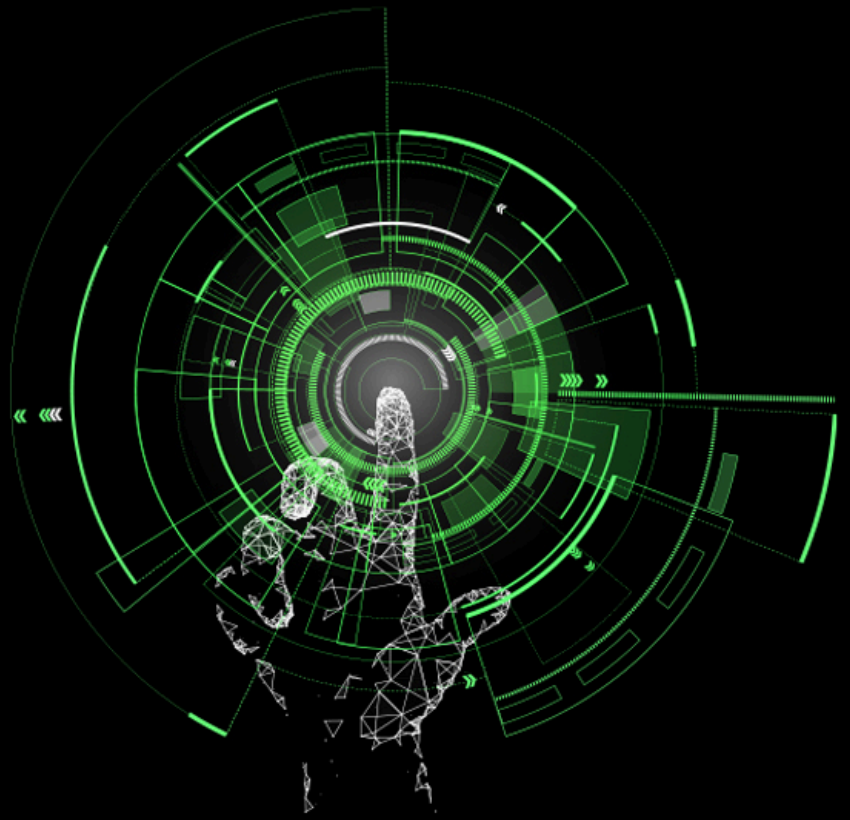
Deloitte.

勤業眾信



將 DAST 整合進入 CI/CD 的正確姿勢

勤業眾信聯合會計師事務所 高于凱

whoami

You can find me at hackercat.org
Facebook @hackercat1215

高于凱 Kai Kao

Deloitte Senior Manager

DevSecOps 社群顧問

Founder of HackerCat

Co-Founder of NOP Lab

Member of UCCU Hacker

出版書籍：

《WebSecurity 網站滲透測試 : Burp Suite 完全學習指南》

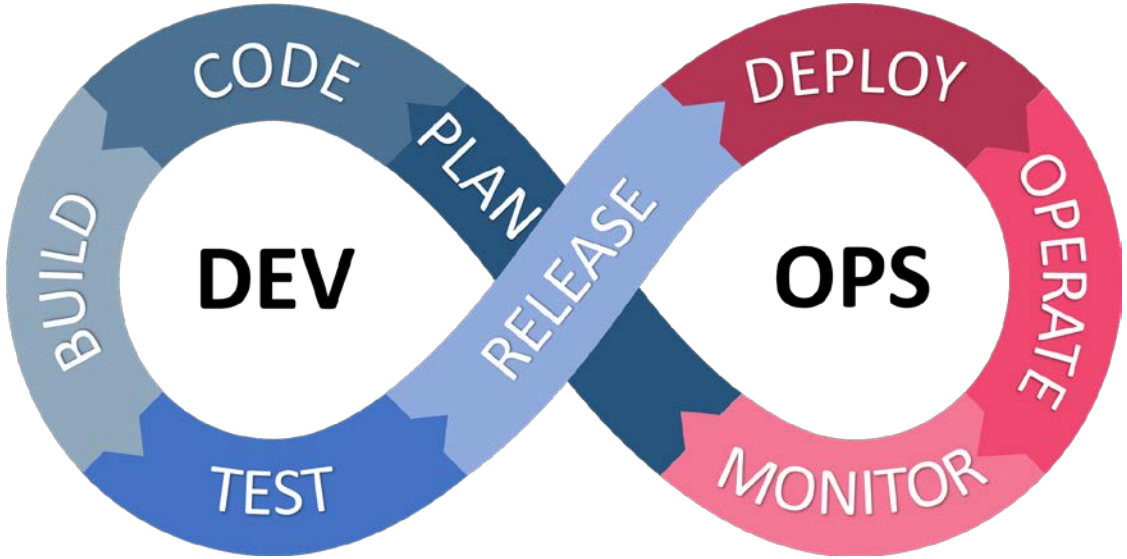
《CI / CD 安全防護大揭密：DevSecOps 最佳實踐指南》

《一個人的藍隊：企業資安防護技術實戰指南》

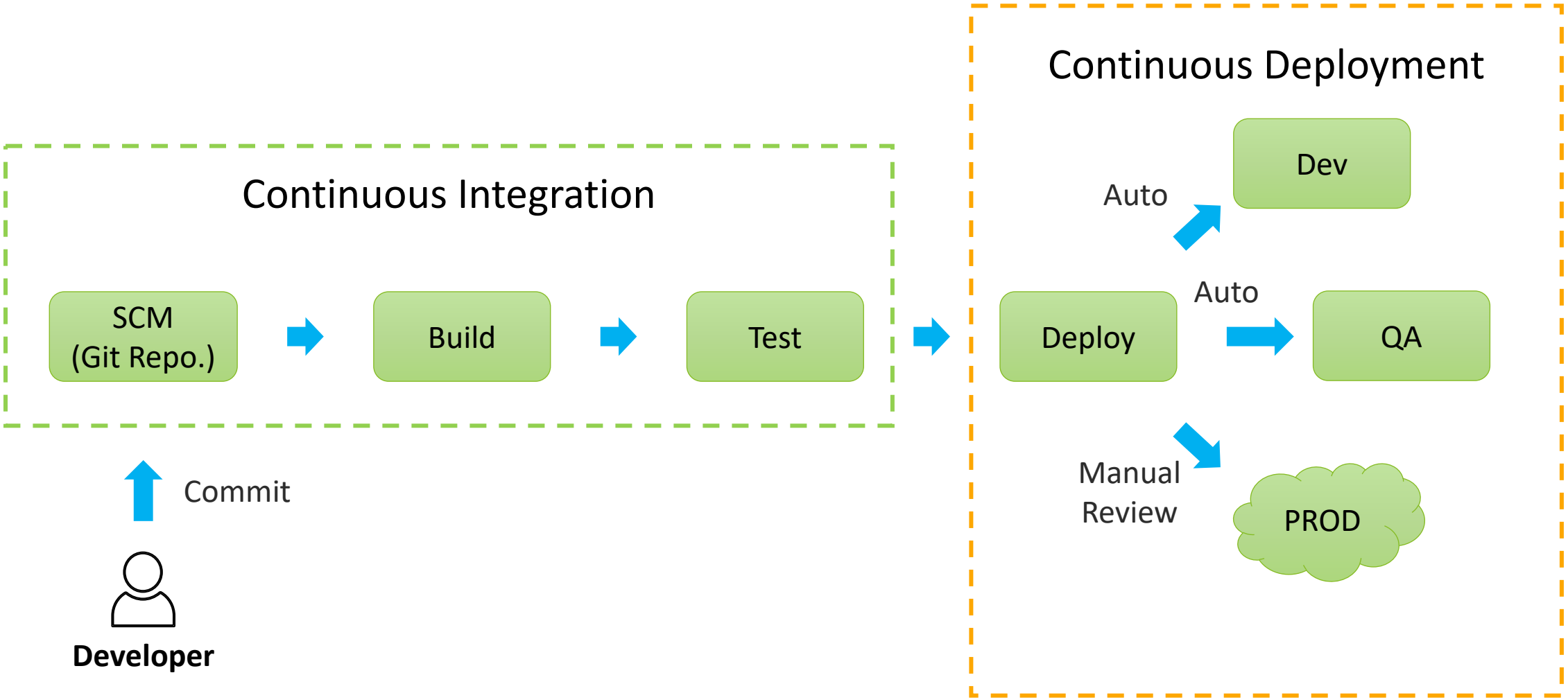
DevSecOps and CI/CD pipeline

DevSecOps 是 DevOps 延伸，核心理念是「安全左移」（Shift Left），概念是將安全融入軟體開發生命週期的每個階段。傳統上，資安通常是在開發完成之後才進行檢查與修補，然而這樣的方式不僅風險高，修復成本也大。在開發初期就導入安全測試與風險評估，讓開發、維運與資安團隊能夠協同作業，共同打造更安全的產品。

