.*

Requirements and Design have their own defect introduction rates and resolution costs, but these tend to be on the lower end of the cost scale (like < 1)

SKILLS SHORTAGE

- Security personnel are in short supply. Typical ratio of devs to general security experts is **10:1.** For Application security, this could be an even higher ratio. This limits test coverage as well as ability to scale.
- *Developers typically do not know much about application security.* General Security is not a core part of most Computer Science programs, and application security is even less commonly taught. Most developers learn security on their own in the field.
- In 2017, number of *unfilled* cyber security jobs in US was 350K. This is estimated to reach 500K by 2021.

- Average time between hacking attacks = 39 sec
- Average data breach cost = $150M

https://www.csoonline.com/article/3200024/cybersecurity-labor-crunch-to-hit-35-million-unfilled-jobs-by-2021.html

MODERN CLOUD NATIVE ARCHITECTURE IS DIVERSE

- Docker images can possibly have embedded vulnerabilities
- Container orchestration tools (aka Kubernetes or Docker Swarm) have their own security issues
- Generally, distributed applications and components are harder to test and monitor than singleton applications

Most application security test tools are geared towards protecting a single point of entry to the application, whereas modern applications have multiple points of entry

A SOLUTION

Change process and culture to address security sooner in the software development life cycle.

- Test *early* - invite the security team to the party early and insist they stay for the whole time.
- Work *cross-functional - as* you are developing your plan, get input from other teams such as System and Application Architects, Infrastructure Engineers, Network Engineers and Governance. These roles all have spheres of influence that critically affect security of applications (both custom and COTS).
- Test *continuously* - throughout the software life cycle

CORE CONCEPTS

- **Empower** team to fix things on their own as much as possible.
- **Assure** the build is not impeded by testing, but also will fail if bonafide security issues are found
- **Protect** the application in production. Make sure you can determine who is attacking. Don't create alert fatigue (false positives)

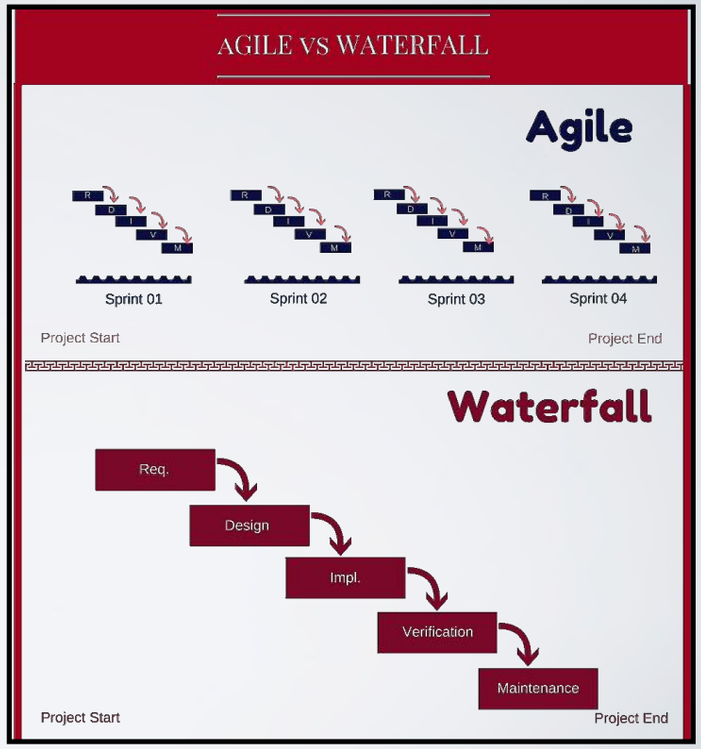
EAP - concepts introduced by Jeff Williams (Contrast Security). Former leader at OWASP org and author of WebGoat vulnerable application.

EMPOWER THE TEAM

RETHINK THE PROJECT LIFECYCLE

Regardless of the approach your team uses, there are opportunities to address AppSecurity ***before*** code is ever committed

This will require trust, collaboration, and probably a bit of developer training



Is “Project End” really the end?

