



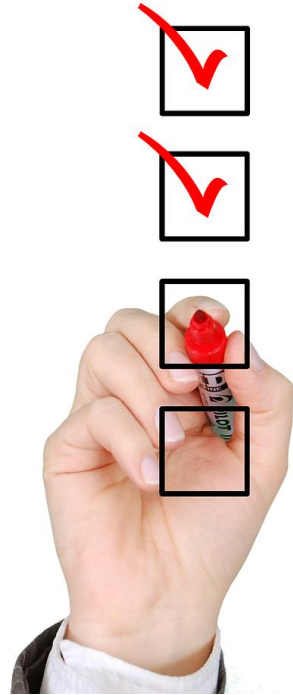
# Maturing Your SDLC: Ch 2. Static & Dynamic Testing

By: William Kiley  
*Managing Consultant - DevSecOps*

# Dynamic vs Static

- What is the difference?

# Know the requirements



# Know the vulnerabilities

- Open Web Application Security Project (OWASP)



T10 OWASP Top 10 Application Security Risks – 2017	
<b>A1:2017-Injection</b>	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
<b>A2:2017-Broken Authentication</b>	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
<b>A3:2017-Sensitive Data Exposure</b>	Many web applications and APIs do not properly protect sensitive data, such as financial healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
<b>A4:2017-XML External Entities (XXE)</b>	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
<b>A5:2017-Broken Access Control</b>	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
<b>A6:2017-Security Misconfiguration</b>	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.
<b>A7:2017-Cross-Site Scripting (XSS)</b>	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML, or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
<b>A8:2017-Insecure Deserialization</b>	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
<b>A9:2017-Using Components with Known Vulnerabilities</b>	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
<b>A10:2017-Insufficient Logging &amp; Monitoring</b>	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

# Automated vs Manual

- In short... both.
- Automated testing offers
  - Definition to the review process
  - Expertise lending to non-experts
  - Speed

# Know the fixes

- Align team knowledge with best practices



## OWASP Top 10 - 2017

The Ten Most Critical Web Application Security Risks

A2  
:2017

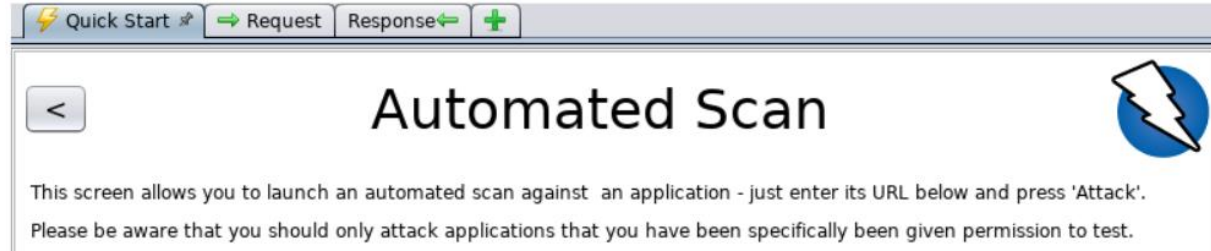
### Broken Authentication

#### How to Prevent

- Where possible, implement multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks.
- Do not ship or deploy with any default credentials, particularly for admin users.
- Implement weak-password checks, such as testing new or changed passwords against a list of the [top 10000 worst passwords](#).
- Align password length, complexity and rotation policies with [NIST 800-63 B's guidelines in section 5.1.1 for Memorized Secrets](#) or other modern, evidence based password policies.
- Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks by using the same messages for all outcomes.
- Limit or increasingly delay failed login attempts. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.
- Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login. Session IDs should not be in the URL, be securely stored and invalidated after logout, idle, and absolute timeouts.

