Venari Ultimate Edition Evaluation Guide

Overview

This evaluation guide takes you through the steps needed to install Venari, onboard a web application and scan that application for security vulnerabilities. There are also sections exploring the major user interface elements.

Evaluation Goals

What is Covered

- Installation and license setup
- (Optional) Downloading and running a vulnerable web application in a Docker container
- Onboarding the web application with basic configuration information
- Starting the scan
- Reviewing the completed scan vulnerabilities (findings)
- Reviewing the scan detailed results in various UI summary and details views
- Onboarding a separate application using pre-created, downloadable templates and workflow files
- Tables of testable Docker images and public internet sites (that are legal to scan) are at the end of the guide

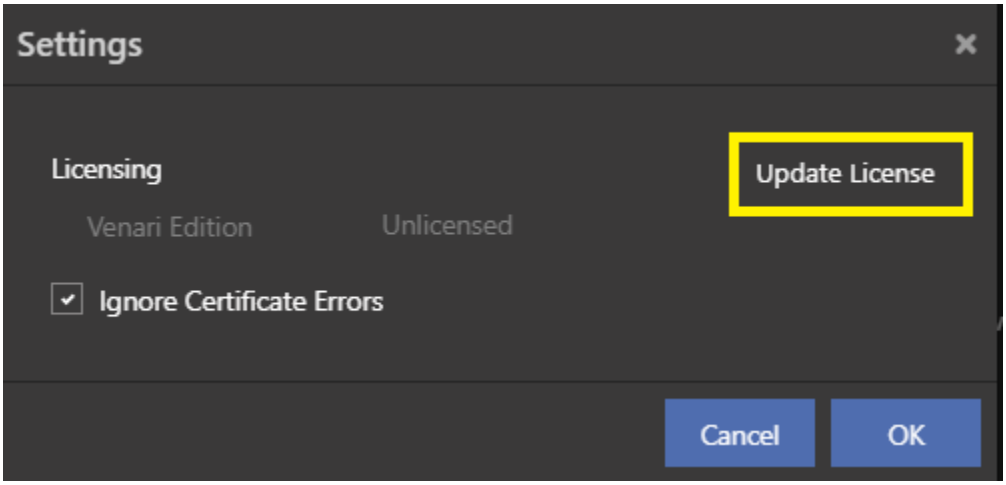
What is Not Covered

- Advanced configuration
- Re-test and triage
- Web API endpoint onboarding
- Manually Creating a login workflow*

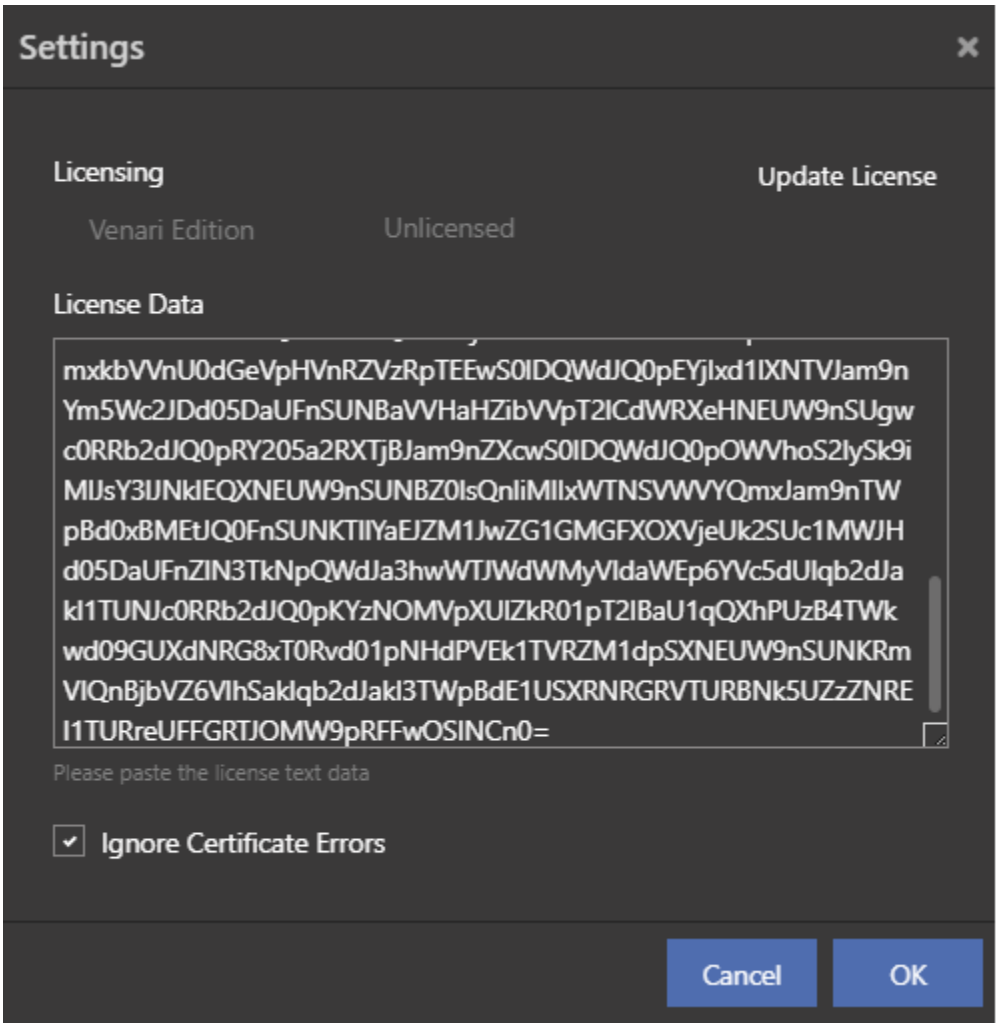
** The example scan configuration in this guide uses **auto-login** which only requires that the username and password be entered in the UI. In cases where auto-login fails, there is a procedure for creating a login workflow. The steps needed to link a login workflow to a template are covered in the last section. The instructions for creating the login workflow are available at: https://assertsecurity.io/venaridocs/quick-starts/tips-and-tricks/record-login-workflow/record-login-workflow/*

Installation and License Setup

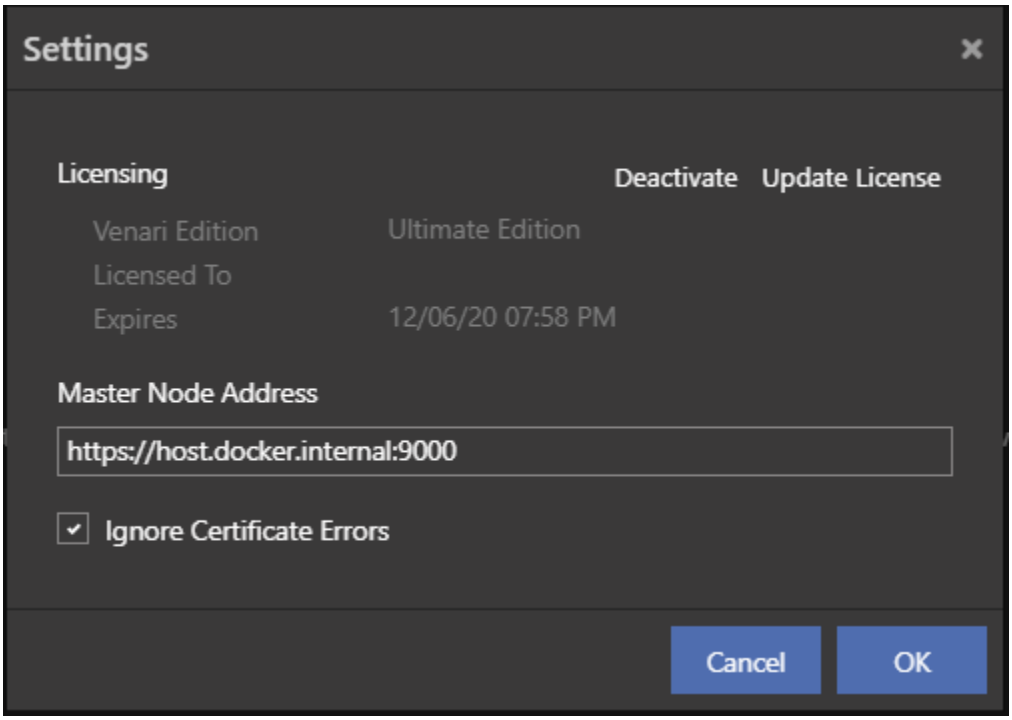
Run the Venari installer. You will see the UI pop up with an active dialog box for entering the license token. Click ‘Update License’.



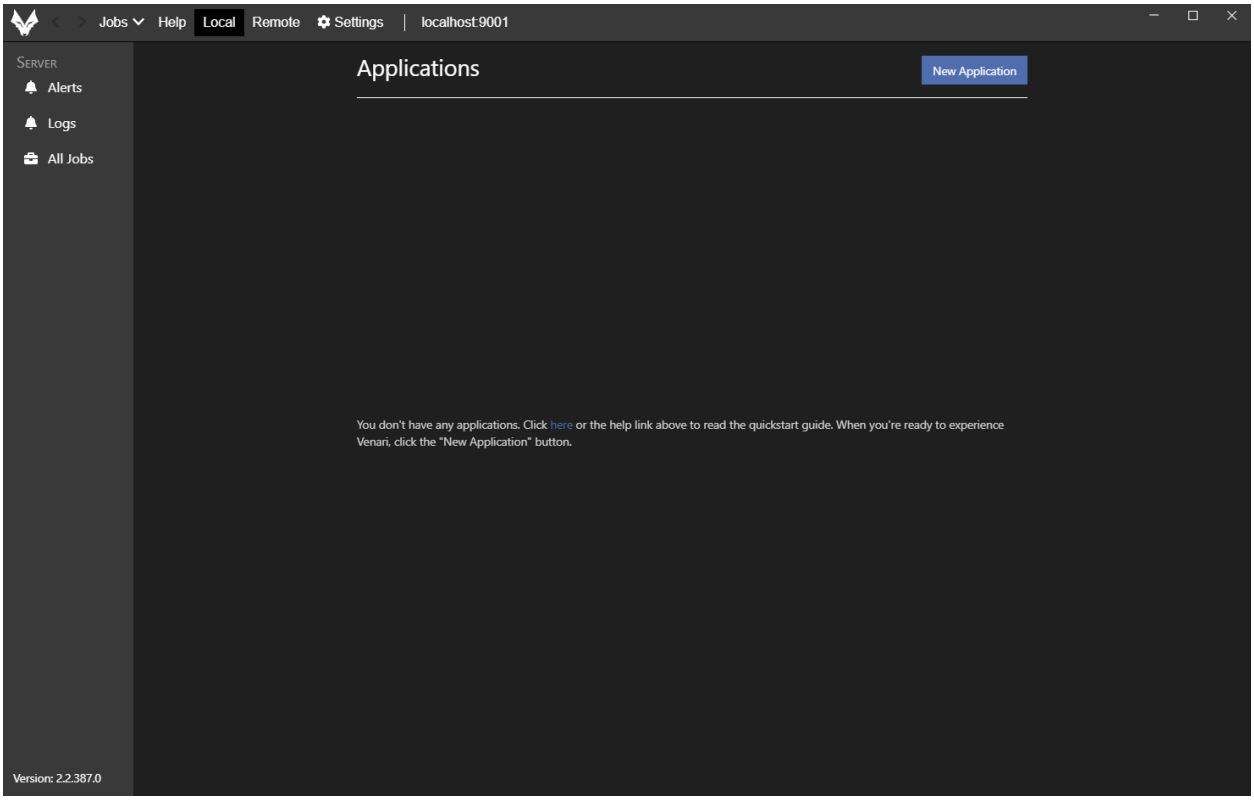
Enter the license text from your evaluation email and click the OK button.



A second dialog box will appear, and it should be pre-populated with the correct information. Click the OK button.



At this point Venari is fully installed and configured for use. The UI should look like the image below. Note that the 'Local' tab is highlighted. The 'Remote' tab will not be described in this evaluation guide since it pertains to Venari DevOps Edition. Clicking the remote tab will produce an error dialog about a connection failure to the remote host. This dialog can be ignored if you happen to click the remote tab.