- From security and quality standpoint….NO

INCLUDE SECURITY MINDED QUESTIONS IN YOUR DESIGN

- What will the deployment environment look like? Where will the database live? Will the application need to go across firewalls or be in the network's DMZ?
- Are security review checklists available for components used by your application?
- What configurations will be needed?
- What classification of *data* will your application handle? Are there regulatory requirements for encryption or other secure handling?

Be especially mindful of Data during the design phase in your SDLC. Most security breaches are actually related to exposing or stealing sensitive data from applications.

THREAT MODELING

Figuring out negative use cases, and ways to defend against them

Basically a brainstorming session with programmers and security to figure out how someone may try to abuse your app

Search your code for these threats

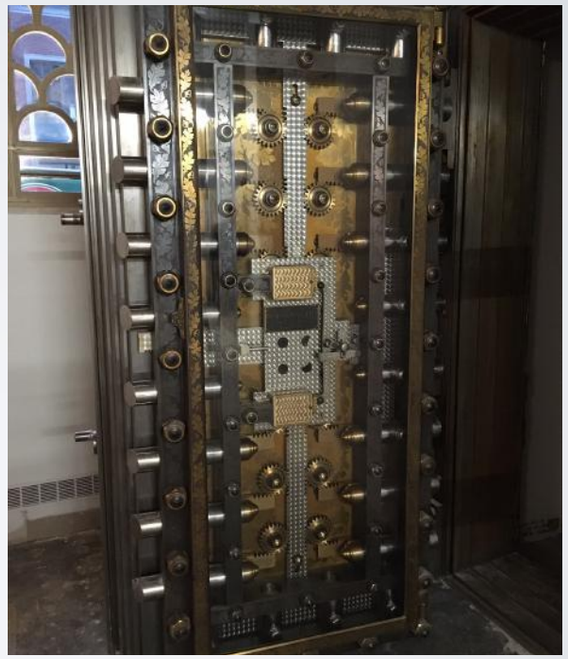
Thinking like an adversary can not only uncover potential issues, it can be fun and educational.

Consider using ideas from formal threat modeling methodologies that are appropriate for your company's security culture.
A few frameworks for threat modeling —
- STRIDE (Microsoft)
- VAST
- OCTAVE
- Trike
- P.A.S.T.A

—
ThoughtWorks has published an open source cue deck to help teams use the STRIDE framework —
https://thoughtworksinc.github.io/sensible-security-conversations/materials/Sensible_Agile_Threat_Modelling_Cards.pdf

SKILL-UP ON SECURITY

- Start with your local OWASP Chapter. Other good sources are *NIST* publications, security advisories for your chosen platform and online learning resources such as cybrary.com, safeCODE.org, Reddit. sqreen.com publishes great content to help developers stay current on security issues.
- As you learn, *shar*e with colleagues. Establish a library of reusable secure components. Consider open-sourcing it if it is not overly proprietary.
- Hold secure code reviews This is a great time to engage with your local security experts. Find out what kinds of things they would look for when attacking your application.
- When searching for code help (ex. StackOverflow), be aware of that the first solution found might not be the most secure. Try using https://security.stackexchange.com/



* If you are a Dev, don't worry! You do not have to become a security expert to write secure code. But it does not hurt to become well versed in the fundamentals so you can avoid making simple mistakes.

* To get started, check out this resource list —
* https://techbeacon.com/security/developers-security-guide-50-online-resources-shift-left

Many security vendors such as *CheckMarx* and *Veracode* offers training in secure code practices. Check with your DevOps or Security team to see if something might be available at your company from an existing account

