# HOWTO: Perform Web Application Security Assessments

## Introduction:

This guide is to provide a summary of how to perform a web application security assessment. Web application penetration testing is carried out in various phases to ensure clear planning and delivery model. When performing external or internal penetration tests, we employ a standard 5-step methodology. This methodology allows for a systematic testing process that ensures all appropriate tests have been applied to the proper devices. The penetration testing methodology is based on industry best practices such as the OSSTMM (Open Source Security Testing Methodology Manual), SANS Top 25, OWASP ASVS (Open Web Application Security Project Application Security Verification Standard) and WASC (Web Application Security Consortium). This ensures that you receive quality and repeatable results, and minimizes the risk to your systems under test.

## Pre-requisite:

We recommend looking over the pre-requisite checklist and the RoE (Rules of Engagement) with the customer or client before starting your assessment. It is important to receive consent from all parties before starting the assessment or at least make sure the client or vendor has a bug bounty program. Be sure to notify your manager before beginning any assessments. Lastly, this guide is written from the perspective and the view point of a seasoned penetration tester, meaning we are assuming this is not your first penetration test of a web application.

### Software Requirements:

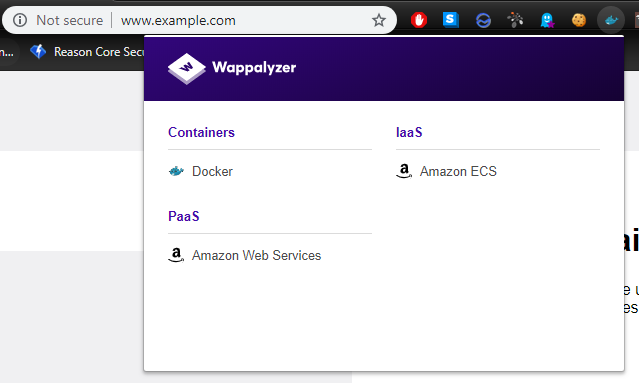
You will need the following services or software installed on or accessible to your machine (you may need to contact support to have these enabled or installed).

#### Recommended Setup:

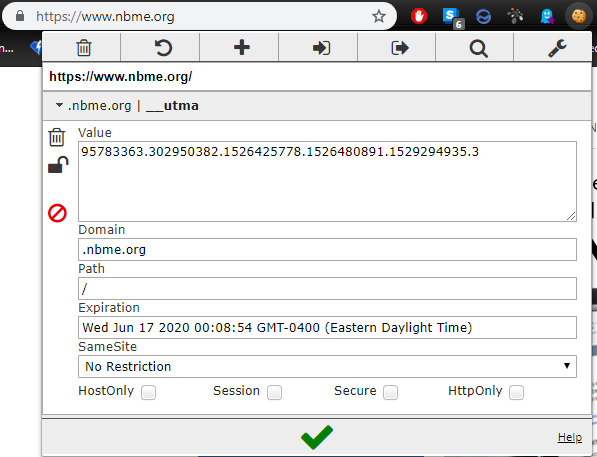
1. Hosted Hypervisor (e.g., VMWare Workstation, Virtual Box, etc.)
2. A Linux distro (pick one and install it in in your hypervisor of choice)
   1. [Black Arch Linux](https://blackarch.org/downloads.html)
   2. [Kali](https://www.kali.org/downloads/) (Recommended)
   3. [Ubuntu](https://www.ubuntu.com/download)
3. Python v2.7 and Python v3.7 (skip if you are using Kali Linux)
4. Git (this should already be preinstalled but check just in case)
5. [OpenVAS](http://www.openvas.org/index.html), [Rapid7 Nexpose](https://www.rapid7.com/products/nexpose/) or [Tenable Nessus](https://www.tenable.com/products/nessus/nessus-professional)
6. [Arachni (Linux)](https://www.arachni-scanner.com/download/) and [BurpSuite v2 Pro (Windows)](https://support.portswigger.net/customer/portal/topics/718317-installing-and-configuring-burp/articles)
7. A web browser with the following extensions installed
   1. [Edit This Cookie (Chrome)](http://www.editthiscookie.com/) or [Cookie Editor (Firefox)](https://addons.mozilla.org/en-US/firefox/addon/cookie-editor/?src=search)
   2. [Proxy SwitchySharp (Chrome)](https://chrome.google.com/webstore/detail/proxy-switchysharp/dpplabbmogkhghncfbfdeeokoefdjegm?hl=en) or [FoxyProxy(Firefox)](https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/)
   3. [VirusTotal](https://support.virustotal.com/hc/en-us/articles/115002700745-Browser-Extensions)
   4. [Vulners Web Scanner(Chrome)](https://chrome.google.com/webstore/detail/vulners-web-scanner/dgdelbjijbkahooafjfnonijppnffhmd) or [unofficial Firefox build](https://addons.mozilla.org/en-US/firefox/addon/vulners-web-scanner/)
   5. [Wappalyzer](https://www.wappalyzer.com/download)
8. Access to the internet
   1. When testing internal web applications, we recommend connecting internally
      1. Be sure to ask permission from the network (security) team before kicking off any scans
   2. When testing external web applications, we recommend connecting externally (e.g., Comcast line or guest Wi-Fi) to avoid false positives
9. Microsoft [Threat Modeling Tool](https://docs.microsoft.com/en-us/azure/security/azure-security-threat-modeling-tool) or OWASP [Threat Dragon](https://www.owasp.org/index.php/OWASP_Threat_Dragon)
10. The Google (is your best friend)