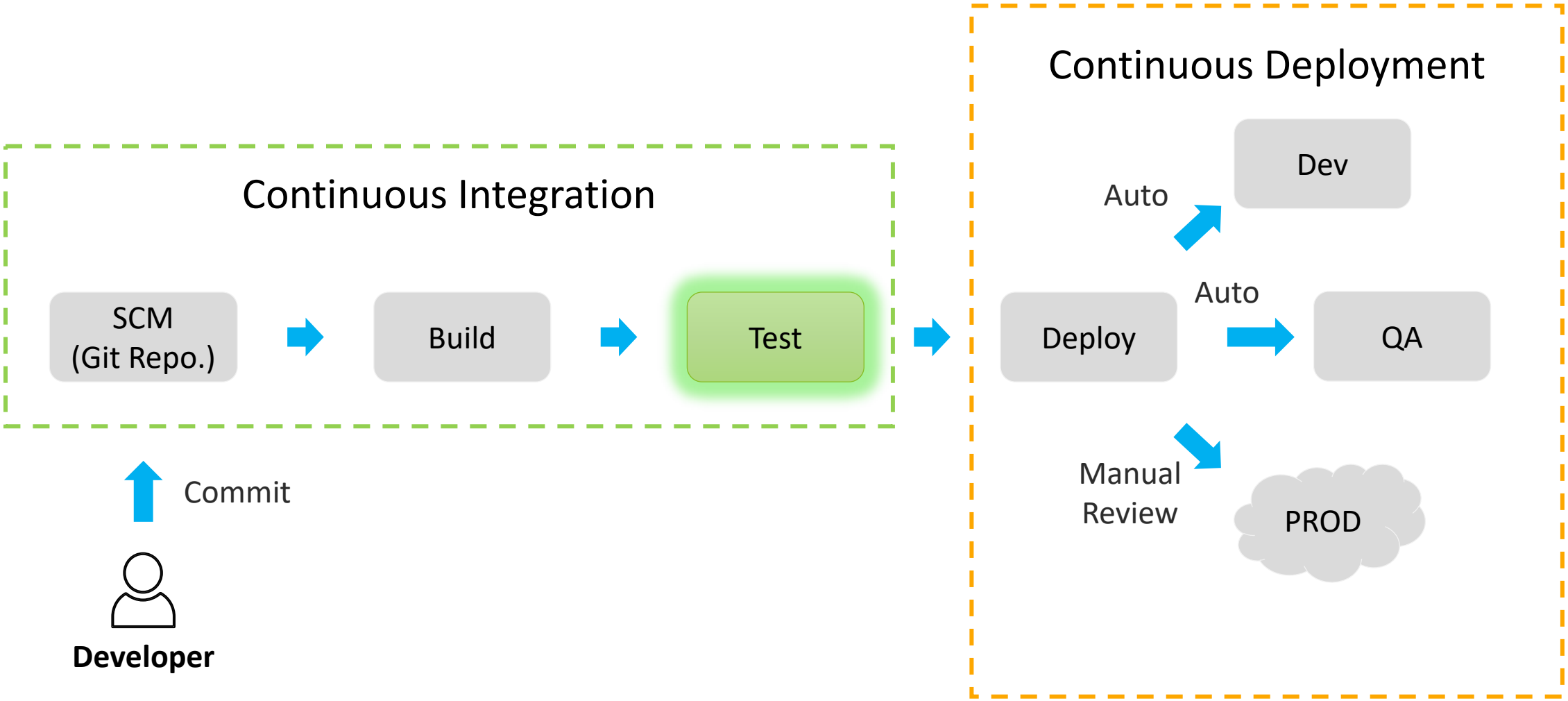
CI/CD Pipeline 則是現代軟體開發流程的自動化核心，透過持續整合（CI）與持續部署（CD），自動化地將程式碼從提交、建置、測試到部署至生產環境。達成縮短交付週期、降低人為錯誤，並確保每次變更都能被有效驗證的目的。



CI/CD Pipeline



CI/CD Pipeline



Functional vs. Non-functional Testing

功能測試 (Functional Testing)	非功能測試 (Non-Functional Testing)
單元測試 (Unit Testing)	性能測試 (Performance Testing)
整合測試 (Integration Testing)	負載測試 (Load Testing)
端對端測試 (End-to-End Testing)	壓力測試 (Stress Testing)
使用者驗收測試 (UAT)	可用性測試 (Usability Testing)
迴歸測試 (Regression Testing)	安全性測試 (Security Testing)
煙霧測試(Smoke testing)	可擴展性測試 (Scalability Testing)

Functional vs. Non-functional Testing

功能測試 (Functional Testing)	非功能測試 (Non-Functional Testing)
單元測試 (Unit Testing)	性能測試 (Performance Testing)
整合測試 (Integration Testing)	負載測試 (Load Testing)
端對端測試 (End-to-End Testing)	壓力測試 (Stress Testing)
使用者驗收測試 (UAT)	可用性測試 (Usability Testing)
迴歸測試 (Regression Testing)	安全性測試 (Security Testing)
煙霧測試(Smoke testing)	可擴展性測試 (Scalability Testing)

Application Security Testing

Penetration test

Expert-led simulated attacks to find exploitable weaknesses.

SAST

Analyzes source code without execution to find security flaws.

Vulnerability Scanning

Automated checks for known security issues and weaknesses.

IAST

Monitors code execution in real-time to detect vulnerabilities.

DAST

Tests running applications by simulating attacks from the outside.

SCA

Identifies vulnerabilities in third-party components and dependencies.

Application Security Testing in CI/CD Pipelines

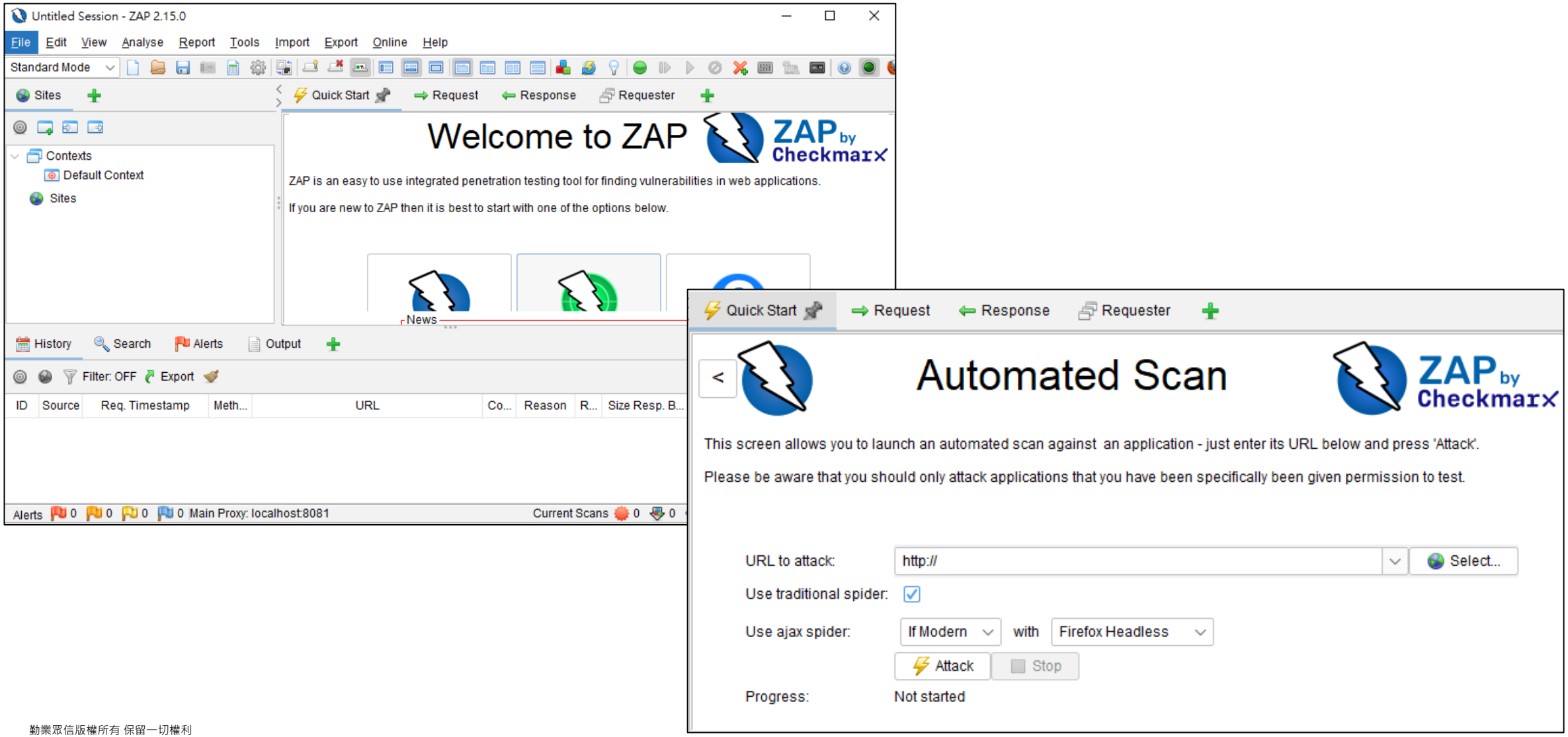
	SAST	DAST	IAST	SCA
Coverage	🔴	🟠	🔴	🔴
Exploitability	🟢	🔴	🔴	🟢
False positive	🔴	🟠	🟢	🟢
Code visibility	🔴	🟢	🔴	🔴
Broad Platform Support	🟠	🔴	🟢	🟢
Integration Difficulty	🔴	🟠	🟢	🟠
Inspects	Source Code	Running Application	Running Application	Dependency