INSTRUMENT THE CODE

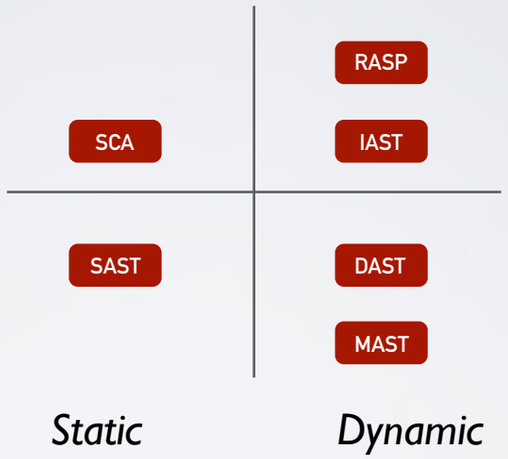
- If possible, add dynamic agents to your application. At the very least, make sure things are logged with security in mind. This also future-proofs your testing strategy.
- Find out what the security team is using and collaborate with them to find the most appropriate tool for your application.
- Some tools can look like a virus, so it is important to keep Security Team in the loop!

THE 'AST' FAMILY

New and emerging

Legacy and established

SAST - Static Application Security Testing
DAST - Dynamic Application Security Testing
SCA - Software Composition Analysis
IAST - Interactive/Instrumented Application Security Testing
RASP - Runtime Application Self Protection
MAST Mobile Application Security

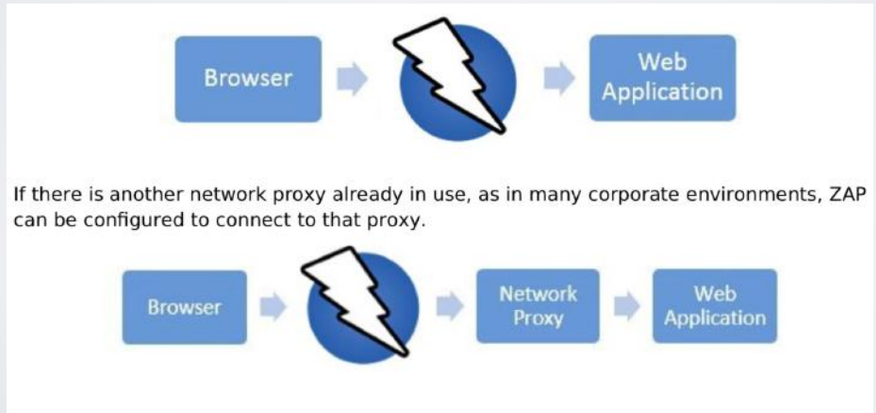


Many of these tools have open source options.

**For those of you looking to create something an open source project, we could use more open source DAST, IAST and RASP options!

MORE ABOUT **DAST**

Many of these types of tools use a 'man-in-the-middle' approach to check application traffic.



NOTES OF CAUTION -

1. Ensure you have permission from your boss before you start, there may be policies against it (ask the security team too!)
2. Be considerate, scanners can hog resources
3. Be careful, scanners can be destructive
4. Back up your data before hand
5. This is an activity that requires some learning before you can start, to ensure you don't cause any damage or tick anyone off
6. Inform security when you start and finish

MORE ABOUT **IAST**

- Instrumented Application Security Tools are embedded to the application under test. This enables *real-time* monitoring of application logic and runtime to look for anomalies using machine learning algorithms. Anything suspicious is confirmed with a central authority hosted on premises or at the tool provider
- Can be used in the developer IDE, in preProduction or in Production
- This class of tools can not only tell you *where* the problem is; it will also suggest *how* to fix it.

This type of test tool is predicted to eventually replace DAST tools

RASP - RUNTIME APPLICATION SELF PROTECTION

- Agents are installed on the application server. Activity is watched and used to build models based on known intents of the application
- Protects the application from the Inside Out
- Can preventively shut down processes, if configured to do so
- Safe for production use

