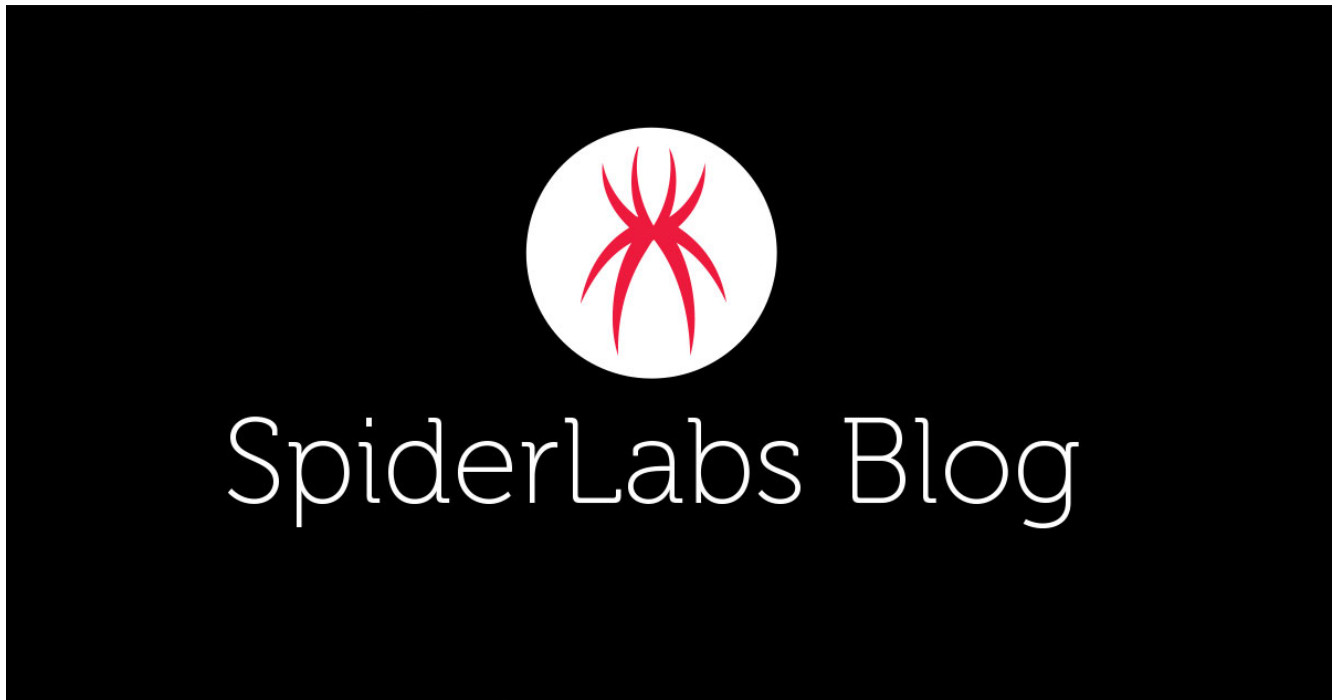


The Attack of the Chameleon Phishing Page

 trustwave.com/en-us/resources/blogs/spiderlabs-blog/the-attack-of-the-chameleon-phishing-page/



Loading...

Blogs & Stories

SpiderLabs Blog

Attracting more than a half-million annual readers, this is the security community's go-to destination for technical breakdowns of the latest threats, critical vulnerability disclosures and cutting-edge research.

Recently, we encountered an interesting phishing webpage that caught our interest because it acts like a chameleon by changing and blending its color based on its environment. In addition, the site adapts its background page and logo depending on user input to trick its victims into giving away their email credentials.

We see an email with the “initial” URLs in the example below:



Figure 1. The raw phishing email showing the URLs, purporting to be a fax message that needs to be accessed.

There are three URLs in the email. The second and third URLs are identical, which we will discuss later. To see what would happen we placed the first URL in a web browser, equating to the email's instructions for the target to click on the provided link to access the "fax document." Doing so brought us to the phishing site shown below. The webpage is fabricated. The victim's email address is already provided, and the site only asks for the victim's password.

