

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

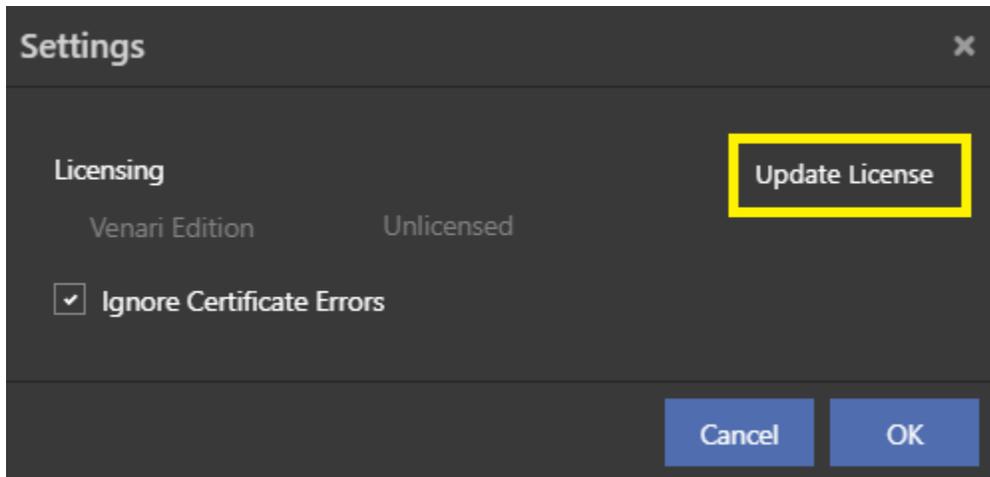
- Advanced configuration
- Re-test and triage
- Web API endpoint onboarding
- Manually Creating a login workflow*

* *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:*

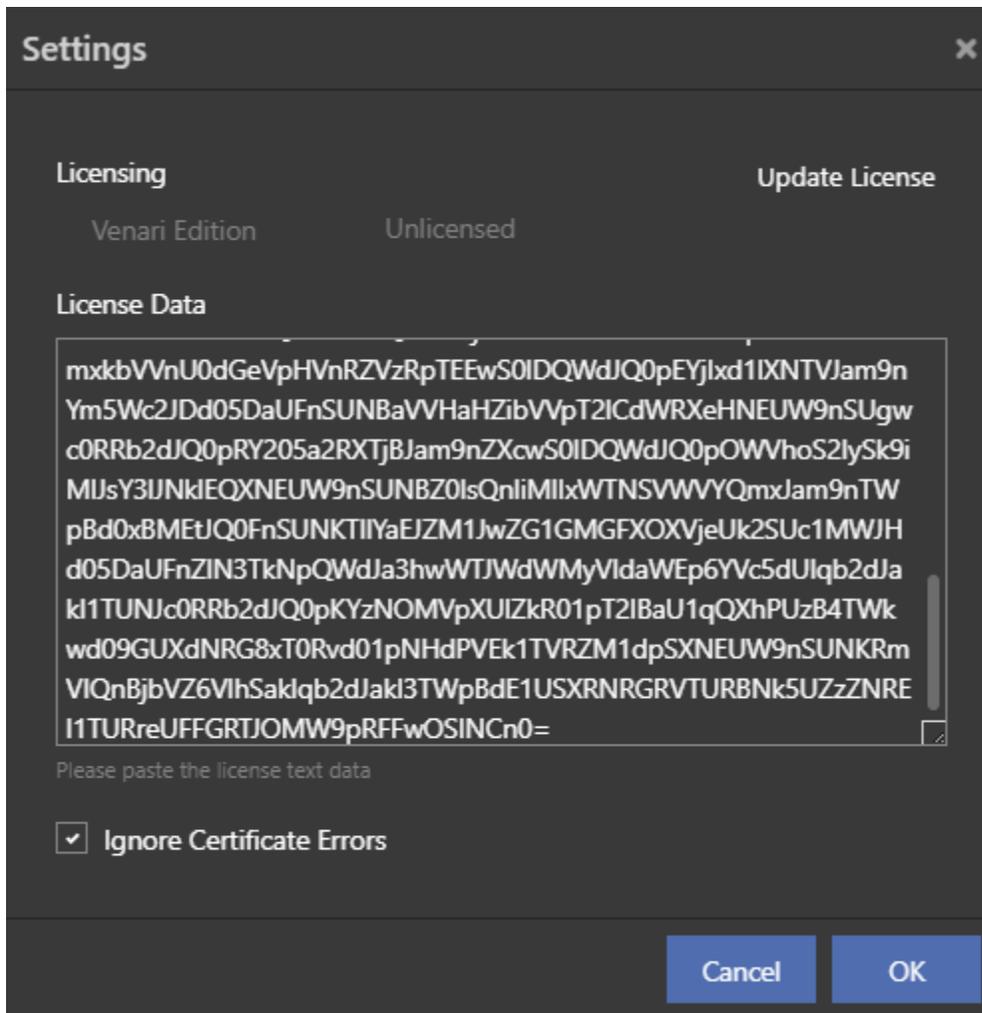
<https://assertsecurity.io/venaridocs/quick-starts/tips-and-tricks/record-login-workflow/record-login-workflow/>

Installation and License Setup

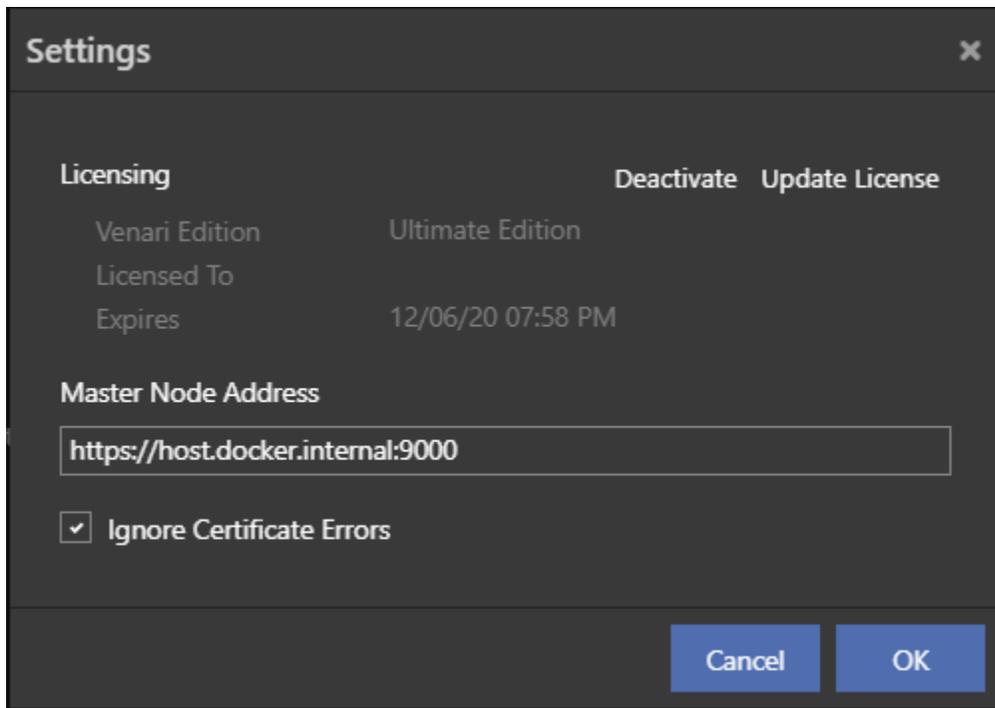
Run the Venari installer. You will see the UI pop up with an active dialog box for entering the license token. Click 'Update License'.