🟢 Low 🟠 Medium 🔴 High

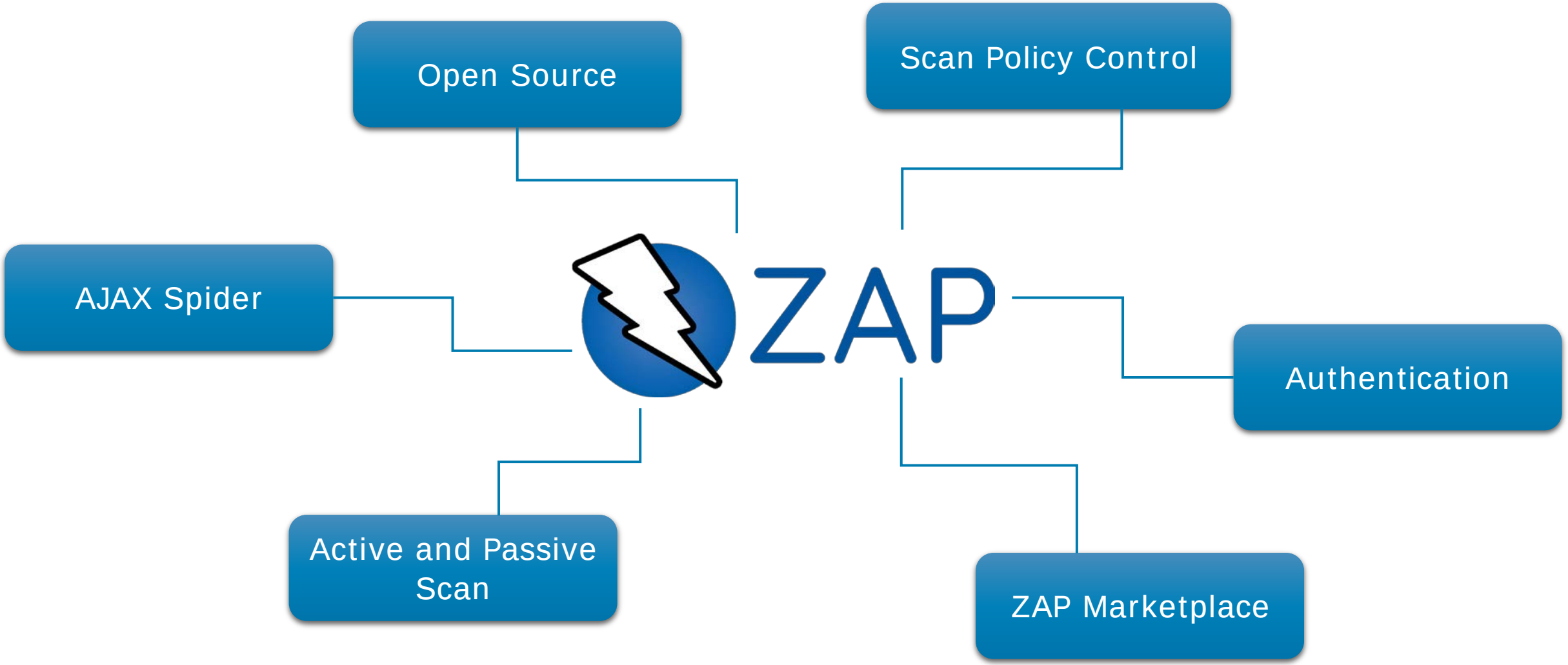
DAST優缺點

Pros	Cons
✅ 能夠檢測運行時漏洞，捕捉靜態分析難以發現的問題	❌ 只能檢測到暴露在應用表面的漏洞，難以發現深層邏輯問題
✅ 不需要原始碼可見性即可進行測試，亦適用於第三方應用或封閉系統	❌ 掃描過程可能較慢，影響CI/CD流程效率
✅ 可模擬真實攻擊者視角，提供漏洞實際可利用性證明	❌ 對特定API或自定義的協定支持有限，需額外配置
✅ 相對低誤報率，識別出的漏洞(identified vulnerabilities)更可靠	❌ 在測試過程中可能意外修改生產數據需謹慎設計測試環境

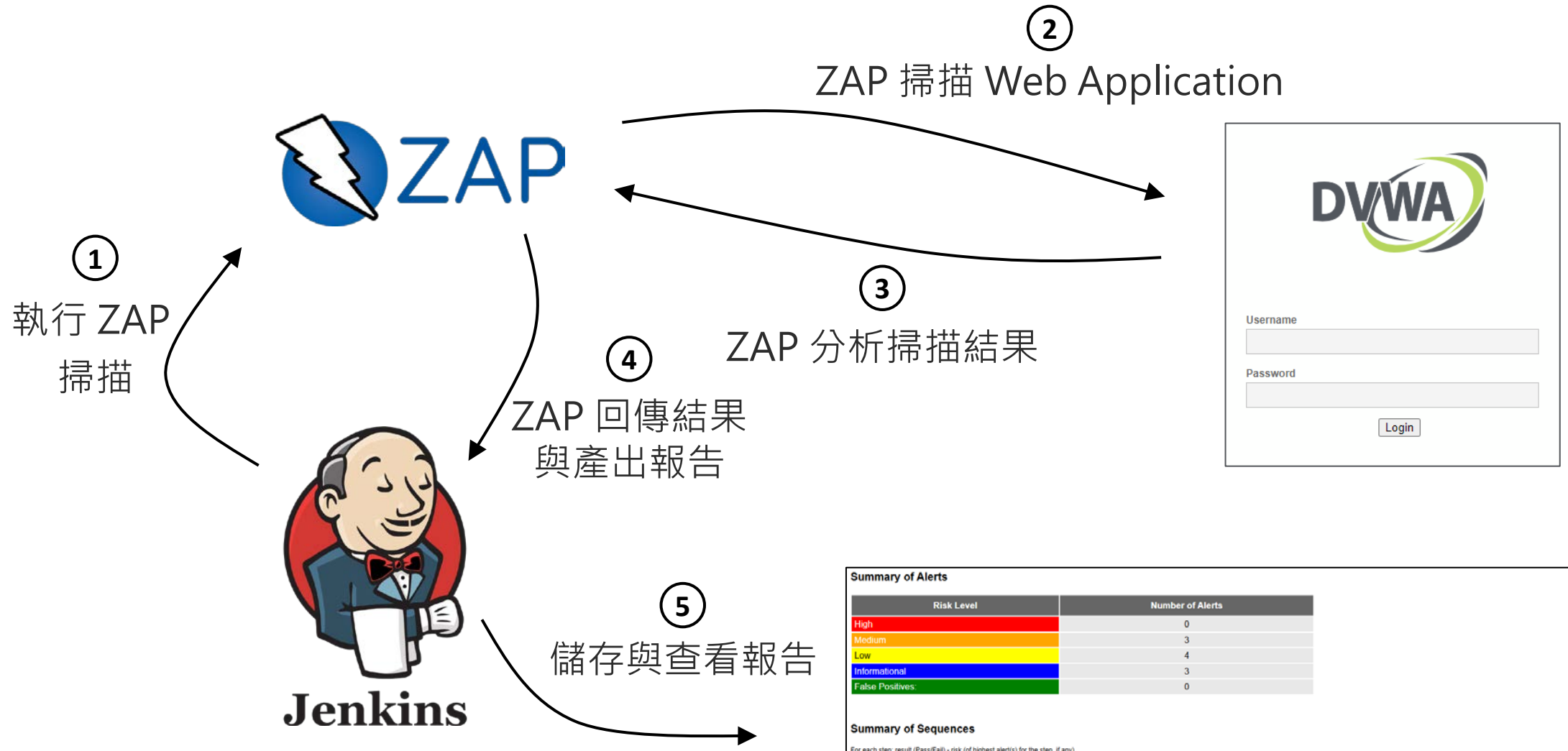
OWASP ZAP



OWASP ZAP



OWASP ZAP with Jenkins



Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	4
Informational	3
False Positives:	0

Summary of Sequences

For each step: result (Pass/Fail) - risk (of highest alert(s) for the step, if any).

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	16
Content Security Policy (CSP) Header Not Set	Medium	12
Missing Anti-clickjacking Header	Medium	12
Cookie No HttpOnly Flag	Low	2
Cookie without SameSite Attribute	Low	2
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	11

OWASP ZAP運行結果

檢測類型	花費時間	發現結果
Baseline Scan	1.5mins	0 High 3 Medium 4 Low
Full Scan	5mins	0 High 4 Medium 4 Low
Full Scan ???	80mins	5 High 7 Medium 8 Low
Full Scan ??? ???	12mins	4 High 3 Medium 5 Low

OWASP ZAP運行結果

檢測類型	花費時間	發現結果
Baseline Scan	1.5mins	0 High / 3 Medium / 4 Low
Full Scan	5mins	0 High / 4 Medium / 4 Low
Full Scan ???	80mins	5 High / 7 Medium / 8 Low
Full Scan ??? ???	12mins	4 High / 3 Medium / 5 Low

What is baselines scan?

```
stage('Integration - OWASP ZAP Scan') {
    steps {
        script {
            catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {
                sh '''
                docker run --user root -w /zap -v /data/jenkins/workspace/webgoat-owasp-zap:/zap/wrk:rw --rm \
                zaproxy/zap-stable:2.16.0 zap-baseline.py \
                -t http://192.168.85.130:81/login.php \
                -J zap-output.json \
                -r zap-report.html
                '''
            }
        }
    }
```

What is baselines scan?

```
stage('Integration - OWASP ZAP Scan') {
    steps {
        script {
            catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {
                sh '''
                docker run --user root -w /zap -v /data/jenkins/workspace/webgoat-owasp-zap:/zap/wrk:rw --rm \
                zaproxy/zap-stable:2.16.0 zap-baseline.py \
                -t http://192.168.85.130:81/login.php \
                -J zap-output.json \
                -r zap-report.html
                '''
            }
        }
    }
```

ZAP - Baseline Scan ZAP - 基線掃描

The ZAP Baseline scan is a script that is available in the ZAP Docker images.

ZAP 基線掃描是 ZAP Docker 映像中提供的腳本。

It runs the ZAP spider against the specified target for (by default) 1 minute and then waits for the passive scanning to complete before reporting the results.

它針對指定目標運行 ZAP 爬蟲（預設情況下）1 分鐘，然後等待被動掃描完成，然後再報告結果。

This means that the script doesn't perform any actual 'attacks' and will run for a relatively short period of time (a few minutes at most).