## Phase I - Reconnaissance & Information Gathering:

1. Navigate the web application in your choice browser
   1. Take note of functionality, input/output fields and client side code
   2. Take note of the technology stack running on the web application (Wappalyzer comes handy here)

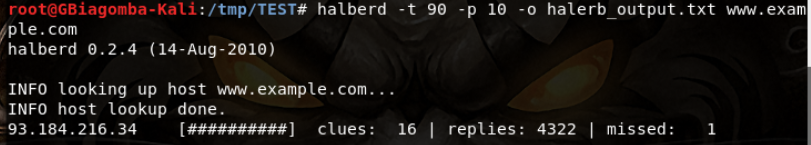


* 1. Take note of what cookies the web application is using (Cookie Editor or Edit This Cookie is perfect for this)

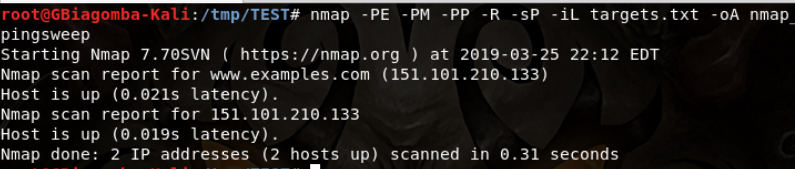


* 1. Take note of the architectural design and data flow (check to see if a threat model exists or make one)
     1. Understand the security objectives of the website and take note of anything an attacker could find of interest
        1. Test for business logic: Session management, authentication vulnerabilities, file upload, etc.
        2. Reach out to architects and lead developers if needed be (used best judgement)

1. Determine the network components running the web application
   1. Find out if the system is load balanced: **halberd -t 90 -p 10 -o “halberd\_output.txt” https://www.example.com**



* 1. Find out systems are alive: **nmap -PE -PM -PP -R -sP -iL targets.txt -oA nmap\_pingsweep**
     1. The “targets.txt” file will consist of the full list of targets from the RoE and whatever you find from running halberd.



* 1. Perform port knocking, services bound to each port, and OS fingerprinting on the aforementioned systems: **nmap -A -iL targets.txt -p 0-65535 -Pn -R --reason --resolve-all -sSUV –T4 -oA nmap\_portknock**
     1. If the scan takes too long (use your best judgement) you might have to expedite the scan by entering one of the following commands:
        1. **nmap -A --top-ports 200 -Pn -R --reason --resolve-all -sSUV -T4 -oA nmap\_portknock**
           1. The above command will only check the top 200 most commonly used ports
        2. **nmap -A -F -Pn -R --reason --resolve-all -sSUV -T4 -oA nmap\_portknock**
           1. The above command will put nmap in fast mode which will only check the top 100 most commonly used ports and it will skip a few host discovery checks.

