如何让安全不再成为DevOps的“绊脚石”

王晓飞

Problem

时间冲突

责任冲突

人力冲突

How

安全下沉 —— 从管理流程、安全责任、安全人员、安全工具、安全测试多个方面把安全因素融入开发团队、测试团队、运维团队。

安全自动化 —— 把安全检测工具嵌入到开发、运维流程中实现安全自动化，提高安全检测效率。

安全培训 —— 对人员进行细粒度分类，不同人员组织不同的培训。

安全下沉

目标：人人参与安全

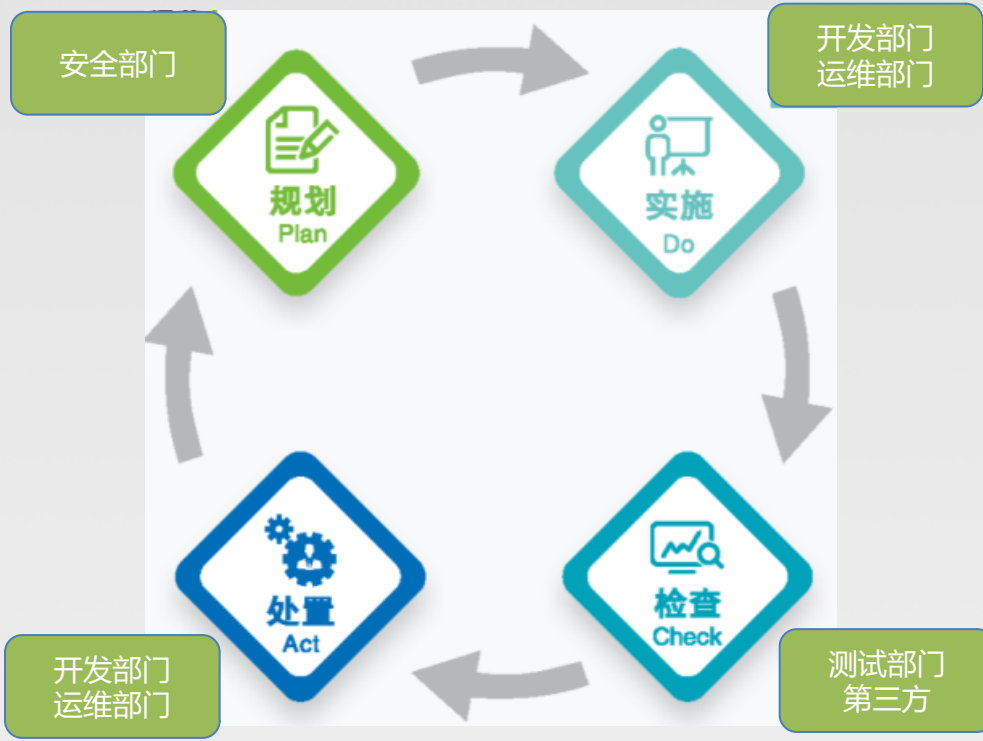
管理流程：放权，角色转变

安全责任：责任分摊，由安全团队、开发团队、测试团队、运维团队共同来承担安全风险。

安全人员：开发部门设立安全员

安全工具：SaaS化安全工具

安全测试： 赋能测试团队，由安全团队和测试团队共同完成



管理制度

责任细化

安全部门：
- 监管职责
- 安全技术更新/指导职责

开发/测试部门：
- 代码层面安全风险
- 第三方插件安全风险

运维部门：
- 操作系统层面安全风险
- 容器安全
- 网络层面安全风险

流程改进

安全部门：
- 监管，定期统计安全态势
- 平台化、自动化研发
- 新安全技术研究和培训

开发部门：
- 代码层面漏洞检测和修复
- 第三方组件漏洞检测修复

测试部门：
- 安全功能测试
- 部分渗透测试

安全人员

安全部门：
- 减少人员
- 专业化/技术化

开发部门：
- 开发经理负责制
- 安全员负责具体实施

测试部门：
- 安全功能测试
- 渗透测试

安全工具平台化


用户层
安全人员
开发人员
运维人员
测试人员
界面层
登陆功能
授权功能
工具启动功能
漏洞统计功能
漏洞详细信息
过期漏洞预警
漏洞审核功能
……
插件层
工具API调用
管理流程控制
工单自动生成
自动发邮件
工具层
SAST
DAST
MAST
Containers
第三方库
渗透测试工具
系统漏洞扫描工具
……