Setting up a Vulnerable Test Application to Scan (Optional)

There are a variety of deliberately vulnerable web applications that have been created as teaching guides. This guide will use such an application for demonstration purposes and the screenshots and text will refer to this application and the results of scanning it. There is an appendix at the end of this document with links to publicly available, known vulnerable web applications and another list of Docker pull commands. The docker images can be pulled to your local machine. The launch commands are included in the appendix as a quick start to getting the applications running.

The following steps will create a running Docker container hosting a vulnerable application called XVWA. Scanning this application will find vulnerabilities from the OWASP Top 10 list.

1. Install and run Docker

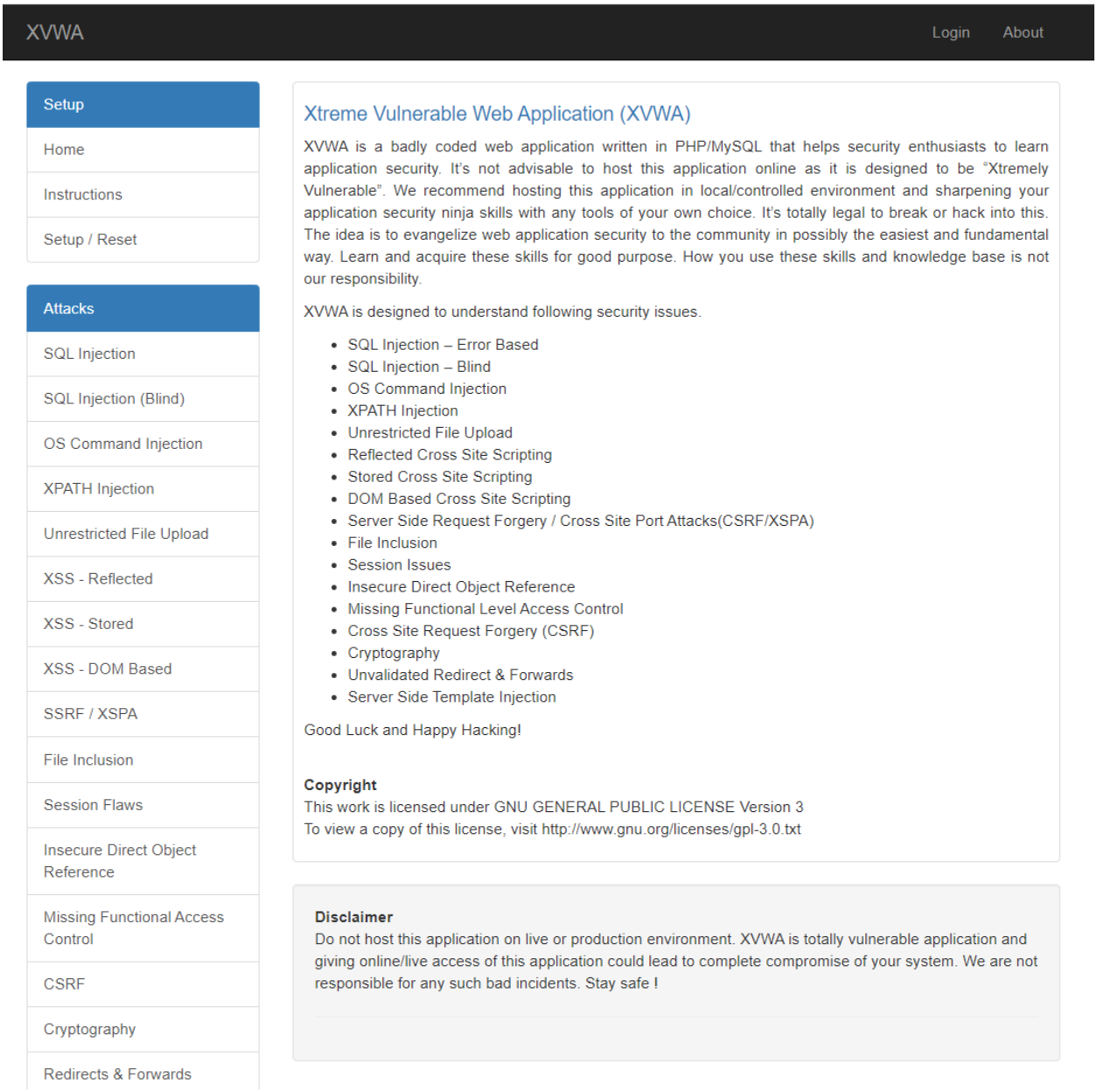
2. Pull the docker image with this command:

   docker pull bitnetsecdave/xvwa

3. Run the Docker image with this command:

   docker run -p 1234:80 -it bitnetsecdave/xvwa

4. Browse to http://localhost:1234/xvwa/ to make sure the application is running and reachable. The browser page should look like the image below.



Onboarding the Test Application

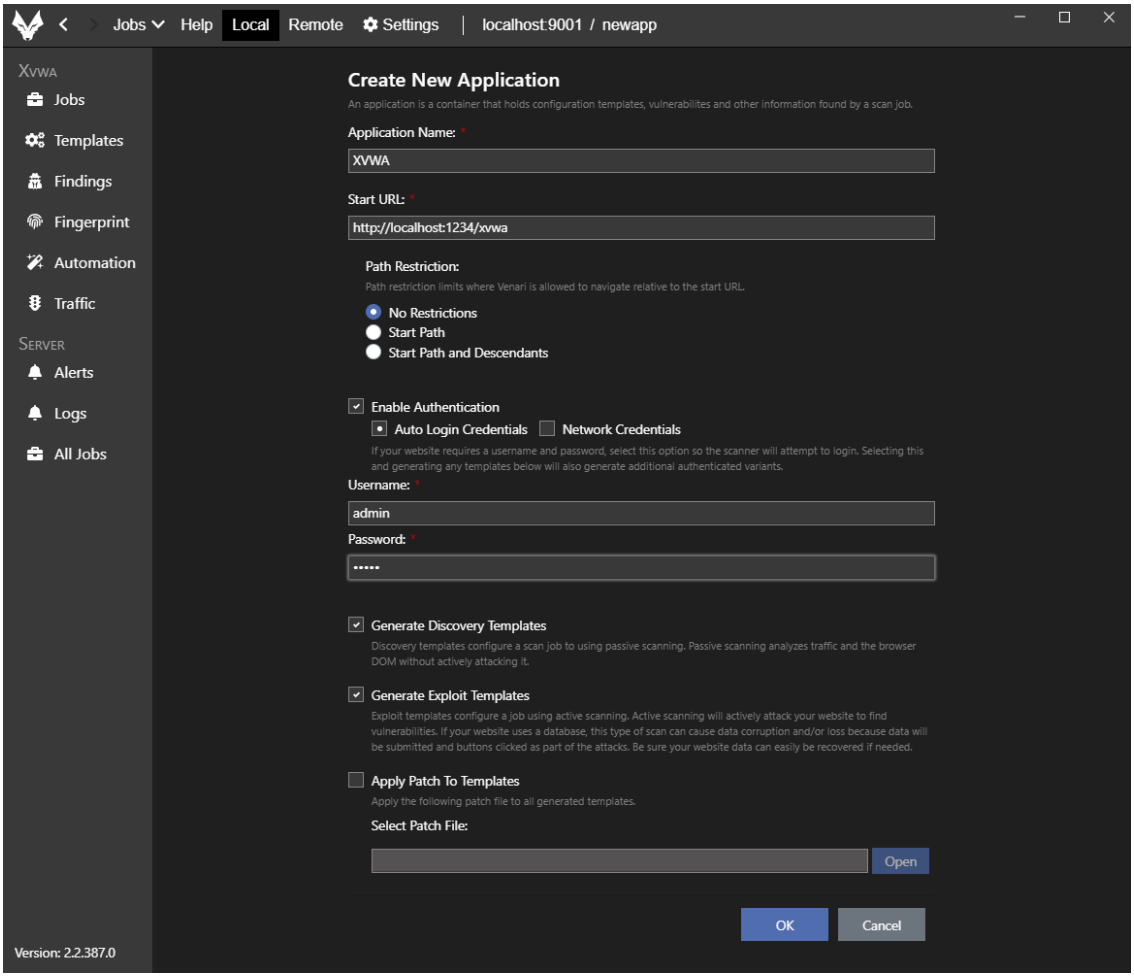
‘Onboarding‘ an application means using the Venari UI to create a named workspace and a default set of job templates that are used in future scans. The named workspaces are referred to as ‘Applications’ in the UI. The steps below show the actions needed to onboard the XVWA application from the previous section.

Note that there is no step to record navigation steps needed for login. Venari has an advanced workflow engine that can take simple username and password credentials and heuristically figure out the navigation and browser actions required. This works in most cases.

If auto-login does not work for an application you are onboarding, see the instructions at https://assertsecurity.io/venaridocs/quick-starts/tips-and-tricks/record-login-workflow/record-login-workflow/ to create the login workflow manually.

Onboarding Steps:

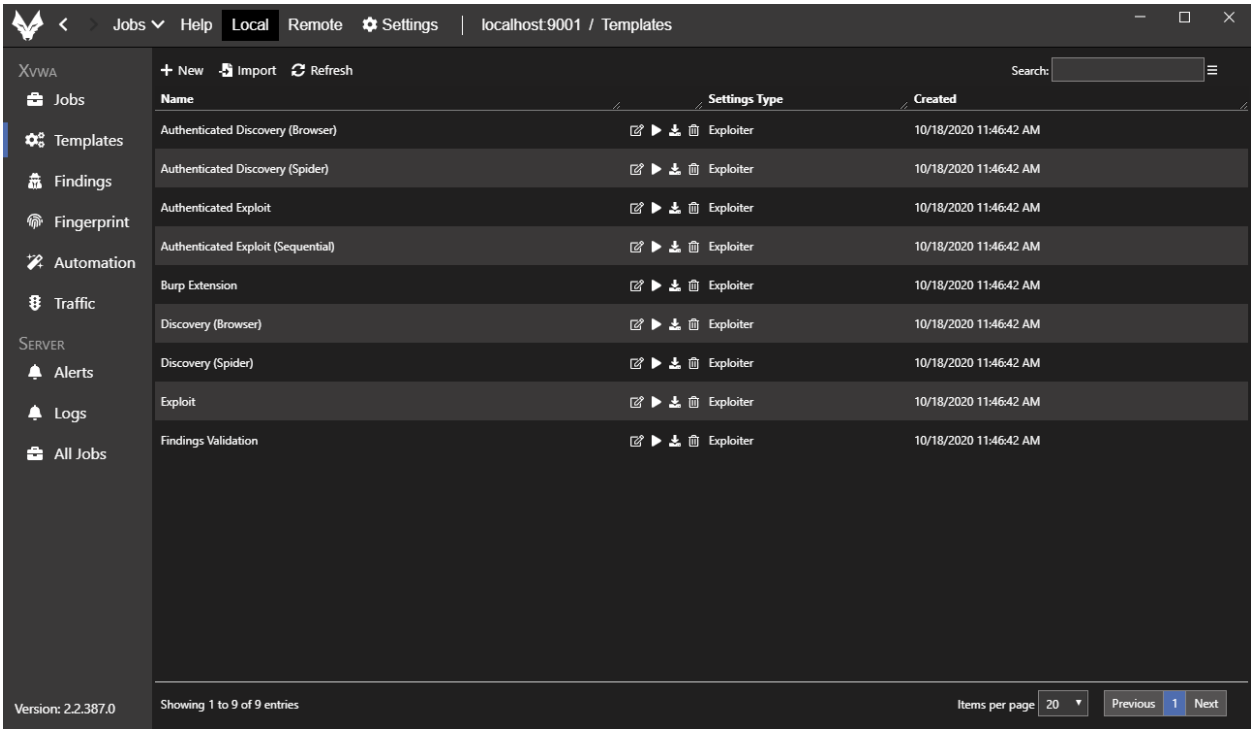
1. Click the ‘New Application’ button on the start screen.
2. Type XVWA into the Application Name field
3. Type http://localhost:1234/xvwa into the start URL field
4. Select the radio button titled ‘Start Path and Descendants’
5. Type ‘admin’ into the username field
6. Type ‘admin’ into the password field. The screen should match the image below



