Enhanced Covert Channel Techniques for C&C Using Social Media Platforms

Kotaiba Alachkar & Peter Prjevara

Hiding in Plain Sight on Twitter

***Abstract*—In this paper, we analyse the current techniques used to embed data into social network posts, especially on Twitter, and develop an enhanced approach of obfuscating commands and payload within a message that looks like a benign, normal message. As a proof of concept, we created a tool called ”twitter-CandC” (Twitter Command & Control) [1]. With this tool we can rely on Twitter to initiate commands on a server. The server runs the parser daemon, which is capable of parsing tweets either by using the Twitter API or out-of-bound services. Herein we discuss in detail the advantages and disadvantages of each option, and the result of several experiments that we used to measure the success rate of our developed obfuscation technique. Lastly we recommend mitigation techniques, and offer directions for future development of the proof of concept, to allow larger payload obfuscation and bi-directional communication between client and server daemons.**

***Index Terms*—social media, covert channel, C&C, obfuscation**

I. Introduction

Over the years, researchers developed many ways to embed data into unused fields of network protocols (like IP, TCP, etc.), in order to send and receive data in a hidden way. Nowadays, malware coders are hiding their communication using social media traffic.

Social networks provide a convenient way of communication - they provide a reliable way to share information for billions of people world wide. One of the downsides of using these channels is that the information is evesdropped by the infrastructure providers - in the recent scandal involving Facebook, it was made clear that private discussions, and privacy sensitive information dispensed over the social network are being monitored, and (ab)used.

This fact doesn't just impact average users, but removes the initiative for system administrators and pentesters to implement privacy dependant system architectures. One potential use case of such privacy oriented system could be obfuscated post-exploitation control of overtaken systems, or bots of pentesters. As highlighted by Wildfire labs ”post-exploitation is a crucial step for successful compromise and further penetration deep inside the network”. [4] A concept that enables a truly private way of using social media platforms can be really powerful tool, for this purpose. In this paper we present a way to successfully hide payload in a tweet that looks perfectly normal, which was developed through analysis of existing techniques and experimentation.

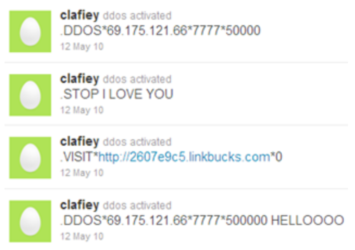
II. Related work

A. Telegram as C&C

Leveraging the standard encryption available in Telegram makes this application a really powerful and reliable C&C controller candidate. Tomas Susanka and his team recently also analysed the security of the application[6], which analysis resulted in no new security vulnerabilities. Unfortunately, the recent discovery of AES-CBC vulnerability might effect the integrity of the application, which is also using AES in CBC mode - this possibility should be furthe examined. Due to the fact that Telegram provides good support for creating messaging bots, it is easy to set up for monitoring. As the setup requires a telegram bot however, anonymity of such C&C infrastructure is difficult to achieve, potentially violating the second requirement defined in Section , Requirements Specification. This does not directly provide enhanced obfuscation, but rather implemented to allow for our demo to take place efficiently. We also believe that there could be different ways devised to create a bot that leaves little or no trace, but these were not investigated as part of this project.

B. Twitter as C&C

Several use cases can be found online that use the Twitter infrastructure as covert communication channel. These solutions can be grouped into two major groups: ones that allow for true command and control functionality and ones that only allow for Open Source Intelligence (OSINT) data collection. A comprehensive survey of Twitter used for true C&C functionality is provided by Lenny Zeltser [11]. This article inspired us, as the commands are (to say the least) fairly easy to detect, as they contain obvious ways of obfuscation. Examples include tweeting an encrypted data block as clear text, or even worse, the command as a hashtag (see one extreme example on Figure 1).



clafiey ddos activated
.DDOS*69.175.121.66*7777*50000
12 May 10

clafiey ddos activated
.STOP I LOVE YOU
12 May 10

clafiey ddos activated
.VISIT*http://2607e9c5.linkbucks.com*0
12 May 10

clafiey ddos activated
.DDOS*69.175.121.66*7777*500000 HELLOOOO
12 May 10

Fig. 1. Commands and targets are visible on Twitter in clear text

1) An API independent tool: The most effective OSINT tool is TWINT. It allows API free access to all tweets, which means that user credentials are not required to access, which supports the second main requirement we set out in Section , Requirements Specification. Due to the implementation specifics, it also allows for unlimited queries (the Twitter API limits to 3200 queries) per day[3]. In conclusion, this tool is a perfect match for our purposes, and it is the library of choice for our PoC, to parse tweets on our.