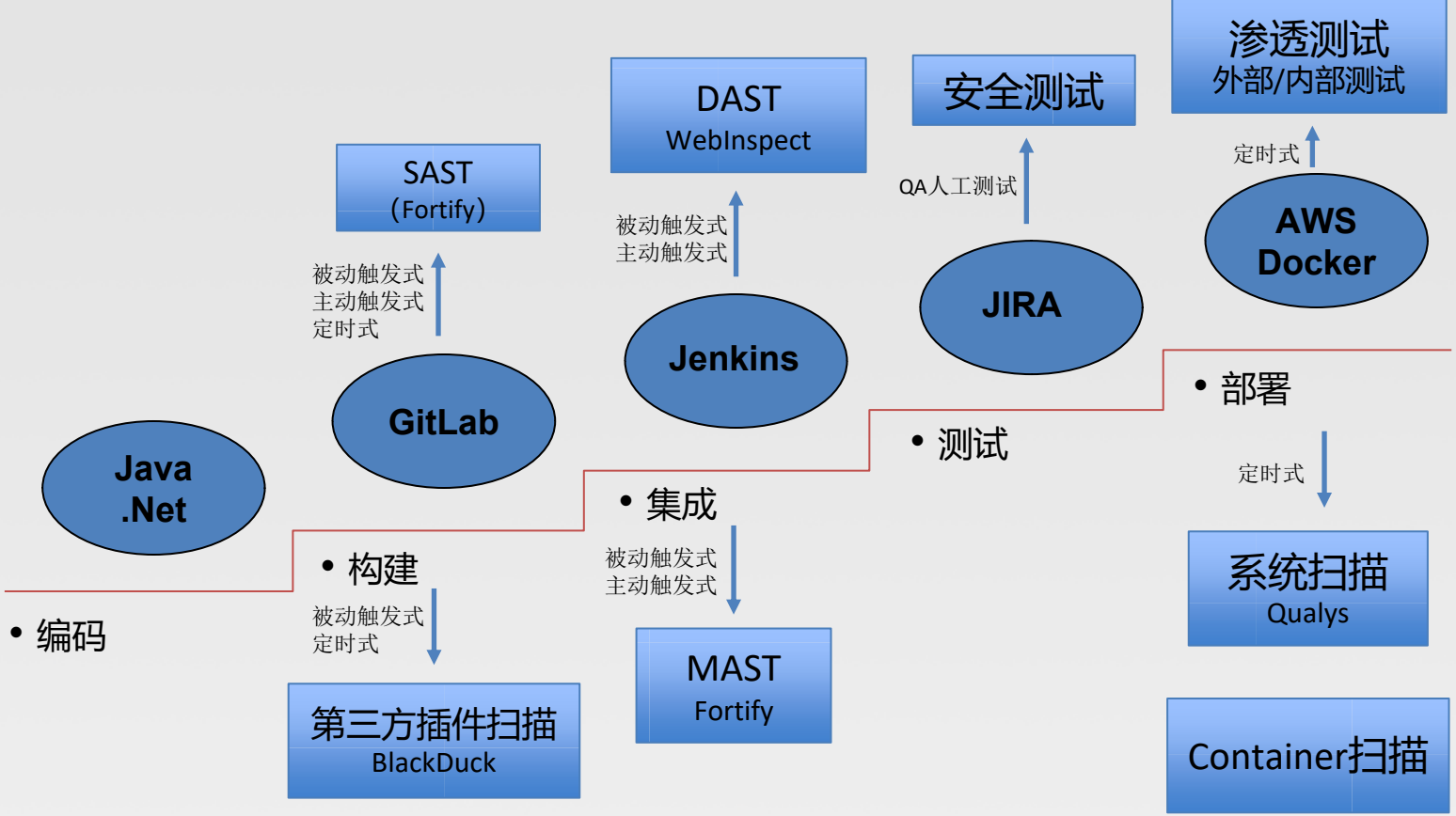
安全自动化

目标：让安全透明无感知

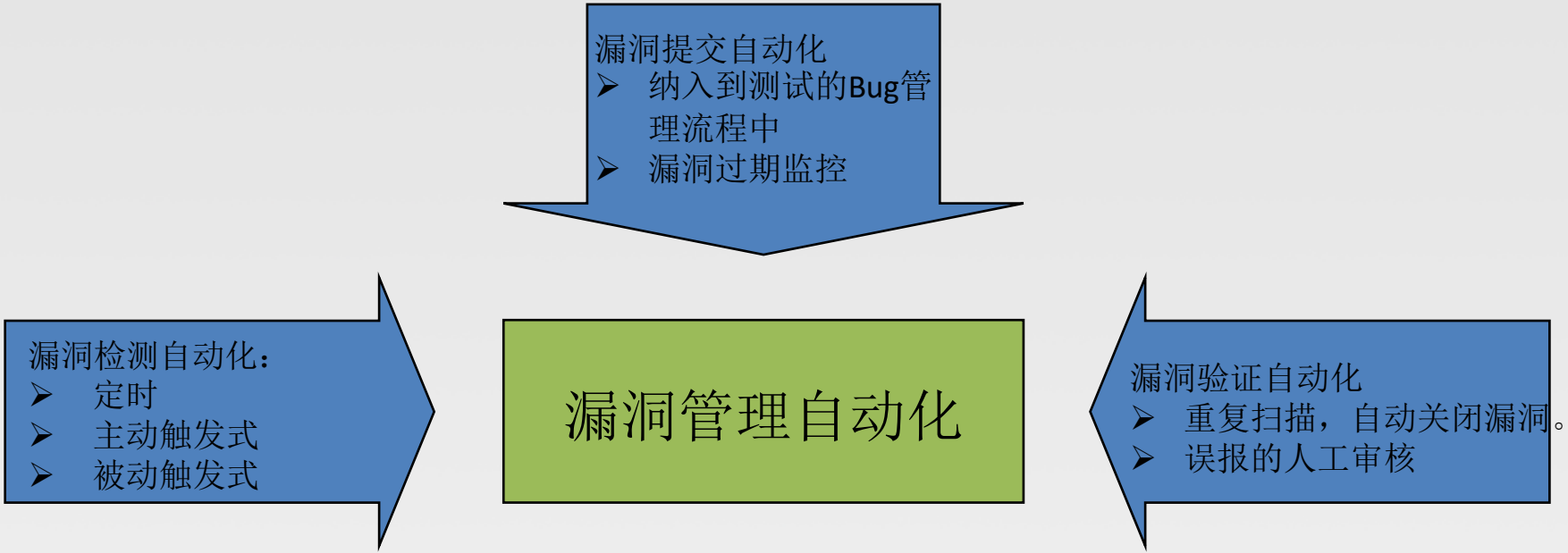
安全检测工具平台化/工具链化

安全工具引擎嵌入到项目管理流程，开发管理流程，运维管理流程中。

安全工具自动运行，自动提交漏洞，自动验证。


渗透测试
外部/内部测试
定时式
AWS
Docker
安全测试
QA人工测试
JIRA
DAST
WebInspect
被动触发式
主动触发式
Jenkins
SAST
(Fortify)
被动触发式
主动触发式
定时式
GitLab
Java
.Net
• 编码
• 构建
被动触发式
定时式
第三方插件扫描
BlackDuck
• 集成
被动触发式
主动触发式
MAST
Fortify
• 测试
• 部署
定时式
系统扫描
Qualys
Container扫描

技术栈： Python/RESTFul API/Selenium/插件化



如何降低误报？

使用安全库函数—— OWASP Enterprise Security API (ESAPI)

ESAPI (The OWASP Enterprise Security API) is a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications. The ESAPI libraries are designed to make it easier for programmers to retrofit security into existing applications.