EACH TYPE OF TOOLS HAS PROS AND CONS

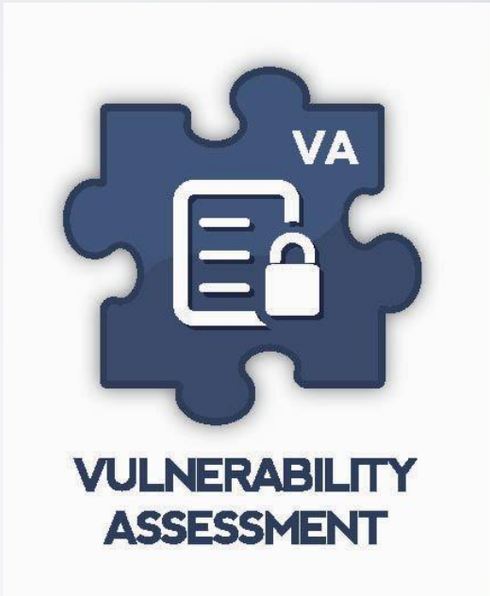
- SAST - Pros: [Easiest to understand, Programming-language-specific]. Cons: [Poor accuracy. Can be slow. Not usable in production. Results are static & tend to become outdated.]
- DAST - Pros: [Programming-language-agnostic, support manual testing]. Cons: [Poor coverage, Requires more specialized knowledge to use, Requires tuning. Reports are static, Can be slow]
- IAST - Pros: [High accuracy. Can be used in preProd as well as prod environment. Realtime results. No tuning required]. Cons: [Require specific language support]

ABOUT TOOLS…

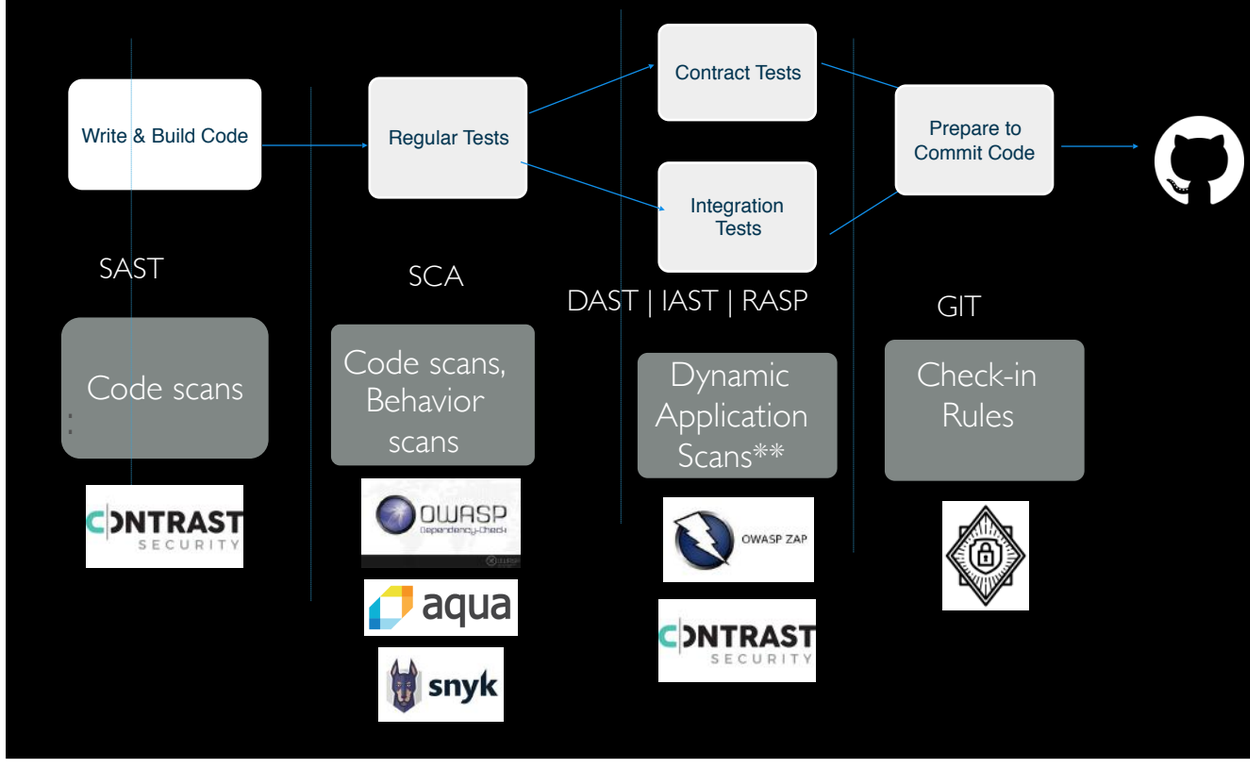
- Tools are fun to explore and use, but by themselves will not make your application secure.
- Security is not something you buy or acquire. It is something you **do**.
- Focus on *people* and *processes* first. Then pick tools that will best support your security requirements.

