Move left in the SDLC

- Jaswinder Kaur

Jaswinder Kaur

- Senior Security Engineer at T-Mobile, Washington
- Worked previously in DC government and Bank of America
- Expertise in web application security testing, architecture risk analysis, vendor assessment and security training solutions

Agenda

- The complexity of Software and Frequency of Releases Continues to Increase
- Risk and Breaches
- Application security Risks
- Move left in the SDLC
- Embedding security into all phases of the SDLC
- What’s Next
- Q/A

The complexity of Software and Frequency of Releases Continues to Increase

- Agile teams deliver working software every few weeks
- Speed of software development is accelerating—and so are software security risks
- DevOps teams are focused on speed and pushing application changes directly to production, sometime committing hundred of builds each day
- What does security look like in a world of continuous change?

Risk and Breaches

- Verizon recently published its 2020 Data Breach Investigations Report (DBIR), which analyzed 32,002 security incidents in 16 different industries and four different world regions
- Majority of breaches – 86 percent – are financially motivated, and most – 70 percent – are caused by outsiders
- Web applications were part of more than 43% of breaches
- State of Software Security (SOSS) report found that in the retail industry, 40 percent of applications are only scanned once a year

Application security Risks

- Though developments in web applications and services have improved the way companies do business, they have also increased the risk of malicious attacks
- Companies have made an unprecedented amount of information available to an unprecedented number of people thanks to the web, and hackers are taking advantage of that
- Let’s think, do you even know what APIs your organization is using? Do you even know what data is being shared via APIs with your trusted customers, partners or vendors?

Application security Risks(cont..)

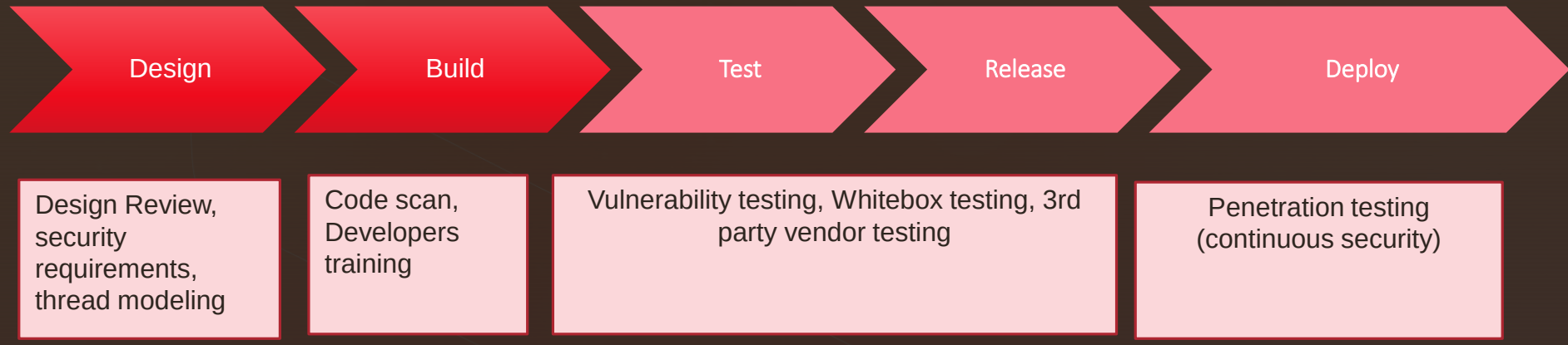
- Broken Authentication
- Sensitive Information Disclosure
- Injection
- Excessive Data Exposure
- Session management
- Authorization Bypass
- Security Misconfiguration
- Server-side request forgery
- Privilege escalation
- Verbose error messages

OWASP API Security Top 10 – 2019
A1:2019 – Broken Object Level Authorization
A2:2019 – Broken Authentication
A3:2019 – Excessive Data Exposure
A4:2019 – Lack of Resources & Rate Limiting
A5:2019 – Broken Function Level Authorization
A6:2019 – Mass Assignment
A7:2019 – Security Misconfiguration
A8:2019 – Injection
A9:2019 – Improper Assets Management
A10:2019 – Insufficient Logging & Monitoring

