Exploring BeEF: The Browser Exploitation Framework

Md. Asif Haider, Mashiyat Mahjabin Prapty

1805112, 1805117

Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology

September 14, 2023

Contents

1 Introduction

This technical report is a part of **CSE 406: Computer Security Sessional** course project. In this project, we explore BeEF: The Browser Exploitation Framework, a powerful and intuitive security tool that facilitates browser penetration testing.

The report is structured as follows: we first present an overview of the tool BeEF. As BeEF is an open-source project, we present a brief, high-level overview of the project source code in the next section. Then we go through the major features of BeEF one by one in the following sections. We demonstrate how to run each of the highlighted features. We also provide snapshots of our demonstration for each of the features presented.

2 BeEF Overview

The **Browser Exploitation Framework (BeEF)** is an open source penetration testing tool focused on exploiting vulnerabilities in the web browser. BeEF is pioneering techniques that provide penetration testers with practical client-side attack vectors. Unlike other security frameworks, BeEF focuses on leveraging browser vulnerabilities to assess the security posture of a target. As mentioned in their official documentation, this project is developed solely for lawful research and penetration testing.

BeEF looks past the hardened network perimeter and client system, and examines exploitability within the context of one open door: the web browser. It hooks one or more web browsers to the application for the launching of directed command modules. Each browser is likely to be within a different security context, and each context may provide a set of unique attack vectors. The framework allows the penetration tester to select specific modules in real time to target each browser, and therefore each context.

3 Source Code Structure

The source code is available here in GitHub. To get started with BeEF, one needs to clone or download this repository in a Linux OS. The code is written in Ruby, thus making it compulsory to install Ruby and the corresponding gems beforehand.

After that, the installation file has to be run which installs all necessary dependencies. If someone wants to use Metasploit or Phonegap, they also must be installed earlier. When the install is finished successfull, we can run './beef' in the terminal and that starts the application. The main files or servers of the application are found inside the **core** folder of the repository.

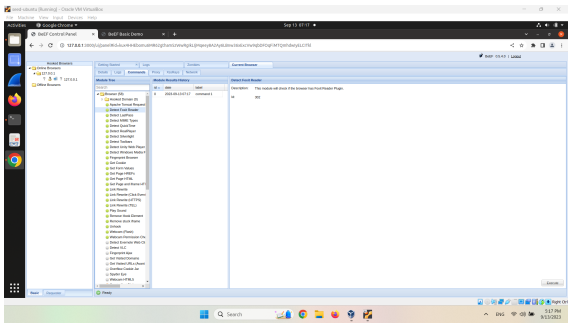
The extensions folder contain different extension files, through which the extensions can be enabled or disabled. The different modules can be found inside the **modules** folder. For every feature that we will discuss afterward, there is a folder inside the **modules** directory. Each of these folders contains three types of files, namely- command.js, config.yaml, module.rb. The command file contains browser specific commands. The config file enables/disables the module as well as configure it. The module file contains the actual code.

4 Key Features

4.1 Browser

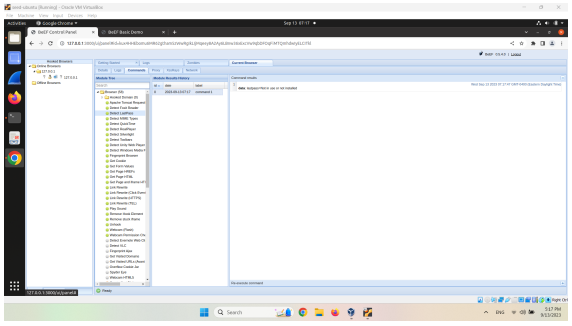
4.1.1 Detect Foxit Reader

This module checks if the browser has Foxit Reader plugin.



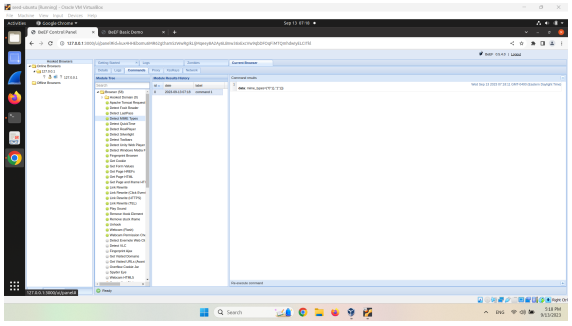
4.1.2 Detect LastPass

This module checks if the LastPass extension is installed and active.



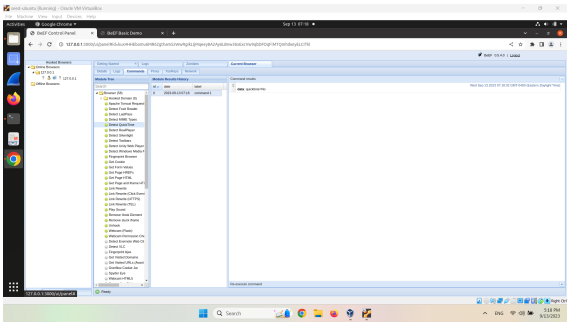
4.1.3 Detect MIME Types

This module retrieves the supported MIME types of the browser.



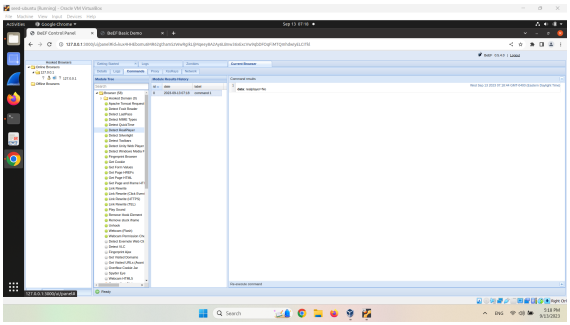
4.1.4 Detect QuickTime

This module will check if the browser has QuickTime support.