C. Combined techniques

An OSINT security researcher Wojciech, dived deeper into the topic, and produced a combination Twitter, Instagram, Youtube and the Telegram bots, to create creative covert channel techniques. In comparison to the previously assessed solutions, the researcher only offered enhancements by implementing a stealthy use case of Instagram as a covert channel. A command through Instagram is executed by placing a like on a specific picture, that contains a reference to a hard coded command in a form of a hasthag (such as location, delete etc.). The server side script is accessing the Instagram API to find the newest likes of a specific user, and executes the command on a first match basis [5]. This same idea is implemented on Twitter, where the administrator retweets an arbitrary tweet that contains one of the predefined commands. Besides all the downsides of using an API (such as traceability and ratelimiting - which has recently been even reduced dramatically [7]) are applicable with this idea, the commands are also mapped to be specific hashtags only. Nevertheless this article of Wojciech has inspired us a lot, and some of the examples are used in our later evaluation of our proof of concept.

D. Hipku IP address encoder

This interesting javascript library allows reversible encoding of IP addresses into verses that adhere to the rules of a Japanese Haiku. These are that a single verse has to be made up of 3 lines, where line one has 5 syllables, line 2 has 7 syllables and line three has 5 syllables, and its of a natural topic. However, to write a good Haiku, one also has to deliver a feeling, by ”strong details and detailed imaginary”. As a simple script of this size is not capable of creating the perfect haiku, the results often lack meaning and true cohesion. For this reason, it would not be an ideal candidate for obfuscation through social networks: it might get really suspicious if a known penetration tester becomes involved with writing Haikus. Still, this method is considered an interesting way to hide information in plain sight, even if it isn't used in our final proof of concept.

III. Requirements Specification

It is clear from the previous section that there are some existing approaches taking advantage of the in-place infrastructure provided by social networks, but to the best of our knowledge each of them lacks full anonymity and perfect obfuscation for one or more reasons. As a result of our preliminary investigation, we have identified the following points to be important for enhanced obfuscation on public channels. The resulting PoC should:

1) Not make posts or tweets that contain anything suspicious (such as clear text or encrypted commands, gibberish or unrelated text to the trends mentioned
2) Even if the daemon is compromised, be difficult to trace back to a specific user

The resulting PoC will:

1) Be fully automated. This is necessary to reduce the probability of mistaken commands introduced by human error

We assume the following hierarchy in place:

- A server (or collection of bots) under control of the implementer of the PoC
- The server or bots are capable of running the python scripts and have the libraries available

IV. Research questions

The following research questions were identified through our preliminary investigation:

- How can the current covert channel uses of Twitter be improved to provide further obfuscation of commands?

Sub-questions:

- How can complex command structures be implemented?
- How can the amount of false positives be reduced?
- Is it possible to detect such covert channel effectively?

V. Proof of Concept

A. Scope

To keep focus on our research topic, we assume that the C&C server is already in place, with a stable internet connection, and the daemon is able to execute the example commands on the server. Our PoC is strictly focusing on implementing improved command and target obfuscation methods.

B. Command Obfuscation

We are using the top 10 trend values from the past hour to encode commands, in the following manner:

- In order to be API independent, the tweets are parsed by the daemon from Trends24 [8]. We are using the BeautifulSoup Pyhton library to interact with the website.
- The *top 3* trends are mapped to 3 specific commands bash commands within our code in the following order. These commands are hard coded with the daemon, only their mapping changes every hour to the new dominating trends.
  1) echo
  2) ping
  3) nslookup

The API independence is also important because the API uses a special, proprietary trend retrieval algorithm. As the details of this algorithm are not known, it increases the possibility of generating false positives and negatives significantly.

