Boten ELISA: A Novel Approach for Botnet C&C in Online Social Networks

Alberto Compagno*, Mauro Conti†, Daniele Lain†, Giulio Lovisotto† and Luigi Vincenzo Mancini*
*Department of Computer Science, Sapienza University of Rome. Via Salaria 113, 00198 Rome, Italy
Email: {compagno, mancini}@di.uniroma1.it
†Department of Mathematics, University of Padua. Via Trieste 63, 35121 Padua, Italy
Email: conti@math.unipd.it, {daniele.lain, giulio.lovisotto}@studenti.unipd.it

***Abstract*—The Command and Control (C&C) channel of modern botnets is migrating from traditional centralized solutions (such as the ones based on Internet Relay Chat and Hyper Text Transfer Protocol), towards new decentralized approaches. As an example, in order to conceal their traffic and avoid blacklisting mechanisms, recent C&C channels use peer-to-peer networks or abuse popular Online Social Networks (OSNs). A key reason for this paradigm shift is that current detection systems become quite effective in detecting centralized C&C.**

In this paper we propose ELISA (Elusive Social Army), a botnet that conceals C&C information using OSNs accounts of unaware users. In particular, ELISA exploits in a opportunistic way the messages that users exchange through the OSN. Furthermore, we provide our prototype implementation of ELISA. We show that several popular social networks can be maliciously exploited to run this type of botnet, and we discuss why current traffic analysis systems cannot detect ELISA. Finally, we run a thorough set of experiments that confirm the feasibility of our proposal.

We have no evidence of any real-world botnets that use our technique to create C&C channels. However, we believe that finding out in advance potential new types of botnets will help to prevent possible future malevolent applications.

I. Introduction

A *botnet* is a network of machines (*bots*) that are compromised by a malware controlled by an attacker (*botmaster*) to perform illegitimate actions. Information and identity theft, denial of service attacks, unsolicited messaging and spreading of new malware are just some examples of goals that botmasters aim to achieve with their botnets. Since their first appearance in 1990, botnets are considered one of the most serious threats against cyber-security. They are difficult to detect, hard to prevent, and their dimension can be as big as millions of infected machines worldwide [1].

The Command and Control channel (hereafter C&C) is the communication channel through which the botmaster sends commands to the bots and coordinates its fraudulent activities. In the past, C&C channels were often built over the well known IRC (Internet Relay Chat) and HTTP (HyperText Transfer Protocol) protocols, which provided a centralized command and control mechanism [2]. Today, such centralized C&C solutions are considered ineffective and new botnets adopt a decentralized communication mechanism. P2P (Peer-to-Peer) botnets [2] are the first attempt of decentralized C&C communication. They use the P2P protocol as C&C channel, which avoids the single point of failure of centralized botnets (i.e., the C&C server). P2P protocols do not provide any proper facility to hide botnets' communications from the current detection techniques (e.g., behaviour-based detection and signature-based detection [3], [4], [5]), forcing attackers to seek other undetectable solutions.

A promising direction to build novel C&C, that is resilient to the aforementioned detection techniques, is the exploitation and manipulation of existing communications on Online Social Networks (OSN) [6]. At the very basis, OSNs allow users to establish their own relationships with other users (such as *friendship* on Facebook and *circles* on Google Plus), as well as posting and sharing content of different nature (e.g., text and pictures) with them. A social botnet leverages relationships and social network communications in order to spread its messages in a stealthy way.

There are many possible techniques that hide information on OSNs [7], [8]. Mainly, these techniques provide privacy-preserving mechanisms to hide user's information from the OSN and from unauthorized users. One of these methods is to create covert channels using steganography techniques (e.g., image steganography and text steganography) [6], [9], [10]. An example of text steganography is given by Unicode text steganography [11]. This technique exploits non-printable characters and characters with identical visual representation in order to hide information within text messages which remains human-readable.

Contribution. In this paper we present ELISA (Elusive Social Army), a novel method of botnet C&C communication. ELISA takes advantage of the diffusion properties of the social network to efficiently spread messages to the botnet, exploiting trust relationships between users of the OSN.

Unlike other social botnets [6], ELISA uses Unicode steganography to build a covert channel, by injecting non-printable characters into the user-generated content that is

Alberto Compagno and Luigi Vincenzo Mancini have been partially supported by the European Commission Directorate General Home Affairs, under the GAINS project, HOME/2013/CIPS/AG/4000005057, and by the European Commission H2020 SUNFISH project, N. 644666.

Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission under the agreement PCIG11-GA-2012-321980. This work is also partially supported by the EU-India REACH Project ICI+/2014/342-896, the TENACE PRIN Project 20103P34XC funded by the Italian MIUR, and by the Project "Tackling Mobile Malware with Innovative Machine Learning Techniques" funded by the University of Padua.

posted on OSNs. This prevents ELISA from generating additional detectable network traffic, and guarantees ELISA to be unnoticed by OSN users.

We implemented a prototype of ELISA, and we run a thorough set of experiments that confirm the feasibility of ELISA and the effectiveness of its communication channel. In particular, our results show that a command message can reach 75% of the botnet in 72 hours. This delivery rate is higher than other social botnets [6], and it further confirms that ELISA might be a real threat in the future. Moreover, we argue that current detection techniques cannot be used to detect ELISA, and possible countermeasures might have some significant drawbacks on the user experience of the OSN.

We believe that this work is an important contribution in the proverbial "fight" against the diffusion of botnets. To the best of our knowledge, there is no evidence in the real world of any botnet using ELISA C&C channel. However, we think that discovering in advance new types of botnet will help in preventing possible future implementations.