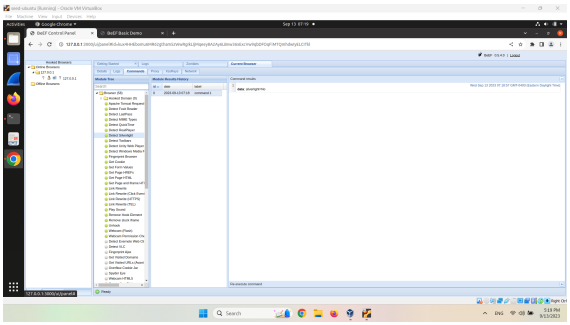
4.1.5 Detect RealPlayer

This module will check if the browser has RealPlayer support.



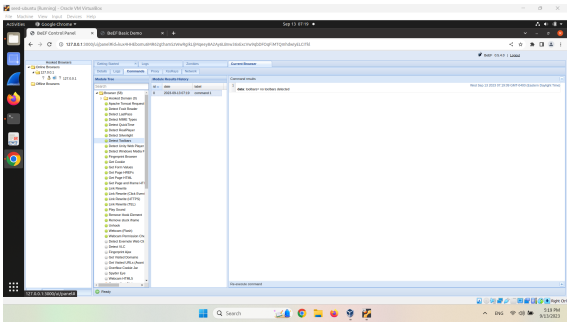
4.1.6 Detect Silverlight

This module will check if the browser has SilverLight support.



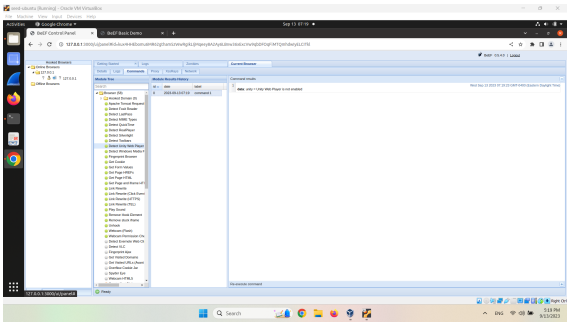
4.1.7 Detect Toolbars

Detects which browser toolbars are installed.



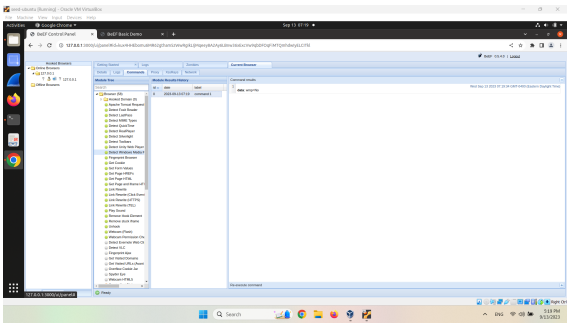
4.1.8 Detect Unity Web Player

Detects Unity Web Player.



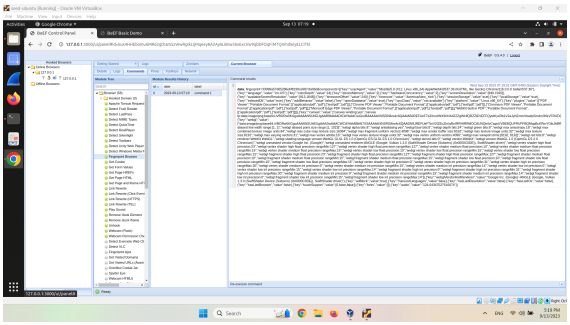
4.1.9 Detect Windows Media Player

This module will check if the browser has Windows Media Player plugin installed.



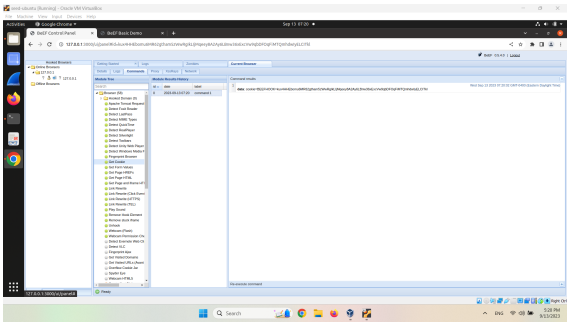
4.1.10 Fingerprint Browser

This module attempts to fingerprint the browser and browser capabilities using FingerprintJS2.



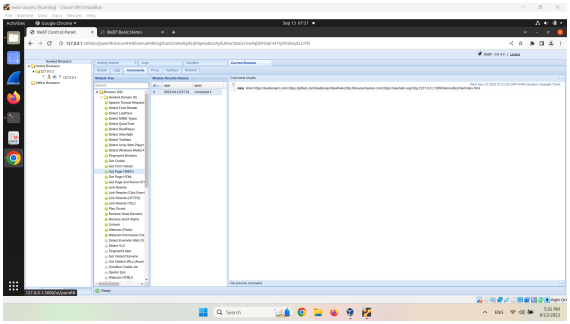
4.1.11 Get Cookie

This module will retrieve the session cookie from the current page.



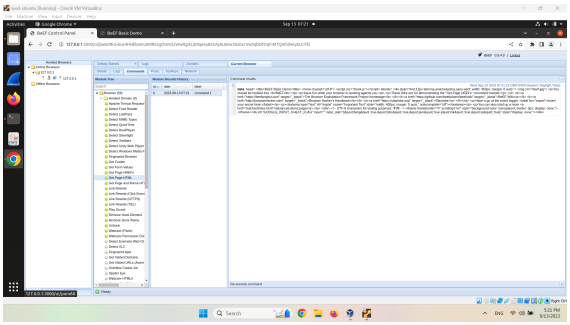
4.1.12 Get Page HREFs

This module will retrieve HREFs from the target page.



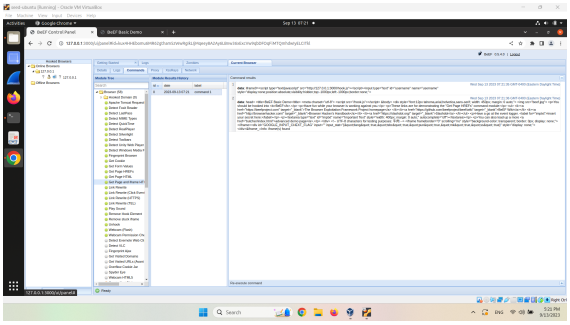
4.1.13 Get Page HTML

This module will retrieve the HTML from the current page.