7. Click the OK button

Scanning the Application

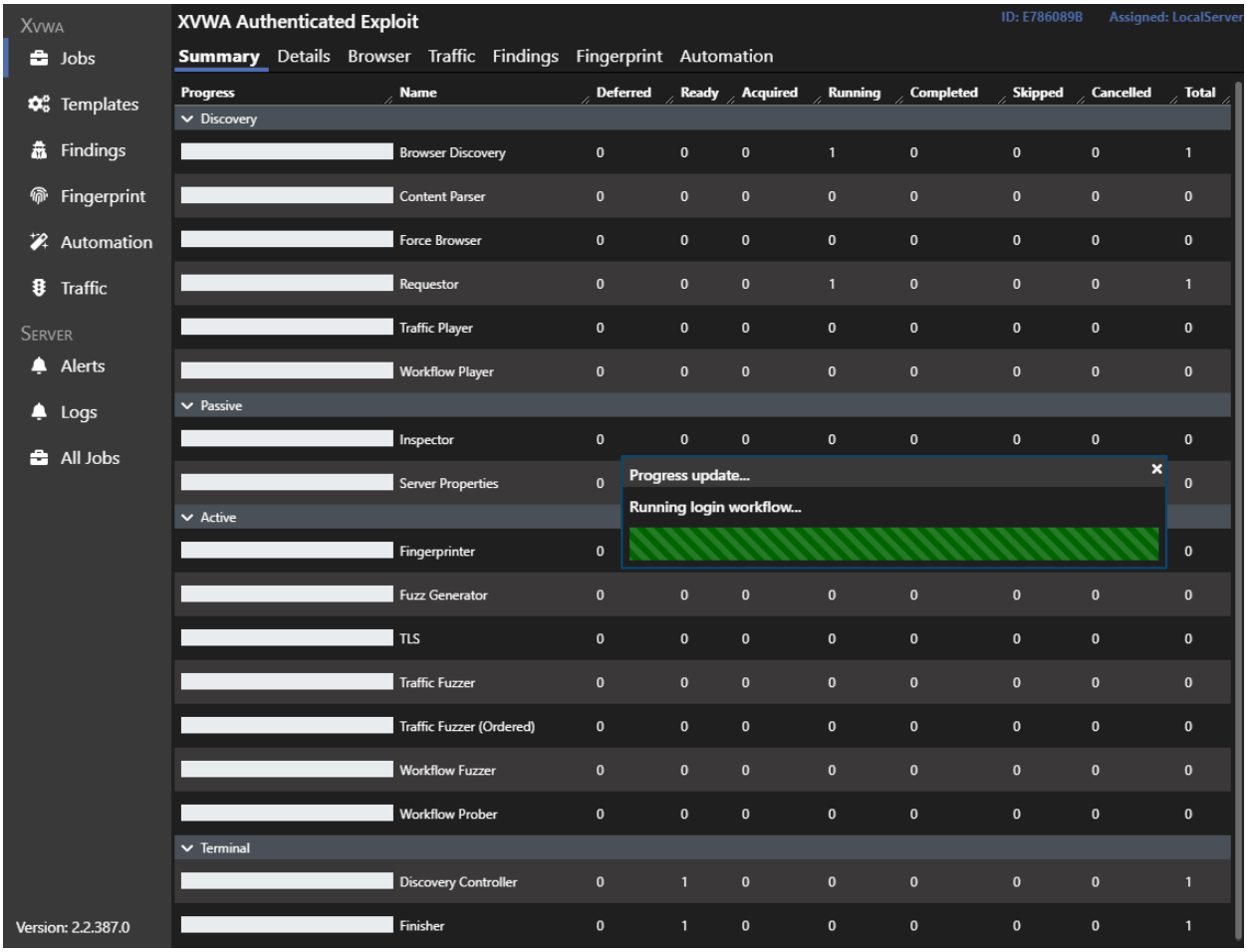
After entering basic information on the start screen, the new application will be created with default job templates. The application page should look like the image below.



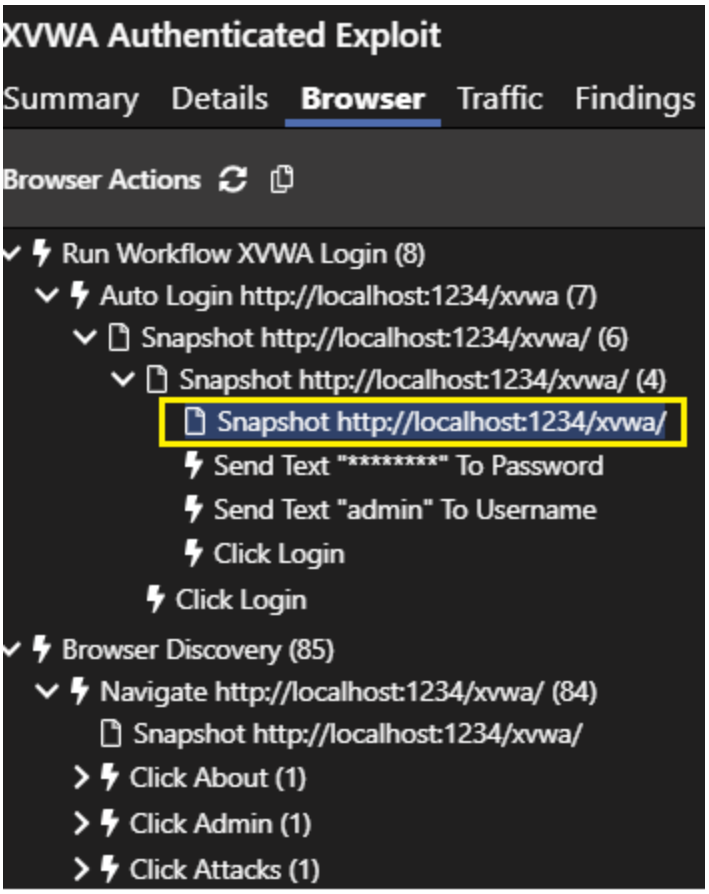
Follow the Steps below to start the authenticated exploit scan.

1. Find the ‘Authenticated Exploit’ row in the UI grid and click the triangle icon (the icon resembles a ‘play’ button)

2. Wait for the scanner to spin up (this may take a few seconds) and observe the auto-login progress bar.



3. The scan is now running. Once the progress dialog disappears you can confirm the that the login was successful by selecting the browser tab and iteratively expanding the top tree node snapshots. The tree will look like the image below and the right-hand side view will show a browser screen capture of the page in a logged in state. The top right of the screenshot will show that the ‘Admin’ user is logged in.



4. Let the scan complete. Scan times vary based on hardware resources, VM containment and other system factors. The authenticated exploit scan time for XVWA while preparing this guide took 16 minutes on a PC running Windows 10. The XVWA application was run in a Docker container and there was no virtual machine hosting involved.

Overview of Vulnerabilities

The scan results are shown on the 'Summary' tab. The vulnerability results are in the lower right panel and should roughly match the image below.

Finding Summary (27)

Severity	Name	Count
Critical	Command Injection	1
Critical	Cross Site Open Redirect	1
Critical	Cross Site Scripting (Reflected)	6
Critical	Unprotected Transport of Credentials (Client)	1
Critical	Unrestricted Failed Logins	1
High	Cross Frame Scripting	1
High	Cross Site Request Forgery (confirmed)	1
High	Local File Inclusion	2
High	Remote File Inclusion	2
High	Unrestricted File Upload (Multi-Part)	1
Medium	Cross Site Request Forgery (possible)	4
Medium	Directory Listing	3
Medium	Javascript CVE-2012-6708	1
Medium	Javascript CVE-2015-9251	1
Medium	Javascript CVE-2019-11358	1

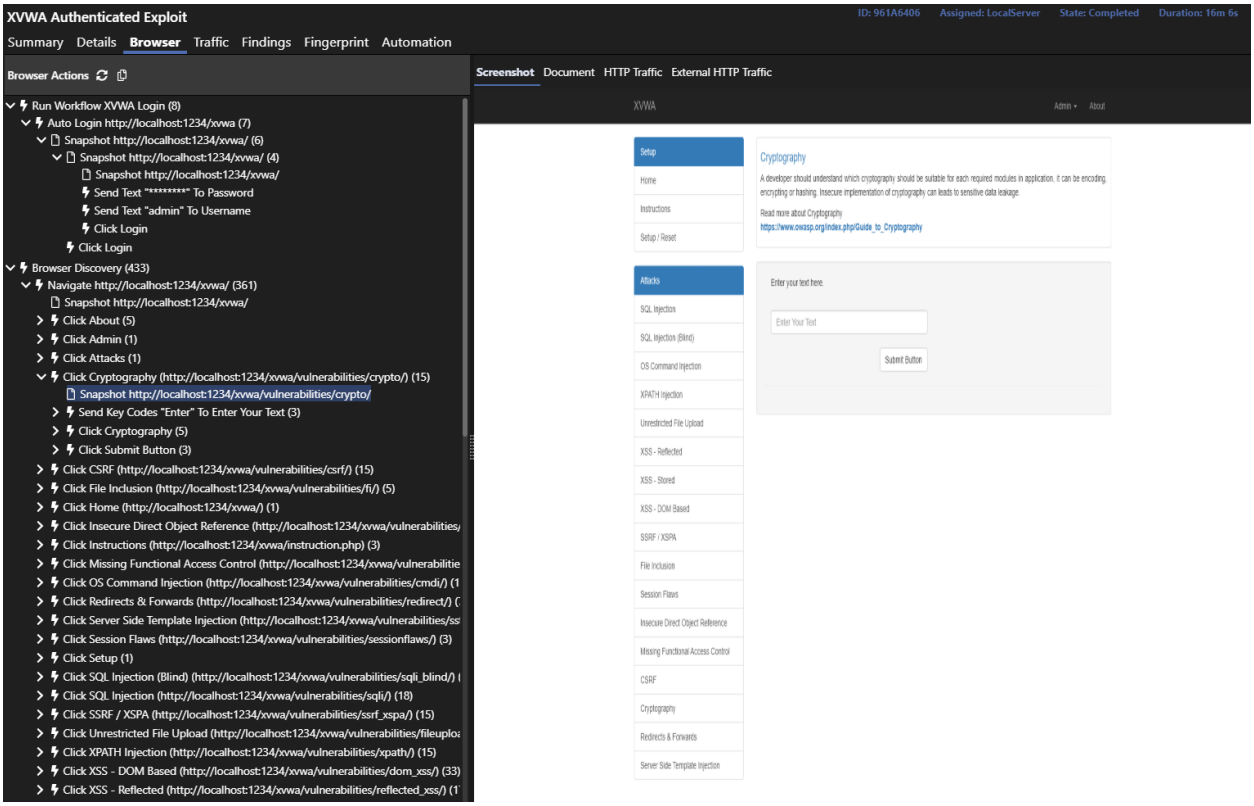
Overview of Other Tabs

Venari accumulates many useful insights about the application while scanning. Information about the browser actions, site URLs and fingerprint information are saved and presented in separate tabs that span the top of the main view. Details on the Findings (Vulnerabilities) and the associated evidence are shown in the 'Findings' tab.