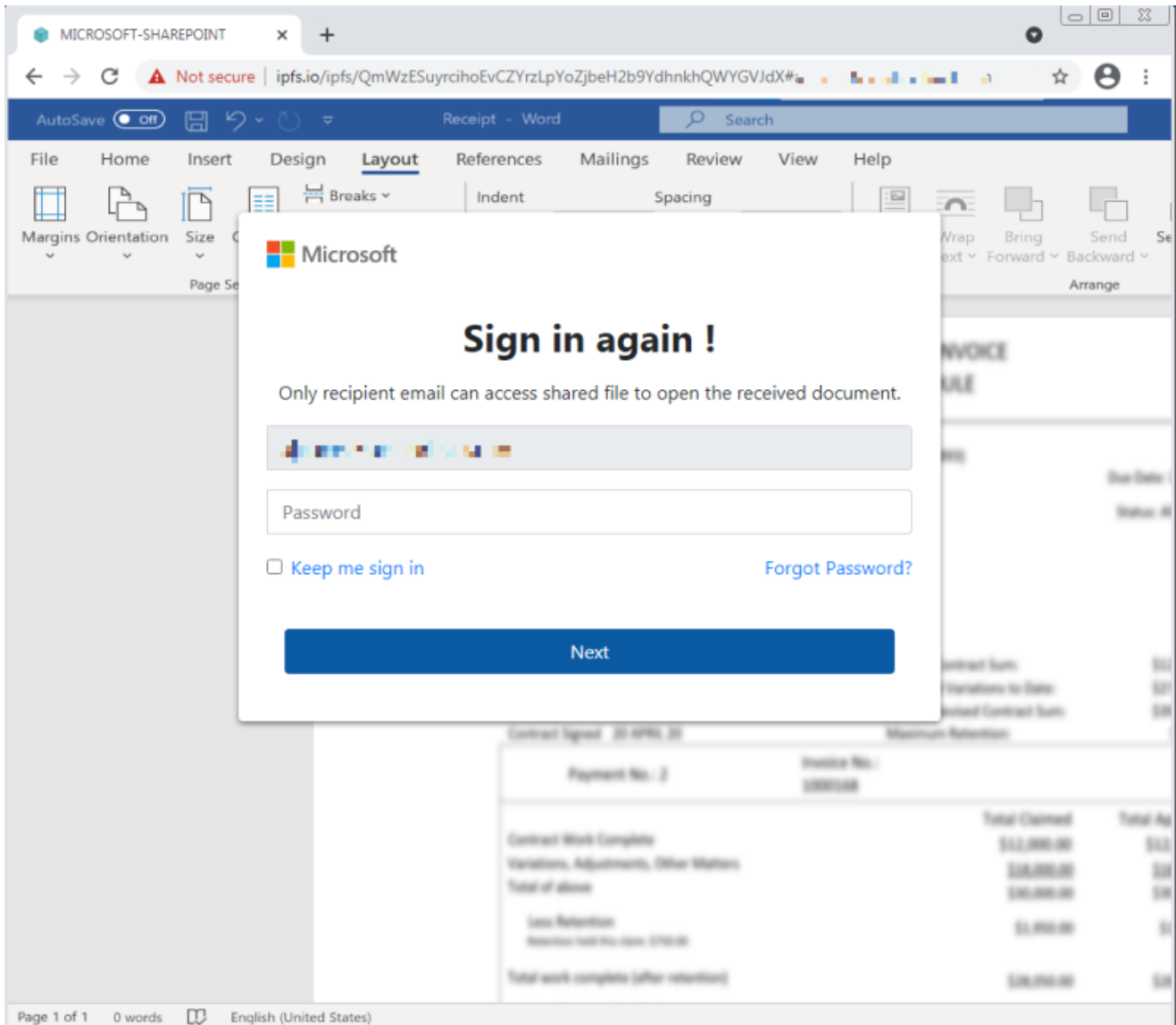


Figure 2. Obviously, a phishing website, but wait there's more.