新建/覆盖/裁剪 扫描工具的规则库——漏洞类型和规则类型

规则库匹配到具体产品/具体开发团队，形成针对特定产品的特定规则库。

安全培训

目标：培养安全文化

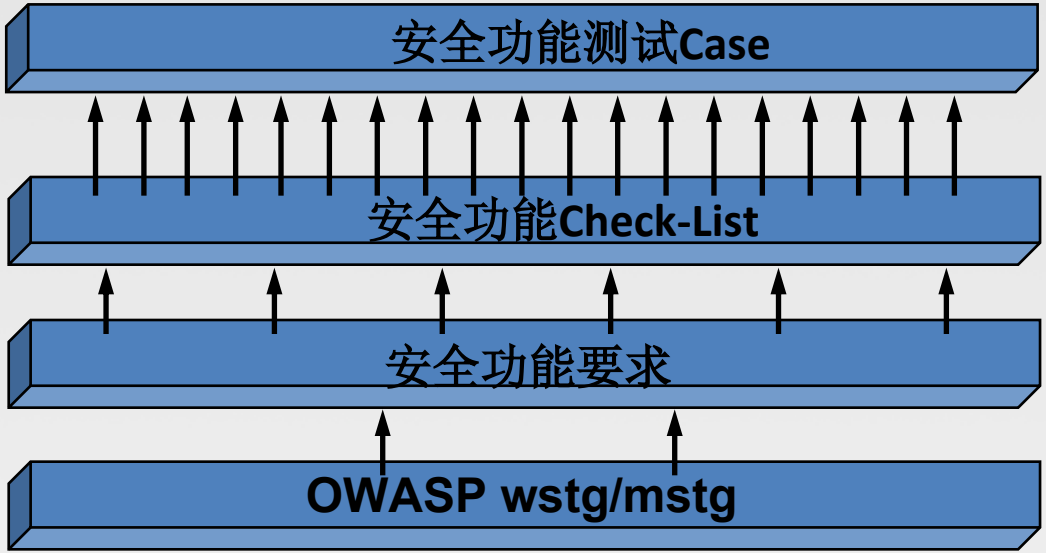
安全意识——领导，经理

安全工具使用——安全员，开发人员

安全功能测试——测试人员

渗透测试——测试人员

安全测试



OWASP Web Security Testing Guide (WSTG)

The WSTG is a comprehensive guide to testing the security of web applications and web services. Created by the collaborative efforts of security professionals and dedicated volunteers, the WSTG provides a framework of best practices used by penetration testers and organizations all over the world.

OWASP Mobile Security Testing Guide (MSTG)

The MSTG is a comprehensive manual for mobile app security testing and reverse engineering. It describes technical processes for verifying the controls listed in the OWASP Mobile Application Verification Standard (MASVS).