It is possible to obfuscate commands further: for instance the 3rd, 4th and 5th most popular trends could be used instead of the top 3, but this would reduce the available space for additional commands. As there are 10 most popular trends for every our, every trend can receive its own command mapping, and up to 10 commands can be implemented with ease with our current model. Further command obfuscation techniques have been investigated, and discussed in Section VIII, Future Work.

C. Target Obfuscation

Individual IP address obfuscation is implemented by encoding the individual octets of an IPv4 address, each with the use of two emojis. As each octet can start with either 1, 2, or 0 in case of a two digit IP address, we decided to use the emojis depicted on Figure 2, to represent these digits. The emojis were selected as they meet the criteria of being context free, meaning that they can be used in positive / negative tweet contexts. Other emojis would also possibly meet this criteria (such as ”fire”, or ”smiling face”), but as our experiments show, our selection provided good enough obfuscation (Section VI, Experiments).

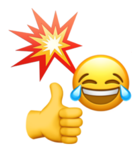
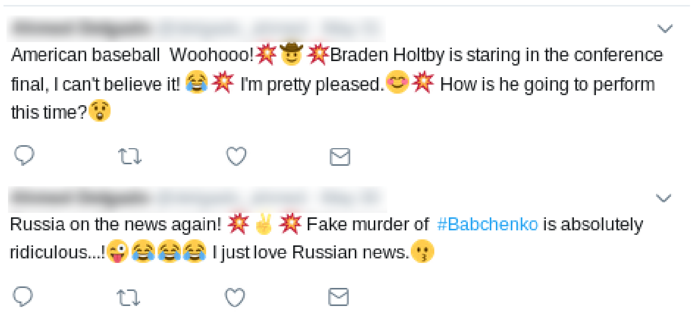


Fig. 2. Emojis used for representing numbers 0, 1 and 2 from an IPv4 octet

The remaining numbers that need to be represented are 3-99. For this we have done research into the 100 most popular emojis [2, 10], unfortunately they are not always fully context free (such as ”heart” or ”love letter”), so at the end we decided to use 97 faces. Through practical experimentation we found this works better, as one can have mixed feelings about a certain trending topic.

In order to further reduce the chance for detection, the following simple randomisation regime has been implemented: depending on the selected trend / command, a different mapping of emoji-to-number mapping is used. The only rule applicable is that the emojis only vary within the number groups - the 3 emojis depicted on Figure 2 are never used for higher numbers, and vice versa.

Figure 3, shows examples of command and target obfuscation at work.



[redacted]

American baseball Woohooo!💥🤠💥Braden Holtby is staring in the conference final, I can't believe it! 😂💥 I'm pretty pleased.😊💥 How is he going to perform this time?😲

[redacted]

Russia on the news again! 💥✌️💥 Fake murder of #Babchenko is absolutely ridiculous...!😜😂😂😂 I just love Russian news.😗

Fig. 3. Example tweets, with successful command and target obfuscation

D. Daemon parameters

We intended to make this daemon as dynamic as possible, in a way that it can be adjusted efficiently. Table I shows all possible configuration parameters that are implemented within the daemon.

Parameter	Description
timeInterval	Trend time offset (e.g. 1 for the past hour)
isDynamicLocation	Specify trends based on tweet location
countryName	Trend’s country of origin
userName	Twitter account username (or empty)
searchQueries	Advanced search query (like AND, OR, and NOT)
commandsList	List of commands with its own trend mapping
FetchTime	How often daemon fetch for tweets and trends

TABLE I
DYNAMIC CONFIGURABLE PARAMETERS WITHIN THE DAEMON

These parameters give the capability to configure the daemon as needed for a specific use case. In addition to keep the door open for future development and improvements.

VI. Experiments and Observations

To be able to measure the effectiveness of the proof of concept we have conducted an experiment, where volunteer students from the SNE group were asked to identify any suspicious tweets from a selection of 9 tweets. 3 tweets out of 9 were created by our PoC (similar as shown on Figure 3), 2 tweets were created with one of the applications mentioned in related work, and 4 did not contain any hidden meaning.

The results shown that large number of the volunteers could only identify the PoC tweets to be suspicious by chance, whereas all volunteers could clearly spot the 2 tweets produced by the PoCs from Related Work to be suspicious. Some users even tried with strategic discovery, without success.