As previously noted, the second and third URLs are identical. This is where things get interesting. When we checked out these URLs in a browser it appeared to look just like another run of the mill phishing site. The phishing URL's format is the victim's email address and is referenced on the fragment part (#). The fragment was used to auto-populate the email address field in the webpage. This [link](#) contains a further explanation of URL fragments. By removing the fragment part of the URL containing the victim's email address, most of the web graphics disappear, making the login page look rather bland.

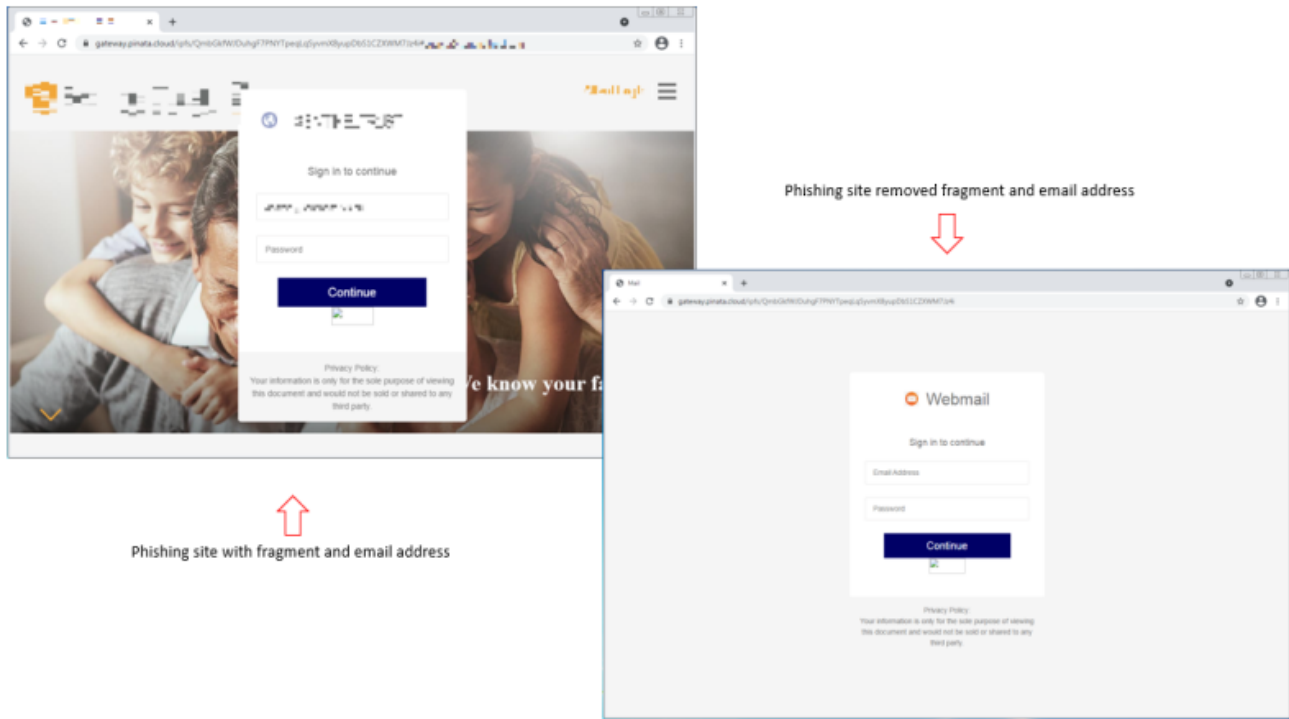


Figure 3. Wait, what? Another phishing page? The original fully dressed up and a second kind of vanilla flavored.

Instead of using the email of the intended recipient, we played around with the URL. We crafted a dummy email address and username and used common email provider domains like gmail.com and outlook.com. The results were interesting.

