Walking an Application

Introduction

In this room you will learn how to manually review a web application for security issues using only the in-built tools in your browser. More often than not, automated security tools and scripts will miss many potential vulnerabilities and useful information.

Browser Tools Overview

Here is a short breakdown of the in-built browser tools you will use throughout this room:

- **View Source** - Use your browser to view the human-readable source code of a website.
- **Inspector** - Learn how to inspect page elements and make changes to view usually blocked content.
- **Debugger** - Inspect and control the flow of a page's JavaScript
- **Network** - See all the network requests a page makes.

The Penetration Testing Approach

As a penetration tester, your role when reviewing a website or web application is to discover features that could potentially be vulnerable and attempt to exploit them to assess whether or not they are. These features are usually parts of the website that require some interactivity with the user.

Finding interactive portions of the website can be as easy as spotting a login form to manually reviewing the website's JavaScript. An excellent place to start is just with your browser exploring the website and noting down the individual pages/areas/features with a summary for each one.

Example Site Review

An example site review for the Acme IT Support website would look something like this:

Feature	URL	Summary
Home Page	`/`	This page contains a summary of what Acme IT Support does with a company photo of their staff.
Latest News	`/news`	This page contains a list of recently published news articles by the company, and each news article has a link with an id number, i.e. `/news/article?id=1`
News Article	`/news/article?id=1`	Displays the individual news article. Some articles seem to be blocked and reserved for premium customers only.
Contact Page	`/contact`	This page contains a form for customers to contact the company. It contains name, email and message input fields and a send button.
Customers	`/customers`	This link redirects to `/customers/login`.
Customer Login	`/customers/login`	This page contains a login form with username and password fields.
Customer Signup	`/customers/signup`	This page contains a user-signup form that consists of a username, email, password and password confirmation input fields.
Customer Reset Password	`/customers/reset`	Password reset form with an email address input field.
Customer Dashboard	`/customers`	This page contains a list of the user's tickets submitted to the IT support company and a "Create Ticket" button.
Create Ticket	`/customers/ticket/new`	This page contains a form with a textbox for entering the IT issue and a file upload option to create an IT support ticket.
Customer Account	`/customers/account`	This page allows the user to edit their username, email and password.
Customer Logout	`/customers/logout`	This link logs the user out of the customer area.

Viewing Page Source

The page source is the human-readable code returned to our browser/client from the web server each time we make a request.

The returned code is made up of HTML (HyperText Markup Language), CSS (Cascading Style Sheets) and JavaScript, and it's what tells our browser what content to display, how to show it and adds an element of interactivity with JavaScript.

For our purposes, viewing the page source can help us discover more information about the web application.

How to View Page Source

There are three main ways to view page source:

1. **Right-click method**: While viewing a website, you can right-click on the page, and you'll see an option on the menu that says "View Page Source".
2. **URL prefix method**: Most browsers support putting `view-source:` in front of the URL, for example: `view-source:https://www.google.com/`
3. **Browser menu**: In your browser menu, you'll find an option to view the page source. This option can sometimes be in submenus such as developer tools or more tools.

What to Look For in Page Source

Comments

At the top of the page, you'll notice some code starting with `<!--` and ending with `-->` — these are comments. Comments are messages left by the website developer, usually to explain something in the code to other programmers or even notes/reminders for themselves. These comments don't get displayed on the actual webpage.

Links

Links to different pages in HTML are written in anchor tags (these are HTML elements that start with `<a`), and the link that you'll be directed to is stored in the `href` attribute.

If you view further down the page source, there may be hidden links to private areas used by the business for storing company/staff/customer information.

External Files and Directory Listings

External files such as CSS, JavaScript and Images can be included using the HTML code. Sometimes these files are all stored in the same directory. If you view this directory in your web browser, there may be a configuration error.

What should be displayed is either a blank page or a 403 Forbidden page with an error stating you don't have access to the directory. Instead, if the directory listing feature has been enabled, it will list every file in the directory. Sometimes this isn't an issue, and all the files in the directory are safe to be viewed by the public, but in some instances, backup files, source code or other confidential information could be stored here.

Framework Detection

Many websites these days aren't made from scratch and use what's called a framework. A framework is a collection of premade code that easily allows a developer to include common features that a website would require, such as blogs, user management, form processing, and much more, saving the developers hours or days of development.