Browser Tab

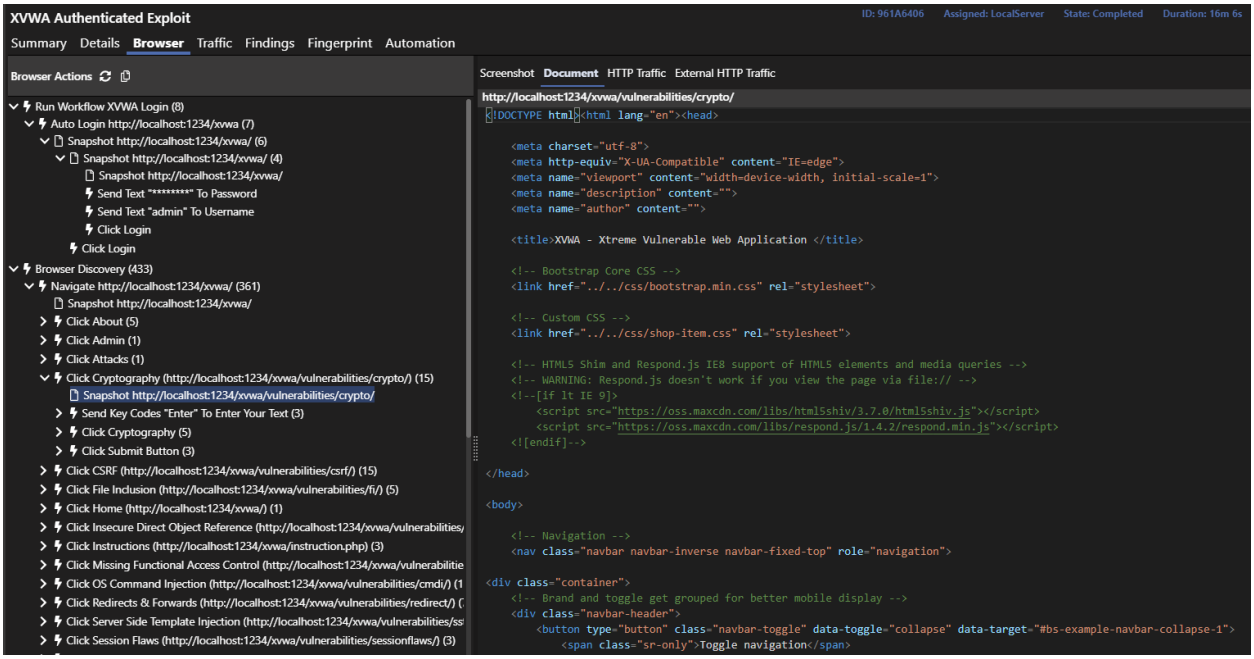
Venari's core scan engine uses a pool of headless browsers to do the vulnerability analysis. This approach yields far better coverage and higher fidelity DOM information than using only HTTP request/response analysis. This architecture also enables Venari to handle modern JS frameworks and to get inside Single Page Application (SPA) surface area. In the process of discovery and exploitation, the browser actions, states and screen renders are collected for evidence and for post-scan review. The screenshots below show the browser tab after the XVWA scan has completed.

Browser Screenshot Sub-view



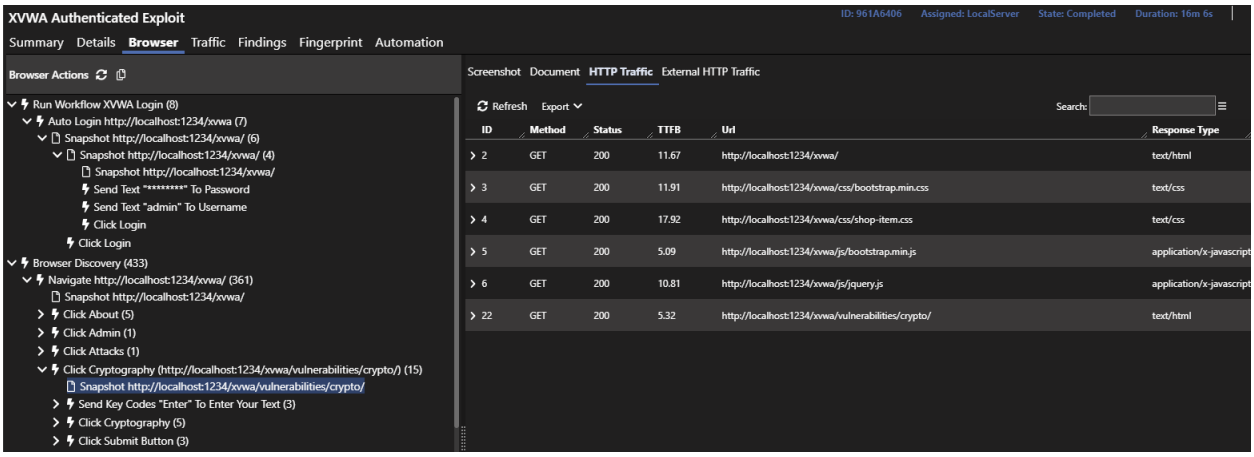
Browser Document Sub-view

The document sub-view shows the DOM state of the fully loaded page and not simply the HTTP response HTML.



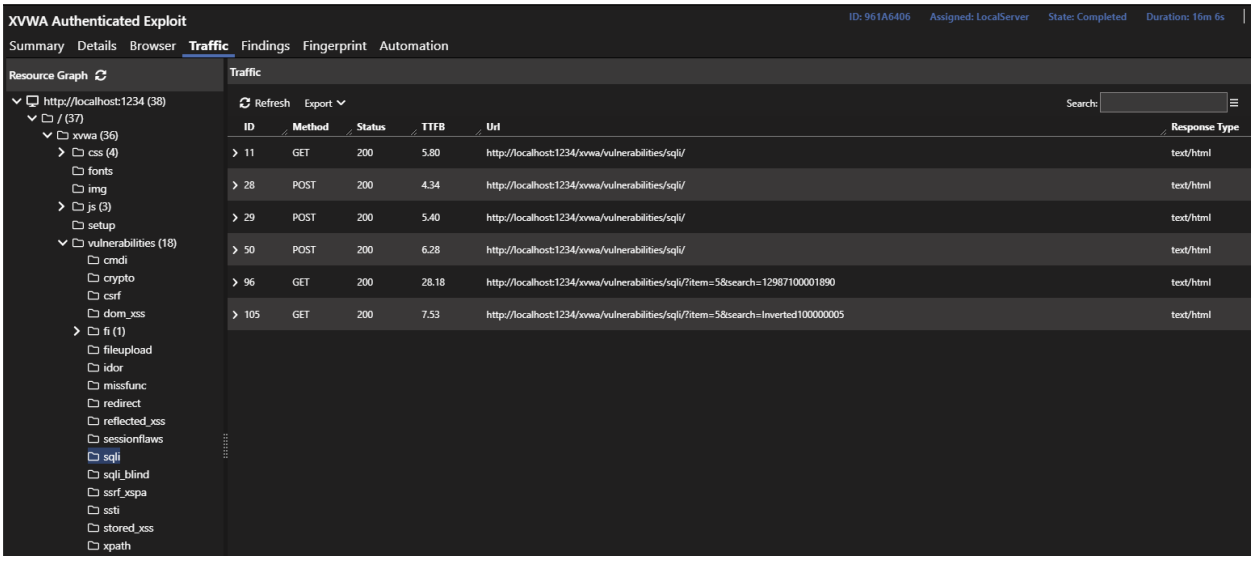
Browser Traffic Sub-view

The traffic view shows all the requests made in the retrieval and loading of the page.

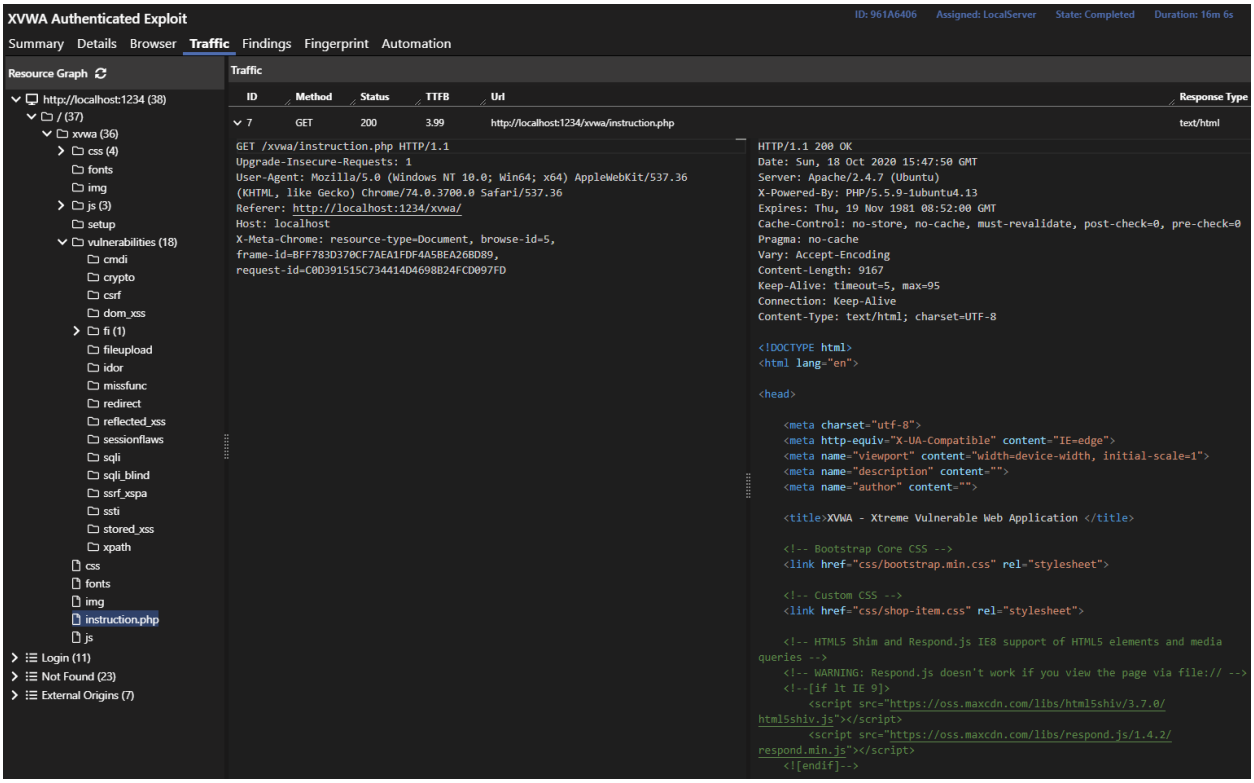


Traffic Tab

The traffic view shows a tree of the file and directory (or route) structure of the application URLs. Selecting a tree node shows the associated HTTP request/response pairs on the right-hand side of the UI.