Mobile Application Security Requirements - Android

ID	MSTG-ID	Detailed Verification Requirement	Level 1	Level 2	Status	
V1		**Architecture, design and threat modelling**				
1.1	MSTG-ARCH-1	All app components are identified and known to be needed.	✓	✓		Architectural Information
1.2	MSTG-ARCH-2	Security controls are never enforced only on the client side, but on the respective remote endpoints.	✓	✓		Injection Flaws (MSTG-ARCH-2 a
1.3	MSTG-ARCH-3	A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture.	✓	✓		Architectural Information
1.4	MSTG-ARCH-4	Data considered sensitive in the context of the mobile app is clearly identified.	✓	✓		Identifying Sensitive Data
1.5	MSTG-ARCH-5	All app components are defined in terms of the business functions and/or security functions they provide.		✓	N/A	Environmental Information
1.6	MSTG-ARCH-6	A threat model for the mobile app and the associated remote services has been produced that identifies potential threats and countermeasures.		✓	N/A	Mapping the Application
1.7	MSTG-ARCH-7	All security controls have a centralized implementation.		✓	N/A	Testing for insecure Configuratic
1.8	MSTG-ARCH-8	There is an explicit policy for how cryptographic keys (if any) are managed, and the lifecycle of cryptographic keys is enforced. Ideally, follow a key management standard such as NIST SP 800-57.		✓	N/A	Cryptographic policy
1.9	MSTG-ARCH-9	A mechanism for enforcing updates of the mobile app exists.		✓	N/A	Testing enforced updating (MST
1.10	MSTG-ARCH-10	Security is addressed within all parts of the software development lifecycle.		✓	N/A	Security Testing and the SDLC
1.11	MSTG-ARCH-11	A responsible disclosure policy is in place and effectively applied.		✓	N/A	
1.12	MSTG-ARCH-12	The app should comply with privacy laws and regulations.	✓	✓		
V2		**Data Storage and Privacy**				
2.1	MSTG-STORAGE-1	System credential storage facilities need to be used to store sensitive data, such as PII, user credentials or cryptographic	✓	✓		Testing Local Storage for Sensitiv
2.2	MSTG-STORAGE-2	No sensitive data should be stored outside of the app container or system credential storage facilities.				Testing Local Storage for Sensitiv

Test Case ID	Test Case Objective	Pre requisite	Steps	Input Data	Expected Output	Actual Output	Status
ATHZ_01	Testing for Horizontal Bypassing Authorization Schema	Two customers' accounts	1. Config the http proxy. 2. Login the application with the two users. And keep the two different sessions active. 3. For every request, change the relevant parameters and the session identifier from token one to token two and diagnose the responses for each token.	Request that is used access resource or operate functions on resources	The response is not same/ 404/ not authorization	Not authorization	PASS
ATHZ_02	Testing for Vertical Bypassing Authorization Schema	An administrator account A normal customer account	1. Config the http proxy. 2. Establish and maintain two different sessions based on the two different roles. 3. For every request in administrator role, change the session identifier from the original to user role's session identifier and evaluate the responses for each.	Request that is only in administrator's menu.	The response is not same/ 404/ not authorization	404	PASS
ATHZ_03	Testing for Access to Resources Assigned to a Different Role	An administrator account A normal customer account	1. Config the http proxy. 2. Establish and maintain two different sessions based on the two different roles. 3. For every request in administrator role, change the session identifier from the original to user role's session identifier. 4. Change the request to PUT or DELETE, then evaluate the responses.	Request that can access files (e.g. pdf, word, xlsx and so on) in administrator role.	The response is not same/ 404/ not authorization	Not authorization	PASS
ATHZ_04	Testing for Special Request Header Handling	N/A	1. Config the http proxy. 2. Access the URL with anonymous. 3. Add X-Original-URL/ X-Rewrite-URL Header in request, then evaluate the response.	X-Original-URL X-Rewrite-URL Header	The response is same	The response is same	
ATHZ_05	Testing for Role/Privilege Manipulation	A normal customer account	1. Config the http proxy. 2. Login the application with the customer account. 3. For the requests/responses that related with privilege manipulations, change the parameters, then check if it has some administrator's functions.	Request/response that is changed ID/Profile	The role/privilege does not change.	The privilege does not change	PASS
ATHZ_06	Testing Directory Traversal File Include	A normal customer account	1. Config the http proxy. 2. Login the application with the customer account. 3. For the GET requests that access some files (e.g. xml, pdf, jpg and so all), change the URL to /../../../etc/passwd or /../../../boot.ini. Then re-send them. 4. Evaluate the response.	/../../../../etc/passwd /../../../../boot.ini	404	404	PASS
ATHZ_07	Weak SessionID	A normal customer account	1. Config the http proxy. 2. Login the application with the customer account. 3. Capture the session. Then try to decode it with Base64 or JWT exploit method. 4. If session can be decoded, change the user ID to another one. Then add the new session into request, re-send it. 5. Evaluate the response.	Changed session	404/ not authorization	Not authorization	PASS
ATHZ_08	Testing for Insecure Direct Object References	2 administrator accounts / 2 normal customer accounts	1. Config the http proxy. 2. Login the application with one account. 3. Choose one location where user input is used to reference objects directly. For example, locations where user input is used to access a database row, a file, application pages and more. Then record the parameter value that is used to test. 4. Login the application with another account. Navigate to the same location. Then change the value of parameter to below value or similar values. re-send it. 5. Evaluate the response.	Deformed value of parameter	404/ no permission	404	PASS