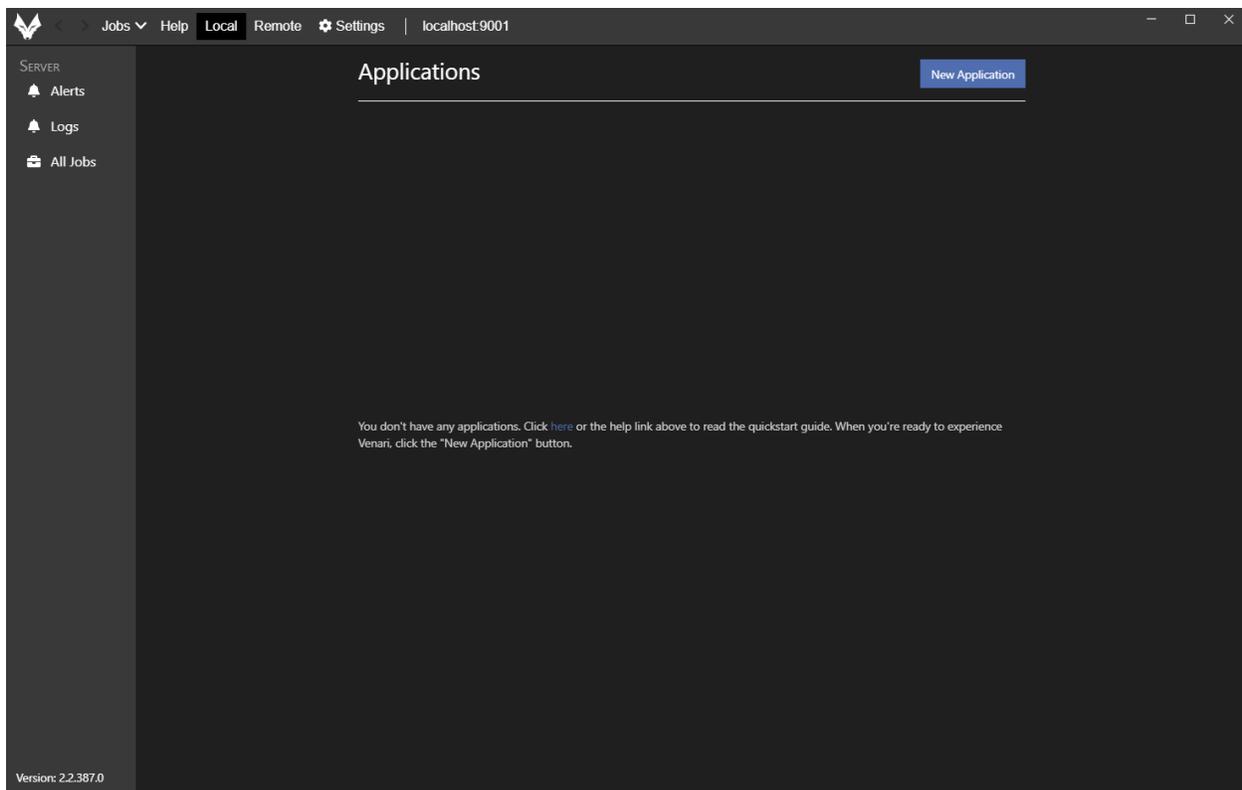
Enter the license text from your evaluation email and click the OK button.



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.

XVWA

Login About

Setup

- Home
- Instructions
- Setup / Reset

Attacks

- SQL Injection
- SQL Injection (Blind)
- OS Command Injection
- XPATH Injection
- Unrestricted File Upload
- XSS - Reflected
- XSS - Stored
- XSS - DOM Based
- SSRF / XSPA
- File Inclusion
- Session Flaws
- Insecure Direct Object Reference
- Missing Functional Access Control
- CSRF
- Cryptography
- Redirects & Forwards

Xtreme Vulnerable Web Application (XVWA)

XVWA is a badly coded web application written in PHP/MySQL that helps security enthusiasts to learn application security. It's not advisable to host this application online as it is designed to be "Xtremely Vulnerable". We recommend hosting this application in local/controlled environment and sharpening your application security ninja skills with any tools of your own choice. It's totally legal to break or hack into this. The idea is to evangelize web application security to the community in possibly the easiest and fundamental way. Learn and acquire these skills for good purpose. How you use these skills and knowledge base is not our responsibility.

XVWA is designed to understand following security issues.

- SQL Injection – Error Based
- SQL Injection – Blind
- OS Command Injection
- XPATH Injection
- Unrestricted File Upload
- Reflected Cross Site Scripting
- Stored Cross Site Scripting
- DOM Based Cross Site Scripting
- Server Side Request Forgery / Cross Site Port Attacks(CSRF/XSPA)
- File Inclusion
- Session Issues
- Insecure Direct Object Reference
- Missing Functional Level Access Control
- Cross Site Request Forgery (CSRF)
- Cryptography
- Unvalidated Redirect & Forwards
- Server Side Template Injection

Good Luck and Happy Hacking!

Copyright
This work is licensed under GNU GENERAL PUBLIC LICENSE Version 3
To view a copy of this license, visit <http://www.gnu.org/licenses/gpl-3.0.txt>

Disclaimer
Do not host this application on live or production environment. XVWA is totally vulnerable application and giving online/live access of this application could lead to complete compromise of your system. We are not responsible for any such bad incidents. Stay safe !

Onboarding the Test Application

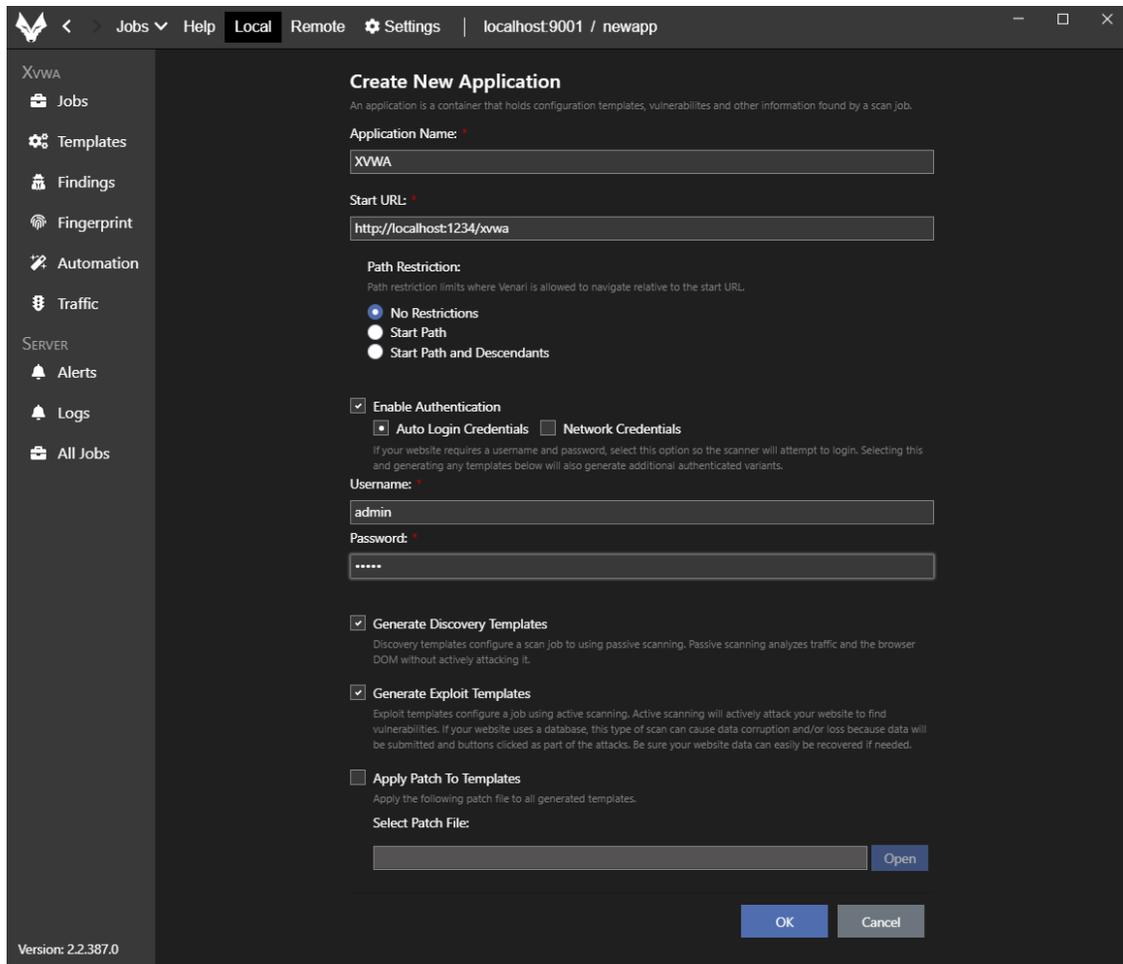
'Onboarding' an application means using the Venari UI to create a named workspace and a default set of job templates that are used in future scans. The named workspaces are referred to as 'Applications' in the UI. The steps below show the actions needed to onboard the XVWA application from the previous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