這意味著該腳本不會執行任何實際的「攻擊」，並且將運行相對較短的時間（最多幾分鐘）。

Full Scan with active scan (full-scan.py)

```
stage('Integration - OWASP ZAP Scan') {
    steps {
        script {
            catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {
                sh '''
                docker run --user root -w /zap -v /data/jenkins/workspace/owasp-zap-dast:/zap/wrk:rw --rm \
                zaproxy/zap-stable:2.16.0 zap-full-scan.py \
                -t http://192.168.85.130:81 \
                -J zap-output.json \
                -r zap-report.html
                '''
            }
        }
    }
```

Full Scan with active scan (full-scan.py)

```
stage('Integration - OWASP ZAP Scan') {
    steps {
        script {
            catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {
                sh '''
                docker run --user root -w /zap -v /data/jenkins/workspace/owasp-zap-dast:/zap/wrk:rw --rm \
                zaproxy/zap-stable:2.16.0 zap-full-scan.py \
                -t http://192.168.85.130:81 \
                -J zap-output.json \
                -r zap-report.html
                '''
            }
        }
    }
```

ZAP - Full Scan ZAP - 完全掃描

The ZAP full scan is a script that is available in the ZAP Docker images.

ZAP 完全掃描是 ZAP Docker 映射中提供的腳本。

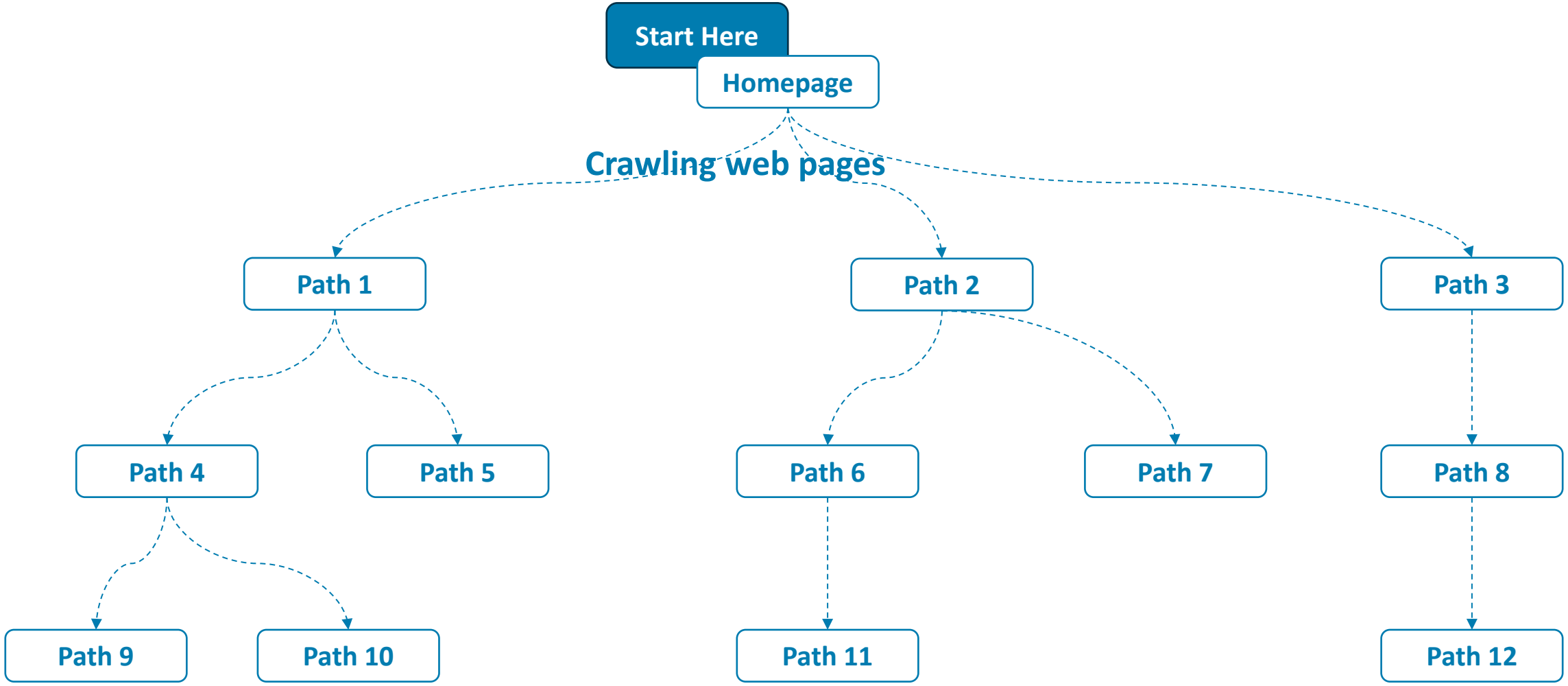
It runs the ZAP spider against the specified target (by default with no time limit) followed by an optional ajax spider scan and then a full active scan before reporting the results.

它針對指定目標運行 ZAP 爬蟲（預設情況下沒有時間限制），然後是可選的 ajax 爬蟲掃描，然後在報告結果之前進行完全主動掃描。

This means that the script does perform actual 'attacks' and can potentially run for a long period of time.

