Honeybot, Your Man in the Middle for Automated Social Engineering

Tobias Lauinger, Veikko Pankakoski, Davide Balzarotti, Engin Kirda
EURECOM
Sophia-Antipolis, France
{*lauinger, pankakos, balzarot, kirda*}*@eurecom.fr*

Abstract

Automated Social Engineering poses a serious information security threat to human communications on the Internet since the attacks can easily scale to a large number of victims. We present a new attack that instruments human conversations for social engineering, or spamming. The detection rate is low, which becomes manifest in link click rates of up to 76.1%. This new attack poses a challenge for detection mechanisms, and user education.

1 Introduction

Instant messaging (IM) spam is a prevailing issue on the Internet. Most often, spam messages contain a link to shady webshops, phishing websites, or malware. Current spam bots either directly distribute those messages, or they contact users and send a link when a user responds. Spamming is an automated process, hence, it can target a large number of victims. However, such spam can often be identified by cautious users because the text of the messages and the interaction with the spam bot are not very natural.

Phishing is another threat to information security. For example, RSA Security reported [9] a particularly sophisticated counterfeit bank website: Once the victims had entered their username and password, they were shown a live chat window where the phishers would interactively ask for the full name, phone number and email address. This human interactivity certainly makes more victims fall for the attack. On the other hand, it also limits the scale of the attack because it is no longer fully automated.

As these examples show, an automated system that supports human-like conversations could pose a great threat from a security perspective. In fact, this is the goal of Automated Social Engineering (ASE): An attack needs to be automated in order to reach a large number of victims, and it should be human-like so that more victims fall for it. Under these assumptions, ASE could be used by an attacker to reach even more complex goals such as automatically convincing users to help money mules with cash transfers, or making employees talk about their company's trade secrets.

Previous work on ASE [8] uses *artificial* conversations where the human victims talk to a computer program that mimics human behaviour. However, such attacks are difficult in practice and these programs can often be identified as such by careful users.

In this paper, we take ASE one step further and show how we can take control of *real* conversations between human users to implement ASE. The approach is similar to a traditional man-in-the-middle attack. Specifically, we are able to

- automatically bootstrap a conversation between two human users,
- influence the topic of the ongoing conversation,
- make the participants click on links that we inserted into the conversation, and
- apply techniques to make conversations last longer.

The design of Honeybot, our proof-of-concept implementation, is outlined in Section 3. To evaluate the bot, we carried out long-term measurements for up to 74 days on several IRC channels. Section 4 contains the setup of the experiment and a discussion of ethical considerations.

We analysed in detail when and why users click on links, and we compared the click rates of Honeybot to current spamming approaches. The results presented in Section 5 show to what extent users trusted our bot: Click rates reached 76.1% when we replaced links in the messages that the users were exchanging. We also carried out a feasibility experiment on Facebook and describe a refinement of the attack in Section 6.

While this paper focuses on a novel class of ASE attacks as a threat, we also elaborate on potential countermeasures against this attack in Section 7.

Altogether, this paper makes the following key contributions:

- We warn against a new threat posed by sophisticated ASE bots,
- we present a new man-in-the-middle attack that would allow effective and stealthy hijacks of human conversations to achieve malicious goals, and
- we provide details on the prototype implementation of this attack and present an experimental validation of its feasibility.

2 Background & Related Work

The attack that we present in this paper is applicable to text-based human to human communication. As representatives of the many existing systems, we will cover Internet Relay Chat (IRC) and Facebook in more detail because we evaluated our attack on these two platforms.

IRC is a server-based chat system where users can post messages on public channels (one-to-many) or exchange private messages between each other (one-to-one). Channels are administered by volunteers to contain spam and to enforce the general rules of conduct of the channel. IRC servers are organised in several distinct networks such as Dalnet, Efnet, Freenode, Undernet and Quakenet. More information about Internet chat systems and on classification of chat traffic can be found in [3].