1. Assuming the web application is hosted over HTTPS, you will need to check for the TLS/SSL configuration: **testssl --csv --fast --parallel --sneaky https://www.example.com**
   1. Placeholder for screenshot
2. Search the target web server for files (e.g., pdf, docx, pptx, xlsx, etc.) of interest: **metagoofil -d www.example.com -l 500 -o metagoofil/evidence -f metagoofil\_output**
3. Gather additional information (e.g., emails, subdomains, hosts, employee names, etc.) from the web server: **theharvester -d www.example.com -l 500 -b all -h**

## Phase II - Network Vulnerability Scanning:

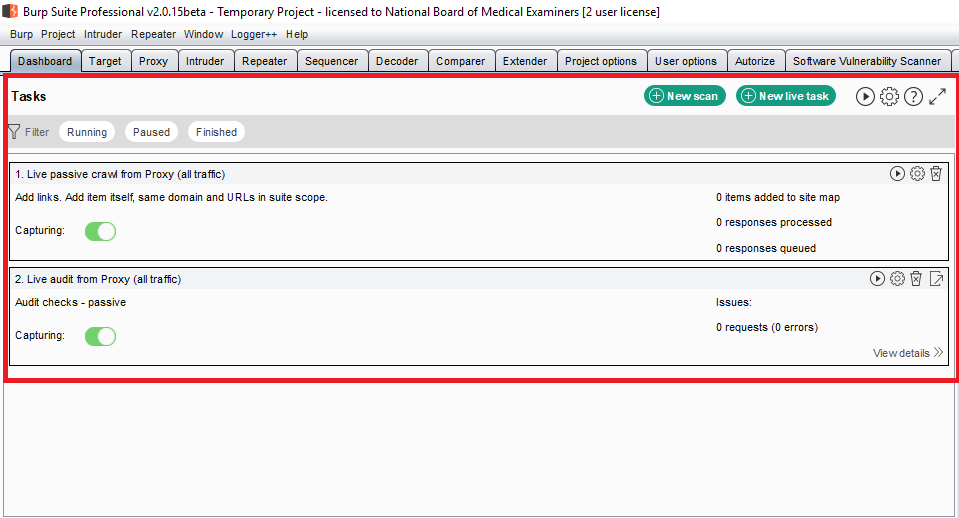
Normally this step is skipped however if you have permission and wish to be more thorough, I would recommend performing this step. You will need to perform two (2) network vulnerabilities scans using two (2) separate tools. We do this for the sake of cross validation and ensuring nothing was missed. Additionally this will simulate a “budget” attacker and a moderately to well-funded attacker.

1. Perform a network vulnerability scan using [OpenVAS](https://docs.greenbone.net/GSM-Manual/gos-4/en/vulnerabilitymanagement.html) or some other [network vulnerability scanner](http://bfy.tw/MwqI)
2. Perform a secondary network vulnerability scan using another network vulnerability scanner (e.g., [Tenable Nessus](https://docs.tenable.com/Nessus.htm), [Rapid7 Nexpose](https://nexpose.help.rapid7.com/docs/security-console-quick-start-guide), etc.)

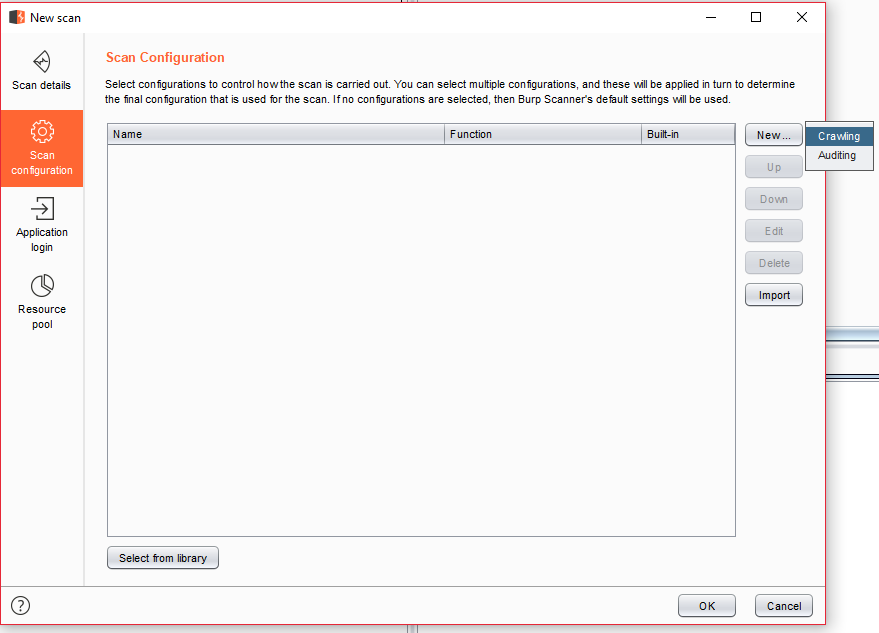
## Phase III - Web Vulnerability Scanning:

Be advised you may have to rerun the below tools (i.e., arachni, burp, etc.) multiple times. The first scan will be unauthenticated, the second scan should be authenticated using an account with least privileges (e.g., standard user) and the last scan should be authenticated using an account with elevated or most privileges (e.g., administrator).

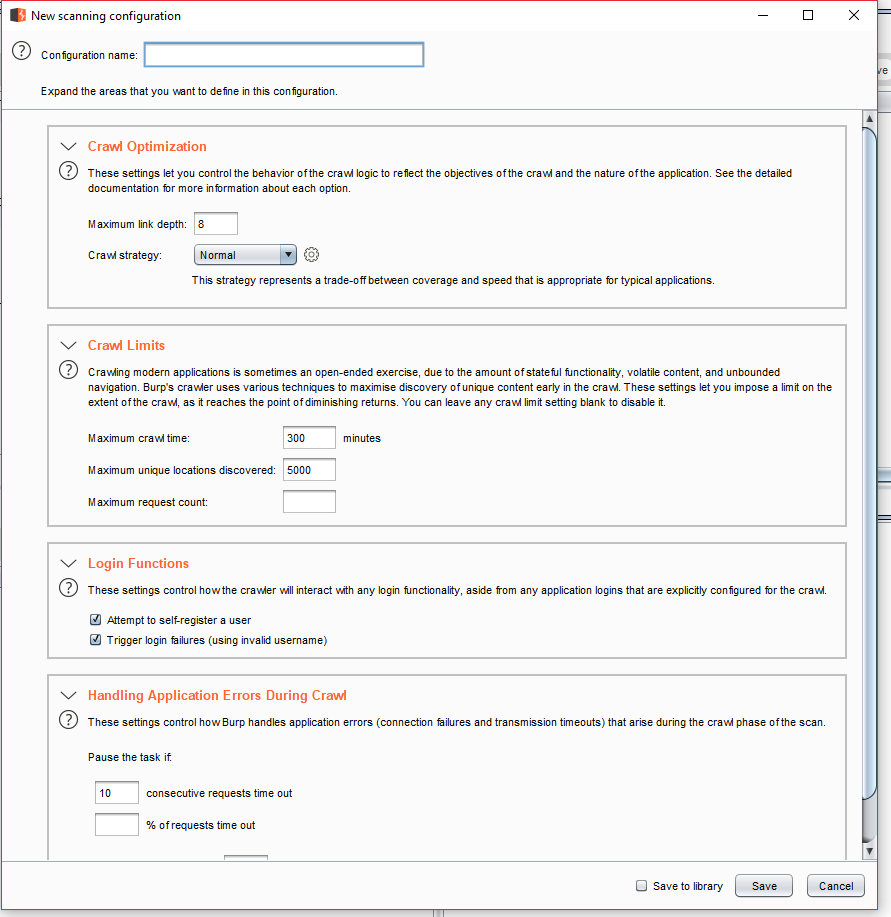
1. Perform subdomain enumeration: **sublist3r -d www.example.com -v -t 5 -o sublist3r\_output.txt**
2. Perform directory, path traversal and filename enumeration: **dirb https://www.example.com:443 /usr/share/dirbuster/wordlists/directory-list-1.0.txt -o dirb\_ output.txt -w**
3. Perform additional web vulnerability scan:
   1. **nikto -C all -h www.example.com -port 443 -o nikto\_output.txt**
   2. **arachni\_multi https://www.example.com http://www.example.com --report-save-path=arachni\_output.afr**
4. Perform additional scans using BurpSuite
   1. When you first open burp, be sure to pause live passive and live audit scanning