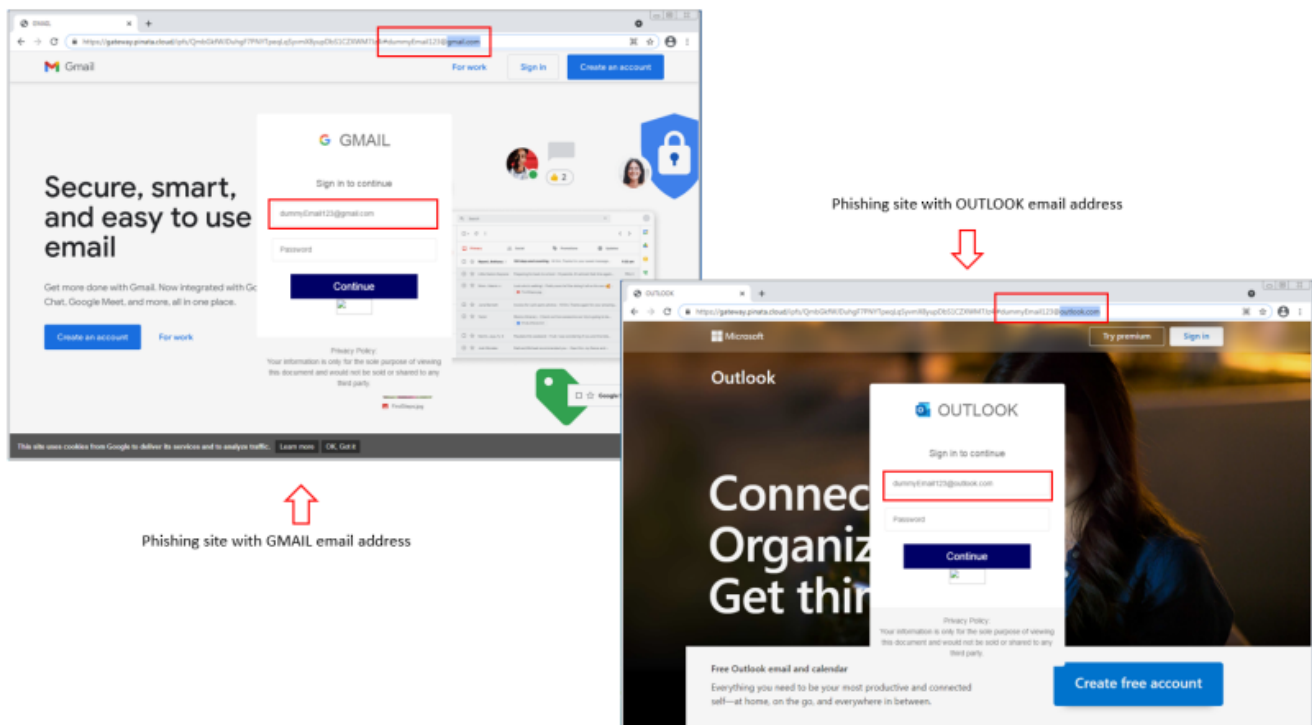


Figure 4. Crafted gmail.com and outlook email address domain name and the results.

This custom phishing site acts like a chameleon, by changing and blending its images to camouflage itself. There were four noticeable web elements that changed whenever we tested a crafted email address in the browser:

- The page's background
- A blurred logo
- The title tab
- The capitalized text of the domain from the email address provider.

We can dive deeper into how these changes happen on the website in the backend by viewing the source code. The site does not allow this action when we do a mouse right-click, but there's a keyboard shortcut for this in Google Chrome's browser, CTRL + U, which opens a new page tab containing the code.

We checked out the scripts in the source code and discovered how the threat actors created their behind-the-scenes trickery. In the JavaScript code, the declared string variable **my_slice** is commonly used. The supplied email address was validated with a regular expression then parsed to extract the domain name.

```
418 {
419   var my_email = email;
420   $('#email').val(my_email);
421   var filter = /^[a-zA-Z0-9_\.\\-]+\@((([a-zA-Z0-9_\.\\-]+\.)+([a-zA-Z0-9]{2,4})+)$/;
422
423   if (!filter.test(my_email)) {
424     $('#error').show();
425     email.focus();
426     return false;
427   }
428   var ind = my_email.indexOf("@");
429   var my_slice = my_email.substr((ind+1));
430   var c = my_slice.substr(0, my_slice.indexOf('.'));
431   var final = c.toLowerCase();
432   var final_u = c.toUpperCase();
```