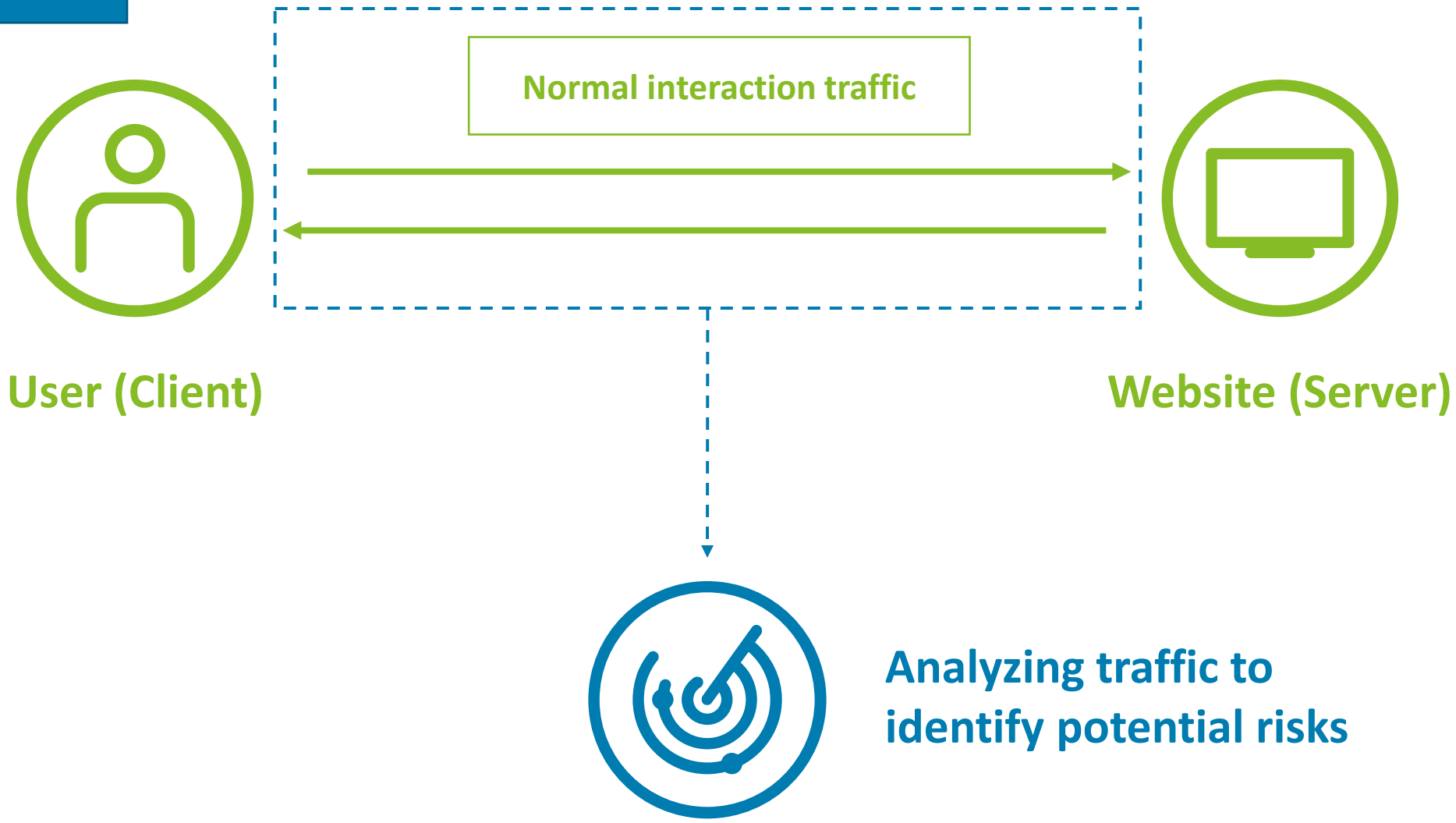
這意味著該腳本確實會執行實際的「攻擊」，並且可能會運行很長時間。

Web Crawler / Spider



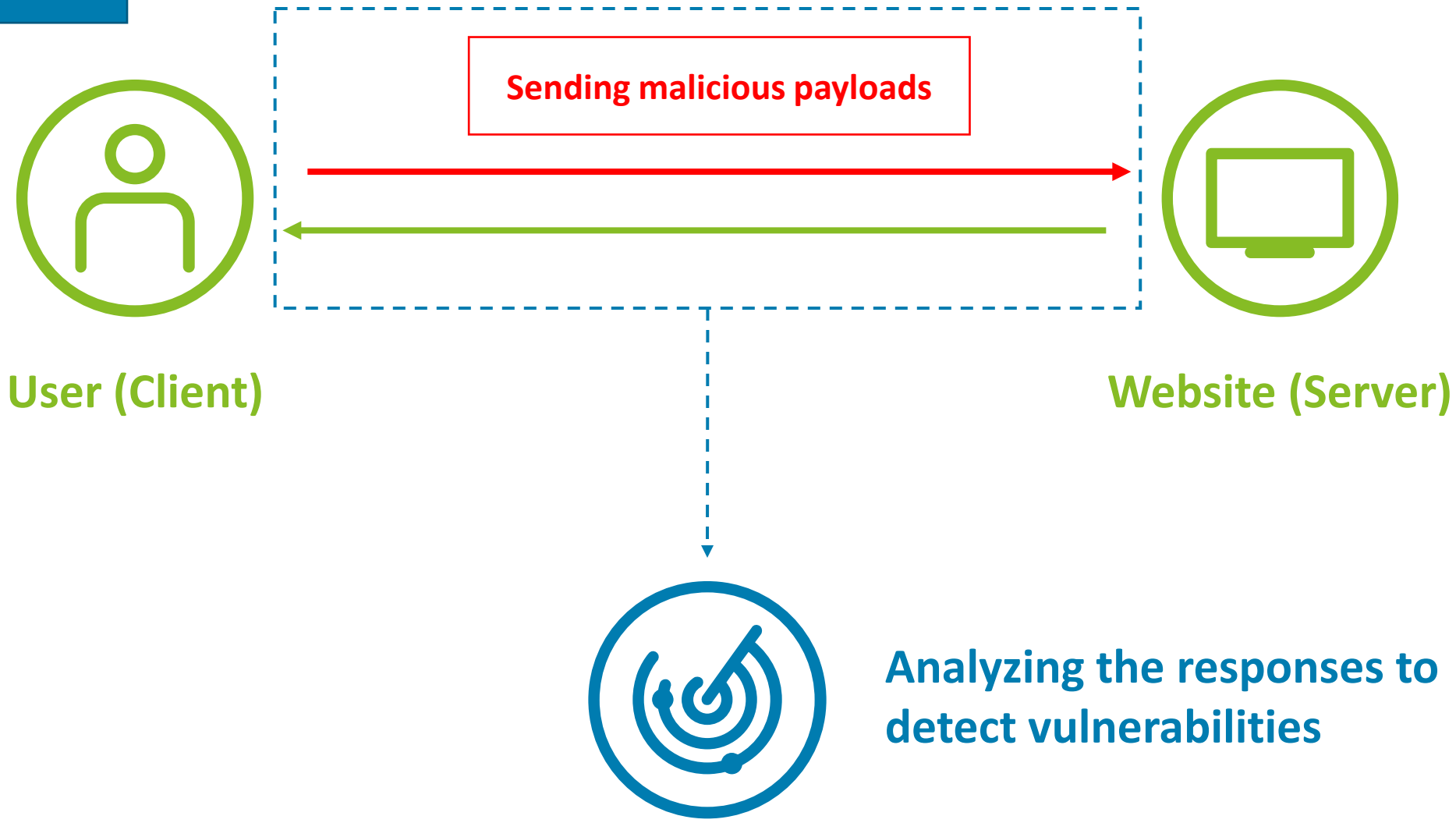
Passive Scan vs Active Scan

Passive Scan

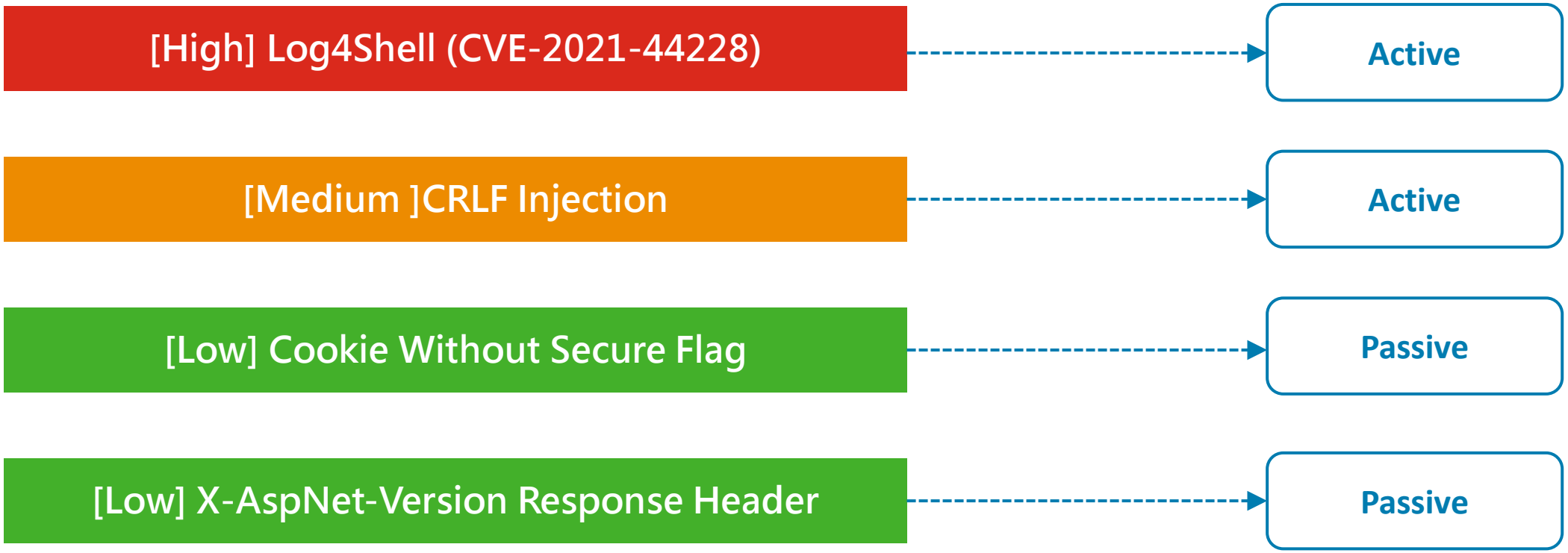


Passive Scan vs Active Scan

Active Scan



Passive Scan vs Active Scan



Passive Scan vs Active Scan

ZAP Alert Details

The CWE and WASC columns are only shown on wider screens - if you are using a mobile phone then try turning your screen sideways if you want to see them.

ID	Alert	Status	Risk	Type	CWE	WASC
0	Directory Browsing	release	Medium	Active	548	48
2	Private IP Disclosure	release	Low	Passive	497	13
3	Session ID in URL Rewrite	release		Passive		
3-1	Session ID in URL Rewrite	release	Medium	Passive	598	13
3-2	Session ID in URL Rewrite	release	Medium	Passive	598	13
3-3	Referer Exposes Session ID	release	Medium	Passive	598	13
6	Path Traversal	release		Active		
6-1	Path Traversal	release	High	Active	22	33

Baseline and Full Scan

1. **Baseline scan**

- **預設只進行一分鐘的爬蟲**
- **預設只進行採用 Passive Scan**

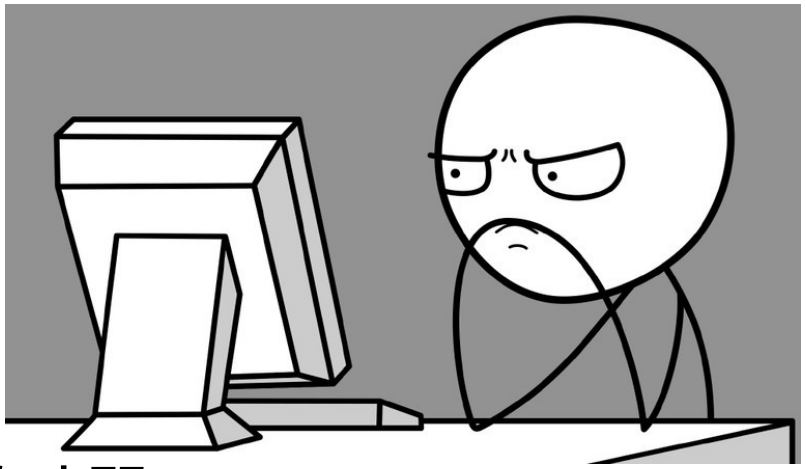
2. **Full Scan**

- **預設進行無時間限制的爬蟲，路徑完整但耗費更多的時間**
- **同時啟用 Active 與 Passive scan，更全面性找到潛在風險，同時也耗費更多時間**