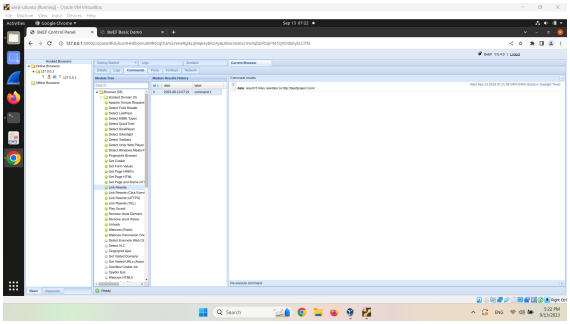
4.1.14 Get Page and iframe HTML

This module will retrieve the HTML from the current page and any iframes(that have the same origin).



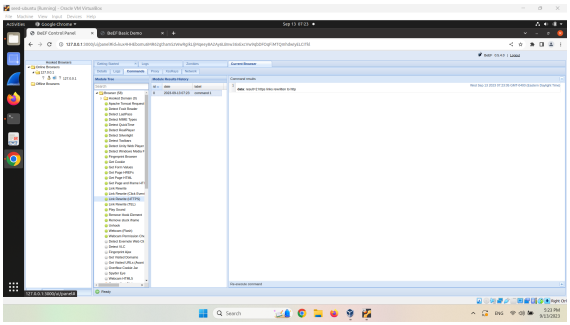
4.1.15 Link Rewrite

This module will rewrite all the href attributes of all matched links.



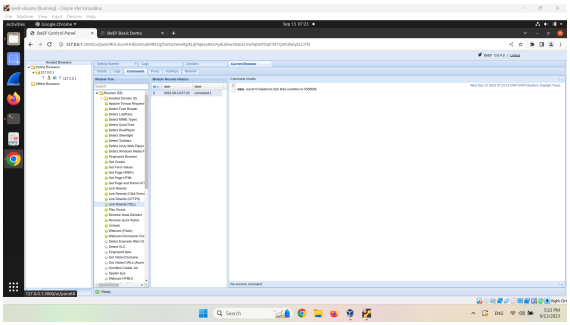
4.1.16 Link Rewrite (HTTPS)

This module will rewrite all the href attributes of HTTPS links to use HTTP instead of HTTPS. Links relative to the web root are not written.



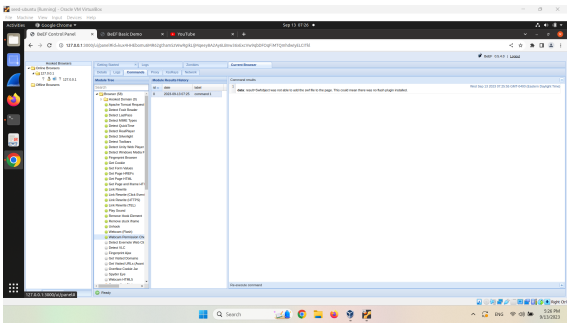
4.1.17 Link Rewrite (TEL)

This module will rewrite all the href attributes to telephone links to call a number of your choice.



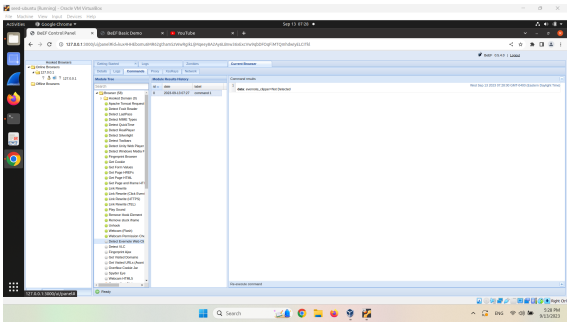
4.1.18 Webcam Permission Check

This module will check to see if the user has allowed BeEf domain(for all domain) to access the Camera and mic with Flash. This module is transparent and should not be detected by the user. Since we did not have flash installed, it return unsuccessful.



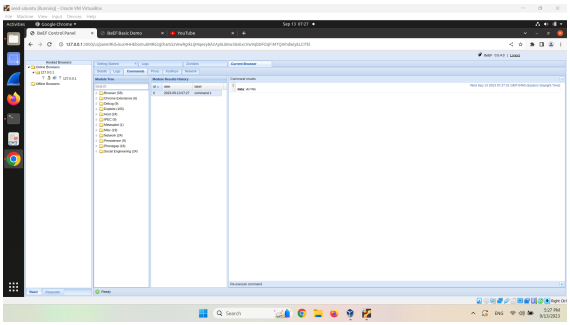
4.1.19 Detect Evernote Web Clipper

This module checks if the Evernote Web Clipper extension is installed and active.



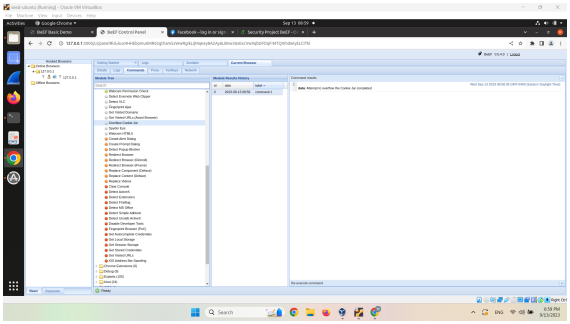
4.1.20 Detect VLC

This module will check if the browser has VLC plugin.



4.1.21 Overflow Cookie Jar

This module attempts to perform John Wilander's CookieJar overflow. With this module, cookies that have the HTTPOnly-flag and/or HTTPS-flag can be wiped.