Organization. The rest of this work is organized as follows. In Section II, we present the related work. In Section III we present the system model and the adversary model considered throughout the paper. In Section IV we present ELISA, we analyze its C&C channel, and we show its feasibility. In Section V, we describe the advantages of our solution. In Section VI, we report on the results of our thorough set of experiments, and analyze possible limitations and countermeasures of our system. Finally, in Section VII we draw some conclusions.

II. Related Work

Scientific literature and real-world malware have already shown different C&C channel techniques. In this section we provide a brief overview of notable examples of traditional C&C protocols (Section II-A) and novel proposals (Section II-B) of C&C over OSN.

A. Traditional C&C Protocol Botnets

In the first stages of development of botnets, the IRC protocol was the C&C medium of choice, since it was widely deployed all over the Internet. Agobot, Spybot and Sdbot [2] are some well known IRC based botnets. Nowadays, IRC is not a viable solution anymore, since its traffic is easily distinguishable from normal traffic.

In the botnet evolution, the next protocol used for the C&C communication was HTTP. Botnets like Rustock, Clickbot and BlackEnergy [2] made use of HTTP server-based C&C. In this type of communication, the bots contact C&C servers periodically to fetch commands and report control information. Using HTTP for the communication has some downsides. The first is that the resulting botnet has a centralized structure that is a single point of failure. The second drawback is that HTTP traffic can be monitored and inspected, and consequently the malicious communication can be easily identified.

To escape the drawbacks of HTTP communication, botnets moved to C&C based on P2P, using existing P2P protocols like BitTorrent, Gnutella and Overnet. Peacomm and Nugache [2] are two recent examples of P2P botnets.

Modern approaches to C&C in recent real-world botnets use fast-flux techniques [2] to prevent detection. However, detection of fast-flux service network has been proved possible in works such as [12].

B. Online Social Network Botnets

Exploiting online social network to create a botnet is not a novel approach. Besides some real-world botnets, as Koobface [13], Naz [14] and SocialNetworkingBot [15], there are several works that address this topic [16], [17], [6], [18] and investigate its potentiality [19]. We can discriminate OSN botnets by the type of the account they rely on. Some solutions use fake accounts to create C&C on the social network and control the bots. In Naz [14], SocialNetworkingBot [15] and DR-SNBot [17] the communication takes place by posting on – and reading from – a known and pre-shared OSN account that serves as a rendezvous-point. However, this approach has several drawbacks: the rendezvous-point becomes a single point of failure; the generation of network traffic makes it detectable with correlation and behavioural analysis. Our proposal differs from these since it is passive with respect to the network, meaning that it does not generate new traffic.

Other proposals use real victims' accounts to establish a communication between bots and botmaster. For example, Stegobot [6] uses image steganography techniques on the pictures the victim uploads, and can only be detected using ad-hoc entropy measures [20]. Unlike Stegobot, we leverage all kind of contents the victims share, obtaining a faster message transmission. SoCellBot [18] is a cellular botnet that aims to propagate and communicate through the social network instant messaging features by disguising the messages to make them appear legitimate. The drawback of this approach is the generation of additional suspicious traffic, which is easily noticeable by the victims, while our proposal is unnoticeable by infected users.

III. System and Adversary Model

In this section, we describe the system model (Section III-A) and the adversary model (Section III-B) considered throughout this paper. On top of these models, we build ELISA, our botnet proposal that stealthily communicates over existing social networks.

A. System model

We consider a model of online social network as the one defined in [21]. We assume the OSN is a centralized platform where a user creates a profile u, and establishes relations with other users, which form a list of profiles $\mathcal{F}_u$. We assume that every account is owned by one individual, and that relationships are bidirectional, which means that if $u \in \mathcal{F}_v$ then $v \in \mathcal{F}_u$. Moreover, the transitive property does not hold between relationships. A third profile $z \in \mathcal{F}_u$ needs to have a direct relation with v to be in $\mathcal{F}_v$.

Users share content or interests with all the other users in their relationship list, and only with them. Therefore, a user u can see the contents shared by every other user $v \in \mathcal{F}_u$ and vice-versa. Conversely, u cannot see any content shared by a user $w \notin \mathcal{F}_u$, since that content is private to w and the users in $\mathcal{F}_w$. This system can be precisely represented as an undirected graph $G = \langle V, E \rangle$, where V is the set of nodes, and E is the set of edges. Nodes represent users of the OSN and edges the relationships between the users. Without loss of generality, we do not take into account other social network entities for our analysis (like *public pages* or *public profiles* in Facebook).

B. Adversary Model

Here we state the properties of the botmaster and address how the malware will spread and establish the botnet.

Adversarial Properties. The attacker, or botmaster, is a malicious entity (individual or system) who has access to one or more social network accounts (henceforth *botmaster accounts*). Through his accounts, the botmaster interacts with the OSN. His actions are bounded by the set of legitimate actions the OSN makes available to any users. In order to avoid the botmaster's identity to be easily tracked, each botmaster account is a fake account (i.e., impersonates another person, or a non existing one).

These fake accounts need to have at least one infected neighbor (a *friend* on Facebook, someone *in your circles* on Google Plus), either by establishing a relation with one of them, or by spreading the malware over the social network itself. As an alternative, the botmaster can buy fake friends [22] to get an initial connection with the social network. For this reason, there is no limit to the number of botmaster accounts the attacker can introduce. Creating new accounts will always improve the propagation speed of the messages. In fact, issuing messages from different starting points in the graph of the social network minimizes the number of hops needed for their transmission. A public key scheme can be used to verify the authenticity of the botmaster.

Malware Spreading. We assume that ELISA's malware is a software that can intercept and modify the information exchanged between a user and the OSN (e.g., by installing malicious browser extensions [23]). Such malware infects a host by compromising the operating system or the web browser.