Baseline and Full Scan

1. **Baseline scan**

- 預設只進行一分鐘的爬蟲
- 預設只進行採用 **Passive Scan**

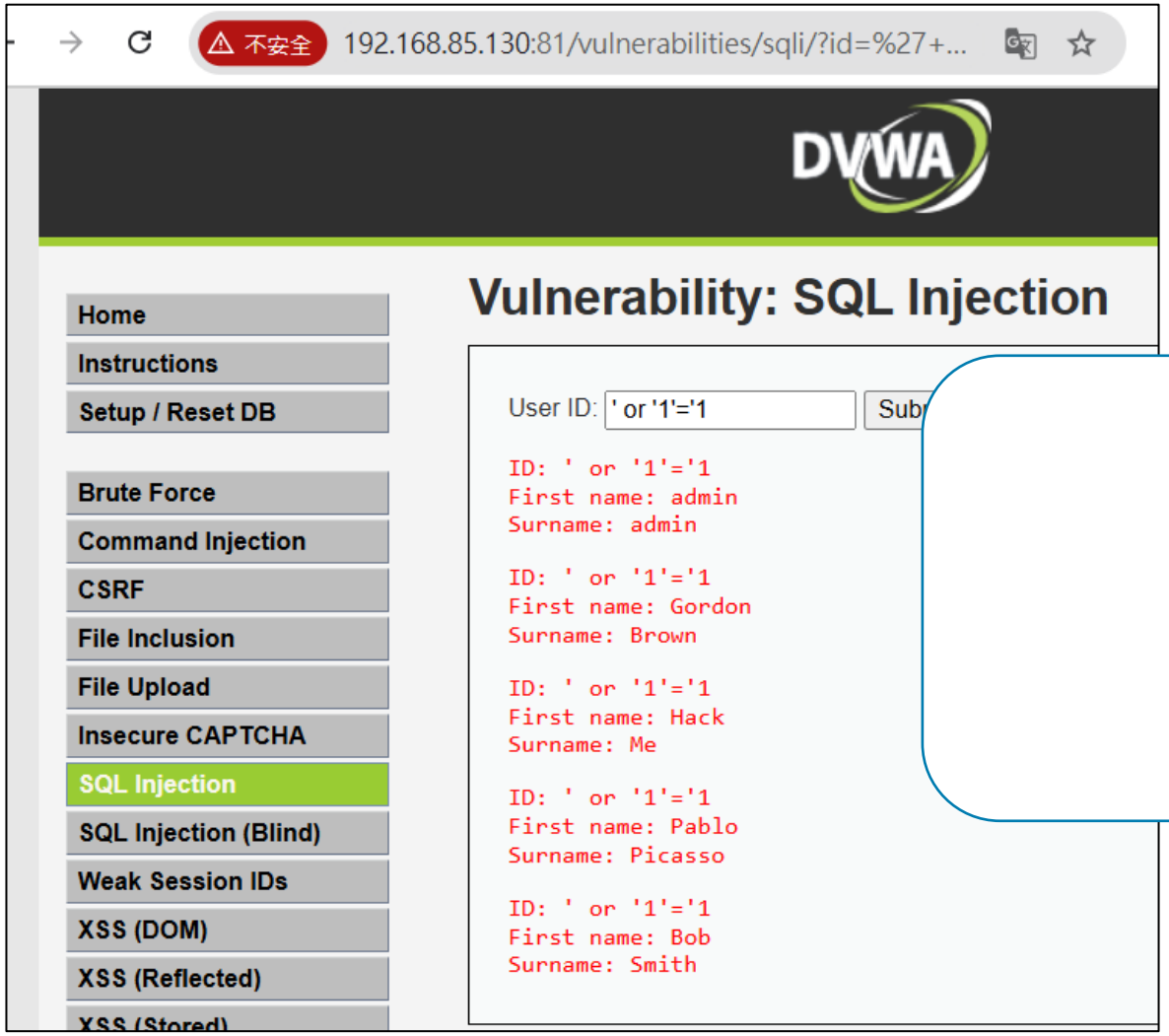


2. **Full Scan**

- 預設進行無時間限制的爬蟲，路徑完整但耗費更多的時間
- 同時啟用 **Active** 與 **Passive scan**，更全面性找到潛在風險，同時也耗費更多時間

檢測類型	花費時間	發現結果
Baseline Scan	1.5mins	0 High / 3 Medium / 4 Low
Full Scan	5mins	0 High / 4 Medium / 4 Low

Auth. 認證機制處理



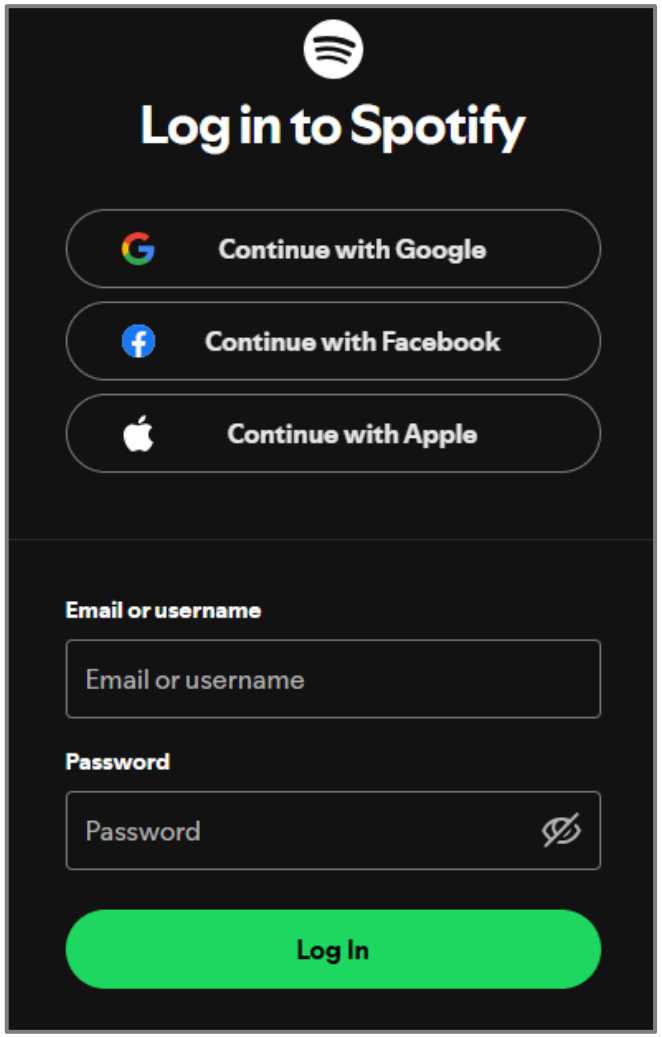
明明有幾個超容易就可以觸發
超好被檢測到的漏洞
但為什麼沒檢測到？？？

Auth. 認證機制處理

雖然是Full scan，雖然有進行Active Scan
但我們主要的功能，都是在登入之後才能使用的
如果我們不登入，怎麼能夠找到漏洞呢？

Auth. 認證機制處理

網站一



網站二

網站三

Auth. 認證機制處理

網站一

```
j_username=test-username&j_password=test-password&j_recaptcha=
03AFcWeA5vakCpSz_APLI44gHjVaW8-um6g5pueexlVGZNL2VRyFiHJkltnAm3zdAGO58
eO3abqZ-cTpj_g218QQHnmlHGJwjp_vXrPGEUz5Nmr-SAgkvfY26iiUMP0xDzFcRa4WR7
lKKN4V2PZGjNsHvZhcXO7nPsOl3HXOxwxhOTo_HkHxXNWDl7B0f6By4W4KE4Hy2M7ViLU
Xtwb5WkCtdifLBPvoDkr9I_P7XcuIDFyFWHF40EGM4BZ6HTXaOXOQQJxb9gJcATSM8sxM
4fK_qMsqRkaNQqkId9g68ZksEr_PY3eBu-N3e_scU4WeHf4wghwYqGo9UZtUvt_fRmRUq
```

網站二

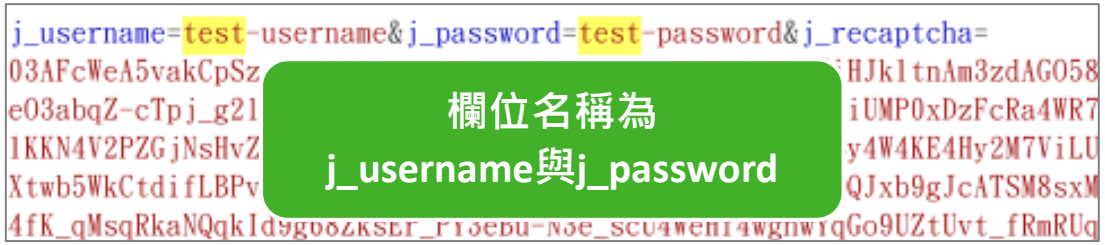
```
enc_password=
%23PWD_INSTAGRAM_BROWSER%3A10%3A1740640924%3AAYZQAAwzXbolzNZckDIt7TOYFztylgrQnP%2FivJwipQYKVXRVqr9cfllS%2BeVpF5vhNgpap8HrZ2
VvOTApHioRKrt6OCGNvS7JcwndnLI2Ct%2BqFnpO%2Fr7rOtaLyDzyn7Vi42F%2Fd40syfJIdJOwN%2FGCp3g%3D&caaF2DebugGroup=0&
loginAttemptSubmissionCount=0&optIntoOneTap=false&queryParams=%7B%7D&trustedDeviceRecords=%7B%7D&username=test-username
```

網站三

```
username=test-username&password=test-password&continue=
https%3A%2F%2Fopen.spotify.com%2F%3Fflow_ctx%3D2f113bbe-7c55-4982-b7c1-3ad8e004863c%253A1740662568&recaptchaToken=
03AFcWeA4NZelvudjdpwB3E2BWgCsvq2nbavi0Bz4YEmRiBYEMn0TtXpwctY0tRCxhFD75o57uiOylFI-69rjEVnHDRcskjUsi2ILoop7wwT2MY8sIABlJa
OfBEMyI_VUC_t6S613JF6WNliQwRAHnM8shRz3xbpxwORWyd2DFzscAEXiiQeqQd8pva2tDSOPcnlTA54ZWV4uRsfxPSsFSzHYAtCqq_7iVetn9MJ0F9NVN
qEciHdJAx9keI2qD0ConGmmYavrYQaAfVsfI8pLjg3M9u2Xn7Da0nBHi88bh7oACel5kRF5kGUQcjwZivYcs3qhVbi8mToZRDLdQlmcObN7B2bWGDKh-2Zf
iaEo3e9tK23yxSdTSVjJsTTEqeZ53pELY3jX8JdGgtsgXLX5yHcR29r-ho7nTYLyHpjQJlxZ0ex68MaxkyPQ_Mqy2OPLJ62ycn4TvmyQWijhtIeSBkY0jbt
kBZazHwe3isUs55uAgxLnZ_U5CfhViBooMy-wbnIsFKwJet5RintafkU6jxlTMavZpJ0PDunl6NqwaIgw3ZNnEOILhNI9lTjXNxCgxylrApQVSUNxlIBTYF
```