Test Case Objective:

Test if user who is not authenticated can access the function of creating an event.

Pre-Requisite:

Admin credentials

Time:

15min

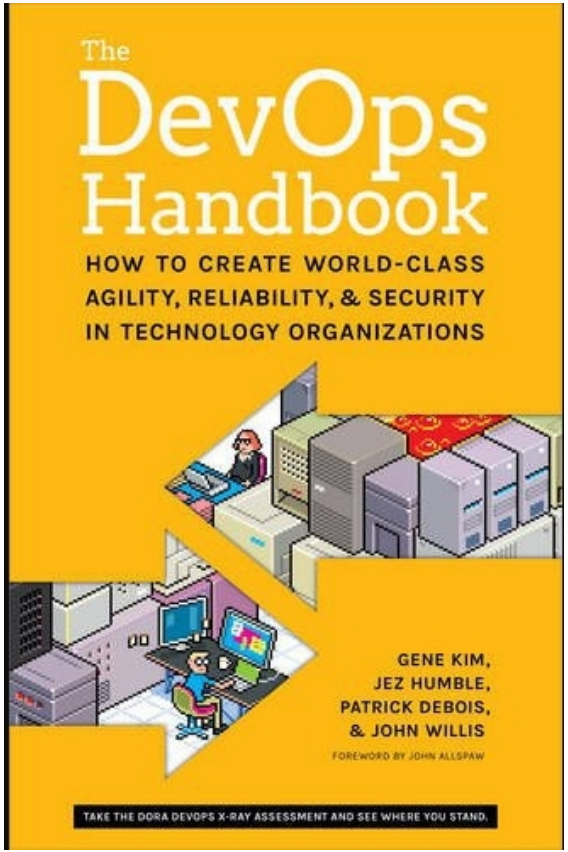
Steps:

1. Set the Http Proxy in browser.
2. Then access the application, login the system with admin's account.
3. Click the Create Event link to create a new event.
4. Capture and save all the requests for creating event.

Request 1:

```
POST /json/ems2EventManagementService/createEvent?nonhtml=true HTTP/1.1
Host: enduranceui-vip.qa.aw.dev.activenetwork.com
Connection: close
Content-Length: 314
sec-ch-ua: "Chromium";v="89", ";Not A Brand";v="99"
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/89.0.4389.90 Safari/537.36
Content-Type: application/json
Origin: https://enduranceui-vip.qa.aw.dev.activenetwork.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://enduranceui-
vip.qa.aw.dev.activenetwork.com/test012345678910111213141516171819202122232425262 7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie:
BIGipServer~activeworks~aw_qa_enduranceui_pool=!Av76/P7dqFWaZIuUGGy2Z8gvqusb6Nda
KN6HYRXIFAeOVrBkXHKOY5Veov6xm36uYFuAg+zZLQ6oCA==;
OPTOUTMULTI=0:0%7Cc1:0%7Cc3:0; s_cc=true; s_fid=592DA36B72ECD6A5-
1E360AA3B3437439;
```

“DevOps Handbook”



Thanks !