For the sake of simplicity, we assume that the malware spreads only over the social network and it infects the machine through which a user (henceforth *victim*) interacts with the OSN. Therefore, the infection spreads along the pre-existing relations (as *friendship* in Facebook), to later allow the malware to communicate. This was proven possible both by research efforts [24] and by real world malware like Koobface and Asprox [25]. We observe that using other means (e.g., emails, social engineering, file sharing) to spread the malware would be possible too. These means could lead to infect some hosts that cannot access the C&C channel, e.g., hosts where the users do not use any OSN, or victims forming graphs that are not connected to any botmaster account on the OSN. However, such hosts would still be useful to perform multi-hop spreading of the malware.

For ease of exposition, in the rest of the paper, we consider the OSN as the only mean for spreading the malware. Therefore, every infected host is used by a victim.

IV. ELISA

In this section we describe our proposal, ELISA, a botnet that uses popular social networking platforms as means to spread its messages. In particular, ELISA uses Unicode steganography to hide its messages inside victims' posts and relies on victims interactions (e.g., a victim posts a message on another victim's wall), to allow communications between the master and his bots. ELISA design meets the following goals (discussed in Section V):

- **Undetectability.** ELISA is not detectable by the current botnet detection techniques.
- **Unawareness by user.** A user is not able to realize if he has been infected by ELISA.
- **Reliability.** ELISA exchanges command and control messages in a reasonable time.
- **Resilience.** ELISA resists to random removal of nodes.
- **Authenticated command messages.** No one, other than the botmaster, is able to issue commands to the botnet.
- **Confidentiality.** Only the botmaster and bots can encrypt and decrypt ELISA's messages.

In the following sections we detail ELISA. In particular, in Section IV-A we describe ELISA's architecture. In Section IV-B we present the Unicode steganography technique used by ELISA. Then, in Section IV-C we analyze how the C&C channel is implemented in the botnet and, in Section IV-D, we describe how ELISA manages C&C channel integrity. Finally, we present our ELISA implementation in Section IV-E.

A. ELISA architecture

ELISA consists of two main components: botmaster and bots. A botmaster is an entity who has the properties described in Section III-B. A bot is a host infected by ELISA malware. To become active in the botnet, a bot needs to be used by a victim who interacts with some social networks.

Unlike C&C server-based botnets, in ELISA there are no components specifically built to manage the C&C channel. The relationships between victims form an overlay network over the OSN which connects the botmaster and the bots together. Hence, botmaster and bots communicate by exchanging command and control messages on the overlay network. Command and control messages are piggybacked on the honest content the victim posts on the OSN, therefore the normal victims interactions on the OSN deliver the messages to the whole botnet. Figure 1 shows the architecture of ELISA.

B. Unicode Steganography and ELISA alphabet

The Unicode standard differentiates between *characters* and *glyphs*. Characters are abstract representation of the smallest component of written language that have semantic value. Every character has its own *code point*, which is the corresponding

value in the Unicode codespace. The glyphs represent the shape that characters can have when they are displayed [26].

In Unicode, one can build two different types of covert channels [26]. The first type of channel uses control code points that have an invisible glyph and will not be displayed during the rendering (i.e., non-printing character). The second type of channel leverages the property that a single glyph can be represented by multiple code points: sorting them in different orders will lead to identical renderings, but the permutations can encode the hidden information.

Our tests show that popular OSNs (e.g., Facebook and Google Plus) are vulnerable to a covert channel that uses control code points with invisible glyphs. These characters are fundamental for internationalization (e.g., adapting web pages such that people with different languages can use them). We tested a set of such non-printing characters on the following operating systems: Microsoft Windows 7 and 8.1, Apple OS X 10.9 and Ubuntu Linux 14.04. We found that 11 characters on Facebook and 23 characters on Google Plus are not stripped away by server-side validation, at the time of writing.

With this set of characters, we build ELISA's alphabet for every target social network. We use the *n-ary Huffman* algorithm [27], where n is the number of non-printing characters the target social network does not strip away.

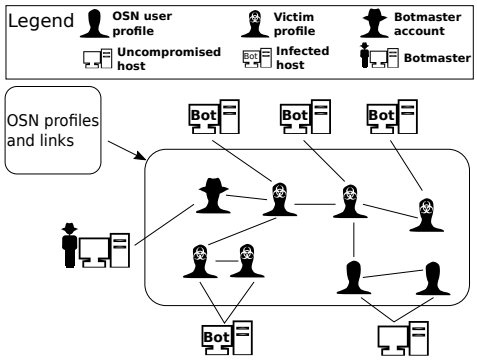

Fig. 1: ELISA architecture.

C. C&C Channel

In ELISA, C&C channel is realized over the online social network. The C&C messages are appended to the regular content that users post on the OSN. The transmission is carried out opportunistically *piggybacking* them on the normal user interaction with the platform. As for any other botnet, ELISA uses two different types of messages: *command* and *control*. *Command* messages are messages containing a command for the bots. Only the botmaster issues such messages. *Control* messages contain information (e.g., status updates, notification that an attack is started, error notification) that bots deliver to the botmaster. In this case, only bots create *control* messages.

Communication between botmaster and bots is secured by means of encryption and signature. In the following, we assume that a pre-shared symmetric key is installed in every bot and botmaster. Moreover, the botmaster owns a pair of public/private key. Confidentiality is obtained by encrypting every message with the symmetric key, while authenticity is obtained by signing command messages with the botmaster's private key.