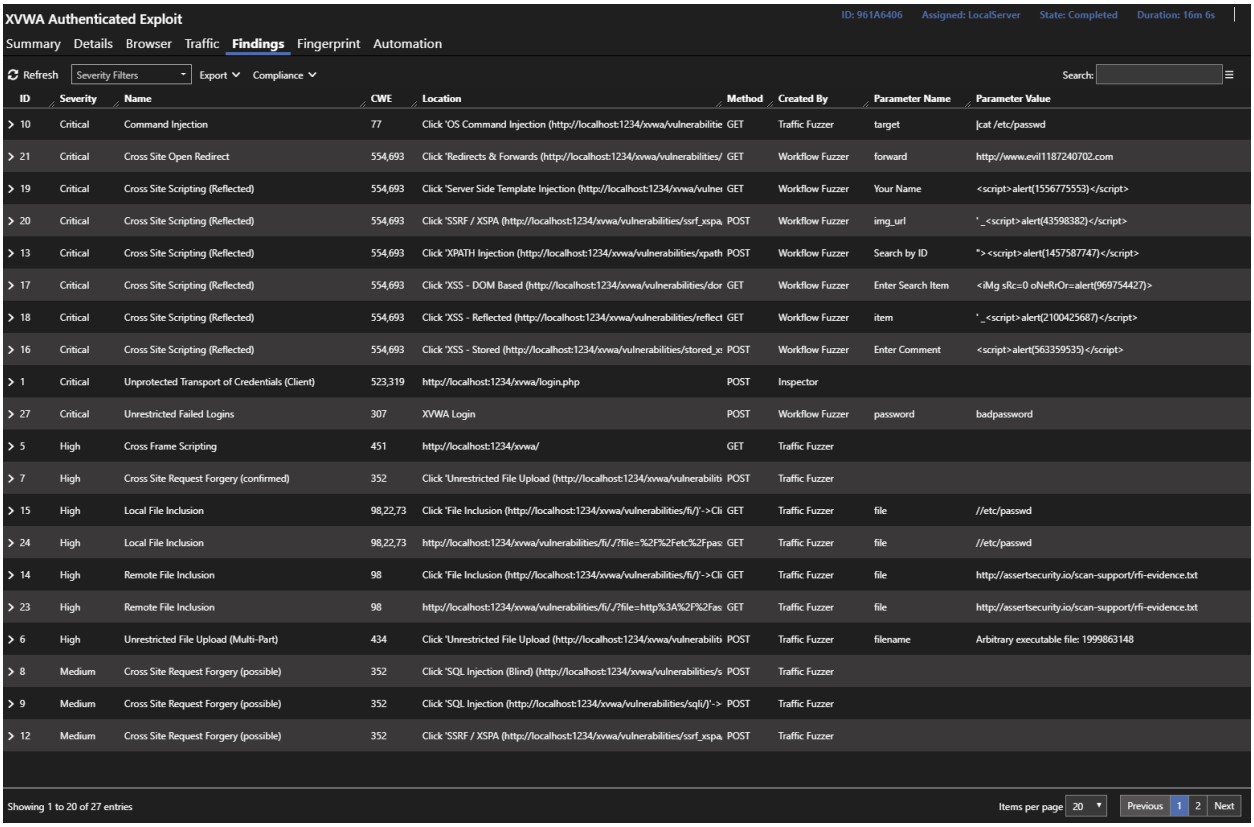


Individual Request/Response items are expandable to see the raw HTTP traffic



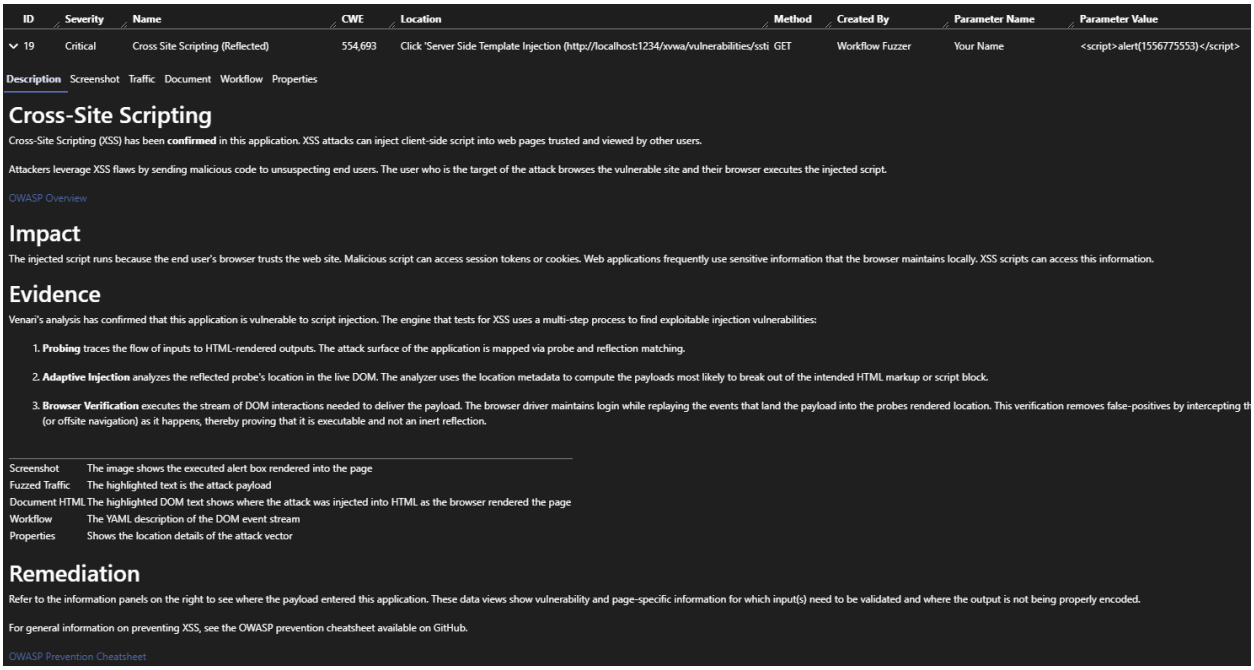
Findings Tab

The Findings tab provides details into each vulnerability or informational finding.



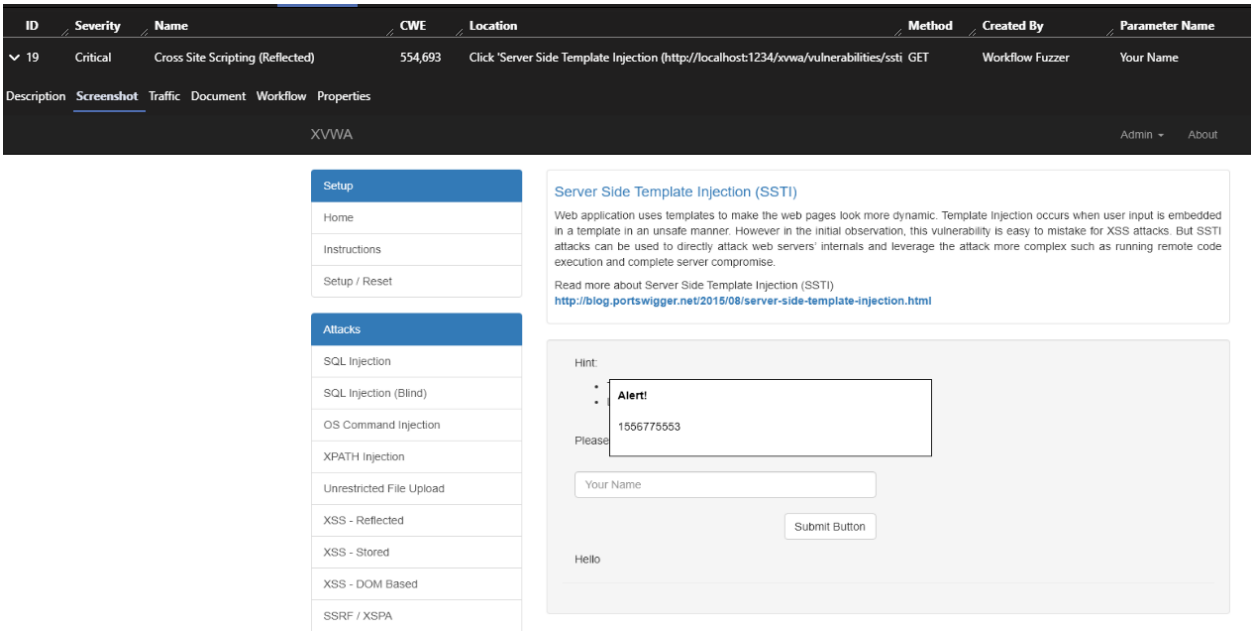
Expanding individual finding rows reveals evidence and full forensic details needed for remediation and triage. The screens shots below show the findings sub-views for a specific XSS vulnerability found on XVWA.

Finding Description Sub-View



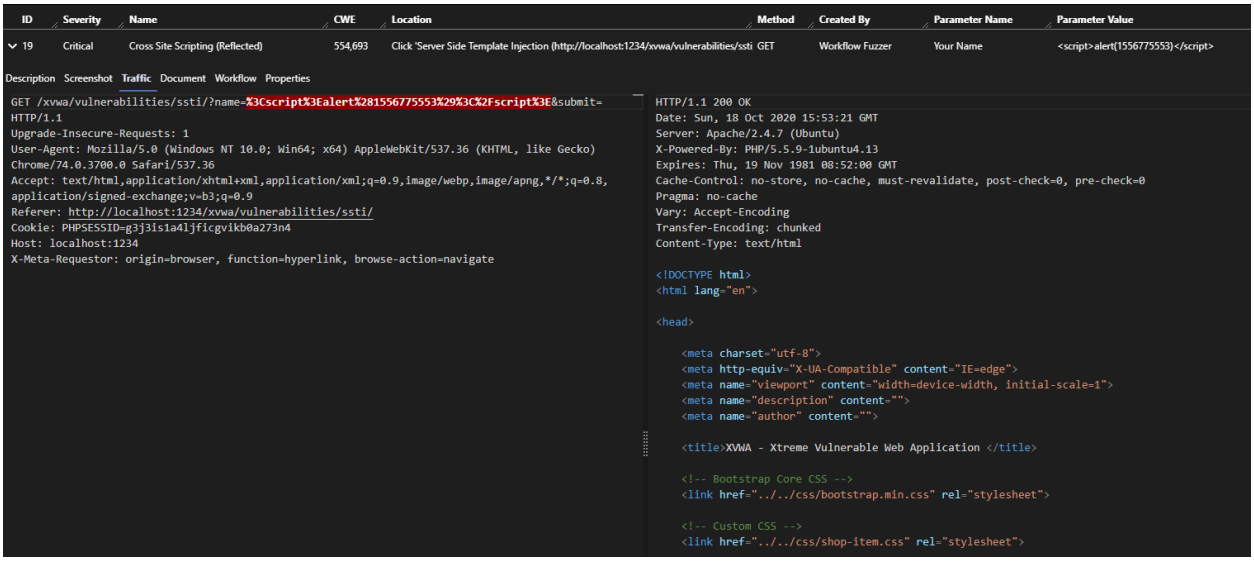
Finding Screenshot Sub-View

In this screenshot example, Venari captures the rendered page state while an injected script alert is popped. This evidence makes XSS testing immune to false positives. Rather than simply recognizing a reflection pattern, the actual script execution is detected for 100% proof of vulnerability.



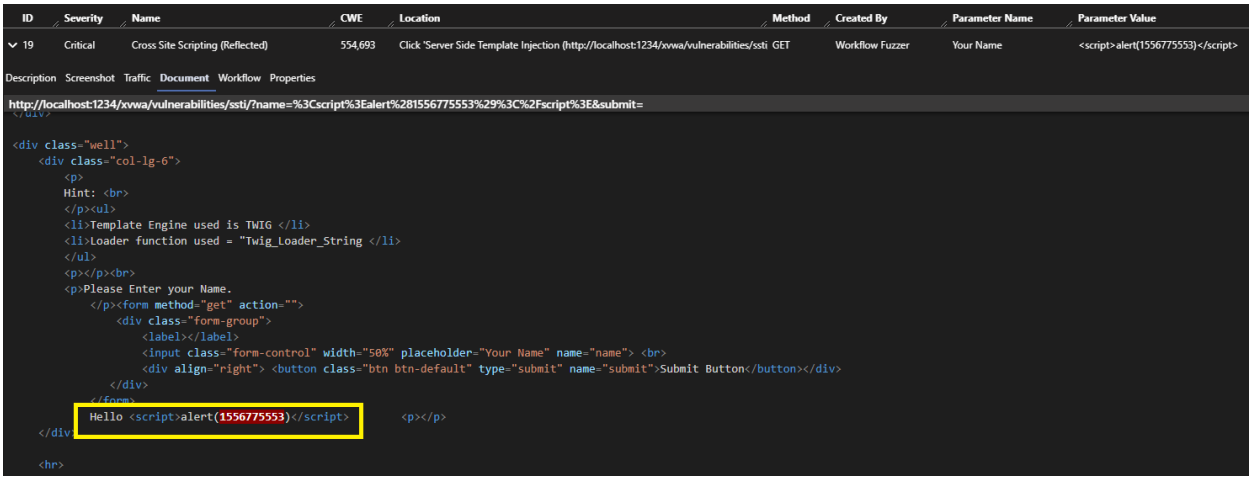
Finding Traffic Sub-View

The traffic sub-view shows the request payload with highlighted text for the attack portion of the request.