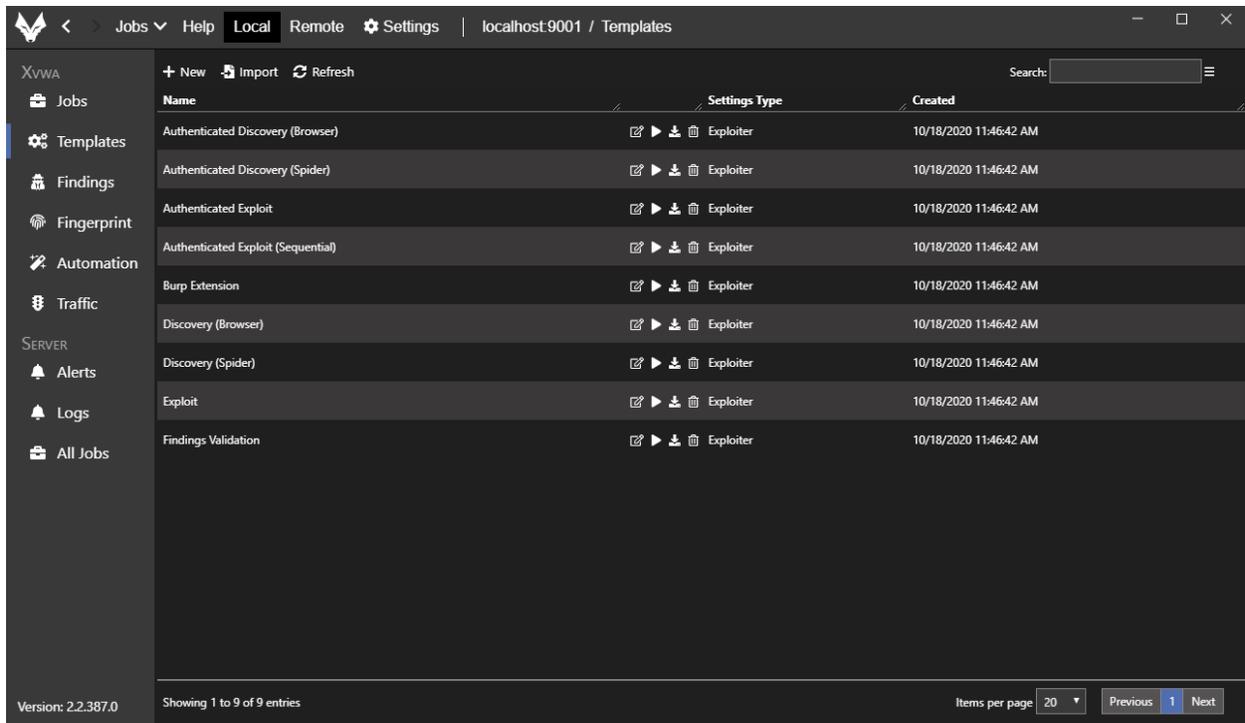
1. Click the 'New Application' button on the start screen.
2. Type XVWA into the Application Name field
3. Type <http://localhost:1234/xvwa> into the start URL field
4. Select the radio button titled 'Start Path and Descendants'
5. Type 'admin' into the username field
6. Type 'admin' into the password 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.

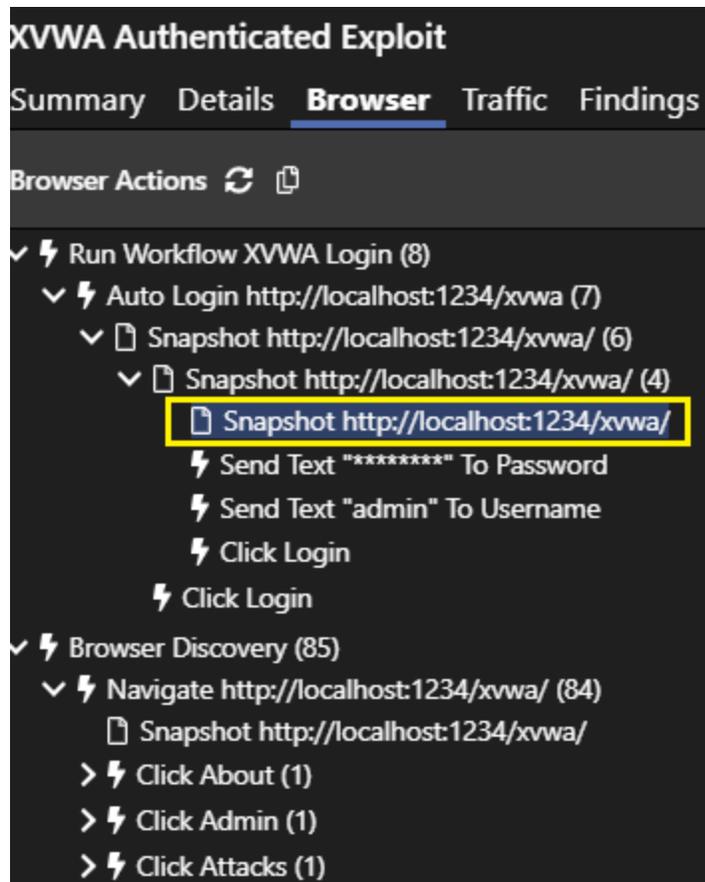
XVWA XVWA Authenticated Exploit ID: E786089B Assigned: LocalServer

Summary Details Browser Traffic Findings Fingerprint Automation

Progress	Name	Deferred	Ready	Acquired	Running	Completed	Skipped	Cancelled	Total	
▼ Discovery										
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Browser Discovery	0	0	0	1	0	0	0	1	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Content Parser	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Force Browser	0	0	0	0	0	0	0	0	
▼ Traffic										
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Requestor	0	0	0	1	0	0	0	1	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Traffic Player	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Workflow Player	0	0	0	0	0	0	0	0	
▼ Passive										
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Inspector	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Server Properties	0	Progress update... Running login workflow...							0
▼ Active										
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Fingerprinter	0	Running login workflow...							0
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Fuzz Generator	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	TLS	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Traffic Fuzzer	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Traffic Fuzzer (Ordered)	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Workflow Fuzzer	0	0	0	0	0	0	0	0	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Workflow Prober	0	0	0	0	0	0	0	0	
▼ Terminal										
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Discovery Controller	0	1	0	0	0	0	0	1	
<div style="width: 100%; height: 10px; background-color: #ccc;"></div>	Finisher	0	1	0	0	0	0	0	1	

Version: 2.2.387.0

- The scan is now running. Once the progress dialog disappears you can confirm 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