Communication in ELISA consists of three different phases:

1) **Issuing phase.** In this phase, the sender of a message creates and injects the message on the botnet. Depending on the message type (i.e., command or control), this phase will take place on the botmaster or on a bot.
2) **Receiving phase.** This phase happens either on the botmaster that notices a control message or on the bots that notice a command or control message posted (by other bots) on the OSN.
3) **Forwarding phase.** In this phase, a bot posts an incoming message to the OSN, hence forwarding it towards the destination.

In the following we describe in detail the three phases forming the communication in ELISA.

Issuing phase. This phase takes place only once for each new message. Depending on the type of message, this phase will occur at the botmaster (commands) or at a bot (controls).

A botmaster creates a command message as follows:

$$cmd\_msg = encode(ENC_{symk}(cmd) \mid SIGN_{sk}(cmd)),$$

where cmd is the command instruction the botmaster wants to deliver to the bots. Therefore, cmd is encrypted with the symmetric key $symk$ and it is signed with the botmaster's private key sk. Then, encrypted command and signature are both encoded using the ELISA's alphabet described in Section IV-B. Finally, the botmaster appends the encoded command to some content, and posts it on the social network.

Control messages have a different structure from command messages. Bots create a control message in the following way:

$$ctrl\_msg = encode(ENC_{symk}(ctrl, d)),$$

where $ctrl$ is the control information and d is the bot distance from the botmaster. Distance information will be used in the forwarding phase to send the control message to the botmaster. As for a command message, the content of a control message $ctrl$ is encrypted and encoded with ELISA's alphabet. When the victim user posts some content on the OSN, ELISA malware intercepts the post, appends the control message to the content, and posts the resulting new content to the OSN.

Receiving phase. The receiving phase can happen on both bots and botmaster. In order to detect messages, ELISA's malware monitors the OSN while the victim is browsing it.

When a bot finds a command message on the OSN, it extracts the message from the post, decrypts the message and verifies the signature. In case the signature verification succeeds, the bot stores the message in a dedicated queue Q_{cmd} and it executes the command. In case the signature verification fails, the message is simply discarded. On the other hand, when the botmaster finds a command message on the OSN, he simply ignores it, since he forwarded it in the first place.

Finding a control message will cause the execution of this phase too. However, if the message is received by a bot, the bot will decrypt the received message and store it in the Q_{ctrl} queue. If the message is received by the botmaster, the botmaster will simply decrypt the control message and use the information for its purposes (e.g., updating its knowledge on the botnet). Both the botmaster and the bots keep a list of messages (control and command) that have been already processed, to prevent multiple executions and/or forwarding.

Forwarding phase. The forwarding phase only occurs at bots and it follows the receiving phase. This phase is triggered in a bot every time its victim posts a content on the OSN and there are messages on Q_{cmd} or Q_{ctrl} to be forwarded.

The main purpose of this phase is to forward command and control messages by appending them to a victim post. Depending on the type of the message, a bot uses two different forwarding strategies: (i) broadcast and (ii) shortest path. The (i) broadcast strategy is used to forward command messages, while the (ii) shortest path strategy is used to forward control messages towards the botmaster. Hence, when a victim tries to post a content on the OSN, ELISA's malware intercepts the content and modifies it appending a message according to the forwarding strategy.

While the broadcast strategy is easily implementable in ELISA (every bot reposts the message in order to forward it to all its neighbours), shortest path strategy needs a deeper explanation. To implement the shortest path strategy, we need to allow only the bot in the shortest path (from the bot issuing the control message to the botmaster) to forward the message. To this end, we assume that every bot knows its distance d from the botmaster (e.g., when a bot is infected the malware contains this information). Hence, before forwarding a message, a bot compares its distance d with the distance d' reported in the message. If $d < d'$, then the bot is at least one hop closer to the botmaster, hence it is in the shortest path from the bot that issued the message and the botmaster. Otherwise it simply discards the message.

D. C&C Channel Integrity Management

During the life of ELISA, it might happen that some events change the topology of the botnet. Changing the topology of the botnet can ruin the integrity of the C&C channel preventing in some cases the normal forwarding of messages. This happens for three main reasons: (i) a new bot joins the botnet; (ii) the botmaster account gets banned; or (iii) a victim account is no more available (e.g., account ban, malware removal, OSN relationship changes).

As explained in Section IV-C, every bot uses distance information from the botmaster for their forwarding decision. Therefore, all the three aforementioned cases make the distance value stored in the bots inconsistent, preventing, in some cases, control messages from being forwarded.

In case (i), every new infection changes the topology of the botnet which might create new shortest paths. Therefore, the actual distance of bots might be different from the distance they are currently storing. In order to keep up-to-date the distance information, during the receiving phase each bot checks its stored distance d with the distance d' contained in the received control messages. If $d' < d + 1$, the bot infers that it is part of a new shortest path towards the botmaster, hence it updates its new distance value with $d' + 1$.

Cases (ii) and (iii) are even more dangerous for the C&C channel integrity. In case (ii), the attacker needs to create another account and get in contact with at least one bot. This action will invalidate every bot's distance. In case (iii), the removal of a bot can make a shortest path no longer available, therefore some control messages might not be forwarded. In both the cases, bots are not able to reconstruct the right distance to the botmaster. To address this problem, ELISA includes the command *reset_distance*. This command forces the bots at one hop from the botmaster to set their distance to one, and all the other bots to the greatest possible value (i.e., infinity). Then, as for case (i), the distance d' contained in the control messages received after the reset will be used by every bot to update its stored distance d at the new value $d' + 1$. The check $d' < d + 1$ will force the update to start on the botmaster's neighbours and propagate towards the bot at the edge of the botnet.