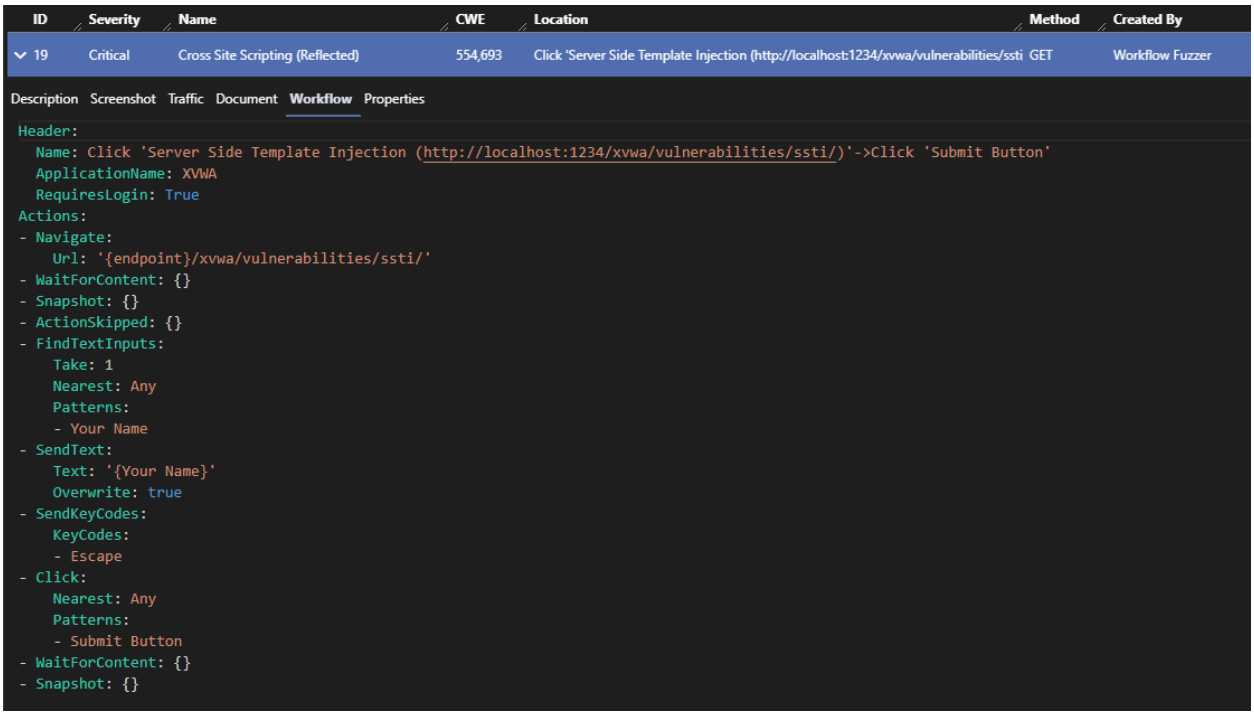
Finding Document Sub-View

In this finding example, the script alert is not reflected in the original response but is created in the changing DOM in response to some browser interaction (click, mouse over, keypress etc.). The document view shows the HTML DOM at the instant the 'reflection' is serialized into the page.



Finding Workflow Sub-View

The workflow sub-view shows the full sequence of browser actions needed to get the page into the state that allowed the attack. The steps are captured from the headless browser engine and expressed in YAML.



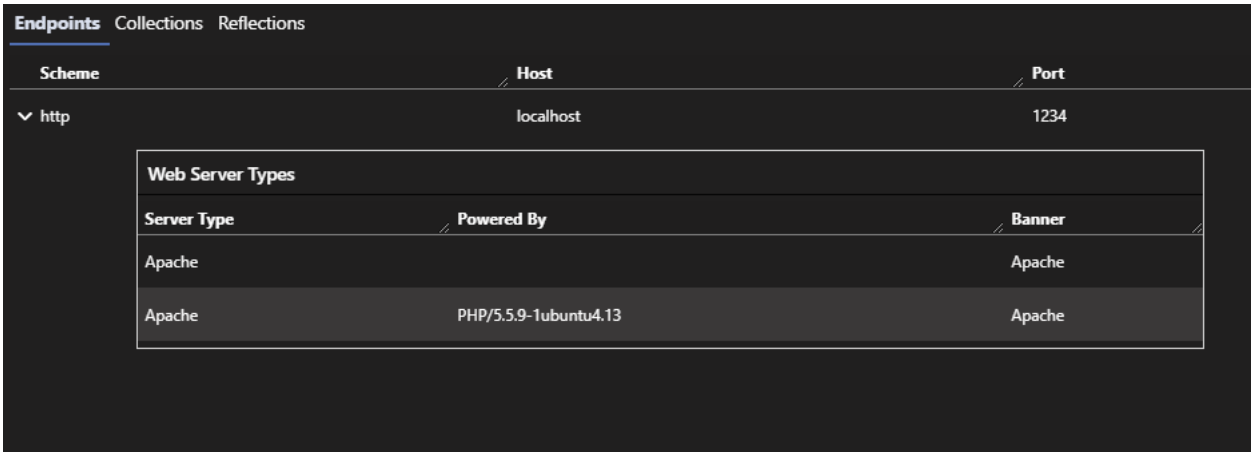
Finding Properties Sub-View

The properties sub-view shows specific information about the URL, parameters, browser actions and vulnerability taxonomy.

Fingerprint Tab

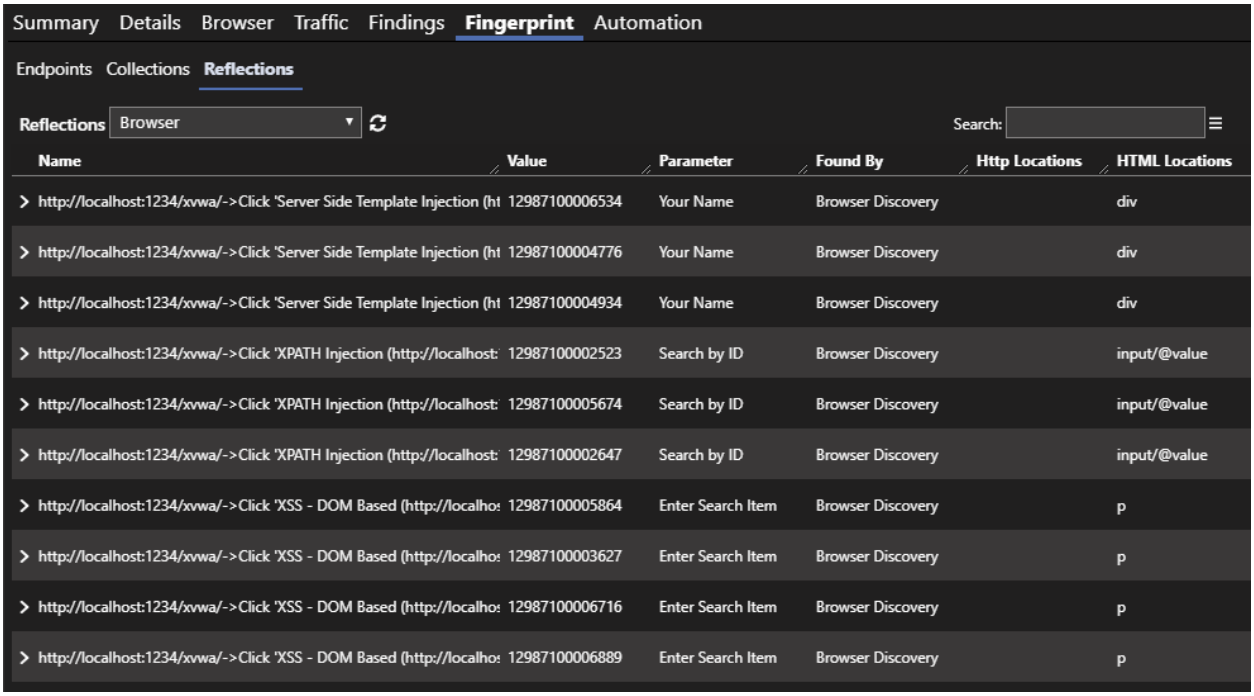
The Fingerprint tab has sub-views for endpoint information, reflections and various collections. The screenshots below show these sub-views.

Fingerprint Endpoints Sub-view

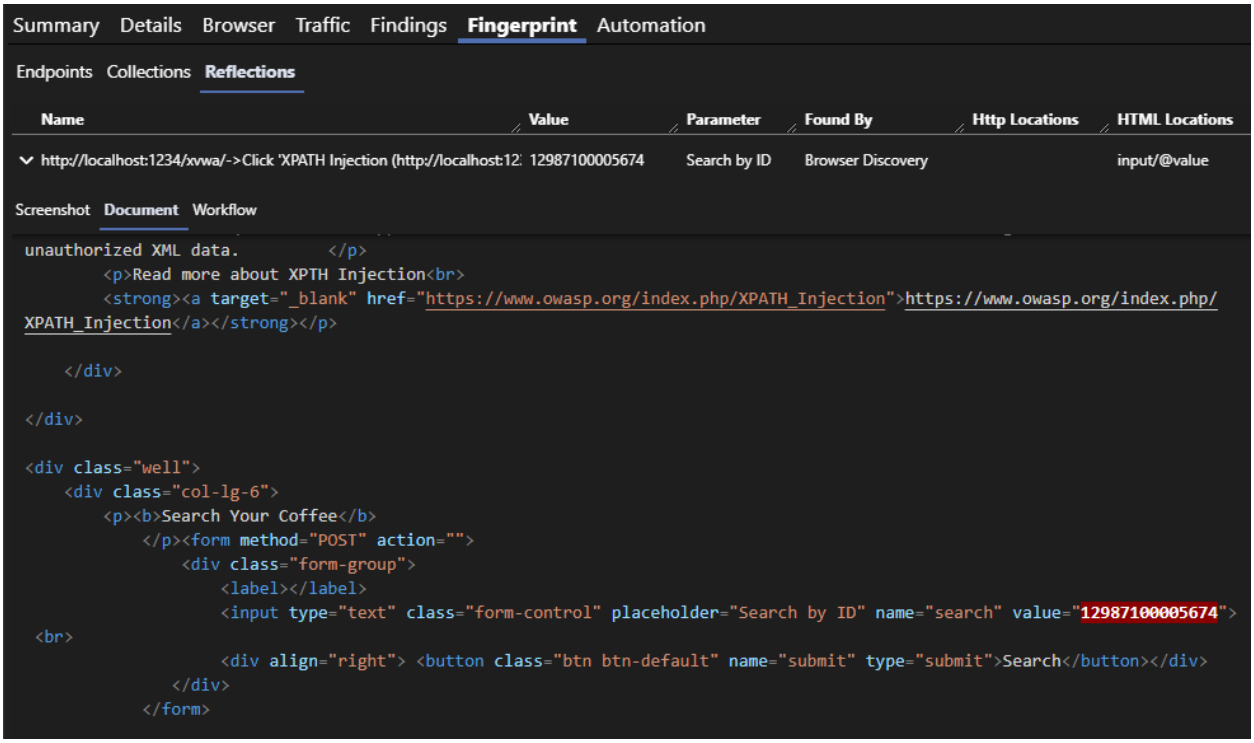


Fingerprint Reflections Sub-view

The reflections sub-view shows all locations where injected payloads were reflected regardless of whether those reflections were exploitable. Both traffic-based reflections (from the HTTP response) and browser-based reflections (from the changing DOM) are shown in this view.