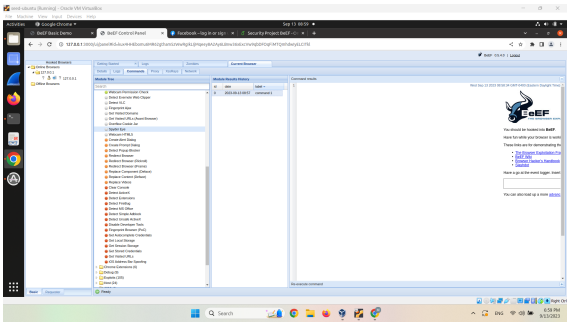


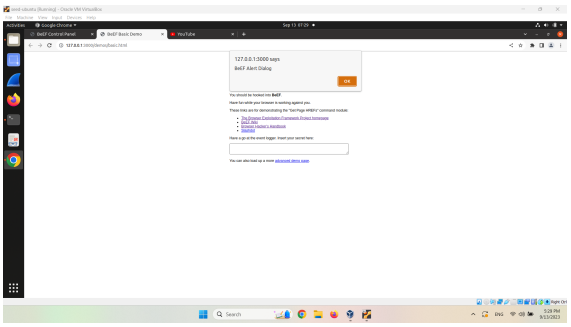
4.1.22 Spyder Eye

This module takes a picture of the victim's browser window. As parameters, we can give the number of times to take pictures as well as the delay.

4.1.23 Create Alert Dialog

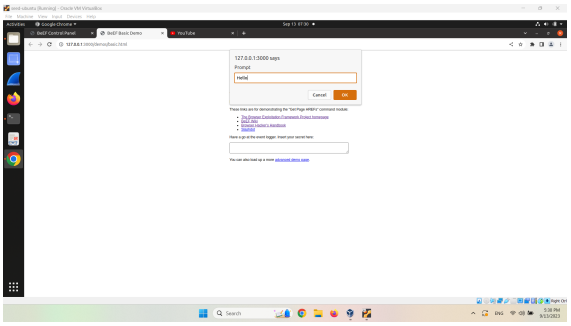
Sends an alert dialog in the hooked browser with the given alert text.

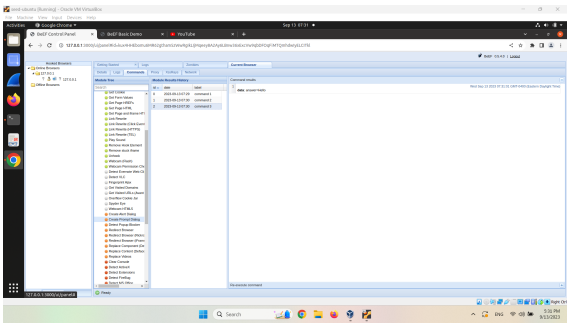




4.1.24 Create Prompt Dialog

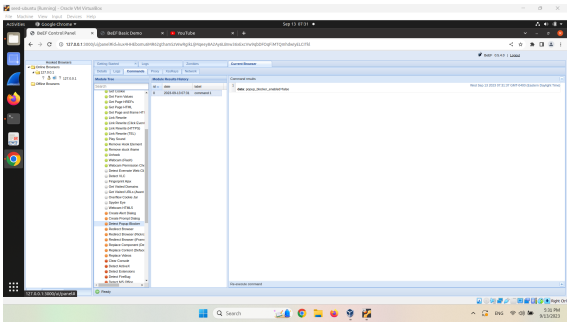
Sends a prompt dialog to the hooked browser.





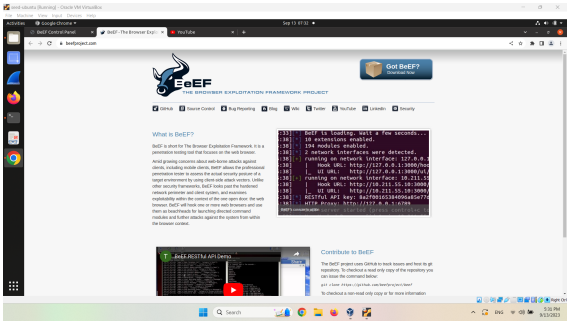
4.1.25 Detect Popup Blocker

Detect if popup blocker is enabled.



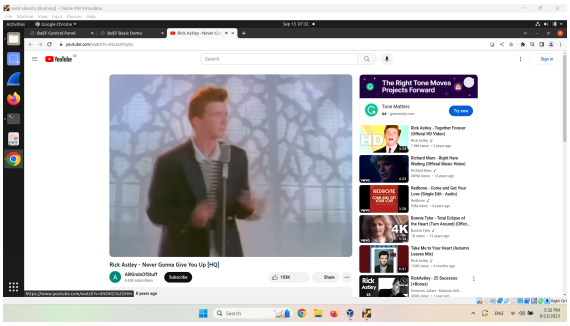
4.1.26 Redirect Browser

This module will redirect the selected hooked browser to the address specified in the 'Redirect URL' input.



4.1.27 Redirect Browser (Rickroll)

Overwrite the body of the page the victim is on with a full screen Rickroll.

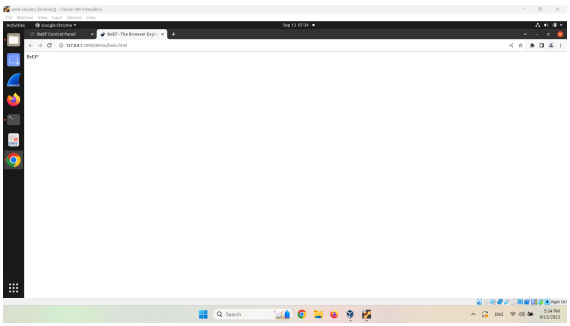


4.1.28 Redirect Browser (iFrame)

This module creates a $100\% * 100\%$ overlaying iframe and keeps the browser hooked to the framework. The content of the iframe, page title, page shortcut icon and the delay are specified in the parameters. The content of the URL bar will not be changed in the hooked browser.

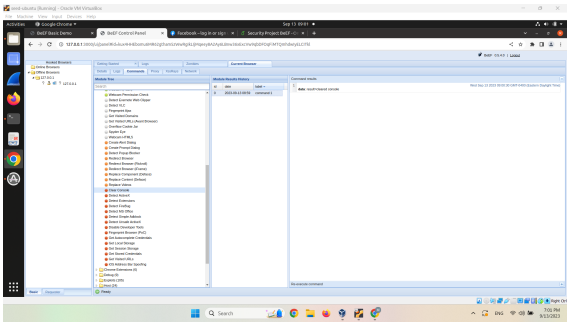
4.1.29 Replace Content (Deface)

Overwrite the page, title and shortcut icon on the hooked page with the values specified.