To figure out if case (iii) occurs, botmaster needs to monitor the status of the C&C channel. For this reason, bots periodically report their infection state and distance to the botmaster. Then, comparing the number of controls received with the number of victims, the botmaster can estimate how many bots are still reachable in the botnet. In case this number reaches an alarmingly low level, botmaster will issue a *reset_distance* command.

E. Implementation

To show that an attacker can effectively set up ELISA's C&C channel, we realized a simple implementation of it that can run on both Facebook and Google Plus. We assumed the malware was already running on the hosts, therefore we only implemented messages transmission.

To emulate the botmaster, we wrote a simple utility that signs, encrypts and encodes the commands as defined in Section IV. This utility generates a set of invisible (non-printing) characters which can be pasted into the OSN post and shared to begin the transmission. To emulate a bot, we created a script which automatically receives and opportunistically retransmits the command and control information. We wrote the implementations in Javascript as a browser extension. The script activates when the user browses any page on the domains `facebook.com` or `plus.google.com`, respectively. We used AES to encrypt the messages and RSA with SHA-2 hashing algorithm to sign the commands. Keys for encryption and signing were both 256 bits length.

Bot script. This script runs on the infected hosts. When the victim is online on the OSN, the script continuously parses the DOM structure of the webpage, looking for C&C messages, and waiting for the user to post content. When the script finds the known invisible separator "|" in the page, it knows that the

separator will be followed in the HTML by a message. Then the script fetches, decodes and decrypts the invisible characters in the message. Finally, the command and the control messages are added to the Q_{cmd} and Q_{ctrl} respectively.

After processing the messages, the script looks for the $textarea$ where the user posts, and injects some code on the $keydown$ event of the $share$ button. The injected code activates when the keydown event is triggered. The code fetches the victim's post text, and then it performs three cleanup operations:

1) It updates bot's distance with the rule described in the previous section (by checking the distances reported by the controls currently in its Q_{ctr} queue).
2) It removes from the control queue Q_{ctr} all the controls which are not to be forwarded.
3) It removes from the command queue Q_{cmd} all the commands which have already expired, all the duplicated ones (the bot keeps a list of the identifiers of the commands which has already been processed), and those with an invalid signature.

Finally, the script takes the messages left in the queues, encrypts and encodes them, and appends the invisible output characters to the original user post. Figure 2 shows an example of the botmaster sending a command. Figure 3 depicts how the corresponding post would look on the victim's page (the message from the botmaster is actually invisible), and what the bot has actually received in the console.

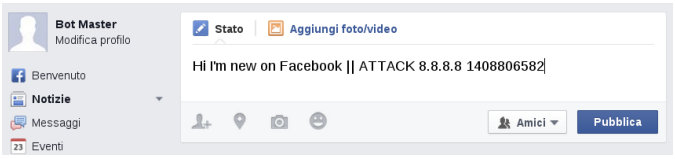

Fig. 2: The botmaster posts a command in Facebook.

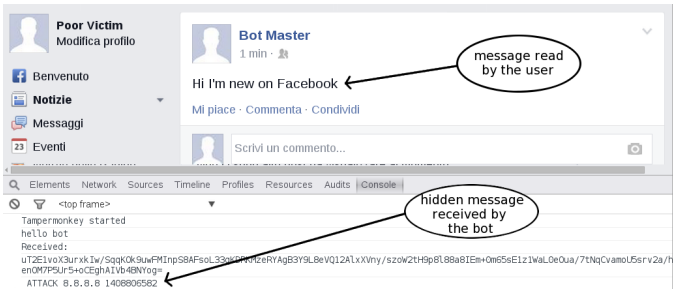

Fig. 3: A malicious message received on the victim's wall: the malicious message is not visible to the user (see top part of the figure), while the bot reads the actual command (see bottom part of the figure).

V. Features of ELISA

We now describe the main features and advantages of ELISA, when compared to botnets which make use of other kinds of C&C communication.

Undetectability. ELISA waits for the normal user interaction with the OSN to forward the messages. This behaviour guarantees the undetectability of the communication with respect to the state-of-the-art C&C detection mechanism. To validate this point, we discuss the ineffectiveness of the common defenses against C&C channels using the taxonomy defined by Khattak et al. [28]. As a prerequisite for the detection, the defender (i.e., the entity who wants to detect the botnet) must be able to access the network traffic. Detection techniques can be either active or passive [28].

The *active* C&C detection aims to take part in the operations of the botnet by *injecting* forged packets into suspicious network flows and observing the replies, or by *suppressing* suspicious packets. The defender then observes whether the potential bots activate some known back-up mechanism (like backup C&C servers, or use of DNS generation algorithms), in that case confirming the infected state. Since the communication in ELISA takes place through the OSN with SSL-encrypted connections, both techniques would ruin the user experience and disrupt his actions if applied on the real user's traffic. Furthermore, these techniques would not observe any active action from the bot, since the communication is passive.

The *passive* C&C detection aims at observing the network traffic, looking for known signatures, analyzing the traffic behaviour, or analyzing the communication graph. Signature-based detection systems would either fail to detect ELISA's C&C channel, or report a false positive. This is because a deep packet inspection is not informative in HTTPS communication, and the malicious content is mixed with the legitimate payload.

Detection systems based on traffic behaviour and correlation also turn out to be ineffective. This kind of analysis consists in extracting a number of features from the network flows, making some assumptions based on a typical bot communications behaviour, like timing patterns or reactions to the reception of command messages. Network flows features can be either based on temporal or size considerations. Once features have been extracted, the system uses machine learning techniques to either classify flows as malicious or legitimate (when training data are available, or obtainable using prior information), or to cluster together nodes which show similar behaviour, this way obtaining one cluster of bots. Works as BotHunter [29], BotFinder [30], Disclosure [31] use this kind of analysis, and BotMiner [32] combines a signature based and traffic behaviour approaches together to improve the detection rate. Although they proved to be very effective on real world traffic, these techniques are not able to detect the communication in ELISA. In fact, in ELISA C&C flows are exactly the flows the user generates with the OSN with his actions, so a system can not discriminate between them and a legitimate flow using communication features as described above. In that case, the detection mechanism would always report a false positive, because it would be reporting a real user action.