Figure 5. One variable that rules them all – *my_slice*

The Page Background

The iframe with ID **mainPage** was concatenated with text protocol `https://` and the variable **my_slice** to be its source attribute. This action pulls in content from the domain in the email address, and this helps make the webpage believable, so the victim won't notice that an incorrect webpage is being accessed.

```

313
314 </head>
315 <body id="top">
316 <iframe id="mainPage" src="" width="100%" scrolling="no" sandbox="allow-forms allow-pointer-lock
317 allow-popups allow-same-origin allow-scripts" style="position: absolute; height: 100%; border: none;"></iframe>
318 <body oncontextmenu="return false;">
319
320 <!
321
439
440 var mainPage = 'https://'+my_slice;
441 // var mainPage = 'https://webmail.staralliancebd.com/';
442 $("#logoimg").attr("src", "https://www.google.com/s2/favicons?domain="+mainPage);
443 document.getElementById('mainPage').src = mainPage;
444 // $("#mainPage").src(mainPage);
445
446
447

```

Figure 6. The iFrame code of the unclickable background.

The Blurred Logo

The code sourced the logo from Google favicon API. The **my_slice** variable was used in the API query to find the matching logo to make the phishing webpage realistic. The sourced logo seemed small, it was stretched, and that's why it looks blurry on the webpage.

```

328
329 <h2 class="text-center"><div id="top"><img src="" alt="" id="logoimg" style="width:28px; height:28px; margin-top:12px;">
330

```

```

493 $("#logoimg").attr("src", "https://www.google.com/s2/favicons?domain="+my_slice);
494 $(".logoimg").attr("src", "https://www.google.com/s2/favicons?domain="+my_slice);

```

Figure 7. The stolen logo's code.

The Tab Title and the Capitalized Text Beside the Logo

The parsed domain name variable, **my_slice**, then undergoes another parsing, disregarding the TLD, extracting the brand, and using it for the **logoname** global variable.

```

5 <title class="logoname">Mail</title>
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7

```



```

330
331 <span style="vertical-align: middle; padding-left: 5px" class="logoname">Webmail</span>
332 <br><br>

```



```

430 var c = my_slice.substr(0, my_slice.indexOf('.'));
431 var final = c.toLowerCase();
432 var finalu = c.toUpperCase();
433
434 $("#logoimg").attr("src", "https://www.google.com/s2/favicons?domain="+my_slice);
435 $("#logoname").html(finalu);
436 $(".logoname").html(finalu);

```

Figure 8. The parsed domain's code.

The code also included various input text field validators to check the text of the email address and password.

```

450 $('#submit-btn').click(function(event){
451   $('#error').hide();
452   $('#msg').hide();
453   event.preventDefault();
454   var email = $('#email').val();
455   var password = $('#password').val();
456   var msg = $('#msg').html();
457   $('#msg').text( msg );
458   ///////////////new injection////////////////////
459   var my_email = email;
460   var filter = /^(a-zA-Z0-9_\.\\ )-\@(((a-zA-Z0-9\\ )\.)|(a-zA-Z0-9){2,4})$/;
461
462   if ( my_email ) {
463     $('#error').show();
464     $('#error').html("Email field is empty...!");
465     email.focus();
466     return false;
467   }
468   if ( filter.test(my_email) ) {
469     $('#error').show();
470     $('#error').html("That account doesn't exist. Enter a valid @ email account");
471     email.focus();
472     return false;
473   }
474   if ( password ) {
475     $('#error').show();
476     $('#error').html("Password field is empty...!");
477     password.focus();
478     return false;
479   }
480   if ( password.length < 5 ) {
481     $('#error').show();
482     $('#error').html("Please enter password atleast 5 digits..!");
483     password.focus();
484     return false;
485   }