Viewing the page source can often give us clues into whether a framework is in use and, if so, which framework and even what version. Knowing the framework and version can be a powerful find as there may be public vulnerabilities in the framework, and the website might not be using the most up to date version.

Developer Tools

Every modern browser includes developer tools — a toolkit used to aid web developers in debugging web applications and gives you a peek under the hood of a website to see what is going on. As a pentester, we can leverage these tools to provide us with a much better understanding of the web application.

We're specifically focusing on three features of the developer toolkit: **Inspector**, **Debugger**, and **Network**.

Opening Developer Tools

The way to access developer tools is different for every browser. If you're not sure how to access it, consult your browser's documentation for specific instructions.

Inspector

The page source doesn't always represent what's shown on a webpage; this is because CSS, JavaScript and user interaction can change the content and style of the page, which means we need a way to view what's been displayed in the browser window at this exact time. Element inspector assists us with this by providing us with a live representation of what is currently on the website.

As well as viewing this live view, we can also edit and interact with the page elements, which is helpful for web developers to debug issues.

Bypassing Paywalls

Floating boxes blocking page contents are often referred to as paywalls as they put up a metaphorical wall in front of the content you wish to see until you pay.

How to bypass using Inspector:

1. Right-click on the premium notice (paywall)
2. Select the "Inspect" option from the menu
3. This opens the developer tools with the elements/HTML that make up the website
4. Locate the `DIV` element with the class `premium-customer-blocker`
5. Find the CSS style `display: block`
6. Click on the word `block` and type `none`
7. The box will disappear, revealing the content underneath

Remember: This is only edited on your browser window, and when you press refresh, everything will be back to normal.

Debugger

This panel in the developer tools is intended for debugging JavaScript, and is an excellent feature for web developers wanting to work out why something might not be working. As penetration testers, it gives us the option of digging deep into the JavaScript code.

> **Note**: In Firefox and Safari, this feature is called "Debugger", but in Google Chrome, it's called "Sources".

Understanding Minified and Obfuscated Code

Many times when viewing JavaScript files, you'll notice that everything is on one line. This is because it has been **minimised**, which means all formatting (tabs, spacing and newlines) have been removed to make the file smaller. Files may also be **obfuscated**, which makes them purposely difficult to read, so they can't be copied as easily by other developers.

Using the Pretty Print Feature

We can return some of the formatting by using the "Pretty Print" option, which looks like two braces `{ }` to make it a little more readable.

Using Breakpoints

Breakpoints are points in the code that we can force the browser to stop processing the JavaScript and pause the current execution.

How to set a breakpoint:

1. Navigate to the JavaScript file you want to debug
2. Click the line number where you want to pause execution
3. The line will turn blue, indicating a breakpoint is set
4. Refresh the page
5. The code execution will pause at that line, allowing you to inspect the current state

Network Tab

The network tab on the developer tools can be used to keep track of every external request a webpage makes. If you click on the Network tab and then refresh the page, you'll see all the files the page is requesting.

Monitoring AJAX Requests

With the network tab open, when you submit forms, you'll notice events in the network tab. Forms are often submitted in the background using a method called **AJAX**. AJAX is a method for sending and receiving network data in a web application background without interfering by changing the current web page.

To monitor form submissions:

1. Open the Network tab
2. Fill in a form on the website
3. Press the submit button
4. Examine the new entry in the network tab
5. View the page the data was sent to and the response received

This can reveal:

- API endpoints
- Data being transmitted
- Server responses
- Hidden parameters or flags

Summary

Manual web application security testing using browser tools is an essential skill for penetration testers. By leveraging View Source, Inspector, Debugger, and Network tools, you can:

- Discover hidden information in comments and source code
- Identify framework versions and potential vulnerabilities
- Bypass client-side restrictions
- Debug and analyze JavaScript code
- Monitor network traffic and AJAX requests
- Find exposed directories and sensitive files

These techniques complement automated tools and often reveal vulnerabilities that automated scanners miss.

Revision #1

Created 2026-01-03 17:35:52 UTC by David Rizzo

Updated 2026-01-03 17:36:12 UTC by David Rizzo