Another approach used by detection mechanisms is to build a communication graph, where nodes are hosts and edges are traffic flows from source node to destination node. Considering the typical bot communication patterns, the defender applies clustering techniques to the graph to find similar nodes, and eventually one of the resulting clusters is composed of infected

hosts. BotTrack [33], BotGrep [3], Entelecheia [4] are works that use this approach. BotyAcc [5] combines behaviour and graph analysis techniques together, and uses a different graph where nodes are traffic flows and edges are similarities between flows, which takes into account spatial information. For the same reasons stated above, these techniques are not able to identify the C&C channel in ELISA. Since the communication takes place through the user interaction with the OSN, the system has no means to discriminate between actions of bots or legitimate users. Therefore, the clustering techniques would not produce a cluster of infected hosts.

Unawareness by users. Unawareness is mainly provided by two different features of ELISA. First, ELISA does not need to post any new content on the OSN, but appends messages to content generated by the victim. This means that a victim will not notice any unexpected content. Second, thanks to the steganography technique described in Section IV-B, ELISA's messages will not be visible to the victim user. One might think that simple operations as copy-pasting a post, inspecting the HTML source code of the web page or a browser plugin can reveal the hidden message. However, ELISA's malware intercepts any user action to the web page and removes the hidden characters. This prevents a victim from knowing about ELISA's messages.

Reliability. Leveraging the OSN friendships graph we obtain a channel that is very efficient in reaching the infected nodes using a broadcasting technique. Our experimental results, which we discuss in Section VI, prove that the average delivery time for a message is adequately low, and that delivery is always guaranteed for most of the network.

Resilience. Using the social network makes our system immune to some common weaknesses of a botnet. First, as proved by Albert et al. [34], the removal of as many as 5% of random nodes in these kind of graphs does not affect the retransmission of messages. Thus, the removal of the malware, or the presence of inactive users do not significantly degrade our system capabilities. Second, the disclosure of the infected state of a bot does not allow the defender to gather relevant information about the botnet topology. Third, the botmaster can use multiple fake accounts to avoid exposure of his identity, and this model is immune to the banning of one or more of the botmaster's fake accounts: bots verify his identity by checking the command signature. Therefore, he could just create new accounts, as long as he manages to get contact with at least one bot that will start the retransmission.

Authenticated command messages. ELISA's design guarantees that botnet command messages can only be generated by the botmaster. In particular, the botmaster is the only one being able to issue command messages. This is obtained by signing each command message with the botmaster private key. At the reception of a command, bots verify the signature with the botmaster public key.

Confidentiality. Using symmetric encryption for the control messages ensures confidentiality to these information. This prevents defenders from gathering control information, such as the infection state. Even if a bot privacy is compromised, and the symmetric key is discovered, defenders would only be able to issue false control messages and read them, without compromising the botnet operations.

VI. Evaluation and Discussion

Analyzing the delivery time of a command message in our C&C proposal is crucial in understanding the effectiveness of the channel. This is somehow complex, because an accurate simulation of message propagation involves deep understandings of user behaviour on social networks. In order to run an analysis of the delivery time of the C&C channel in ELISA, we used the WOSN Facebook dataset [35]. This dataset represents an undirected graph $G = \langle V, E \rangle$, where nodes represent users, and edges represent friendships between users. We also assumed that interactions only take place between nodes connected by an edge, as described in Section III. Table I reports the following relevant properties of the corresponding graph: the number of nodes, the number of edges, the average degree, the diameter, and the *effective diameter* (i.e., the average number of hops required to reach 90% of the other nodes from a starting node [36]).

We describe the experiment in detail in Section VI-A. We report the results of our experiment in Section VI-B, and discuss them in Section VI-C.

Nodes	Edges	Average degree	Diameter	Effective diameter
63731	817035	25.640	15	4.97

TABLE I: Properties of the WOSN dataset.

A. Experiment design

One run of the experiment consists of the following steps: (i) simulating a simple malware infection on the graph; (ii) obtaining a new graph (*botnet graph*) by removing the non-infected users from the initial graph; (iii) simulating the issuing of a command message, its retransmission, and measuring the delivery time in the botnet graph. We now describe the first and the third phases in more detail.

Infection. To simulate the infection, we use the *decreasing cascade model* [37] on the graph, with a decreasing factor that reduces the probability of a node to be infected after every attempt, and makes cascades fit empirical observation on real-world social networks [38]. The *seed nodes* of the infection process are one random node with a degree higher than the average degree of the graph, selected as the botmaster, and $1/8$ of his friends selected at random. We pick the *transmissibility* factor λ (i.e, the probability for a node to be infected) randomly in the range $[0.041, 0.080]$. In particular, the lower bound 0.041 guarantees that the transmissibility is larger than the *epidemic threshold*, which means that the infection does not die out quickly [39]. We set the decreasing factor f to 1.5. When the infection process dies out, the nodes that have been infected form the botnet graph. We report the averages

over 1000 runs of some relevant properties of the resulting botnet graphs: (i) the average number of infected nodes was 1530; (ii) the average botmaster *eccentricity* was 6.759; and (iii) the average botmaster effective diameter was 4.881. The botmaster eccentricity is the maximum number of hops the botmaster needs to reach any other node [36]. Botmaster effective diameter is the effective diameter computed starting from the botmaster.