VULN SCAN != VULN ASSESSMENT

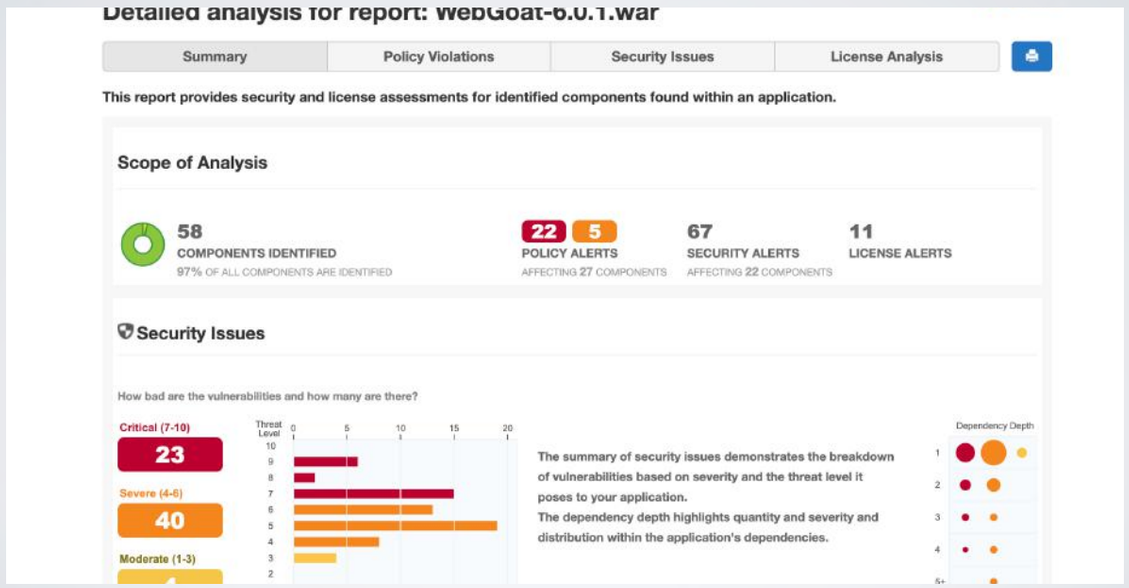
- Results of security scans should undergo an assessment of what is most relevant for the application.
- Be sure to keep track of dependencies
- Engage a security expert to do the assessment and *schedule lots of time* for the analysis