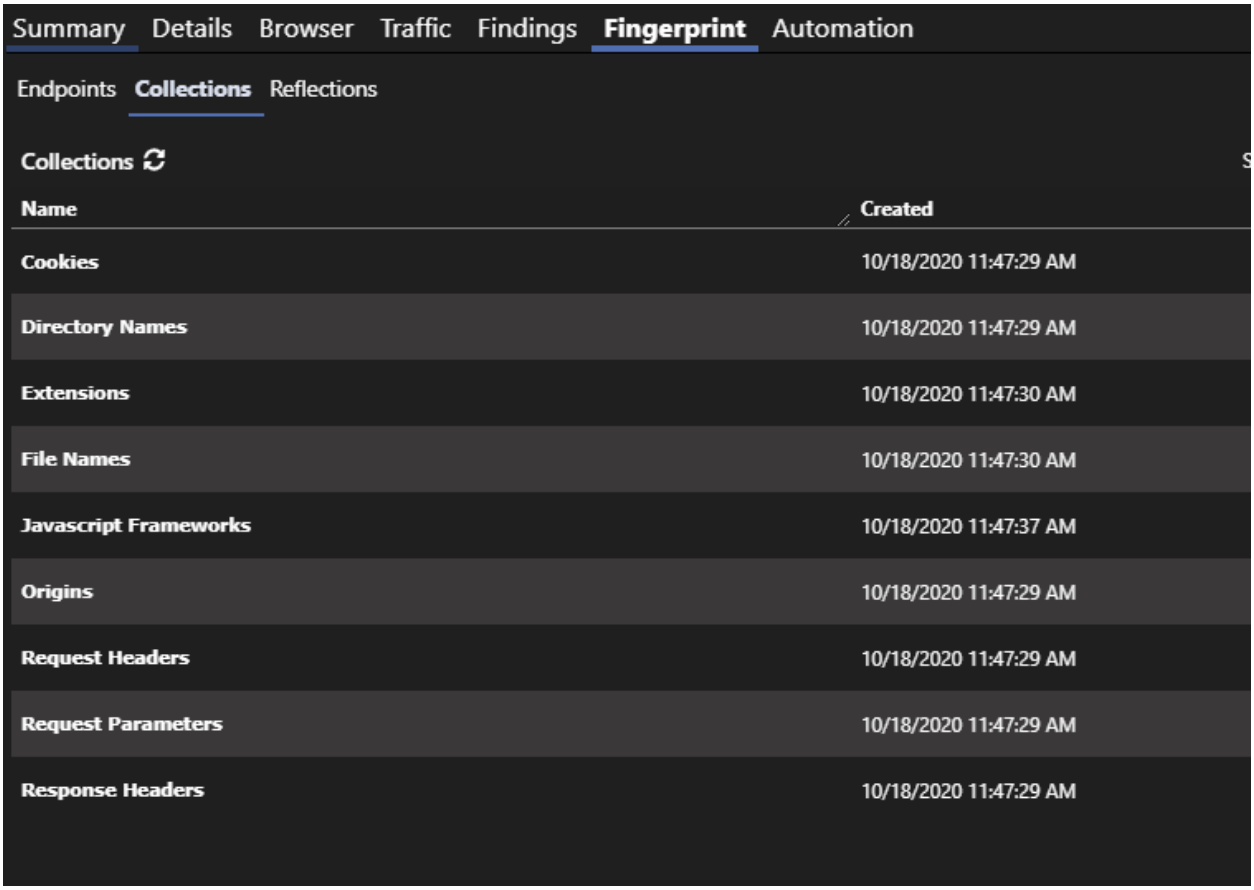


The example below shows a browser reflection into the DOM.



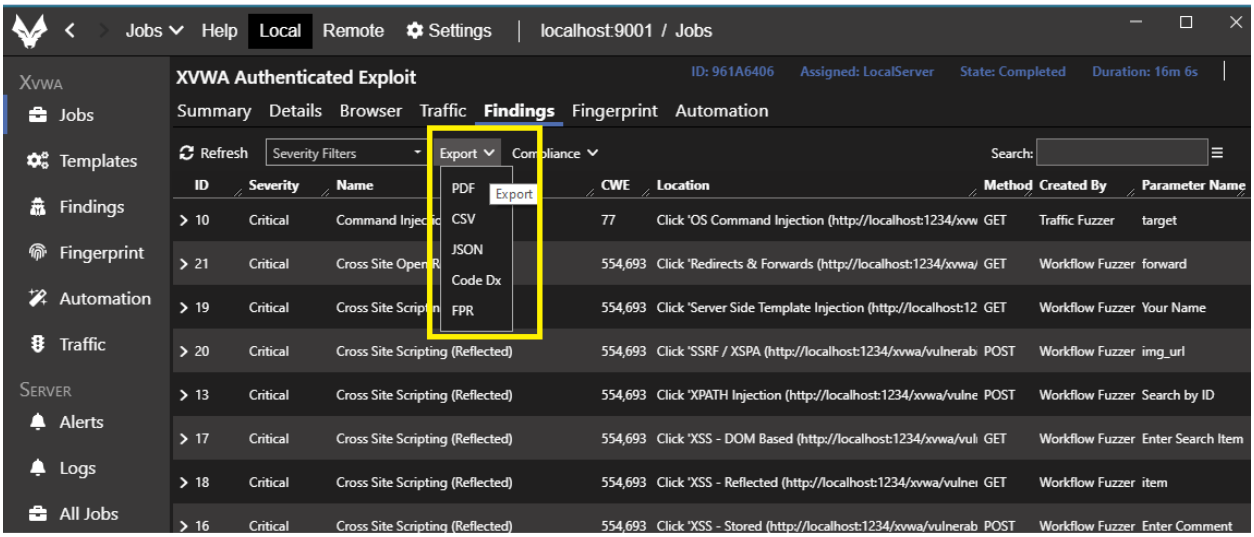
Fingerprint Collections Sub-view

The collections sub-view shows aggregated statistical information about the composition of the application URLS, parameters, external origins etc.



Exporting Results and Viewing Reports

From the findings tab, select the export dropdown and choose the type of export.



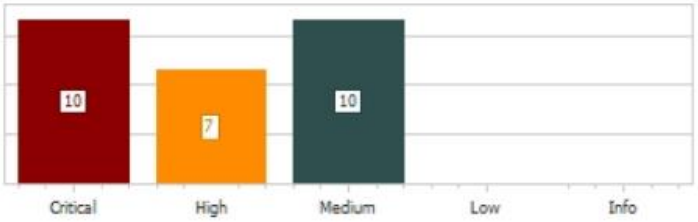
An example PDF report summary page is shown below

Venari Findings Report

Application - XVWA

Job:	XVWA Authenticated Exploit
ID:	961A6406
Created:	Sunday, October 18, 2020 3:46 PM
Duration:	16m 6s

Findings By Severity



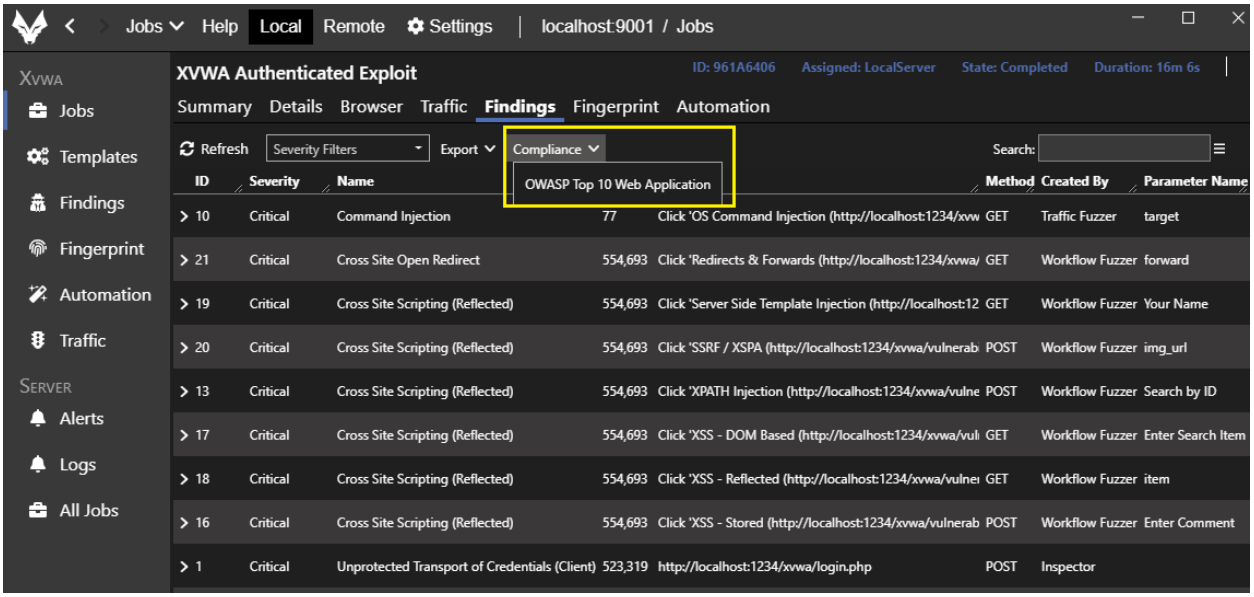
Critical	Command Injection	1
Critical	Cross Site Open Redirect	1
Critical	Cross Site Scripting (Reflected)	6
Critical	Unprotected Transport of Credentials (Client)	1
Critical	Unrestricted Failed Logins	1
High	Cross Frame Scripting	1
High	Cross Site Request Forgery (confirmed)	1
High	Local File Inclusion	2
High	Remote File Inclusion	2
High	Unrestricted File Upload (Multi-Part)	1
Medium	Cross Site Request Forgery (possible)	4
Medium	Directory Listing	3
Medium	Javascript CVE-2012-6708	1
Medium	Javascript CVE-2015-9251	1
Medium	Javascript CVE-2019-11358	1

Command Injection

Name:	Command Injection
Severity:	Critical
CWE:	77
Repro Steps:	Click 'OS Command Injection (http://localhost:1234/xvwa/vulnerabilities/cmdi/)' Click 'Submit Button'
Url:	http://localhost:1234/xvwa/vulnerabilities/cmdi/?target=%7Ccat%20%2Fetc%2Fpasswd
Parameter Name:	target
Parameter Value:	|cat /etc/passwd

Exporting Compliance Reports

From the findings tab, select the compliance dropdown to export compliance reports.



An example page from the OWASP Top 10 compliance report is show below

Venari Compliance Report

Application - XVWA

Job:	XVWA Authenticated Exploit
ID:	961A6406
Created:	Sunday, October 18, 2020 3:46 PM
Duration:	16m 6s

Broken Access Control

High	Cross Frame Scripting	Found (1)
High	Insecure Direct Object Reference	Not Found
Medium	Cross Site Request Forgery (possible)	Found (5)
Medium	Incomplete Logout Functionality	Not Found

Broken Authentication

Low	Session ID Unchanged After Authentication	Not Found

Cross-Site Scripting XSS

Critical	Cross Site Scripting (Reflected)	Found (7)
Low	Cross Site Scripting Weakness (Reflection in Response)	Not Found

Injection

Critical	Command Injection	Found (1)
Critical	Command Injection (Time-Based)	Not Found
Critical	PHP Code Injection	Not Found
Critical	PHP Code Injection (Time-Based)	Not Found
Critical	Server Side Include	Not Found
Critical	SQL Injection	Not Found
Critical	SQL Injection (Time-Based)	Not Found
High	Local File Inclusion	Found (2)
High	Remote File Inclusion	Found (2)

Onboarding from Existing Template Files

Venari can load pre-made job templates and workflows for use in scans. There are three types of files relevant to this guide.

1. Job Template. Example: exploit.jobtemplate.json. This file contains the main configuration for a scan. The template may contain auto-login credentials and in these cases, this is the only file that needs to be imported.

2. Login Workflow. Example: login.workflow.yaml. This file contains browser actions needed to achieve login. This file is only necessary in cases where auto-login did not work. Login workflows need to be imported and then linked to a job template. See the sections below for this two-step process.

3. Setup Workflow. Example: registeruser.workflow.yaml. This file contains browser actions that are pre-requisites needed for a useful scan. For example, imagine a staging application that is part of a CI/CD build pipeline. When the application is launched for test purposes there are no user accounts so one must be created. The setup workflow can drive the page actions needed to create a user account.