4.1.30 Clear Console

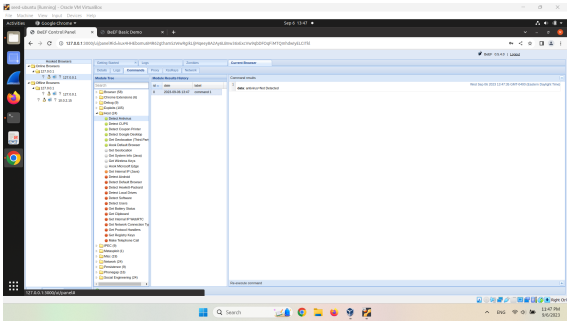
This module clears the Chrome developer console.



4.2 Host

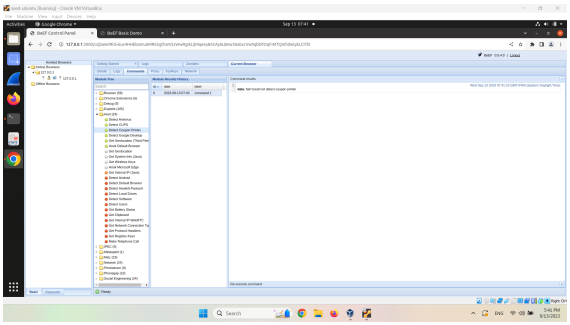
4.2.1 Detect Antivirus

This module detects if there is any antivirus in the localhost.



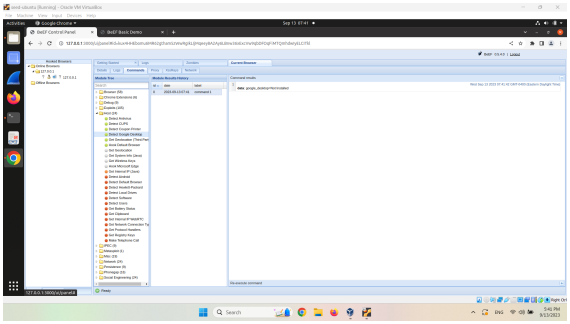
4.2.2 Detect Coupon Printer

This module attempts to detect Coupon Printer on localhost on the default WebSocket port 4004.



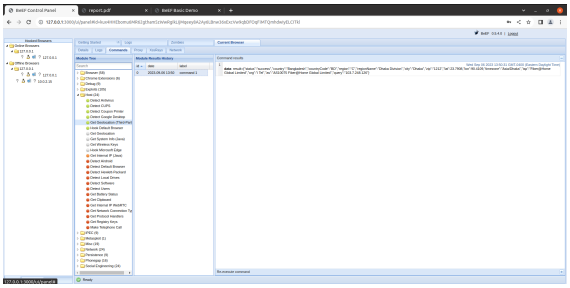
4.2.3 Detect Google Desktop

This module attempts to detect Google Desktop running on the default port 4664.



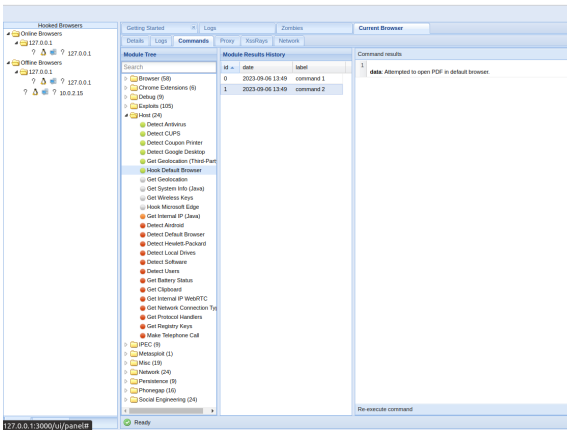
4.2.4 Get Geolocation (Third-Party)

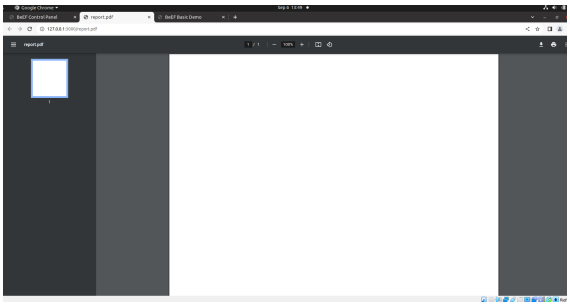
Gets the location of the browser using a third-party API.



4.2.5 Hook Default Browser

Hooks the default browser with BeEF and in this process tries to open a PDF in that browser. If the PDF can be opened successfully that means the browser is hooked.

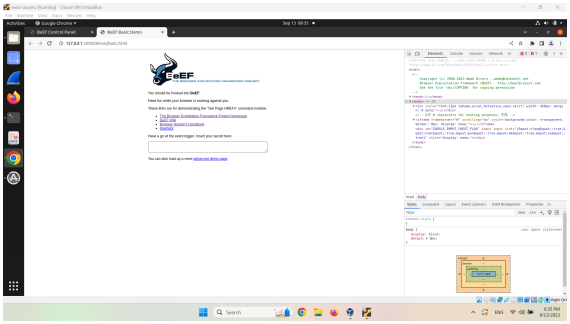




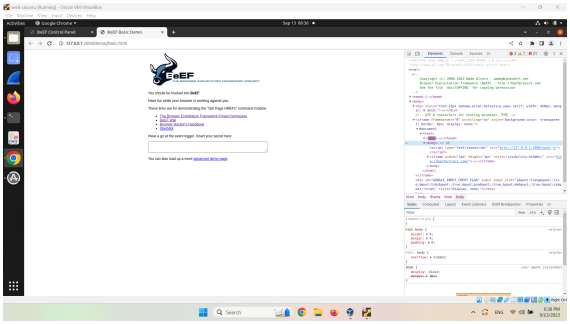
4.3 Misc

4.3.1 Create Invisible Iframe

Creates an invisible iFrame. We first see the original HTML code from console.