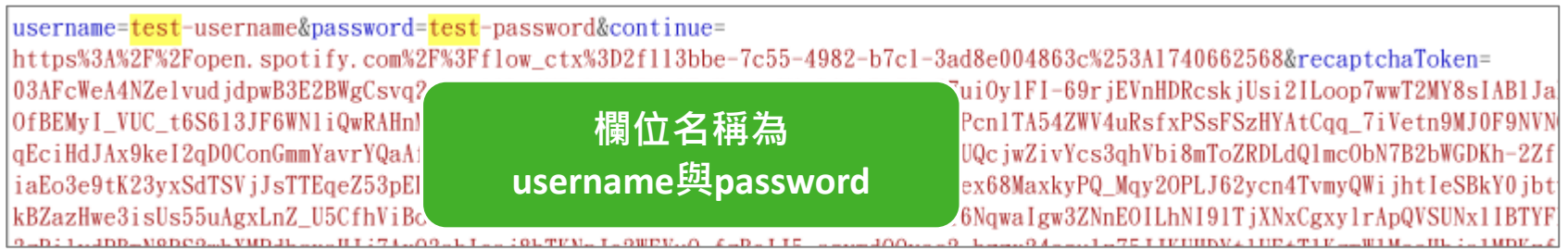
Auth. 認證機制處理

網站一



網站二

網站三



OWASP ZAP Authentication

Authentication Methods in OWASP ZAP

1. ~~Manual Authentication~~ (Not Applicable for CI/CD Pipeline)
2. Form-Based Authentication
3. JSON-Based Authentication
4. HTTP/NTLM Authentication (Authorization header with the Bearer token)
5. Script Based
6. Autorun config file

OWASP ZAP Authentication

Authentication Methods in OWASP ZAP

1. ~~Manual Authentication~~ (Not Applicable for CI/CD Pipeline)
2. Form-Based Authentication
3. JSON-Based Authentication
4. HTTP/NTLM Authentication (Authorization header with the Bearer token)
5. Script Based
6. Autorun config file (NEW!)

OWASP ZAP Authentication

老傢伙
你的替身最沒用了

沒有斑紋
沒有通透

沒有左手

拒絕成為版本更新受害者

Form-Based & Script-Based

Form Based

```
docker run --user root -w /zap \
-v /data/jenkins/workspace/owasp-zap-dast:/zap/wrk:rw --rm \
zaproxy/zap-stable:2.16.0 zap-full-scan.py \
-t http://192.168.85.130:81 \
-J zap-output.json \
-r zap-report.html \
--hook=/zap/auth_hook.py \
-z "auth.loginurl=http://192.168.85.130:81/login.php \
    auth.username=admin \
    auth.password=password \
    auth.username_field=username \
    auth.password_field=password \
    auth.submit_field=Login \
    auth.auto=1 \
    auth.loggedin=Welcome \
    http.cookie=security=low" \
-m 60
```

Script Based

```
def authScript = '''
    function authenticate(helper, paramsValues, credentials) {
        var requestUri = new URI(paramsValues.get("loginUrl"), false);
        var requestMethod = "POST";
        var requestBody = "username=" + credentials.get("username") + "&pas

        var requestMessage = helper.prepareMessage();
        requestMessage.getRequestHeader().setURI(requestUri);
        requestMessage.getRequestHeader().setMethod(requestMethod);
        requestMessage.setRequestBody(requestBody);
        helper.sendAndReceive(requestMessage);

        var responseMessage = requestMessage.getResponseBody().toString();
        if (responseMessage.contains("Login Successful")) {
            return true;
        } else {
            return false;
        }
    }
'''
```

Form-Based Auth. using Autorun config

```
stage('Integration - OWASP ZAP Scan') {
    steps {
        script {
            catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {
                sh '''
                docker run --user root -v /data/jenkins/workspace/owasp-zap-dast:/zap/wrk:rw --rm \
                zaproxy/zap-stable:2.16.0 \
                zap.sh -cmd -autorun /zap/wrk/zap-auth-config.yaml
                '''
            }
        }
    }
```

Form-Based Auth. using Autorun config

```yaml
env:
  contexts:
  - name: "DVWA-form"
    urls:
    - "http://192.168.85.130:81"
    includePaths:
    - "http://192.168.85.130:81.*"
    - "http://192.168.85.130:81/vulnerabilities/"
    excludePaths:
    - "http://192.168.85.130:81/logout.php"
    - "http://192.168.85.130:81/login.php"
    - "http://192.168.85.130:81/setup.php"
    - "http://192.168.85.130:81/security.php"
    - "http://192.168.85.130:81/vulnerabilities/csrf.*"
    authentication:
      method: "form"
      parameters:
        loginRequestBody: "username={%username%}&password=password&Login=Login&user_token=f4c2be9fba02186111829f053797c917"
        loginPageUrl: "http://192.168.85.130:81/login.php"
        loginRequestUrl: "http://192.168.85.130:81/login.php"
      verification:
        method: "poll"
        loggedInRegex: "\\Qadmin\\E"
        pollFrequency: 60
        pollUnits: "requests"
        pollUrl: "http://192.168.85.130:81/instructions.php"
        pollPostData: ""
```