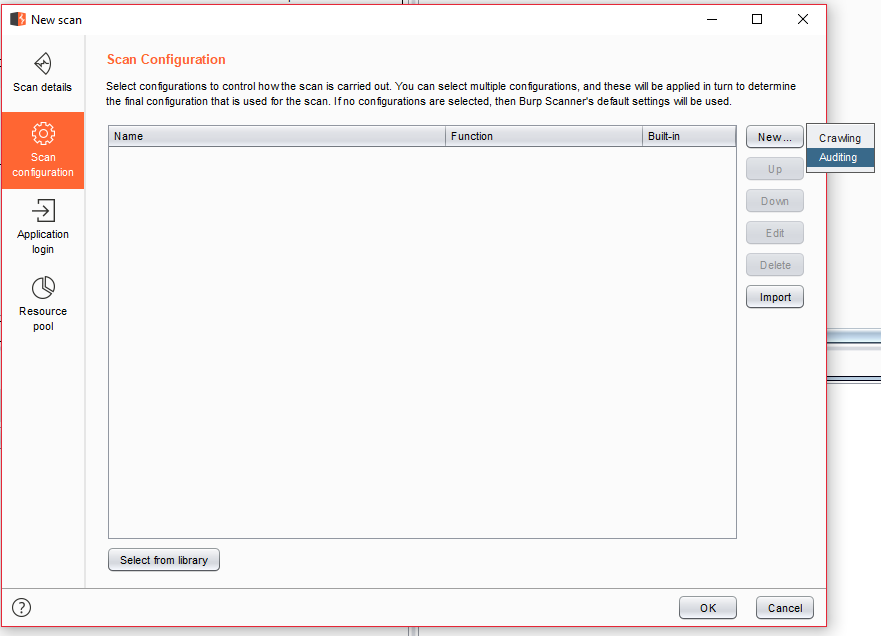
1. You might want to install additional extensions before starting your scan, [click here](https://support.portswigger.net/customer/portal/articles/1965930-how-to-install-an-extension-in-burp-suite) for more details
   * 1. You may have to download [Jython](https://www.jython.org/downloads.html) (python based extensions) and [JRuby](https://www.jruby.org/download) (ruby based extensions)
2. Visit the target web application in your browser and redirect your browser traffic to Burp: [click here](https://portswigger.net/burp/documentation/desktop/penetration-testing/configuring-your-browser) for more detail
3. Make sure you turn intercept off (for now) by going to: Proxy > Intercept > Intercept on/off
4. Add your target web application to the scan by going to: Proxy > HTTP History > Look for the target website > Right click > Add to Scope > Yes
5. Set up the target scan you wish to perform a scan of
   * 1. Go to dashboard > click “New scan”
     2. Select “Crawl and Audit”
     3. Click “Scan configuration” (located on the left-hand side) > New > Crawling



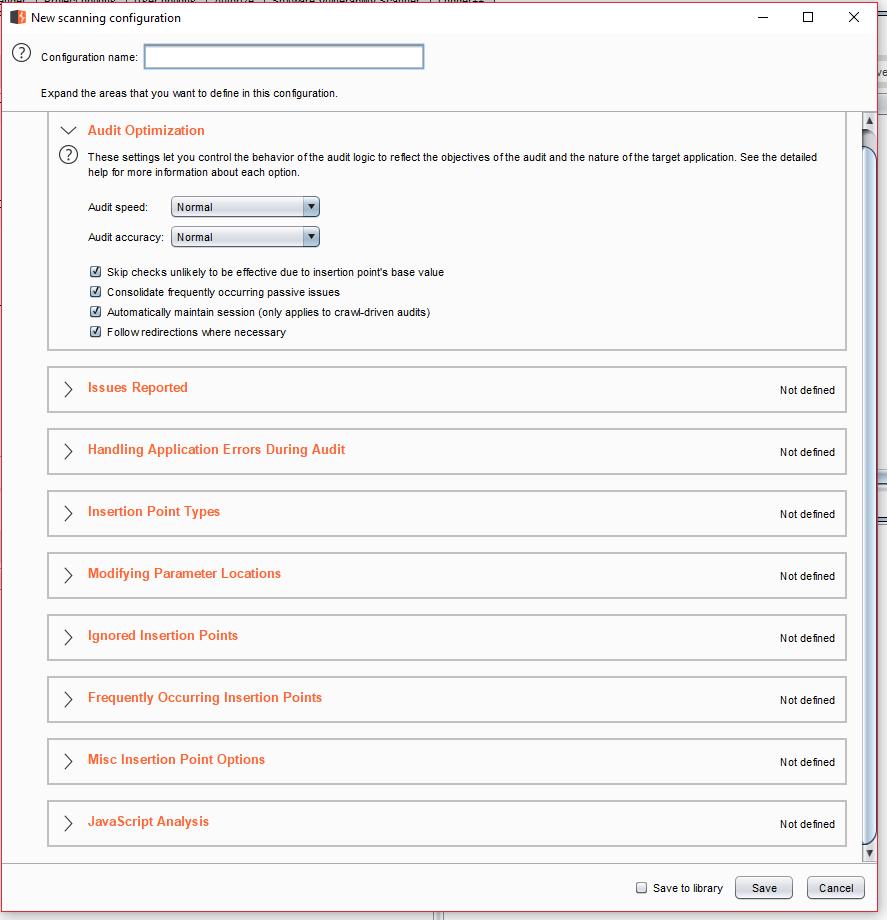
* + 1. Go through all the crawl setting by click “>” to expand the options
       1. To keep any changes you make to the profile, keep the drop down expanded (I usually keep the default crawling settings)
       2. Provide a name for the configuration name (input field located at the top left)
       3. Save the changes by clicking “save” at the bottom right