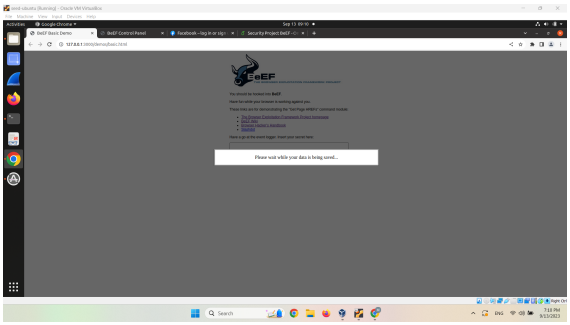


After the creation of the iFrame, we get the following state of the HTML code of the page.



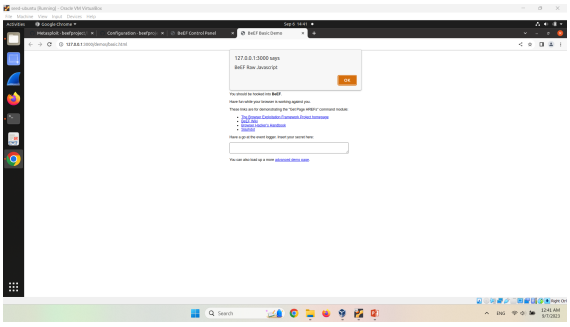
4.3.2 BlockUI Modal Dialog

This module uses jQuery BlockUI to block the window and display the message given as input.



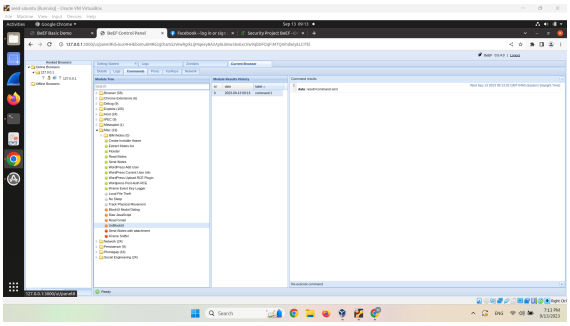
4.3.3 Raw JavaScript

This module will send the code entered in the 'JavaScript Code' section to the selected hook browsers where it will be executed. Code is run inside an anonymous function and the return value is passed to the framework. Multiline scripts are allowed, no special encoding is required.



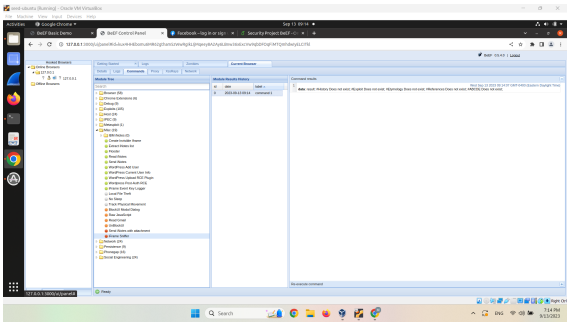
4.3.4 UnBlockUI

This module removes all jQuery BlockUI dialogs.



4.3.5 iFrame Sniffer

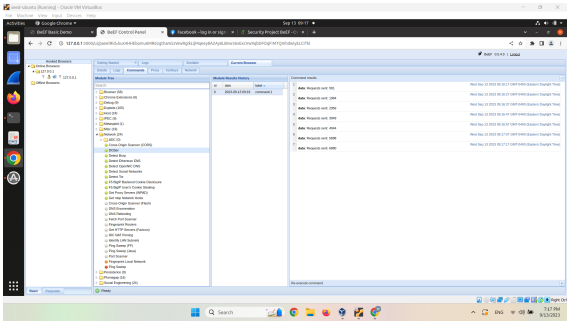
This module attempts to do framesniffing(aka Leaky Frame). It will append leakyframe.js(written by Paul Stone) to the DOM and check for specified anchors to be present on a URL.



4.4 Network

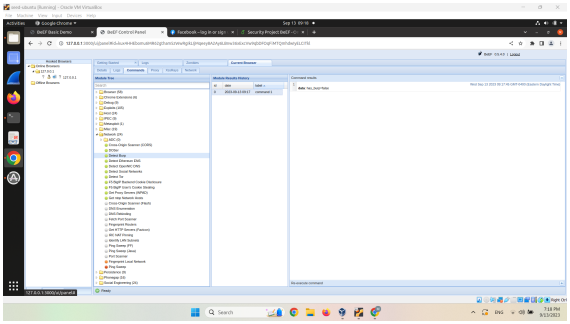
4.4.1 DOSer

Do infinite GET or POST requests to a target, spawning a WebWorker in order to slow down the hooked page.



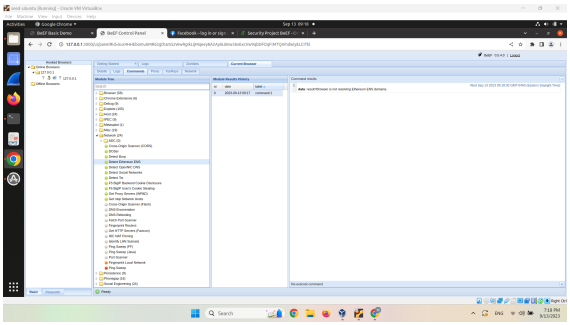
4.4.2 Detect Burp

This module checks if the browser is using Burp. The Burp web interface must be enabled(default). The proxy IP address is returned to BeEf.



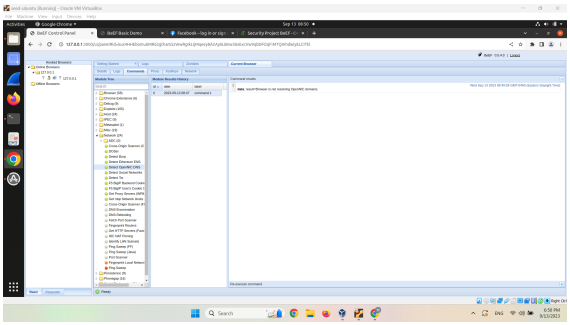
4.4.3 Detect Ethereum ENS

This module will detect is the zombie is currently using Ethereum ENS resolvers.