```

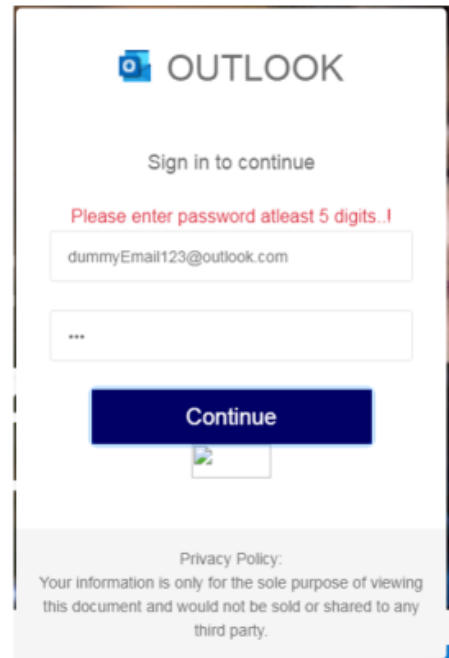


Figure 9. The text validators on the textbox fields.

As the victim keys in their password, a notification will appear, “Invalid Details, Please try again.” The submit button’s text shifts from **Continue** to **Sign in**. Unknowingly to the user, each time the button is clicked, the email and password data are forwarded to the attacker’s server. After three tries, it finally redirects the victim to the correct website. Once more, the variable **my_slice** is used by concatenating with “http://www.” to be the final landing page destination.

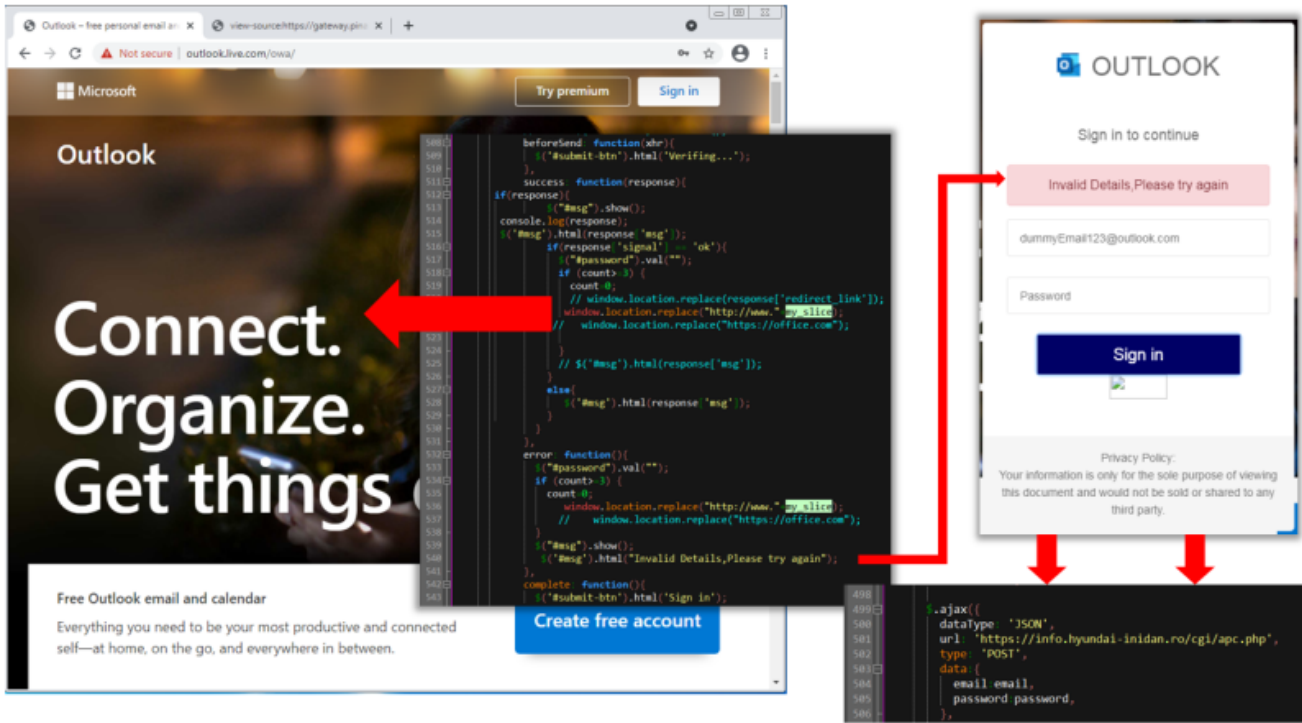


Figure 10. This is where the target's credentials are taken away and sent to the final landing page.