Message retransmission. This phase is carried out in the botnet graph created in the infection phase. We run a loop over ten simulated days, starting from midnight of the first day. We split the time of the day into fixed slices of 30 minutes, to perform a discrete-event simulation. We observed that increasing or decreasing the slices size does not significantly alter the outcome. At the start of every day, we draw the maximum number of daily posts for every victim from a Poisson distribution with a mean of 0.6, which is the average number of posts per day per user on Facebook [40]. To compute on how many time slices every victim will be online, we draw values from a Poisson distribution with a mean of 3. With this value, we represent an average online time of $3 * 30 = 90$ minutes, which rounds down the average time spent on Facebook by users (around 100 minutes [41]). Finally, to choose on which time slices every victim will be online, we set the probability to be online in a determined slice accordingly to the values measured in [40] for slices falling between hours 18-21 and 21-24. For the rest of the slices we set a very low probability. For simplicity, we suppose that all the victims fall into the same timezone.

To take into account that almost all of a user's interactions happen only with 60% of his friends, that we refer to as *relevant friends* [35], we stochastically pre-compute for every node who are the friends he will interact with. This does not change over the simulation. We assume that every user reads all the *news feed* posts from his relevant friends, and does not see posts from his other friends. For every simulated day, the message propagation is reproduced in the following way. We iterate over the time slices. On every iteration, every victim who is online reads what his relevant friends posted. If the bot has not received a command yet, and some of the victim's relevant friends posted it previously in time, the bot will receive it as soon as the victim is online. The victim will then post with respect of his drawn daily maximum number of posts. If this is the first time he posts since the reception of the command message, the bot retransmits it.

B. Results

The results of the experiment over 1000 runs are averaged and reported in Figure 4, along with standard deviation in errorbars. Analyzing the results of the experiment, we can see that the message spreading is reasonably fast. After 72 hours, the message already propagated to 75% of the botnet on average. We can see how low user activity during the nights and the first part of the days leads to plateaus on the message propagation. Notable increments happen during the evenings, when most of the users are online on the platform. The convergence of the spreading towards 80% of the botnet is fast, and happens on average on the fourth day of propagation, while it takes some more time to reach 90% of the botnet. However, the number of delivered messages is still notably higher compared to other passive C&C approaches such as StegoBot [6].

These results show that ELISA's C&C channel is able to deliver a command message in a relatively small time. We recall this is a simulation on a simple case of a botnet with only one botmaster account. Further investigation would be required to analyze the effects of more botmasters on the delivery time, and to test the channel on bigger datasets.

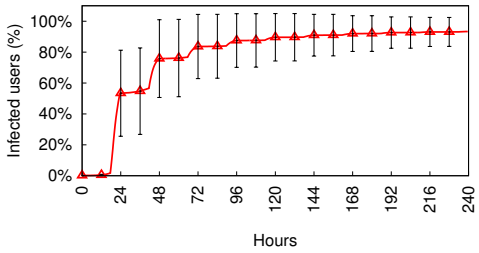

Fig. 4: Message reception on the infected component of the WOSN dataset. Values are the cumulative percentage of the total number of infected users who received the command message.

C. Discussions and Limitations

The C&C channel of ELISA has many advantages which make it undetectable and resilient to disruption from external observers. However, it is still vulnerable to countermeasures that OSNs can adopt. The easiest countermeasure is filtering the characters we use to build our covert channel. While simple blacklisting would be an effective solution to block ELISA, we argue that this is hardly the case. Many of the characters we use can not be removed from OSNs without impacting their features, in particular internationalization. For example, the bi-directional characters (characters code 206a, 206b, 206c, 206d) are fundamental for right-to-left alphabets like the Arabic and Hebrew ones. Other characters carry semantic meaning, which substantially changes the visualization of the surrounding characters (e.g., characters code 200c, 200d). Therefore, removing all the possible characters is probably not convenient to OSNs, since it may affect its functionalities.

Other possible countermeasures would then be more complicated, and involve statistical analysis of every post. Even though these techniques would be effective, we believe they are computationally intensive approaches.

A defender might also consider that the size of every post slightly increases due to the additional information we append. Further analysis would be required to understand whether this could be an effective detection method or if disguising the increased packet size as photos and other media uploads would suffice to prevent this countermeasure.

Finally, all of these actions need knowledge of the C&C channel we use. Given ELISA's stealthy nature, even understanding

where the covert channel is would probably require reverse-engineering, which is an expensive and difficult operation.

VII. Conclusions and Future Work

In this work we presented ELISA, a new OSN based botnet that propagates itself and the C&C messages using victims' social accounts. ELISA shows that this covert C&C channel approach is both feasible and not easily detectable. Our results demonstrate that the time needed for ELISA to spread messages is relatively small and acceptable in practical scenarios. In particular, our simulations reveal that in about five days, a message is delivered to its destination. Using a delayed approach, we think that a botmaster will be able to use ELISA to perform destructive attacks on the Internet.

We argue that a simple filtering approach on characters is not convenient. In fact, if such solution is implemented to counter ELISA, it will also have a bad impact on fundamental OSN functionalities such as internationalization.

As future work, we plan to empirically prove the undetectability of ELISA's covert channel by modern detection systems. We want to test those systems using a real world IP traffic dataset containing our malicious C&C packets. Moreover we plan to provide a more detailed messages propagation analysis, studying the impact of multiple botmaster accounts on the command propagation, using real traffic and posts from OSNs, analyzing the control messages propagation and comparing the performance of ELISA to real botnets.

References

[1] W. Chang, A. Mohaisen, A. Wang, and S. Chen, "Measuring botnets in the wild: Some new trends," in *ASIACCS*, 2015, pp. 645–650.