zap-auth-config.yaml

Auth. 認證機制處理挑戰

During scan

- What if session time out or expired
- What if trigger logout

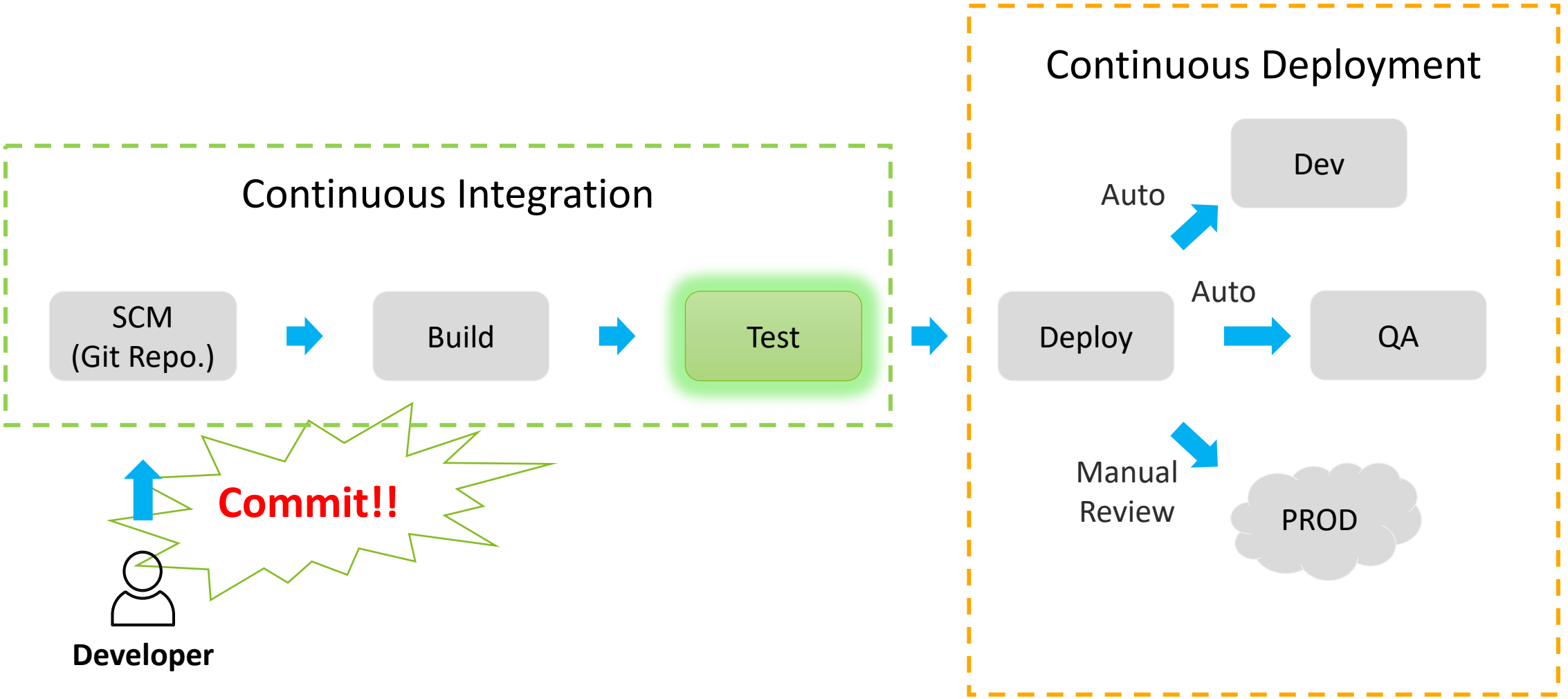
When login

- What if there is captcha
- What if there is OTP MFA

OWASP ZAP運行結果

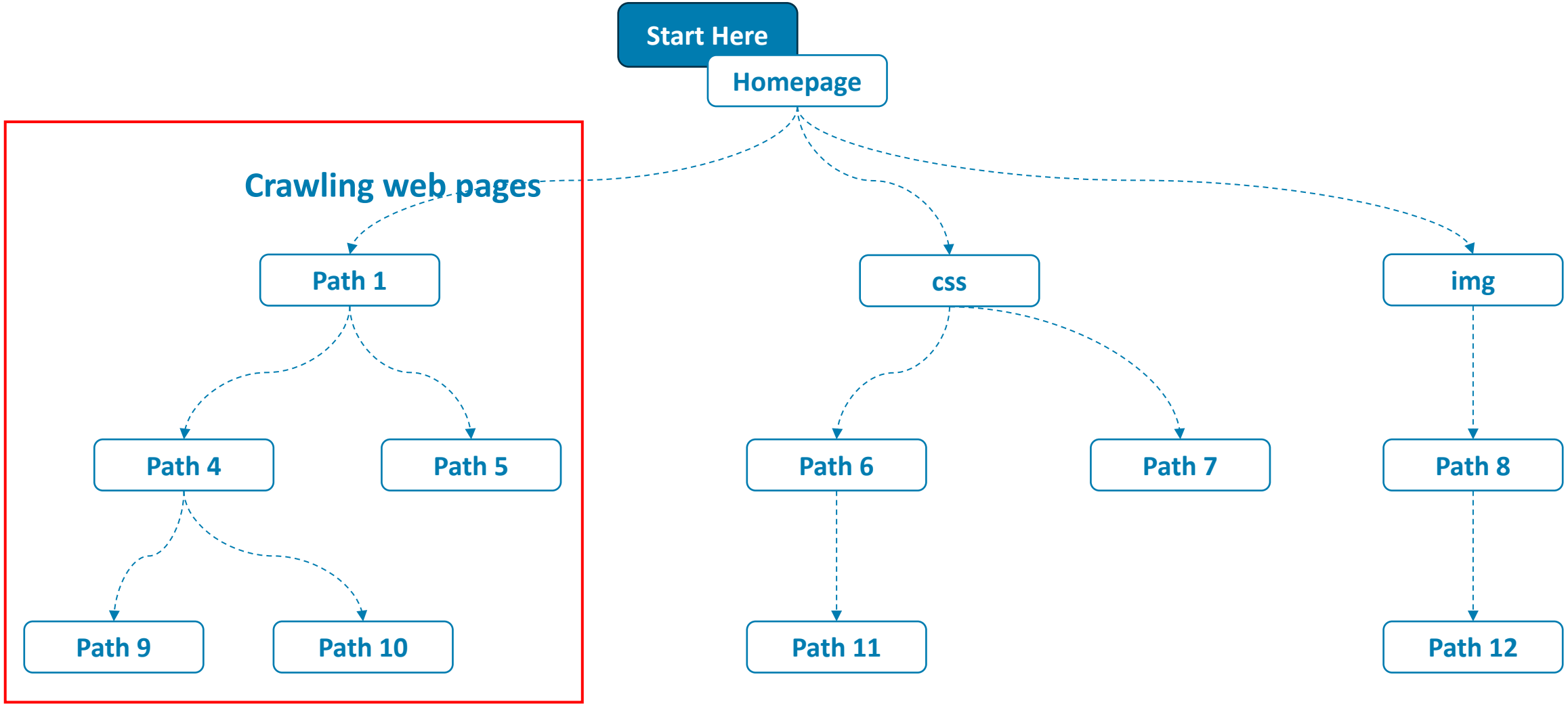
檢測類型	花費時間	發現結果
Baseline Scan	1.5mins	0 High / 3 Medium / 4 Low
Full Scan	5mins	0 High / 4 Medium / 4 Low
Full Scan with Auth.	80mins	5 High / 7 Medium / 8 Low
Full Scan with Auth. with Policy	12mins	4 High / 3 Medium / 5 Low

CI/CD Pipeline



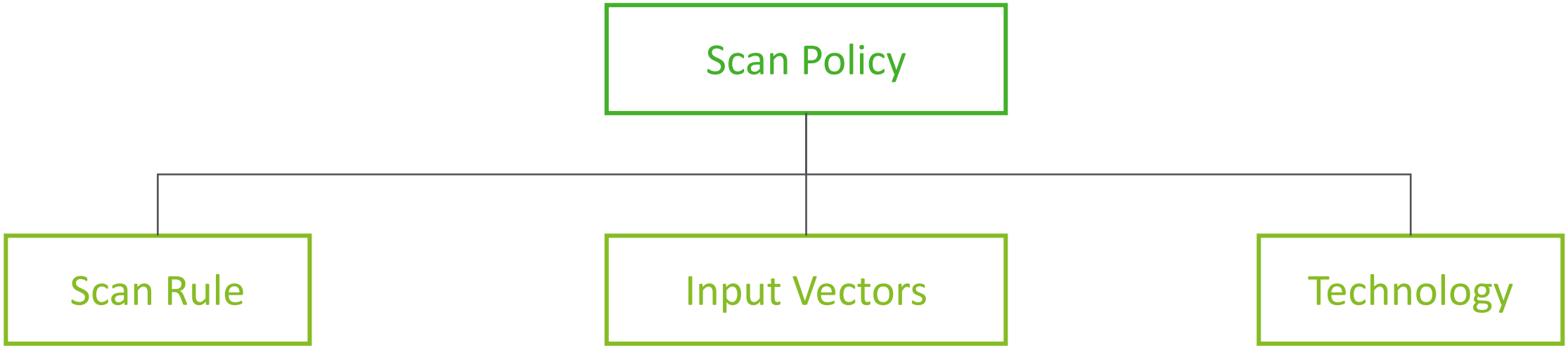

80
MINUTES
LATER...

Web Crawler / Spider - Limited Scope



Limit the scope of crawlers

Best Practices - Scan Policy



- SQL Injection
- Path Traversal
- LDAP Injection
- Log4Shell
- JWT None Exploit

- URL Query Parameter
- URL Path
- POST Data
- HTTP Headers
- Cookie Data
- Multipart Form Data

- Operating System
- Web Server
- Language
- Database
- SCM

Technology

- Operating System
  - Linux
  - MacOS
  - Windows
- Web Server
  - Tomcat
  - IIS
  - Apache
- SCM
  - Git
  - SVN
- Language
  - C
  - XML
  - PHP
  - JavaScript
  - Python
  - ASP
  - JSP/Servlet
  - Java
  - Ruby
- Database
  - SQLite
  - CouchDB
  - Microsoft SQL Server
  - MongoDB
  - SAP MaxDB
  - PostgreSQL
  - Sybase
  - IBM DB2
  - MariaDB
  - Oracle
  - Firebird

Finally Autorun config

```
stage('Integration - OWASP ZAP Scan') {
    steps {
        script {
            catchError(buildResult: 'SUCCESS', stageResult: 'FAILURE') {
                sh '''
                docker run --user root -v /data/jenkins/workspace/owasp-zap-dast:/zap/wrk:rw --rm \
                zaproxy/zap-stable:2.16.0 \
                zap.sh -cmd -autorun /zap/wrk/zap-limited-config.yaml
                '''
            }
        }
    }
```

Form-Based Auth. using Autorun config

```
contexts:
- name: "DVWA-form"
  urls:
  - "http://192.168.85.130:81"
  includePaths:
  - "http://192.168.85.130:81/vulnerabilities/xss_r.*"
  - "http://192.168.85.130:81/vulnerabilities/sqli.*"
  - "http://192.168.85.130:81/vulnerabilities/fi.*"
  - "http://192.168.85.130:81/vulnerabilities/exec.*"
  excludePaths:
  - "http://192.168.85.130:81/logout.php"
  - "http://192.168.85.130:81/login.php"
  - "http://192.168.85.130:81/setup.php"
  - "http://192.168.85.130:81/robots.txt"
  - "http://192.168.85.130:81/sitemap.xml"
  - "http://192.168.85.130:81/security.php"
  - "http://192.168.85.130:81/dvwa/css"
  - "http://192.168.85.130:81/dvwa/images"
  - "http://192.168.85.130:81/dvwa/js"
  - "http://192.168.85.130:81/docs"
  - "http://192.168.85.130:81/instructions.php"
  - "http://192.168.85.130:81/vulnerabilities/csp/"
  - "http://192.168.85.130:81/vulnerabilities/upload/"
  - "http://192.168.85.130:81/vulnerabilities/weak_id"
  - "http://192.168.85.130:81/vulnerabilities/javascript.*"
  - "http://192.168.85.130:81/vulnerabilities/captcha/"
  - "http://192.168.85.130:81/vulnerabilities/brute/"
  - "http://192.168.85.130:81/vulnerabilities/csrf.*"
  - "http://192.168.85.130:81/vulnerabilities/page.*"
  authentication:
    method: "form"
    parameters:
      Cookie: "PHPSESSID={%cookie:PHPSESSID%}; security=low"
  technology:
    exclude:
    - C
    - MacOS
    - SQLite
    - CouchDB
    - XML
    - Microsoft SQL Server
    - MongoDB
    - SAP MaxDB
    - SVN
    - JavaScript
    - PostgreSQL
    - Sybase
    - Python
    - Windows
    - ASP
    - IBM DB2
    - MariaDB
    - Oracle
    - JSP/Servlet
    - Firebird
    - HypersonicSQL
    - Ruby
```

zap-limited-config.yaml

OWASP ZAP運行結果

檢測類型	花費時間	發現結果
Baseline Scan	1.5mins	0 High / 3 Medium / 4 Low
Full Scan	5mins	0 High / 4 Medium / 4 Low
Full Scan with Auth.	80mins	5 High / 7 Medium / 8 Low
Full Scan with Auth. with Policy	12mins	4 High / 3 Medium / 5 Low

Takeaway

DAST in CI/CD Pipeline

- OWASP ZAP 基準掃描 (Baseline Scan)：僅執行被動式掃描 (Passive Scan)
- 評估 OWASP ZAP 是否應使用完整掃描(Full Scan)：依風險需求決定是否啟用主動式掃描(Active Scan) ，在Full Scan當中主動式掃描是啟用的。
- 實作身分驗證 (Implement Authentication)：確保掃描涵蓋身分驗證與授權流程，否則會遺漏驗證後才能存取的功能風險與漏洞。
- 限制爬蟲範圍 (Limit Crawler Scope)：聚焦目標應用，避免無關路徑。可在掃描中大幅減少掃描時間。
- 設定掃描策略 (Set Scanning Policy)：自訂規則以平衡效率與覆蓋率。在安全性及效率當中取得最佳平衡。

Deloitte.

勤業眾信