LOCAL SECURITY TESTING EXAMPLE

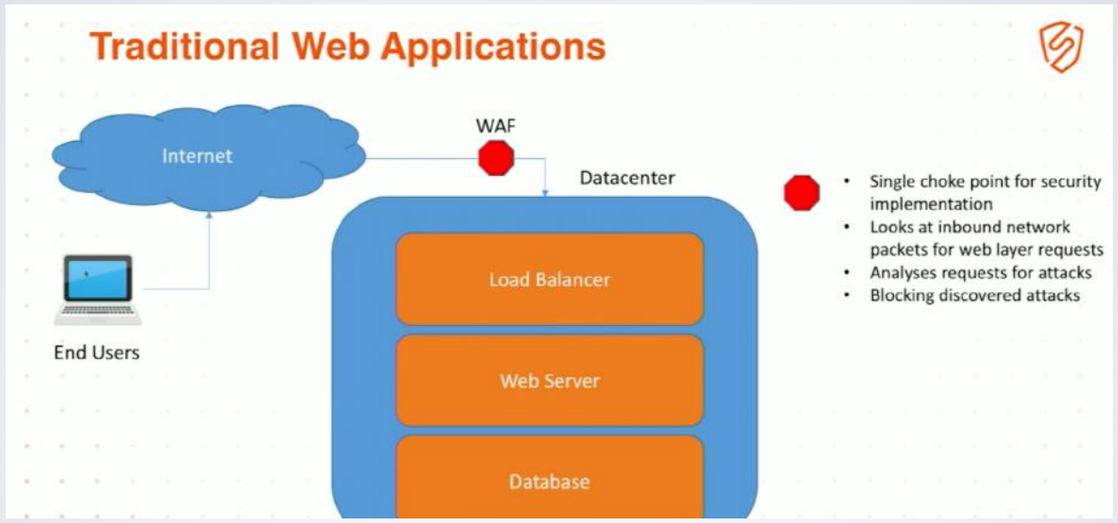


SAMPLE REPORTS



Sonatype - SCA - Dependencies

OLD PROTECTION VERSUS NEXT GEN PROTECTION

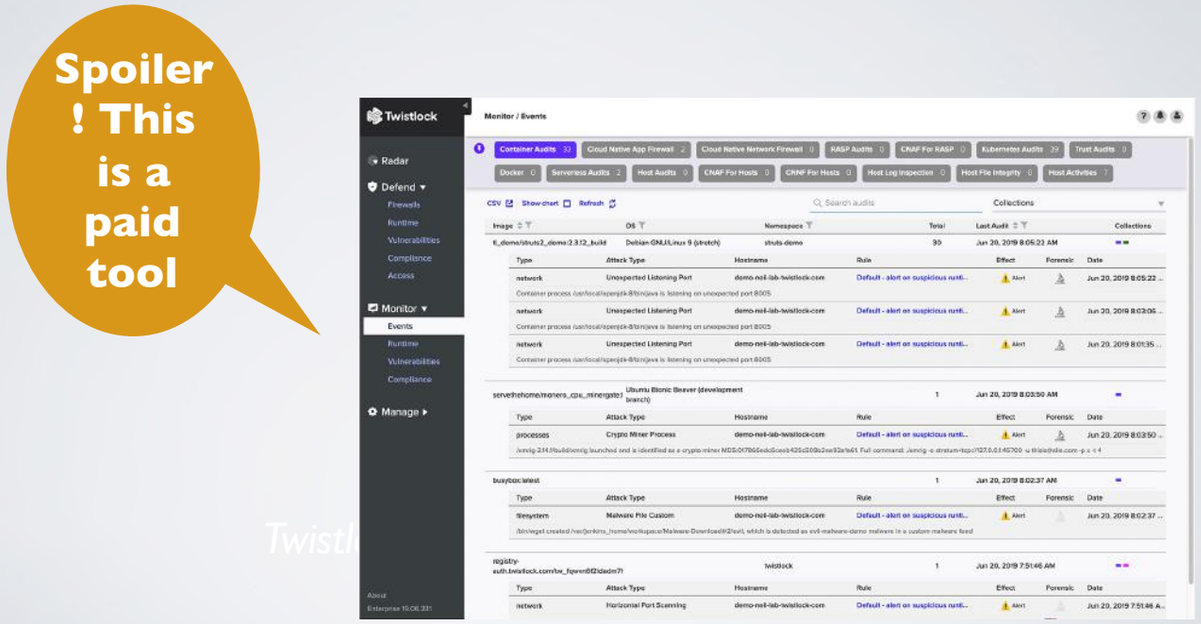


Legacy = Outside In (Scanning, Firewalls, WAF)

OPEN SOURCE RASP

Baidu	Installed to web application server

RASP REPORTING



This is a paid tool called Twistlock that specializes in containerized applications.

Screenshot shows a Twistlock container audit showing ‘unexpected listening ports’. This is actually a crypto miner process running in the application’s production environment.

1. You can configure it to shut these processes down, or just record these incidents and alert.

It looks at policy files, Docker files, and identifies anomalous behavior based on it’s understanding of the application.

Runtime protection (not runtime self protection)
Twistlock is more of a full lifecycle tool.
Vulnerability management goes into the runtime environment as well.

ASSURE

Continuous testing should happen once code is committed.