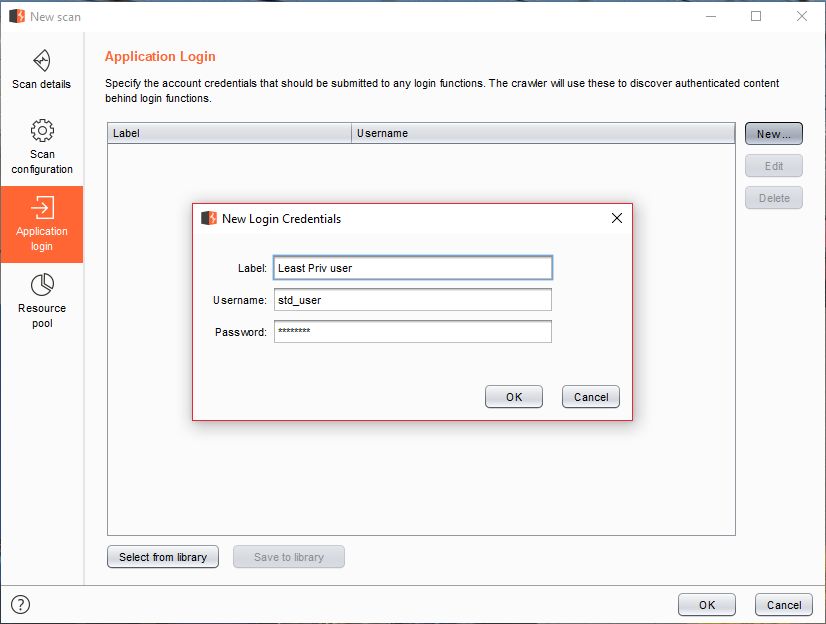
* + 1. Click “Scan configuration” (located on the left-hand side) > New > Auditing



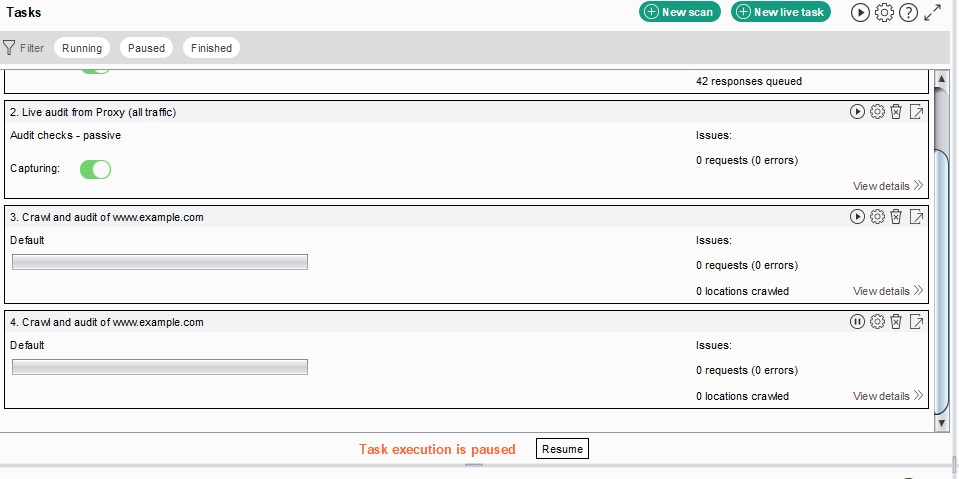
1. Go through all the audit setting by click “>” to expand the options
   * + 1. To keep any changes you make to the profile, keep the drop down expanded (I usually keep the default audit settings)
       2. Provide a name for the configuration name (input field located at the top left)
       3. Save the changes by clicking “save” at the bottom right