The screenshot shows the XVWA Authenticated Exploit interface. The top bar includes the title 'XVWA Authenticated Exploit', ID '961A6406', 'Assigned: LocalServer', 'State: Completed', and 'Duration: 16m 6s'. Below the title bar are tabs for 'Summary', 'Details', 'Browser', 'Traffic', 'Findings', 'Fingerprint', and 'Automation'. The 'Browser' tab is active, showing 'Browser Actions' on the left and a 'Screenshot' sub-view on the right. The 'Browser Actions' list includes: 'Run Workflow XVWA Login (8)', 'Auto Login http://localhost:1234/xvwa/ (7)', 'Snapshot http://localhost:1234/xvwa/ (6)', 'Snapshot http://localhost:1234/xvwa/ (4)', 'Snapshot http://localhost:1234/xvwa/ (4)', 'Send Text ***** To Password', 'Send Text "admin" To Username', 'Click Login', 'Click Login', 'Browser Discovery (433)', 'Navigate http://localhost:1234/xvwa/ (361)', 'Snapshot http://localhost:1234/xvwa/ (361)', 'Click About (5)', 'Click Admin (1)', 'Click Attacks (1)', 'Click Cryptography (http://localhost:1234/xvwa/vulnerabilities/crypto/) (15)', 'Snapshot http://localhost:1234/xvwa/vulnerabilities/crypto/ (15)', 'Send Key Codes "Enter" To Enter Your Text (3)', 'Click Cryptography (5)', 'Click Submit Button (3)', 'Click CSRF (http://localhost:1234/xvwa/vulnerabilities/csrf/) (15)', 'Click File Inclusion (http://localhost:1234/xvwa/vulnerabilities/fi/) (5)', 'Click Home (http://localhost:1234/xvwa/) (1)', 'Click Insecure Direct Object Reference (http://localhost:1234/xvwa/vulnerabilities/idor/) (1)', 'Click Instructions (http://localhost:1234/xvwa/instruction.php) (3)', 'Click Missing Functional Access Control (http://localhost:1234/xvwa/vulnerabilities/mfac/) (1)', 'Click OS Command Injection (http://localhost:1234/xvwa/vulnerabilities/cmd/) (1)', 'Click Redirects & Forwards (http://localhost:1234/xvwa/vulnerabilities/redirect/) (1)', 'Click Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/ssti/) (1)', 'Click Session Flaws (http://localhost:1234/xvwa/vulnerabilities/sessionflaws/) (3)', 'Click Setup (1)', 'Click SQL Injection (Blind) (http://localhost:1234/xvwa/vulnerabilities/sql_injection_blind/) (1)', 'Click SQL Injection (http://localhost:1234/xvwa/vulnerabilities/sql_injection/) (18)', 'Click SSRF / XSPA (http://localhost:1234/xvwa/vulnerabilities/ssrf_xspa/) (15)', 'Click Unrestricted File Upload (http://localhost:1234/xvwa/vulnerabilities/ufu/) (1)', 'Click XPATH Injection (http://localhost:1234/xvwa/vulnerabilities/xpath/) (15)', 'Click XSS - DOM Based (http://localhost:1234/xvwa/vulnerabilities/xss_dom/) (33)', 'Click XSS - Reflected (http://localhost:1234/xvwa/vulnerabilities/xss_reflected/) (1)'. The 'Screenshot' sub-view shows the XVWA application with a 'Setup' section containing a 'Cryptography' page and an 'Attacks' section containing a 'Enter your text here' form.

Browser Document Sub-view

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

XVWA Authenticated Exploit ID: 961A6406 Assigned: LocalServer State: Completed Duration: 16m 6s

Summary Details **Browser** Traffic Findings Fingerprint Automation

Browser Actions

- Run Workflow XVWA Login (8)
 - Auto Login http://localhost:1234/xvwa (7)
 - Snapshot http://localhost:1234/xvwa/ (6)
 - Snapshot http://localhost:1234/xvwa/ (4)
 - Send Text "*****" To Password
 - Send Text "admin" To Username
 - Click Login
 - Click Login
- Browser Discovery (433)
 - Navigate http://localhost:1234/xvwa/ (361)
 - Snapshot http://localhost:1234/xvwa/
 - Click About (5)
 - Click Admin (1)
 - Click Attacks (1)
 - Click Cryptography (http://localhost:1234/xvwa/vulnerabilities/crypto/) (15)
 - Snapshot http://localhost:1234/xvwa/vulnerabilities/crypto/
 - Send Key Codes "Enter" To Enter Your Text (3)
 - Click Cryptography (5)
 - Click Submit Button (3)
 - Click CSRF (http://localhost:1234/xvwa/vulnerabilities/csrf/) (15)
 - Click File Inclusion (http://localhost:1234/xvwa/vulnerabilities/fi/) (5)
 - Click Home (http://localhost:1234/xvwa/) (1)
 - Click Insecure Direct Object Reference (http://localhost:1234/xvwa/vulnerabilities/idor/) (1)
 - Click Redirects & Forwards (http://localhost:1234/xvwa/vulnerabilities/redirect/) (1)
 - Click Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/sssti/) (3)
 - Click Session Flaws (http://localhost:1234/xvwa/vulnerabilities/sessionflaws/) (3)

Screenshot Document HTTP Traffic External HTTP Traffic

http://localhost:1234/xvwa/vulnerabilities/crypto/

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="">
  <meta name="author" content="">
  <title>XVWA - Xtreme Vulnerable Web Application </title>
  <!-- Bootstrap Core CSS -->
  <link href="../../css/bootstrap.min.css" rel="stylesheet">
  <!-- Custom CSS -->
  <link href="../../css/shop-item.css" rel="stylesheet">
  <!-- HTML5 Shim and Respond.js IEB support of HTML5 elements and media queries -->
  <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
  <!-- [if lt IE 9] -->
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
  <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
  </endif-->
</head>
<body>
  <!-- Navigation -->
  <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="container">
      <!-- Brand and toggle get grouped for better mobile display -->
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
          <span class="sr-only">Toggle navigation</span>
      </div>
    </div>
  </nav>

```

Browser Traffic Sub-view

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

XVWA Authenticated Exploit ID: 961A6406 Assigned: LocalServer State: Completed Duration: 16m 6s

Summary Details **Browser** Traffic Findings Fingerprint Automation

Browser Actions

- Run Workflow XVWA Login (8)
 - Auto Login http://localhost:1234/xvwa (7)
 - Snapshot http://localhost:1234/xvwa/ (6)
 - Snapshot http://localhost:1234/xvwa/ (4)
 - Send Text "*****" To Password
 - Send Text "admin" To Username
 - Click Login
 - Click Login
 - Browser Discovery (433)
 - Navigate http://localhost:1234/xvwa/ (361)
 - Snapshot http://localhost:1234/xvwa/
 - Click About (5)
 - Click Admin (1)
 - Click Attacks (1)
 - Click Cryptography (http://localhost:1234/xvwa/vulnerabilities/crypto/) (15)
 - Snapshot http://localhost:1234/xvwa/vulnerabilities/crypto/
 - Send Key Codes "Enter" To Enter Your Text (3)
 - Click Cryptography (5)
 - Click Submit Button (3)

Screenshot Document **HTTP Traffic** External HTTP Traffic

Refresh Export Search:

ID	Method	Status	TTFB	Url	Response Type
> 2	GET	200	11.67	http://localhost:1234/xvwa/	text/html
> 3	GET	200	11.91	http://localhost:1234/xvwa/css/bootstrap.min.css	text/css
> 4	GET	200	17.92	http://localhost:1234/xvwa/css/shop-item.css	text/css
> 5	GET	200	5.09	http://localhost:1234/xvwa/js/bootstrap.min.js	application/javascript
> 6	GET	200	10.81	http://localhost:1234/xvwa/js/jquery.js	application/javascript
> 22	GET	200	5.32	http://localhost:1234/xvwa/vulnerabilities/crypto/	text/html

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.

XVWA Authenticated Exploit ID: 961A6406 Assigned: LocalServer State: Completed Duration: 16m 6s

Summary Details Browser **Traffic** Findings Fingerprint Automation

Resource Graph

- http://localhost:1234 (38)
 - / (37)
 - xvwa (36)
 - css (4)
 - fonts
 - img
 - js (3)
 - setup
 - vulnerabilities (18)
 - cmdi
 - crypto
 - csrf
 - dom_xss
 - fi (1)
 - fileupload
 - idor
 - missfunc
 - redirect
 - reflected_xss
 - sessionflaws
 - sqli
 - sqli_blind
 - ssrf_xspa
 - sssti
 - stored_xss
 - xpath

ID	Method	Status	TTFB	Url	Response Type
> 11	GET	200	5.80	http://localhost:1234/xvwa/vulnerabilities/sqli/	text/html
> 28	POST	200	4.34	http://localhost:1234/xvwa/vulnerabilities/sqli/	text/html
> 29	POST	200	5.40	http://localhost:1234/xvwa/vulnerabilities/sqli/	text/html
> 50	POST	200	6.28	http://localhost:1234/xvwa/vulnerabilities/sqli/	text/html
> 96	GET	200	28.18	http://localhost:1234/xvwa/vulnerabilities/sqli/?item=5&search=12987100001890	text/html
> 105	GET	200	7.53	http://localhost:1234/xvwa/vulnerabilities/sqli/?item=5&search=Inverted100000005	text/html

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

XVWA Authenticated Exploit ID: 961A6406 Assigned: LocalServer State: Completed Duration: 16m 6s

Summary Details Browser **Traffic** Findings Fingerprint Automation

Resource Graph

- http://localhost:1234 (38)
 - / (37)
 - xvwa (36)
 - css (4)
 - fonts
 - img
 - js (3)
 - setup
 - vulnerabilities (18)
 - cmdi
 - crypto
 - csrf
 - dom_xss
 - fi (1)
 - fileupload
 - idor
 - missfunc
 - redirect
 - reflected_xss
 - sessionflaws
 - sqli
 - sqli_blind
 - ssrf_xspa
 - sssti
 - stored_xss
 - xpath

ID	Method	Status	TTFB	Url	Response Type
> 7	GET	200	3.99	http://localhost:1234/xvwa/instruction.php	text/html