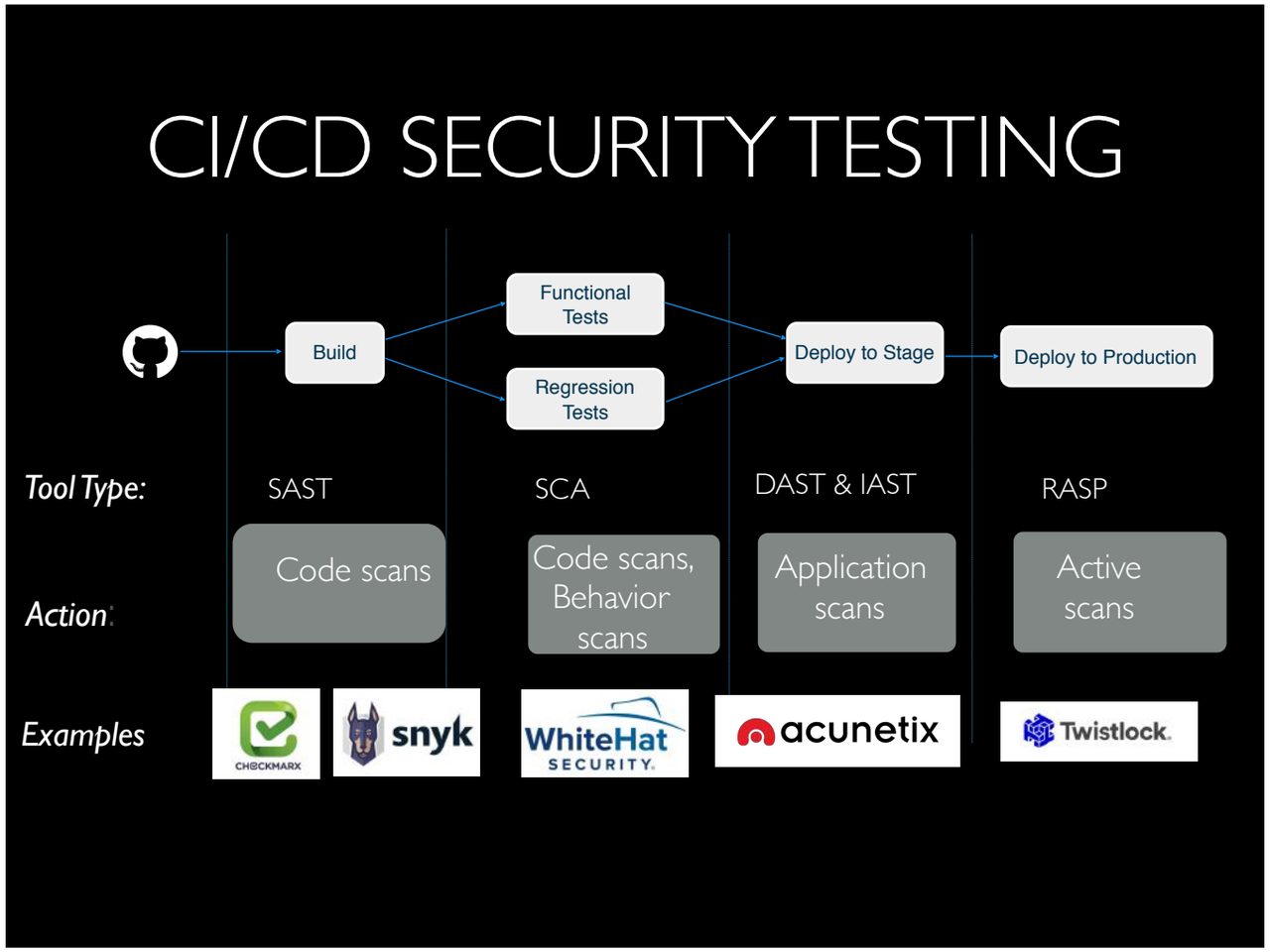
If application security has been seriously addressed in early stages of the project, the testing here should only find critical security issues.