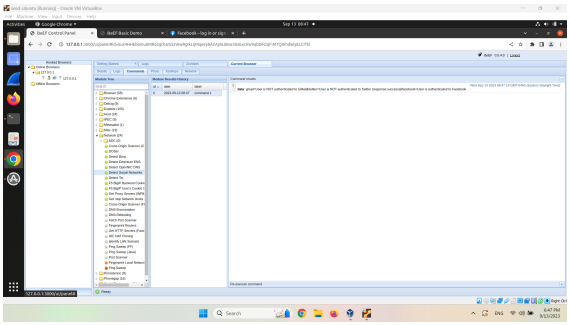
4.4.4 Detect OpenNIC DNS

This module will detect if the zombie is currently using OpenNIC DNS resolvers.



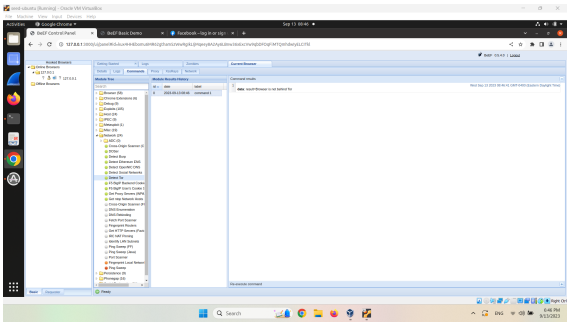
4.4.5 Detect Social Networks

This module will detect if the hooked browser is currently authenticated to GMail, Facebook and Twitter.



4.4.6 Detect Tor

This module will detect if the zombie is currently using Tor.

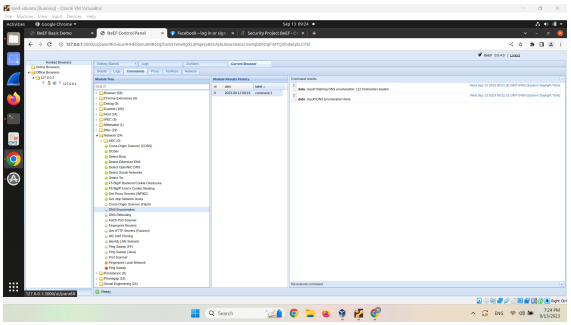


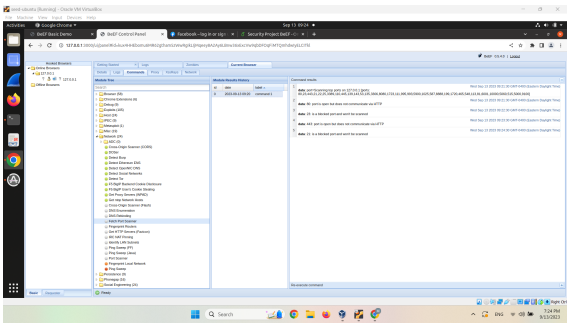
4.4.7 DNS Enumeration

Discover DNS hostnames within the victim's network using dictionary and timing attacks.

4.4.8 Fetch Port Scanner

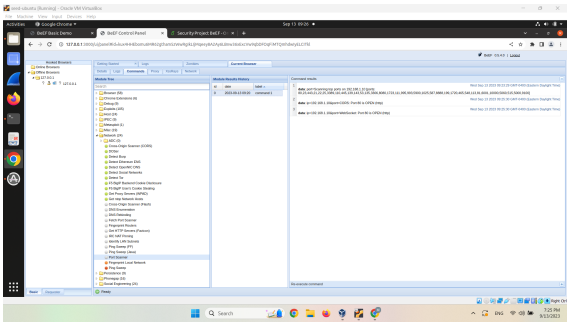
Uses fetch to test the response in order to determine if a port is open or not.





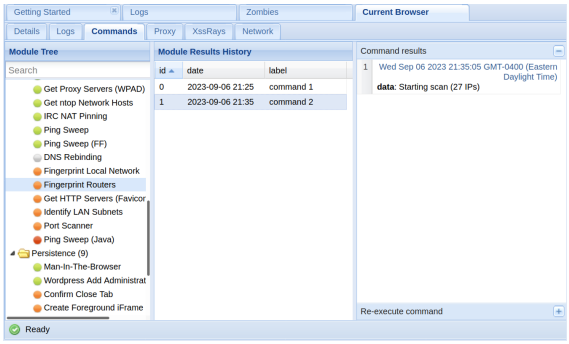
4.4.9 Port Scanner

Scan ports in a given hostname, using WebSockets, CORS and img tags. It uses the three methods to avoid blocked ports or Same Origin Policy.



4.4.10 Fingerprint Routers

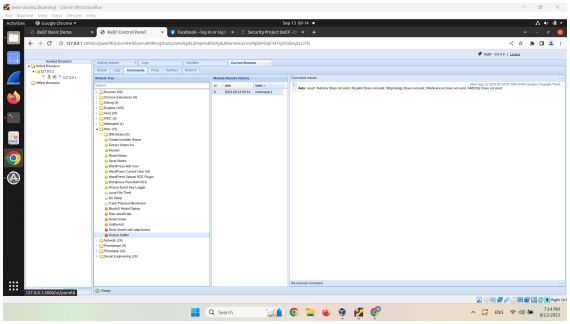
This module attempts to discover network routers on the local network of the hooked browser. It scans for web servers on IP addresses commonly used by routers.



4.5 Persistence

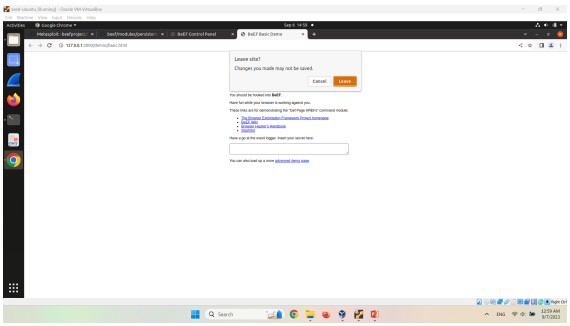
4.5.1 Man-In-The-Browser

An MitB attack works by infecting a browser with a Trojan horse, which enables an attacker to intercept and modify data sent from a browser to a server. In this case, new tabs are opened infinitely until the hooked tab is closed.