```

GET /xvwa/instruction.php HTTP/1.1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3700.0 Safari/537.36
Referer: http://localhost:1234/xvwa/
Host: localhost
X-Meta-Chrome: resource-type=Document, browse-id=5, frame-id=BFF783D378CF7AE1FDF4A5BEA26BD89, request-id=C00391515C734414D4698B24FCD097FD

```

```

HTTP/1.1 200 OK
Date: Sun, 18 Oct 2020 15:47:50 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.13
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 9167
Keep-Alive: timeout=5, max=95
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="description" content="">
<meta name="author" content="">
<title>XVWA - Xtreme Vulnerable Web Application </title>
<!-- Bootstrap Core CSS -->
<link href="css/bootstrap.min.css" rel="stylesheet">
<!-- Custom CSS -->
<link href="css/shop-item.css" rel="stylesheet">
<!-- HTML5 Shim and Respond.js IEB support of HTML5 elements and media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->

```

Findings Tab

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

XVWA Authenticated Exploit									
ID: 961A6406 Assigned: LocalServer State: Completed Duration: 16m 6s									
Summary Details Browser Traffic Findings Fingerprint Automation									
Refresh Severity Filters Export Compliance Search: <input type="text"/>									
ID	Severity	Name	CWE	Location	Method	Created By	Parameter Name	Parameter Value	
> 10	Critical	Command Injection	77	Click 'OS Command Injection (http://localhost:1234/xvwa/vulnerabilit	GET	Traffic Fuzzer	target	cat /etc/passwd	
> 21	Critical	Cross Site Open Redirect	554,693	Click 'Redirects & Forwards (http://localhost:1234/xvwa/vulnerabilities/	GET	Workflow Fuzzer	forward	http://www.evil1187240702.com	
> 19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulne	GET	Workflow Fuzzer	Your Name	<script>alert(1556775553)</script>	
> 20	Critical	Cross Site Scripting (Reflected)	554,693	Click 'SSRF / XSPA (http://localhost:1234/xvwa/vulnerabilities/ssrf_xspa	POST	Workflow Fuzzer	img_url	'_<script>alert(43598382)</script>	
> 13	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XPATH Injection (http://localhost:1234/xvwa/vulnerabilities/xpath	POST	Workflow Fuzzer	Search by ID	'><script>alert(1457587747)</script>	
> 17	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - DOM Based (http://localhost:1234/xvwa/vulnerabilities/dor	GET	Workflow Fuzzer	Enter Search Item		
> 18	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - Reflected (http://localhost:1234/xvwa/vulnerabilities/reflect	GET	Workflow Fuzzer	item	'_<script>alert(2100425687)</script>	
> 16	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - Stored (http://localhost:1234/xvwa/vulnerabilities/stored_x	POST	Workflow Fuzzer	Enter Comment	<script>alert(563359535)</script>	
> 1	Critical	Unprotected Transport of Credentials (Client)	523,319	http://localhost:1234/xvwa/login.php	POST	Inspector			
> 27	Critical	Unrestricted Failed Logins	307	XVWA Login	POST	Workflow Fuzzer	password	badpassword	
> 5	High	Cross Frame Scripting	451	http://localhost:1234/xvwa/	GET	Traffic Fuzzer			
> 7	High	Cross Site Request Forgery (confirmed)	352	Click 'Unrestricted File Upload (http://localhost:1234/xvwa/vulnerabilit	POST	Traffic Fuzzer			
> 15	High	Local File Inclusion	98,22,73	Click 'File Inclusion (http://localhost:1234/xvwa/vulnerabilities/lf/)->Ch	GET	Traffic Fuzzer	file	//etc/passwd	
> 24	High	Local File Inclusion	98,22,73	http://localhost:1234/xvwa/vulnerabilities/lf//?file=%2F%2Fetc%2Fpas	GET	Traffic Fuzzer	file	//etc/passwd	
> 14	High	Remote File Inclusion	98	Click 'File Inclusion (http://localhost:1234/xvwa/vulnerabilities/lf/)->Ch	GET	Traffic Fuzzer	file	http://assertsecurity.io/scan-support/ff-evidence.bt	
> 23	High	Remote File Inclusion	98	http://localhost:1234/xvwa/vulnerabilities/lf//?file=http%3A%2F%2Fas	GET	Traffic Fuzzer	file	http://assertsecurity.io/scan-support/ff-evidence.bt	
> 6	High	Unrestricted File Upload (Multi-Part)	434	Click 'Unrestricted File Upload (http://localhost:1234/xvwa/vulnerabilit	POST	Traffic Fuzzer	filename	Arbitrary executable file: 1999863148	
> 8	Medium	Cross Site Request Forgery (possible)	352	Click 'SQL Injection (Blind) (http://localhost:1234/xvwa/vulnerabilities/s	POST	Traffic Fuzzer			
> 9	Medium	Cross Site Request Forgery (possible)	352	Click 'SQL Injection (http://localhost:1234/xvwa/vulnerabilities/sqli/)->	POST	Traffic Fuzzer			
> 12	Medium	Cross Site Request Forgery (possible)	352	Click 'SSRF / XSPA (http://localhost:1234/xvwa/vulnerabilities/ssrf_xspa	POST	Traffic Fuzzer			

Showing 1 to 20 of 27 entries

Items per page: 20 Previous 1 2 Next

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

ID	Severity	Name	CWE	Location	Method	Created By	Parameter Name	Parameter Value
19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/ssti)	GET	Workflow Fuzzer	Your Name	<script>alert(1556775553)</script>

Description Screenshot Traffic Document Workflow Properties

Cross-Site Scripting

Cross-Site Scripting (XSS) has been **confirmed** in this application. XSS attacks can inject client-side script into web pages trusted and viewed by other users.

Attackers leverage XSS flaws by sending malicious code to unsuspecting end users. The user who is the target of the attack browses the vulnerable site and their browser executes the injected script.

[OWASP Overview](#)

Impact

The injected script runs because the end user's browser trusts the web site. Malicious script can access session tokens or cookies. Web applications frequently use sensitive information that the browser maintains locally. XSS scripts can access this information.

Evidence

Venari's analysis has confirmed that this application is vulnerable to script injection. The engine that tests for XSS uses a multi-step process to find exploitable injection vulnerabilities:

- Probing** traces the flow of inputs to HTML-rendered outputs. The attack surface of the application is mapped via probe and reflection matching.
- Adaptive Injection** analyzes the reflected probe's location in the live DOM. The analyzer uses the location metadata to compute the payloads most likely to break out of the intended HTML markup or script block.
- Browser Verification** executes the stream of DOM interactions needed to deliver the payload. The browser driver maintains login while replaying the events that land the payload into the probes rendered location. This verification removes false-positives by intercepting the (or offsite navigation) as it happens, thereby proving that it is executable and not an inert reflection.

Screenshot: The image shows the executed alert box rendered into the page
 Fuzzed Traffic: The highlighted text is the attack payload
 Document HTML: The highlighted DOM text shows where the attack was injected into HTML as the browser rendered the page
 Workflow: The YAML description of the DOM event stream
 Properties: Shows the location details of the attack vector

Remediation

Refer to the information panels on the right to see where the payload entered this application. These data views show vulnerability and page-specific information for which input(s) need to be validated and where the output is not being properly encoded.

For general information on preventing XSS, see the OWASP prevention cheatsheet available on GitHub.

[OWASP Prevention Cheatsheet](#)

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.

ID	Severity	Name	CWE	Location	Method	Created By	Parameter Name
19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/ssti)	GET	Workflow Fuzzer	Your Name

Description Screenshot Traffic Document Workflow Properties

XVWA Admin - About

Setup

- Home
- Instructions
- Setup / Reset

Attacks

- SQL Injection
- SQL Injection (Blind)
- OS Command Injection
- XPATH Injection
- Unrestricted File Upload
- XSS - Reflected
- XSS - Stored
- XSS - DOM Based
- SSRF / XSPA

Server Side Template Injection (SSTI)

Web application uses templates to make the web pages look more dynamic. Template Injection occurs when user input is embedded in a template in an unsafe manner. However in the initial observation, this vulnerability is easy to mistake for XSS attacks. But SSTI attacks can be used to directly attack web servers' internals and leverage the attack more complex such as running remote code execution and complete server compromise.

Read more about Server Side Template Injection (SSTI)
<http://blog.portswigger.net/2015/08/server-side-template-injection.html>

Hint:

Alert!

1556775553

Please:

Your Name

Hello

Finding Traffic Sub-View

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

ID	Severity	Name	CWE	Location	Method	Created By	Parameter Name	Parameter Value
19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/sssti GET	Workflow Fuzzer	Your Name		<script>alert(1556775553)</script>

Description Screenshot Traffic Document Workflow Properties