Many of the open source tools previously mentioned can be repurposed for for automated testing of code branches

THOUGHTS ON AUTOMATED SECURITY TESTING — AKA "DEV-SEC-OPS"

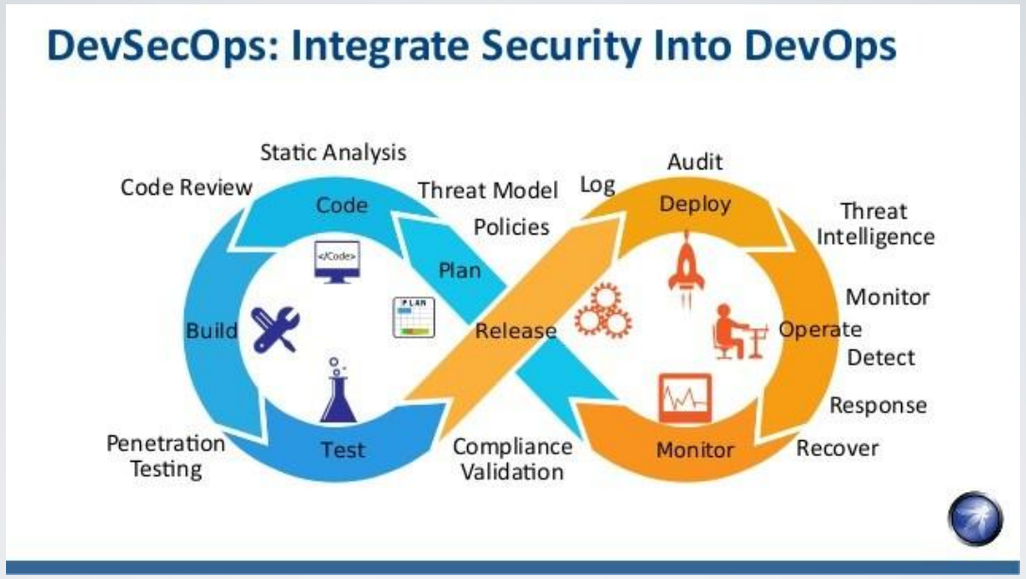
- Security tests should not slow down the build.
- Combine browser automation tools like Selenium with DAST tools like Zap Proxy to expand test coverage.
- Issues found should break the build and be documented in your issue tracker.
- Write regression tests for all issues found and fixed at this stage.

CI/CD SECURITY TESTING



AUTOMATION OPTIONS

- Use open source automation frameworks such as `reapsaw` or `test-cafe` to invoke security test tools and aggregate results
- Consider using a cloud-native build service with fully integrated build services. GitLab has a free Community edition that is open source. The paid version has fully integrated security tests, monitoring, issue tracking and test environments. Both can be hosted on premises, cloud or their managed service
- Integrate security tests into existing build systems such as GoCD, CircleCI, Travis, Jenkins, etc



LS^2

Shift Left Security enables transition into DevSECOps

Borrowed from reshift.com
https://www.softwaresecured.com/open-source-apps-in-devsecops-pipeline/

PROTECT

THOUGHTS ON POST-RELEASE SECURITY

- The project is not "over" when the software is released
- Monitoring in production needs to be adaptable to new threats and have reasonable alert triggers
- Make sure you know who is attacking
- Use real-time test tools such as RASP and IAST

PATCHING

- Check with Network Team/ Security Team to find out patching schedule(s)
- Document all dependencies (especially open source components) with version, where and how used in your application. Sometimes patches are only needed for specific usages of a component.
- Develop a testing strategy to understand the effect of patches on your application in a preProduction environment

SUMMARY

Security is more about People and Process than Tools

Advocate for addressing security concerns early and often

Never stop learning. Join OWASP to meet like-minded people and promote eduction about application security

TO LEARN MORE

- All the tools and concepts featured in this slide deck are documented in my GitHub repo —

  https://github.com/infomaven/LS2-ShiftingApplicationSecurityLeft

- If you have suggestions for improvement, please feel free to fork and make a pull request or file an Issue. :)

- Join OWASP


Thank You!