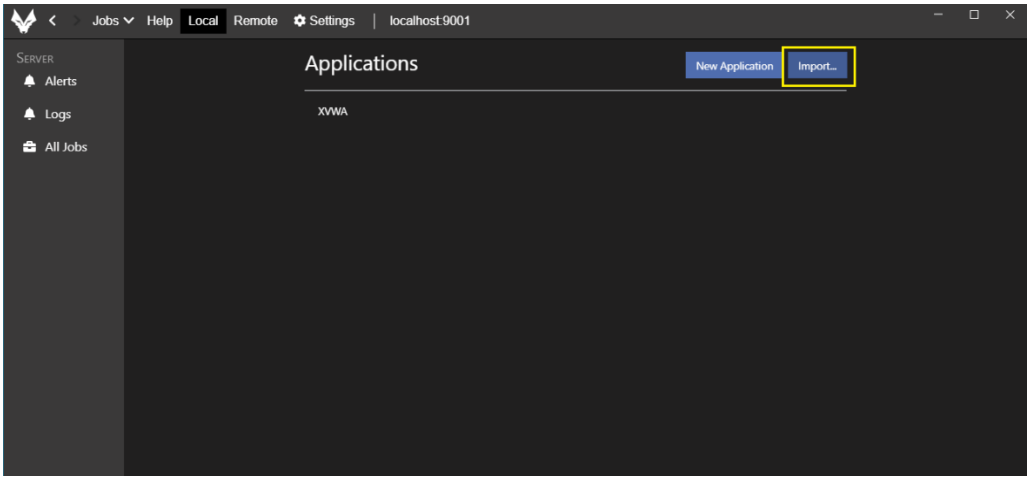
Sometimes onboarding an app is as simple as importing a single template file with auto-login information already embedded in the template. A more complex case would be an application that needs all three file types mentioned above, such as a site that needs a user account created and also has a non-trivial login, hence the need for the login workflow.

There are URLs and docker pull commands for vulnerable test sites at the end of this document. Each site has pre-created, downloadable job templates and setup or login workflows as well.

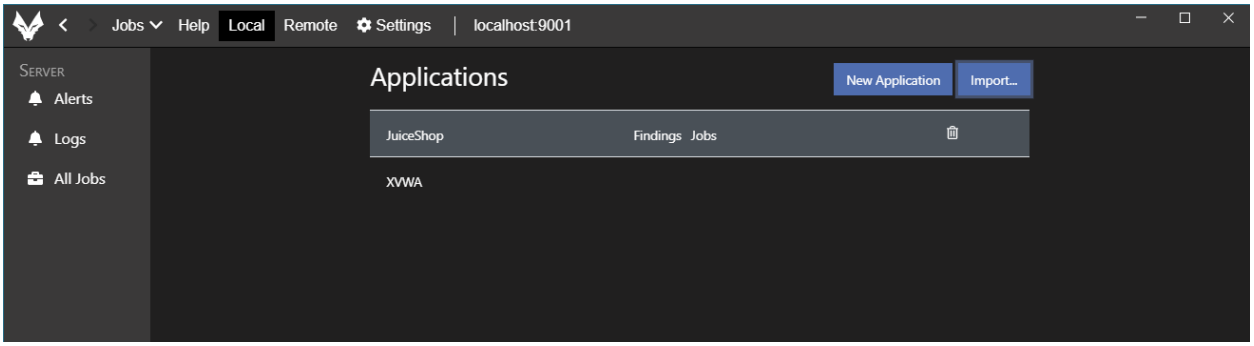
Scenario1: Import Job Template

Follow these steps to onboard an application from a template.

1. Find a test application in the tables at the end of the document
2. Download the template file.
3. Import the job template by clicking the Import button on the start page of the Venari UI and selecting the downloaded file.



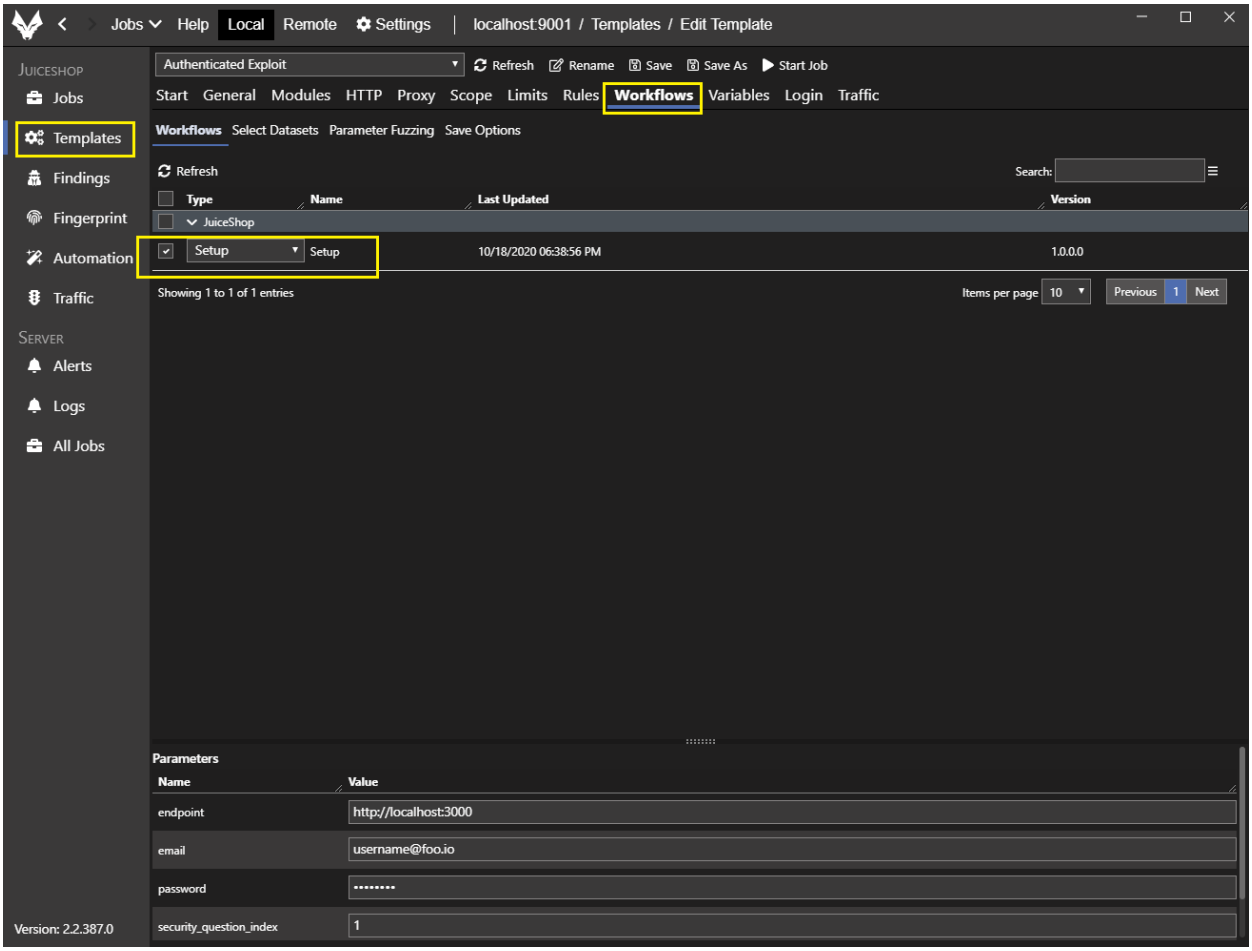
4. Observe that the application now appears on the start page



Scenario2: Import Job Template and Setup Workflow

Follow these steps to onboard an application from a template and link in a setup workflow. An example site that requires both is Juice Shop. See the table at the end of the document for download links.

1. Find the test application in the tables at the end of the document
2. Download the template file and the setup workflow file.
3. Import the job template by clicking the Import button on the start page of the Venari UI and selecting the downloaded file.
4. Click the Juice Shop row on the start page and observe the navigation to the template list
5. Click the automation icon on the left-hand side of the UI
6. Click the Import icon
7. Select the setup workflow file
8. Click the Templates icon on the left
9. Click the authenticated exploit template
10. Click the workflows tab
11. Observe that the setup template is checked and associated with the template



Publicly Available Test Applications

Venari has an optimization feature that tunes scan template settings based on fingerprinted information about the site, such as technologies, code constructs and versions. The optimizer has built-in rules to make evaluation easier on the test sites listed below. The generated templates will include special setup workflows which register user accounts (if needed) or reset databases to clean state for certain Docker images.

To use the optimizer to set up one of the applications below, follow these steps:

1. Click the ‘New Application’ button
2. Enter a name and the start URL
3. Uncheck the authentication box
4. Click OK
5. Click the play icon for the ‘Exploit’ template row in the grid

The optimizer will do two things:

1. Start a scan and apply optimization rules.
2. Create an optimized template for use in later scans. The name will be 'Optimized: NAME'

The optimized template will include any needed auto-login credentials or special login workflows and also any setup workflows, such as registering a new user or resetting a test database.

The Optimized template will provide the best point and shoot quick start experience to allow the user to see the results that Venari produces and to review the findings and various data views.

Publicly Available Test Applications (Docker)

The table below summarizes freely downloadable Docker images that contain intentionally vulnerable web applications. These containers can be used as a quick method to legally test Venari's scan capabilities and features.

<table>
<tr><th>Application</th><th>Notes</th></tr>
<tr><td>WebGoat 8</td><td>
<table>
<tr><td>Docker Pull Command</td><td>docker pull webgoat/webgoat-8.0</td></tr>
<tr><td>Docker Run Command</td><td>docker run -d -p 8080:8080 -t webgoat/webgoat-8.0</td></tr>
<tr><td>Start URL</td><td>http://localhost:8080/WebGoat</td></tr>
</table>
</td></tr>
<tr><td>Juice Shop</td><td>
<table>
<tr><td>Docker Pull Command</td><td>docker pull bkimminich/juice-shop</td></tr>
<tr><td>Docker Run Command</td><td>docker run -d -p 3000:3000 bkimminich/juice-shop</td></tr>
<tr><td>Start URL</td><td>http://localhost:3000</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>bWapp</td><td>
<table>
<tr><td>Docker Pull Command</td><td>docker pull raesene/bwapp</td></tr>
<tr><td>Docker Run Command</td><td>docker run -d -p 80:80 raesene/bwapp</td></tr>
<tr><td>Start URL</td><td>http://localhost/</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Multillidae</td><td>
<table>
<tr><td>Docker Pull Command</td><td>docker pull szsecurity/mutillidae</td></tr>
<tr><td>Docker Run Command</td><td>docker run -d -p 80:80 szsecurity/mutillidae</td></tr>
<tr><td>Start URL</td><td>http://localhost</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>DVWA</td><td>
<table>
<tr><td>Docker Pull Command</td><td>docker pull vulnerables/web-dvwa</td></tr>
</table>
</td></tr>
</table>

	Docker Run Command: docker run --rm -it -p 80:80 vulnerables/web-dvwa Start URL: http://localhost/index.php
DSVW	Docker Pull Command: docker pull appsecco/dsvw Docker Run Command: docker run -d -p 1235:8000 -it appsecco/dsvw Start URL: http://localhost:1235
XVWA	Docker Pull Command: docker pull bitnetsecdave/xvwa Docker Run Command: docker run -p 1234:80 -it bitnetsecdave/xvwa Start URL: http://localhost:1234
Hackazon	Docker Pull Command: docker pull mutzel/all-in-one-hackazon:postinstall Docker Run Command: docker run -d -p 80:80 mutzel/all-in-one-hackazon:postinstall supervisord -n Start URL: http://127.0.0.1/
WAVSEP 1.5	Docker Pull Command: docker pull owaspvwad/wavsep Docker Run Command: docker run -d -p 8080:8080 -i -t owaspvwad/wavsep Start URL: http://localhost:8080/wavsep/index-active.jsp

Publicly Available Test Applications (Internet)

The table below summarizes public-facing, intentionally vulnerable web applications. These sites are deployed for the purpose of tool evaluation and/or AppSec education (hacking labs).

Application	**Notes**
Google Firing Range	Google Firing Range is a testbed site that exposes XSS vulnerabilities of almost every known variety.

	[Start URL] https://public-firing-range.appspot.com/
Testfire	Testfire (Altoro Mutual) is a fake banking application with vulnerabilities baked in. [Start URL] http://demo.testfire.net/
TestSparker	[Start URL] http://aspnet.testsparker.com
VulnWeb	[Start URL] http://testphp.vulnweb.com/
WebScantest	[Start URL] http://www.webscantest.com/login.php