1. Click “Application Login” > New > enter the test credentials you wish to use & label it accordingly > Click “OK”



1. Click “Resource pool” > feel free to modify this to your hearts contempt > Click “OK”
2. The scan should automatically start on its own from there



## Phase IV - Manual Testing & False Positive Analysis:

During this phase the tester will gather all the scan data from all the tools and go through them. Do to the high variety of outcomes; it is difficult to provide some test cases and validation, therefore I would recommend using the following testing guides:

1. OWASP Top 10 testing using [Burp Suite](https://support.portswigger.net/customer/portal/articles/1969845-using-burp-to-test-for-the-owasp-top-ten)
2. OWASP [ASVS](https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project) Testing Guideline
3. Miscellaneous OWASP [Testing Guide](https://www.owasp.org/index.php/OWASP_Testing_Project)
4. Testing Restful API Testing – [Click here](https://support.portswigger.net/customer/portal/articles/2898216-using-burp-to-test-a-rest-api)
5. Testing session management
   * Testing token handling using burp – [Click here](https://support.portswigger.net/customer/portal/articles/1964140-using-burp-to-test-session-token-handling)
   * Testing session token generation – [Click Here](https://support.portswigger.net/customer/portal/articles/1964169-using-burp-to-test-session-token-generation)
   * Testing session handling rules – [Click Here](https://support.portswigger.net/customer/portal/articles/2363088-configuring-burp-s-session-handling-rules)
6. Testing for CSRF – [Click Here](https://support.portswigger.net/customer/portal/articles/1965674-using-burp-to-test-for-cross-site-request-forgery-csrf-)
   * You can use the [CSRF PoC](https://portswigger.net/burp/documentation/desktop/functions/generate-csrf-poc)
7. Testing for Clickjacking – [Click Here](https://portswigger.net/burp/documentation/desktop/tools/clickbandit)
   * Hunting for the vulnerability – [Click Here](https://support.portswigger.net/customer/en/portal/articles/2363105-using-burp-to-find-clickjacking-vulnerabilities)
8. Testing for Cross Site Scripting – [Click Here](https://support.portswigger.net/customer/portal/articles/1965737-using-burp-scanner-to-find-cross-site-scripting-xss-issues)
   * Reflected XSS – [Click Here](https://support.portswigger.net/customer/portal/articles/2325939-using-burp-to-manually-test-for-reflected-xss)
   * Stored XSS – [Click Here](https://support.portswigger.net/customer/portal/articles/2325938-using-burp-to-manually-test-for-stored-xss)
   * Injecting into direct HTML – [Click Here](https://support.portswigger.net/customer/portal/articles/2325935-exploiting-xss---injecting-into-direct-html)
9. Testing for SQL Injection – [Click Here](https://support.portswigger.net/customer/en/portal/articles/2791005-using-burp-to-find-sql-injection-flaws.html)
10. Testing for code injection – [Click Here](https://support.portswigger.net/customer/en/portal/articles/2590642-using-burp-to-test-for-code-injection-vulnerabilities)
11. Using burp testing methodology – [Click here](https://support.portswigger.net/customer/portal/articles/2326039-the-burp-methodology-)
12. All miscellaneous testing methodologies using burp – [Click here](https://support.portswigger.net/customer/portal/topics/792273-burp-testing-methodologies/articles)

## Phase V - Reports & Deliverables:

All reports and deliverables are located in “I:\DataSecurity\Application Security Assessments Reports”. Additionally, the template report document (titled “[QX-YYYY - CLIENT-Web-Application-Security-Assessment-Report-Draft.dotx](file:///I:\DataSecurity\Application%20Security%20Assessments%20Reports\QX-YYYY%20-%20CLIENT-Web-Application-Security-Assessment-Report-Draft.dotx)”) is located in the root of the same folder.