```
GET /xvwa/vulnerabilities/sssti/?name=%3Cscript%3Ealert%281556775553%29%3C%2Fscript%3E&submit=
HTTP/1.1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3700.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.9
Referer: http://localhost:1234/xvwa/vulnerabilities/sssti/
Cookie: PHPSESSID=g3j3is1a4ljficgvikb0a273n4
Host: localhost:1234
X-Meta-Requestor: origin-browser, function-hyperlink, browse-action-navigate

HTTP/1.1 200 OK
Date: Sun, 18 Oct 2020 15:53:21 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.13
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="">
  <meta name="author" content="">

  .....

  <title>XVWA - Xtreme Vulnerable Web Application </title>

  <!-- Bootstrap Core CSS -->
  <link href="../../css/bootstrap.min.css" rel="stylesheet">

  <!-- Custom CSS -->
  <link href="../../css/shop-item.css" rel="stylesheet">
```

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.

ID	Severity	Name	CWE	Location	Method	Created By	Parameter Name	Parameter Value
19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/sssti GET	Workflow Fuzzer	Your Name		<script>alert(1556775553)</script>

Description Screenshot Traffic Document Workflow Properties

```
http://localhost:1234/xvwa/vulnerabilities/sssti/?name=%3Cscript%3Ealert%281556775553%29%3C%2Fscript%3E&submit=
...

<div class="well">
  <div class="col-lg-6">
    <p>
      Hint: <br>
    </p><ul>
      <li>Template Engine used is TWIG </li>
      <li>Loader function used = "Twig_Loader_String" </li>
    </ul>
    <p></p><br>
    <p>Please Enter your Name.
      </p><form method="get" action="">
        <div class="form-group">
          <label></label>
          <input class="form-control" width="50%" placeholder="Your Name" name="name"> <br>
          <div align="right"> <button class="btn btn-default" type="submit" name="submit">Submit Button</button></div>
        </div>
      </form>
    </div>
    Hello <script>alert(1556775553)</script> <p></p>
  </div>
</div>
```

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.

ID	Severity	Name	CWE	Location	Method	Created By
19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/ssti)	GET	Workflow Fuzzer

Description Screenshot Traffic Document Workflow Properties

Header:

- Name: Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/ssti)'->Click 'Submit Button'
- ApplicationName: XVWA
- RequiresLogin: True

Actions:

- Navigate:
 - Url: '{endpoint}/xvwa/vulnerabilities/ssti/'
- WaitForContent: {}
- Snapshot: {}
- ActionSkipped: {}
- FindTextInputs:
 - Take: 1
 - Nearest: Any
 - Patterns:
 - Your Name
- SendText:
 - Text: '{Your Name}'
 - Overwrite: true
- SendKeyCodes:
 - KeyCodes:
 - Escape
- Click:
 - Nearest: Any
 - Patterns:
 - Submit Button
- WaitForContent: {}
- Snapshot: {}

Finding Properties Sub-View

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

ID	Severity	Name	CWE	Location	Method	Created By	Parameter Name
19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/ssti)	GET	Workflow Fuzzer	Your Name

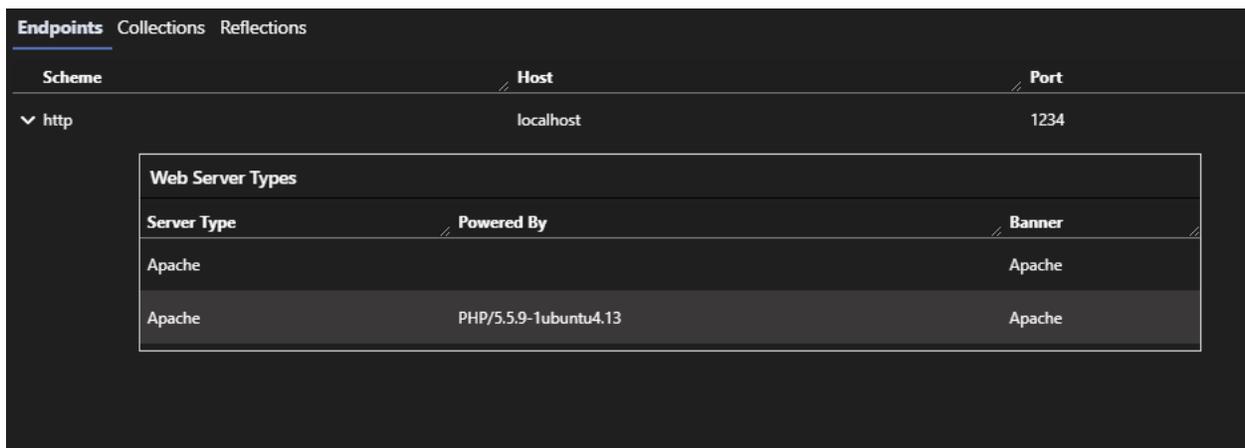
Description Screenshot Traffic Document Workflow Properties

- ID: 19
- Severity: Critical
- Name: Cross Site Scripting (Reflected)
- CWE: 554,693
- Location: Click 'Server Side Template Injection (http://localhost:1234/xvwa/vulnerabilities/ssti)'->Click 'Submit Button'
- Url: http://localhost:1234/xvwa/vulnerabilities/ssti/?name=%3Cscript%3Ealert%281556775553%29%3C%2Fscript%3E&submit=
- Method: GET
- Created By: Workflow Fuzzer
- Traffic Parameter Name: name
- Parameter Location: QueryString
- Parameter Name: Your Name
- Parameter Value: <script>alert(1556775553)</script>
- Parameter Type: Normal

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



Scheme	Host	Port
▼ http	localhost	1234
Web Server Types		
Server Type	Powered By	Banner
Apache		Apache
Apache	PHP/5.5.9-1ubuntu4.13	Apache

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.

Summary Details Browser Traffic Findings Fingerprint Automation					
Endpoints Collections Reflections					
Reflections Browser ↻ Search: <input type="text"/> ☰					
Name	Value	Parameter	Found By	Http Locations	HTML Locations
> http://localhost:1234/xvwa/->Click 'Server Side Template Injection (ht	12987100006534	Your Name	Browser Discovery		div
> http://localhost:1234/xvwa/->Click 'Server Side Template Injection (ht	12987100004776	Your Name	Browser Discovery		div
> http://localhost:1234/xvwa/->Click 'Server Side Template Injection (ht	12987100004934	Your Name	Browser Discovery		div
> http://localhost:1234/xvwa/->Click 'XPATH Injection (http://localhost:	12987100002523	Search by ID	Browser Discovery		input/@value
> http://localhost:1234/xvwa/->Click 'XPATH Injection (http://localhost:	12987100005674	Search by ID	Browser Discovery		input/@value
> http://localhost:1234/xvwa/->Click 'XPATH Injection (http://localhost:	12987100002647	Search by ID	Browser Discovery		input/@value
> http://localhost:1234/xvwa/->Click 'XSS - DOM Based (http://localho:	12987100005864	Enter Search Item	Browser Discovery		p
> http://localhost:1234/xvwa/->Click 'XSS - DOM Based (http://localho:	12987100003627	Enter Search Item	Browser Discovery		p
> http://localhost:1234/xvwa/->Click 'XSS - DOM Based (http://localho:	12987100006716	Enter Search Item	Browser Discovery		p
> http://localhost:1234/xvwa/->Click 'XSS - DOM Based (http://localho:	12987100006889	Enter Search Item	Browser Discovery		p

The example below shows a browser reflection into the DOM.

Summary Details Browser Traffic Findings Fingerprint Automation					
Endpoints Collections Reflections					
Name	Value	Parameter	Found By	Http Locations	HTML Locations
✓ http://localhost:1234/xvwa/->Click 'XPATH Injection (http://localhost:12.	12987100005674	Search by ID	Browser Discovery		input/@value

Screenshot Document Workflow