Move left in the SDLC

- it’s faster, easier, and cheaper to find and fix software issues early in the development process
- Moving left eliminates unnecessary work later
- Instead of sifting through and prioritizing long lists of security issues generated by penetration test post-release, development team should instead work closely with their security team to eliminate vulnerabilities earlier in the SDLC. Security team should be involved from the start of the SDLC cycle

Embedding security into all phases of the SDLC



Each step in the SDLC requires its own security enforcements and tools. Throughout all phases, automated detection, prioritization, and remediation tools can be integrated with your team's IDEs, code repositories, build servers, and bug tracking tools to address potential risks as soon as they arise

Design Phase

- Involve security team to review the design documents, perform threat modeling and write down all the security requirements that needs to be consider
- Security requirements should cover security policies (CIA) and mechanisms to enforce those security polices (AuthN/AuthZ, auditing)
- Don’t wait, Involve your security team right from the beginning
- Security engineer will review your design document and work with the application team to improve the security posture of the application

Design Phase (cont..)

Here is the high-level guide for the review:

- Is there dataflow/architectural diagram of the application?
- What's data classification applicability? (eg, Restricted, confidential, PCI)
- How is the data being protected? (eg. Encryption - storage location- access)
- Is there API involved? How are those APIs exposed? (eg. Internal, external, dedicated partner)
- For external facing, how api monitoring is being handled? (eg. API Gateway, Bot detection)
- Are vendors or 3rd-parties involved within the application, and did those vendors have a security evaluation?
- Is there a regular patch management, code scan, internal/external testing being done?
- Is there any regulatory impact? (eg. SOX, CCPA)
- What's the purpose of the application?
- Who will be the end user? (eg. Public, private/internal)
- Review the infrastructure components (eg. on prem, web server, database)
- Overall network structure, data centers and cloud environments (on prem, Azure, AWS)
- What are the security controls in place? (WAF, Firewall, SIEM, MFA)
- What are the Authentication methods? (eg. SSO, mfa)
- What are the Secrets Management Practices? (eg. credentials, keys, certificates)
- Is there logging and monitoring in place?
- How is disaster recovery and business continuity handled with the application?

Build Phase

- Development team to make sure make sure they use secure coding standards
- Pay attention to any security vulnerabilities in the code
- Essential to assess source code regularly and the best way is to integrate code scanning with CI/CD pipeline
- Many high critical issues such as hardcoded passwords can be identified in build phase through code review. Fixing such issues are lot quicker in the build phase without causing any delay in the release cycle.
- Track remediation activity for identified security flaws
- Security awareness training is managed through education and supportive process modifications, as a precursor to making security a functional requirement of the application.

Test and Release Phase

- Include security testing, using automated DevSecOps tools to improve application security
- Test web application url, APIs before deploying the code in production
- Important to remember that the DevOps approach calls for continuous testing throughout the SDLC
- Testing sooner and testing often is the best way to make sure that your products and SDLC are secure from the get-go
- Should start testing in the earliest stages of development,

Deploy phase

- Security testing doesn’t stop at the deployment and implementation
- Even after deployment and implementation, security practices need to be followed throughout software maintenance
- Products need to be continuously updated to ensure it is secure from new vulnerabilities and compatible with any new tools you may decide to adopt
- Perform regular penetration testing on production environment

What's Next

- Implementing secure SDLC helps you follow security best practices, integrating security activities and checkups across the development cycle. This will help to increase your product and company security posture
- Create security culture: work with dev teams, not against them
- Make it fun, team completions, hackathons
- Measure the practices. Eg. High risk apps, number of open security issues, time to fix security issues and top 10 security issues in your organization
- Measure, plan and improve

Q/A

@21kaurjaswinder