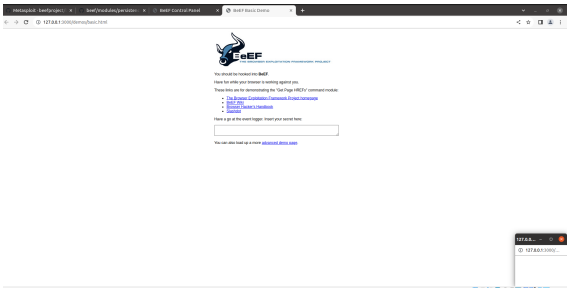
4.5.2 Confirm Close Tab

Shows a confirm dialog to the user when they try to close a tab. If they click yes, re-display the confirmation dialog.



4.5.3 Create Pop Under

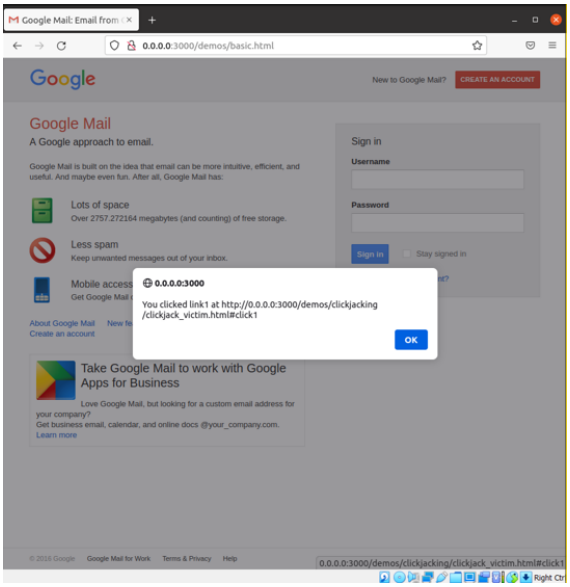
This module creates a new discreet pop under window with the BeEF hook icluded. Another browser node will be added to the hooked browser tree.



4.6 Social Engineering

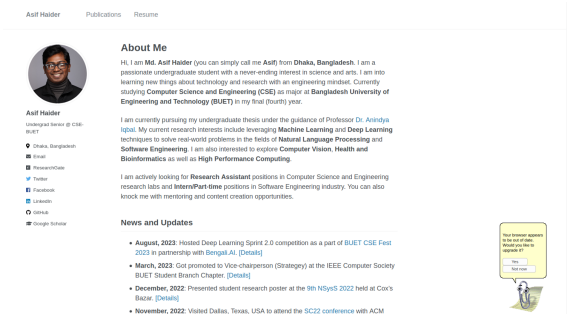
4.6.1 Clickjacking

Allows someone to perform basic multi-click clickjacking. The iframe follows the mouse, so anywhere the user clicks on the page will be over x-pos, y-pos. The optional JS configuration values specify local Javascript to execute when a user clicks, allowing the page to give visual feedback.



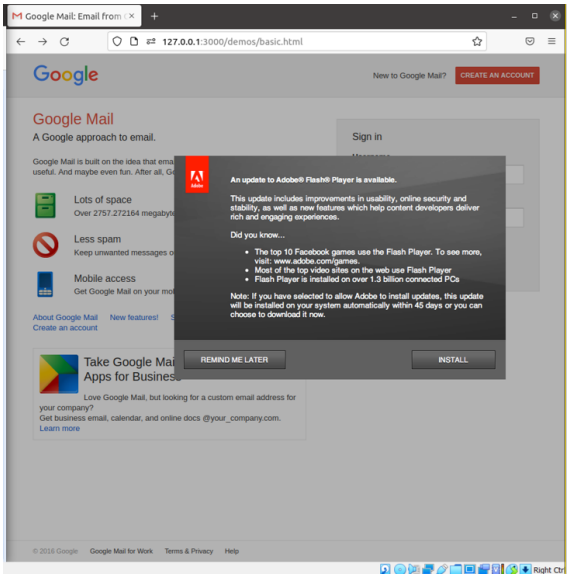
4.6.2 Clippy

An executable link is provided with the Clippy with a custom text.



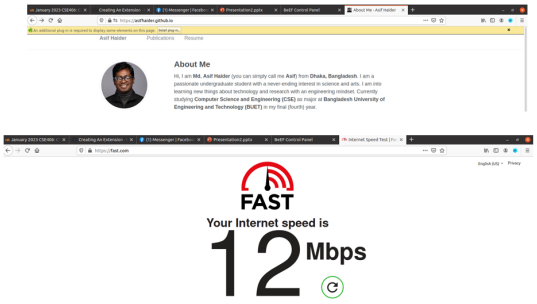
4.6.3 Fake Flash Update

Prompts the user to install an update to Adobe Flash Player. The delivered payload could be a custom file, a browser extension or any specific URL.



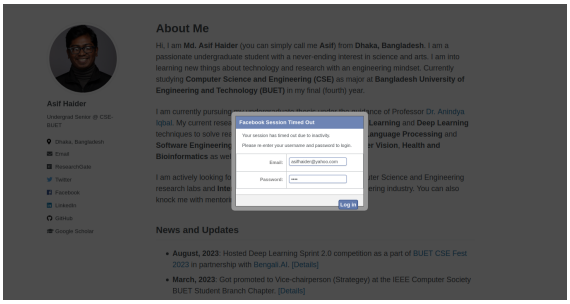
4.6.4 Fake Notification Bar

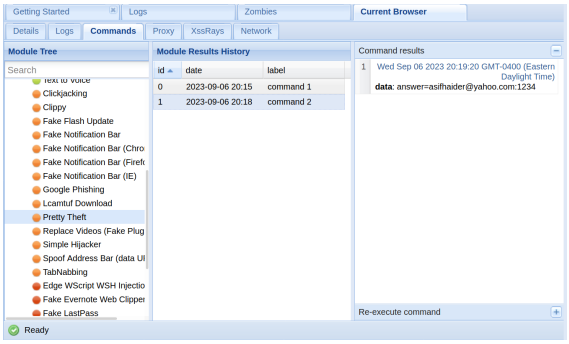
Displays a fake notification bar at the top of the screen, similar to those present in the specific browser. If the user clicks the notification they will be prompted to download a file from the specified url.



4.6.5 Pretty Theft

Asks the user for their username and password using a floating div of a prominent site.
In the following demo we have used a Facebook login prompt.





5 Demonstration Video Link

Link: BeEF Demonstration