We tried sending a dummy email and password via POST monitored by the network monitoring application, Fiddler; unfortunately, the server is inaccessible.

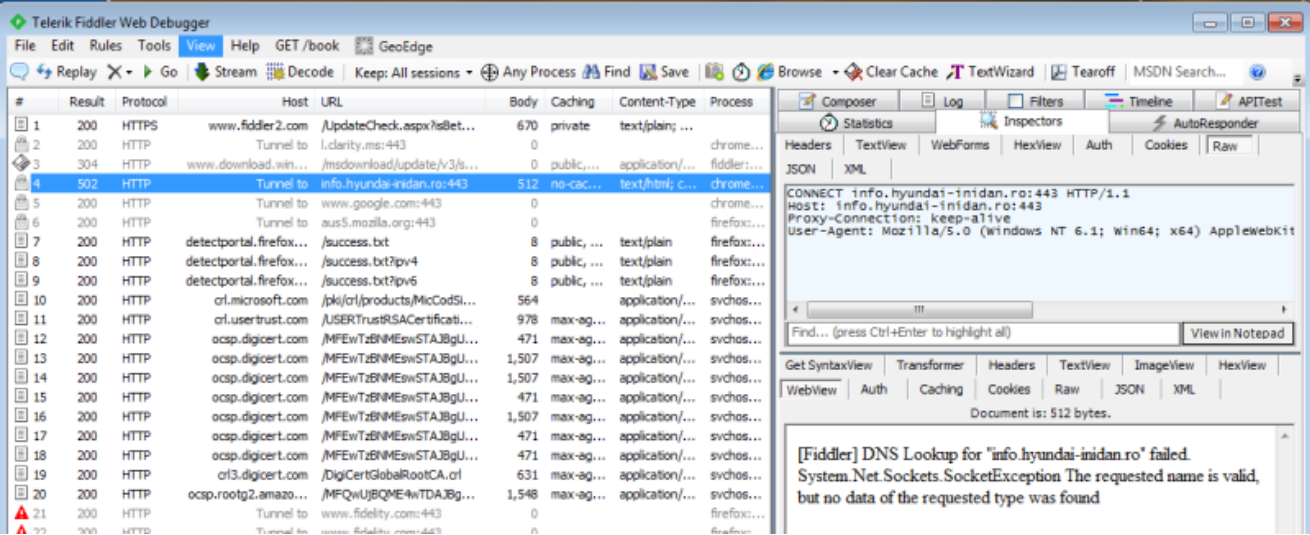


Figure 11. The attacker's server is currently offline, for now.

Phishing webpages are often taken down in a matter of minutes or become unavailable as soon as information security companies detect them as being malicious. These templated, or so-called chameleon phishing sites, are used repeatedly by malware authors using the clever tricks we just detailed to fool the user into thinking these pages are real. The phishers can easily customize the template and use other domains to host these scripts, allowing attackers to prey on unsuspecting users over and over again.

Trustwave MailMarshal defends against this phishing campaign.

IOCs:

URLs

- *hxxps://ipfs[.]io/ipfs/QmWzESuycihoEvCZYrzLpYoZjbeH2b9YdhnkhQWYGVJdX#[emailAddress]*
- *hxxps://gateway[.]pinata[.]cloud/ipfs/QmbGkfWJDuhgF7PNYTpeqLqSyvmX8yupDbS1CZXWM7Jz4i#[emailAddress]*
- *https://info[.]hyundai-inidan[.]ro/cgi/apc[.]php*