[2] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.

[3] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: Finding p2p bots with structured graph analysis." in *USENIX Security*, 2010, pp. 95–110.

[4] H. Hang, X. Wei, M. Faloutsos, and T. Eliassi-Rad, "Entelecheia: Detecting p2p botnets in their waiting stage," in *IFIP Networking*, 2013, pp. 1–9.

[5] S. Nagaraja, "Botyacc: Unified p2p botnet detection using behavioural analysis and graph analysis," in *ESORICS*, 2014, pp. 439–456.

[6] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov, "Stegobot: A covert social network botnet," in *IH*, 2011, pp. 299–313.

[7] F. Beato, M. Conti, B. Preneel, and D. Vettore, "Virtualfriendship: Hiding interactions on online social networks," in *CNS*, 2014.

[8] M. Conti, A. Hasani, and B. Crispo, "Virtual private social networks and a facebook implementation," *Transactions on the Web (TWEB)*, vol. 7, no. 3, p. 14, 2013.

[9] F. Beato, E. De Cristofaro, and K. B. Rasmussen, "Undetectable communication: The online social networks case," in *PST*, 2014, pp. 19–26.

[10] J. Ning, I. Singh, H. V. Madhyastha, S. V. Krishnamurthy, G. Cao, and P. Mohapatra, "Secret message sharing using online social media," in *CNS*, 2014, pp. 319–327.

[11] A. E. Ali, "A new text steganography method by using non-printing unicode characters," *Eng. & Tech. Journal*, vol. 28, no. 1, 2010.

[12] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton, "Real-time detection of fast flux service networks," in *CATCH*, 2009, pp. 285–292.

[13] K. Thomas and D. M. Nicol, "The koobface botnet and the rise of social malware," in *MALWARE*, 2010, pp. 63–70.

[14] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu, "Social network-based botnet command-and-control: emerging threats and countermeasures," in *ACNS*, 2010, pp. 511–528.

[15] A. Singh, A. H. Toderici, K. Ross, and M. Stamp, "Social networking for botnet command and control," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 5, no. 6, p. 11, 2013.

[16] L. Cao and X. Qiu, "Asp2p: An advanced botnet based on social networks over hybrid p2p," in *WOCC*, 2013, pp. 677–682.

[17] T. Yin, Y. Zhang, and S. Li, "Dr-snbot: A social network-based botnet with strong destroy-resistance," in *NAS*, 2014, pp. 191–199.

[18] M. Faghani and U. T. Nguyen, "Socellbot: A new botnet design to infect smartphones via online social networking," in *CCECE*, 2012, pp. 1–5.

[19] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, "On the impact of social botnets for spam distribution and digital-influence manipulation," in *CNS*, 2013, pp. 46–54.

[20] V. Natarajan, S. Sheen, and R. Anitha, "Detection of stegobot: A covert social network botnet," in *SecurIT*, 2012, pp. 36–41.

[21] N. B. Ellison *et al.*, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[22] M. Conti, R. Poovendran, and M. Secchiero, "Fakebook: Detecting fake profiles in on-line social networks," in *ASONAM*, 2012, pp. 1071–1078.

[23] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, "Hulk: Eliciting malicious behavior in browser extensions," in *USENIX Security*, 2014, pp. 641–654.

[24] G. Yan, G. Chen, S. Eidenbenz, and N. Li, "Malware propagation in online social networks: nature, dynamics, and defense implications," in *ASIACCS*, 2011, pp. 196–206.

[25] R. Borgaonkar, "An analysis of the asprox botnet," in *SECURWARE*, 2010, pp. 148–153.

[26] F. J. Mabry, J. R. James, and A. J. Ferguson, "Unicode steganographic exploits: maintaining enterprise border security," *Security & Privacy*, vol. 5, no. 5, pp. 32–39, 2007.

[27] D. A. Huffman *et al.*, "A method for the construction of minimum redundancy codes," *IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[28] S. Khattak, N. Ramay, K. Khan, A. Syed, and S. Khayam, "A taxonomy of botnet behavior, detection, and defense," *Communications Surveys Tutorials*, vol. 16, no. 2, pp. 898–924, 2014.

[29] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation." in *USENIX Security*, 2007, pp. 1–16.

[30] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, "Botfinder: Finding bots in network traffic without deep packet inspection," in *CoNEXT*, 2012, pp. 349–360.

[31] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *ACSAC*. ACM, 2012, pp. 129–138.

[32] G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection." in *USENIX Security*, 2008, pp. 139–154.

[33] J. François, S. Wang, T. Engel *et al.*, "Bottrack: tracking botnets using netflow and pagerank," in *NETWORKING 2011*. Springer, 2011, pp. 1–14.

[34] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, 2000.

[35] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *WOSN*, 2009.

[36] C. R. Palmer, G. Siganos, M. Faloutsos, C. Faloutsos, and P. B. Gibbons, "The connectivity and fault-tolerance of the internet topology," *NRDM*, 2001.

[37] D. Kempe, J. Kleinberg, and É. Tardos, "Influential nodes in a diffusion model for social networks," in *ICALP*, 2005, pp. 1127–1138.

[38] R. Ghosh and K. Lerman, "A framework for quantitative analysis of cascades on networks," in *WSDM*, 2011, pp. 665–674.

[39] G. Ver Steeg, R. Ghosh, and K. Lerman, "What stops social epidemics?" in *ICWSM*, 2011.

[40] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *EuroSys*, 2009, pp. 205–218.

[41] R. Junco, "The relationship between frequency of Facebook use, participation in Facebook activities, and student engagement," *Computers and Education*, vol. 58, no. 1, pp. 162–171, 2012.