In order to conduct software testing, a new twitter account was created to ensure that our personal accounts are not poisioned with test material. With regards to the account the following observations were made:

- Several bots have started to follow our test account. Bots could easily be identified by their account details being unrealistic
- Eventually the account was locked, possibly due to the meaningless test tweets

After the PoC was well tested, we sent some command containing tweets with on one of our personal twitter accounts, without observing any of the above side effects.

VII. Mitigation Techniques

In a corporate environment it is beneficial to use the Twitter API for the daemon, as an API call is less suspicious compared to an HTTP call to trends24. As a result, the simplest mitigation technique would be to disallow the use of Twitter and monitor for Twitter API flows or calls to trend summarising websites. If one knows the Twitter user account that the daemon is controlled from, and complex command structures are implemented, once can potentially monitor the tweet activity behaviour of the user, and notice any sudden increase to estimate payload conversion that is based on concatenated tweets.

VIII. Future Work

A. Issues during development

Due to the character encoding used in Twint [3], our daemon is only able to use trends that written in English. In addition, the advanced query search will only work with trends and subjects that do not contain space character as shown in Table II. This results in decreasing the number of possible trends that can be mapped to commands.

Example	Search
#abdc	Yes
Hello World	No
Football	Yes

TABLE II
Hashtags and Subject examples

In case of using the API dependent daemon, the mechanism used to display a list of possible trends for each user is not stable. A possible reason is that Tweepy (Twitter for Python) [9] is using its own method to generate the trends list. As a result, the trends list on the server is not perfectly matched with the one advertised on Twitter's trends section.

B. General payload obfuscation and bidirectional message sending

A more general payload obfuscation could be achieved by using base64 encoding on the payload, before using a similar conversion technique, as for the IPv4 addresses, discussed in Section V, Proof of Concept. If we for instance take 2 base64 characters (each encoding 6 bits, 12 bits in total), using a similar encoding scheme as desribed for the IP address encoding, we can encode the message with 4 emojis, where each emoji represents a digit between 0-9, so only 10 distinctive emojis would be required. As the number of emojis are dependant on the number of combinations, a mapping of the possible different combinations to their indicies would also have to be in place.

If we implement the highest number of combinations depicted in Table 2, then with only 7 tweets, that are each containing 10 emojis, we would be able to deliver 25 bytes of payload.

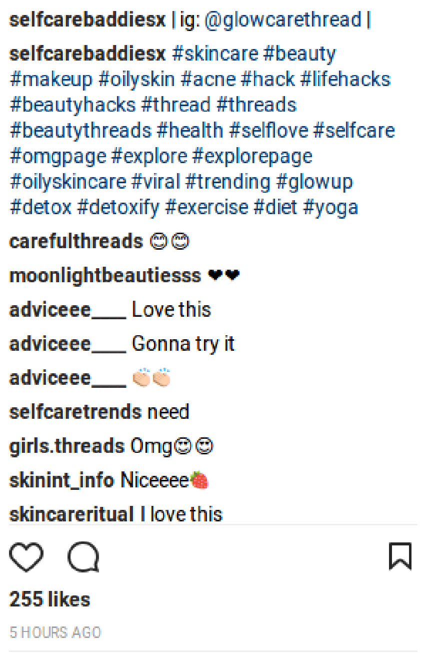
# of symbols	# of bits	# of combinations	# of emojis required
2	12	4096	4
3	18	262144	6
4	24	16777216	8
8	30	1073741824	10

TABLE III
Payload obfuscation

Bidirectional communication can be possible several ways: a simple like on the tweet by the daemon could confirm command execution success. Otherwise, to be able to extract payload, the bot could respond with several emojis following the encoding scheme - however this might become suspicious if the emojis are out of context, hence careful consideration is required when selecting the emojis containing the payload. The amount of information stored within the emojis would also be limited by the above described limitations.