```

unauthorized XML data.      </p>
  <p>Read more about XPTH Injection<br>
  <strong><a target="_blank" href="https://www.owasp.org/index.php/XPATH_Injection">https://www.owasp.org/index.php/
XPATH_Injection</a></strong></p>

  </div>

</div>

<div class="well">
  <div class="col-lg-6">
    <p><b>Search Your Coffee</b>
    </p><form method="POST" action="">
      <div class="form-group">
        <label></label>
        <input type="text" class="form-control" placeholder="Search by ID" name="search" value="12987100005674">
      <br>
        <div align="right"> <button class="btn btn-default" name="submit" type="submit">Search</button></div>
      </div>
    </form>
  
```

Fingerprint Collections Sub-view

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

Name	Created
Cookies	10/18/2020 11:47:29 AM
Directory Names	10/18/2020 11:47:29 AM
Extensions	10/18/2020 11:47:30 AM
File Names	10/18/2020 11:47:30 AM
Javascript Frameworks	10/18/2020 11:47:37 AM
Origins	10/18/2020 11:47:29 AM
Request Headers	10/18/2020 11:47:29 AM
Request Parameters	10/18/2020 11:47:29 AM
Response Headers	10/18/2020 11:47:29 AM

Exporting Results and Viewing Reports

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

The screenshot shows the XVWA interface with the 'Findings' tab selected. The 'Export' dropdown menu is open, showing options for PDF, CSV, JSON, Code Dx, and FPR. The 'PDF' option is highlighted with a yellow box.

ID	Severity	Name	CWE	Location	Method	Created By	Parameter Name
> 10	Critical	Command Injection	77	Click 'OS Command Injection (http://localhost:1234/xvw	GET	Traffic Fuzzer	target
> 21	Critical	Cross Site Open R	554,693	Click 'Redirects & Forwards (http://localhost:1234/xvwa/	GET	Workflow Fuzzer	forward
> 19	Critical	Cross Site Scriptin	554,693	Click 'Server Side Template Injection (http://localhost:12	GET	Workflow Fuzzer	Your Name
> 20	Critical	Cross Site Scripting (Reflected)	554,693	Click 'SSRF / XSPA (http://localhost:1234/xvwa/vulnerab	POST	Workflow Fuzzer	img_url
> 13	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XPath Injection (http://localhost:1234/xvwa/vulne	POST	Workflow Fuzzer	Search by ID
> 17	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - DOM Based (http://localhost:1234/xvwa/vul	GET	Workflow Fuzzer	Enter Search Item
> 18	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - Reflected (http://localhost:1234/xvwa/vulne	GET	Workflow Fuzzer	item
> 16	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - Stored (http://localhost:1234/xvwa/vulnerab	POST	Workflow Fuzzer	Enter Comment

An example PDF report summary page is shown below

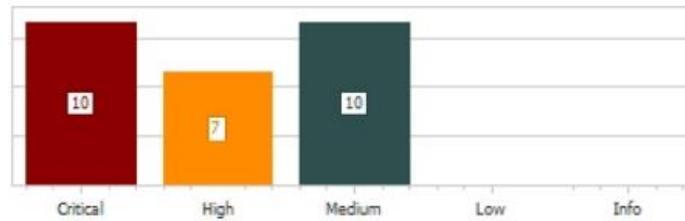


Venari Findings Report

Application - XVWA

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

Findings By Severity



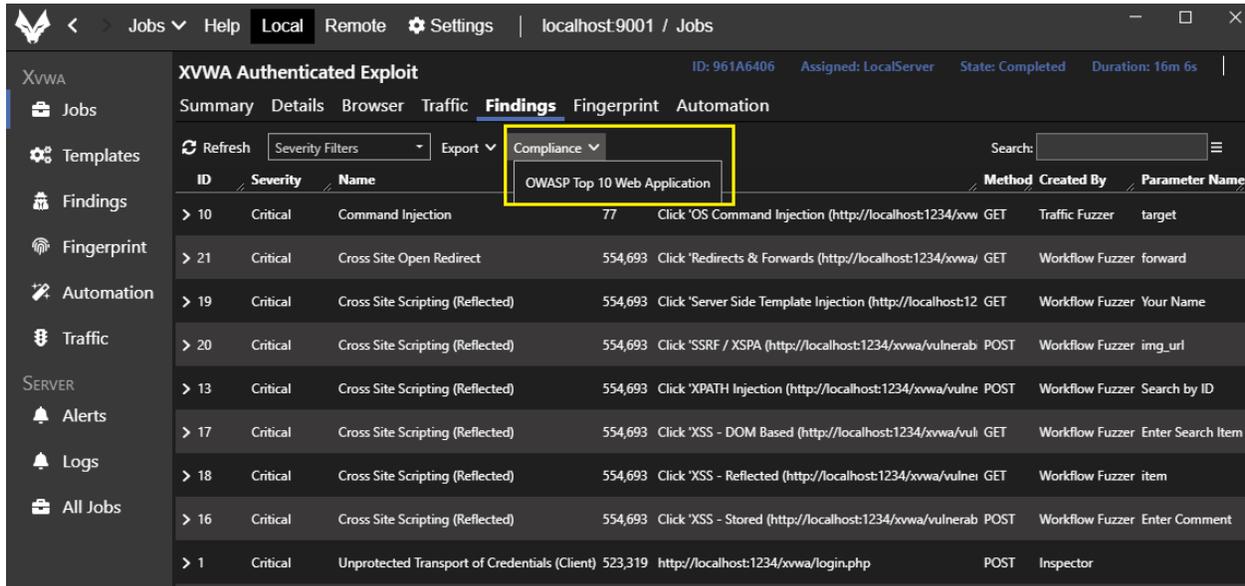
Critical	Command Injection	1
Critical	Cross Site Open Redirect	1
Critical	Cross Site Scripting (Reflected)	6
Critical	Unprotected Transport of Credentials (Client)	1
Critical	Unrestricted Failed Logins	1