# Dynamic Tools

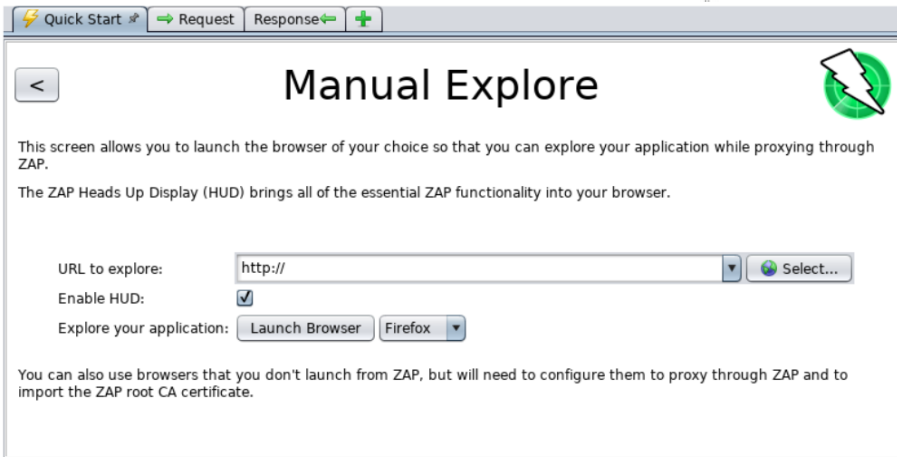
- Open Source 



Quick Start \* Request Response +

## Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'. Please be aware that you should only attack applications that you have been specifically given permission to test.



Quick Start \* Request Response +

## Manual Explore

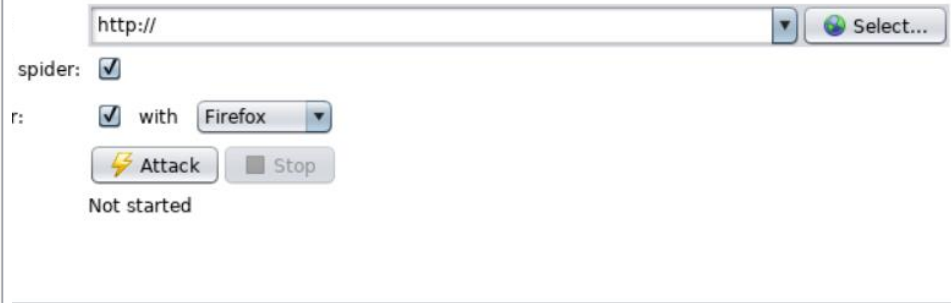
This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP. The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.

URL to explore:

Enable HUD:

Explore your application:

You can also use browsers that you don't launch from ZAP, but will need to configure them to proxy through ZAP and to import the ZAP root CA certificate.



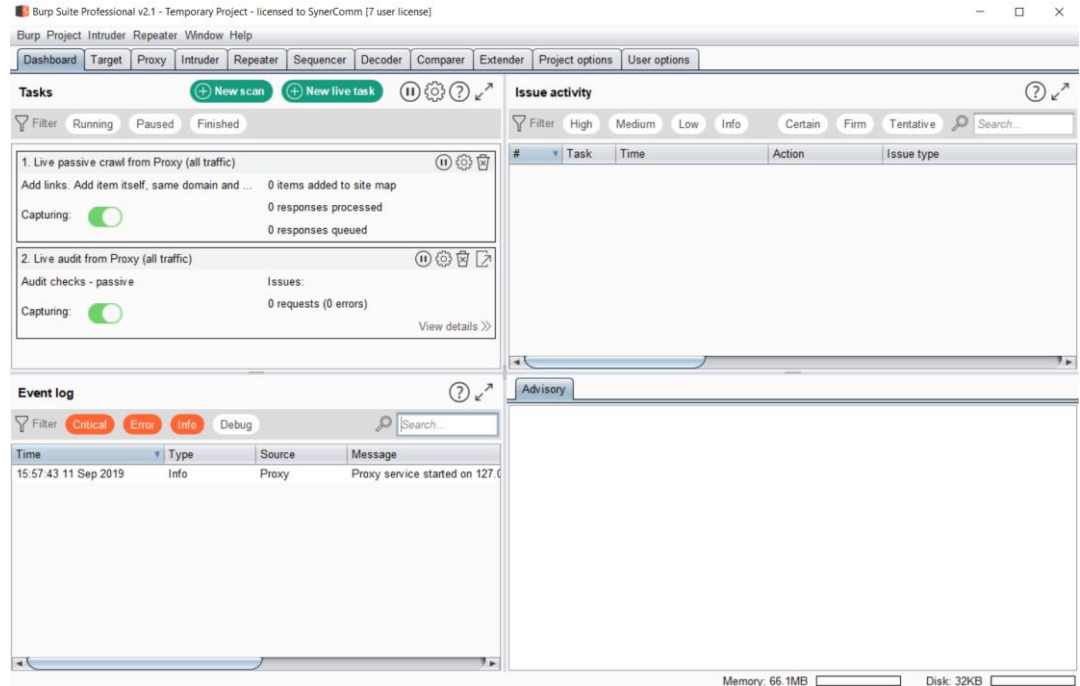
spider:

r:  with

Not started

# Dynamic Tools

- Professional:





# Static Tools

- Open Source  
**sonarqube**

The screenshot displays a code analysis interface for a Java file. The code snippet is as follows:

```
246 if (Provider.class == roleTypeClass) {
247     Type providedType = ReflectionUtils.getLastTypeGenericArgument(dependen
248     2 Class providedClass = 1 ReflectionUtils.getTypeClass(providedType);
249
250     if (this.componentManager.hasComponent(providedType, dependencyDescript
251         || 3 providedClass.isAssignableFrom(List.class) || providedClass.
```

A red tooltip highlights a bug: "A 'NullPointerException' could be thrown; 'providedClass' is nullable here." The bug is categorized as "Major" and is associated with the component "cert.cwe".

The dashboard summary on the right shows the following metrics:

			New code Since last release	
<b>Reliability</b>				
🚩 Bugs	2	B	1	B
<b>Security</b>				
🔒 Security Vulnerabilities	0	A	0	A
🔒 Security Hotspots	39	-	0	-
<b>Maintainability</b>				
🕒 Technical Debt	6 days	C	0	A
🐛 Code Smells	319	-	0	-

# Static Tools

- Professional



APPLICATIONS DASHBOARD REPORTS ADMINISTRATION

Your Applications

Managed 62 Discovered 830 Ignored 206

62 found

NAME	PRODUCTION RISK & POLICY COMPLIANCE					SCAN & SECURITY STATUS					MOST RECENT CHANGE
	CRITICAL	HIGH	MEDIUM	LOW		STATIC	DYNAMIC	NETWORK	MONITORING	APP DEF	
<b>PGAdmin</b> 1 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	370	201	2	20	✓	⊖	⊖	⊖	⊖	08/17/2017 New Monitoring Vulnerabilities Detected
<b>Advantage Online Banking</b> 2 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	88	17	5	55	✓	⊖	✓	⊖	✓	10/01/2017 New Static Vulnerabilities Detected
<b>Demo App for Deletion</b> 5 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	86	89	21	81	✓	✓	✓	⊖	✓	10/23/2017 New Static Vulnerabilities Detected
<b>everywhere.com</b> 4 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	22	75	441	14	⊖	⊖	✓	⊖	⊖	08/11/2017 Release Failing Security Policy
<b>Cobol Sample</b> 2 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	50	4	0	0	✓	✓	⊖	✓	✓	04/28/2017 New Dynamic Vulnerabilities Detected
<b>Zero</b> 2 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	3	7	3	16	⊖	⊖	⊖	⊖	✓	08/24/2017 New Monitoring Vulnerabilities Detected
<b>Custom Banking App</b> 1 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	1	0	28	2	✓	✓	⊖	⊖	✓	01/18/2017 New Mobile Vulnerabilities Detected
<b>emea.hpfd.com</b> 1 RELEASES Business Criticality: HIGH	FAIL ★☆☆☆☆	0	1	2	7	⊖	⊖	⊖	⊖	⊖	06/23/2017 New Monitoring Vulnerabilities Detected
<b>Drupal</b> 1 RELEASES Business Criticality: HIGH	FAIL ★★★★★	0	0	0	6	✓	⊖	⊖	⊖	⊖	08/11/2017 Release Failing Security Policy
<b>Microservices App</b> 6 RELEASES Business Criticality: MEDIUM	FAIL ★☆☆☆☆	187	49	99	1624	✓	⊖	⊖	⊖	⊖	08/15/2017 Release Failing Security Policy

Display: 25 50 100

expand all | collapse all

SEARCH Text

+ NEW APPLICATION

▼ SORT

- Production Risk
- AGENCY
  - (Not Set) 4
  - Acme 13
  - Coders Inc 16
  - DevStrez 6
  - Internal Team 18
  - Pinnacle 5
- APPLICATION DEFENDER
- APPLICATION MONITORING
- APPLICATION TYPE
- BUSINESS CRITICALITY
- BUSINESS UNIT
- COMPLIANCE REQUIREMENT
- DYNAMIC SCAN STATUS
- MOBILE SCAN STATUS
- MOST RECENT CHANGE
- PASS/FAIL
- REGION
- SCAN TYPE
- STAR RATING
- STATIC SCAN STATUS
- SVP

# Questions

- Thank you for attending!