C. Complex Command / Payload Structures

Since we are relying on the top 10 trends, use of multiple trends in the same tweet (which is common practice on Instagram - see Figure 4) *should not* be implemented, as the top trending values are rarely related, and verification of any valid relationships would require much effort if assessed dynamically.



selfcarebaddiesx | ig: @glowcarethread |

selfcarebaddiesx #skincare #beauty #makeup #oilyskin #acne #hack #lifehacks #beautyhacks #thread #threads #beautythreads #health #selflove #selfcare #omgpage #explore #explorepage #oilyskincare #viral #trending #glowup #detox #detoxify #exercise #diet #yoga

carefulthreads 😊😊

moonlightbeautiesss ❤❤

adviceee___ Love this

adviceee___ Gonna try it

adviceee___ 👏🏻👏🏻

selfcaretrends need

girls.threads Omg😊😊

skinint_info Niceeee🍓

skincareritual I love this

255 likes

5 HOURS AGO

Fig. 4. Multiple hashtags could allow sending complex datastructures

Instead however, if multiple tweets are sent where each of them is valid individually, the concatenation of such tweets can contain much more information. The trends change extremely frequently: at least 50% of the top 10 trends for the United Kingdom are different at the moment of observation, than what they were 4 hours ago, as it can be observed on trends24.in. This allows for an ability of continuous tweeting of new information, and hence sending significant amounts of payload, if combined with the above encoding scheme.

D. Daemon anonymity

As shown in Table I, the daemon allows to either specify a Twitter account username or by keeping it empty. In the former way, the daemon will go through that specific user wall and obtain all posted tweet from it. This is called the semi anonymous approach. Although this will only specify the username, which is totally public, without any extra credentials. But gaining direct access to the server might leak the actual user. The latter way is by not defining any user. The daemon will go through all tweets of a specific hashtag and search for a tweet that contains command-shaped presence. This way called a fully anonymous approach. The main advantage of it is that even earning direct access on the control server will not leak any traces or information about the actual user. However, it might generate additional false negative.

IX. Conclusion

In this paper we first presented a short survey of recent and existing techniques that allow hiding information in plain sight using social media. After carefully analysing these, we have came up with a set of requirements, that improves on the obfuscation techniques, anonymisation techniques and general detectability of the techniques analysed. We then presented a highly configurable proof of concept solution, that relies on the Twitter infrastructure and satisfies the requirements criteria. The source code of the PoC is modular, which enables fast deployment and easy configuration of the various different settings. We can conclude that the API based solution presented can be more beneficial to hide information in large corporate environments, where monitoring of the Twitter API is difficult, but the API independent solution provides better anonymity, and efficiency. We have verified the effectiveness of the PoC through a demo, and a survey, and we conclude our PoC can be safely implemented on personal accounts as it provides undetectable obfuscation. Besides implementing target and IP address obfuscation, we also made recommendations for improvements that can be implemented in the future to allow for hiding larger payloads - up to 30 bits / tweet using 10 emojis, and complex command structures.

References

[1] Kotaiba Alchkar and Pter Prjevara, 2018. Enhanced covert channel techniques for CC server using Twitter - https://github.com/kalachkar/twitter-CandC.

[2] Mona Chalabi, 2014. The 100 Most-Used Emojis - FiveThirtyEight - https://fivethirtyeight.com/features/the-100-most-used-emojis/.

[3] haccer, 2017. TWINT - An advanced Twitter scraping OSINT tool - https://github.com/haccer/twint.

[4] Wildfire Labs, 2016. bt2: leveraging Telegram as a command control platform - https://wildfire.blazeinfosec.com/bt2-leveraging-telegram-as-a-command-control-platform/.

[5] Wojciech OSINT Security Researcher, 2017. Command and control server in social media (Twitter, Instagram, Youtube + Telegram) - https://medium.com/@wojciech/command-and-control-server-in-social-media-twitter-instagram-youtube-telegram-5206ce763950.

[6] Tomáš Sušánka and Josef Kokeš. Security analysis of the telegram im. In *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium*, page 6. ACM, 2017.

[7] TechCrunch, 2017. Instagram suddenly chokes off developers as Facebook chases privacy - https://techcrunch.com/2018/04/02/instagram-api-limit/.

[8] Trends24, 2018. Amsterdam, Netherlands — Twitter Trending Topics Today — trends24.in - https://trends24.in/netherlands/amsterdam/.

[9] tweepy, 2018. Tweepy: Twitter for Python - https://github.com/tweepy/tweepy.

[10] Twitter, 2018. Real Time Emoji Tracker Twitter - http://emojitracker.com/.

[11] Lenny Zeltser, 2015. When Bots Use Social Media for Command and Control - https://zeltser.com/bots-command-and-control-via-social-media/.