High	Cross Frame Scripting	1
High	Cross Site Request Forgery (confirmed)	1
High	Local File Inclusion	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/cmd/)' Click 'Submit Button'
Uri:	http://localhost:1234/xvwa/vulnerabilities/cmd/?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.



The screenshot shows the Xvwa application interface. The top navigation bar includes 'Jobs', 'Help', 'Local', 'Remote', and 'Settings'. The main header displays 'XVWA Authenticated Exploit' with details like 'ID: 961A6406', 'Assigned: LocalServer', 'State: Completed', and 'Duration: 16m 6s'. The 'Findings' tab is active, showing a table of findings. A yellow box highlights the 'Compliance' dropdown menu in the 'Export' section, which is currently set to 'OWASP Top 10 Web Application'.

ID	Severity	Name	Count	Method	Created By	Parameter Name
> 10	Critical	Command Injection	77	Click 'OS Command Injection (http://localhost:1234/xvw	Traffic Fuzzer	target
> 21	Critical	Cross Site Open Redirect	554,693	Click 'Redirects & Forwards (http://localhost:1234/xvwa/	Workflow Fuzzer	forward
> 19	Critical	Cross Site Scripting (Reflected)	554,693	Click 'Server Side Template Injection (http://localhost:12	Workflow Fuzzer	Your Name
> 20	Critical	Cross Site Scripting (Reflected)	554,693	Click 'SSRF / XSPA (http://localhost:1234/xvwa/vulnerab	Workflow Fuzzer	img_url
> 13	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XPath Injection (http://localhost:1234/xvwa/vulne	Workflow Fuzzer	Search by ID
> 17	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - DOM Based (http://localhost:1234/xvwa/vul	Workflow Fuzzer	Enter Search Item
> 18	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - Reflected (http://localhost:1234/xvwa/vulne	Workflow Fuzzer	item
> 16	Critical	Cross Site Scripting (Reflected)	554,693	Click 'XSS - Stored (http://localhost:1234/xvwa/vulnerab	Workflow Fuzzer	Enter Comment
> 1	Critical	Unprotected Transport of Credentials (Client)	523,319	http://localhost:1234/xvwa/login.php	POST	Inspector

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.

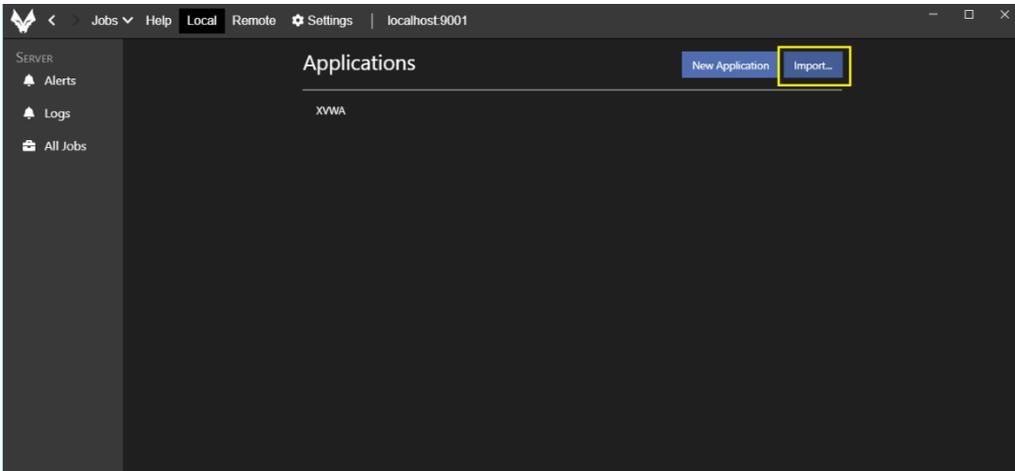
Sometimes onboarding an app is as simple as importing a single template file with auto-login information already embedded in the template. A more complex case would be an application that needs all three file types mentioned above, such as a site that needs a user account created and also has a non-trivial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

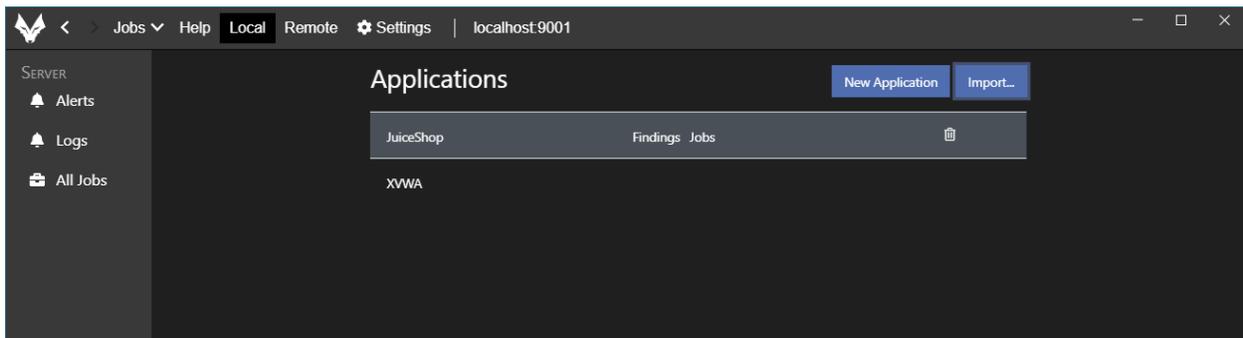
Scenario1: Import Job Template

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.



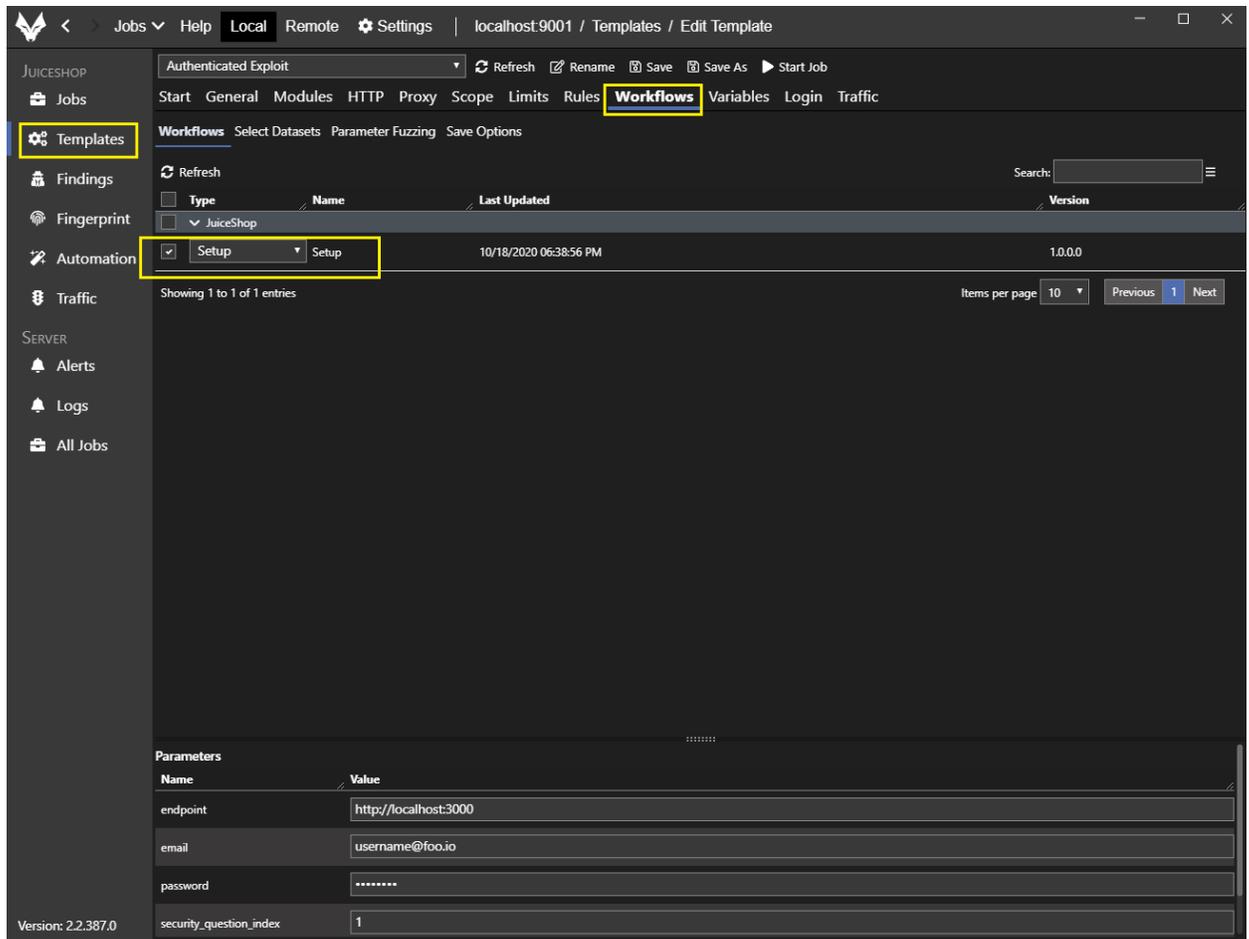
4. Observe that the application now appears on the start page



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.

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

	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 owaspwvad/wavsep
	Docker Run Command	docker run -d -p 8080:8080 -i -t owaspwvad/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