Facebook is a social networking platform that allows users to foster social relationships. Users can send each other email-like messages, or exchange instant messages using an embedded chat feature. Communication is restricted to users that have previously established a friendship relation. However, as demonstrated in an earlier study [2], this process can be automated. Another study [1] suggests that it is relatively common for Facebook users to accept friend requests from strangers.

Related Work

Bots are computer programs that populate chat rooms and carry out automated tasks. In the following, we concentrate on bots used for spamming. Gianvecchio et al. [6] carried out measurements on Yahoo! chat. Considering only public messages, they detected four different types of spam bots:

- *Periodic bots* post messages at regular time intervals.
- *Random bots* post messages at random time intervals.
- *Responder bots* post automated replies to questions found in other users' messages.
- *Replay bots* record messages exchanged on a set of channels and replay them on the targeted channel when they match the current topic.

The authors also propose bot detection mechanisms based on entropy and machine learning. However, they do not analyse bots that send private messages, such as the bot that we have implemented. Click analysis of the links sent by bots has not been done so far, either.

Automated Social Engineering (ASE) is the process of automatically executing social engineering attacks. Social engineering targets human weaknesses of the user instead of vulnerabilies of a technical system. Spamming can be seen as a very simple form of social engineering (making users click), but usually ASE tasks are considered to be more complex.

As an example of ASE, Robert Epstein reports in the Scientific American Mind [4] how he was fooled for a considerable amount of time by a computer program that pretended to be a Russian woman. He exchanged romantic emails with "her" during several months and realised the hoax only because she was constantly evasive in her answers.

Huber et al. [8] describe an ASE attack cycle and implement a bot that chats with users on Facebook to recruit them for a malicious online survey. Their bot generates artificial replies using the Artificial Intelligence Markup Language. The authors assess in a Turing test [10] how human-like the bot's chatting capabilities are: The test subjects rated 80% the likelihood of talking to a bot after having exchanged three messages with the bot, as opposed to 4% when they were talking to a human. This supports our hypothesis that users can often easily distinguish computer programs and real humans after a very limited number of messages. In this paper, we therefore propose to instrument entirely human conversations.

3 The Attack

The general attack principle works with any chat system that allows the exchange of private messages. It is based on the traditional man-in-the-middle concept. Every instance of the attack involves two human users and a bot in the middle. Both users believe that they are talking to the bot, but in reality, their messages are forwarded back and forth as shown in the following example:

bot → Alice: *Hi!*
Alice → bot: *hello*
bot → Carl: *hello*
Carl → bot: *hi there, how are you?*
bot → Alice: *hi there, how are you?*
Alice → bot: ...

The bot looks perfectly human to both users because the entire conversation is reflected off the bot in the middle.

Furthermore, as all messages pass through it, the bot can eavesdrop on the communication, and it can influence the conversation by dropping, inserting, or modifying messages. We assert that if links (or questions) are inserted into such a conversation, they will seem to originate from a human user. Hence, the click (or response) probability will be higher than in artificial conversation approaches.

Our prototype, Honeybot, implements the attack outlined above. In the following, we examine each component of Honeybot in more detail.

Conversation Bootstrapping

To initiate a new conversation, Honeybot contacts users when they join the channel by sending a message such as *hello, wanna chat?* or simply *hi there*. Replies to this message are forwarded to a second user chosen at random from the entire channel population[1]. If no reply is received from that correspondent user after 180 s, Honeybot marks this user as unresponsive and contacts another user with a new *hello* message.

Maintaining Conversations

During the conversation, Honeybot normally forwards all messages between the users. It replaces nicknames in the messages as given in the example:

Alice → bot: *Hi, I'm Alice*
bot → Carl: *Hi, I'm bot*
Carl → bot: *hello bot!*
bot → Alice: *hello Alice!*

Honeybot is capable of modifying the perceived gender of the users talking to each other. This feature is useful to form more matching pairs on channels with an unbalanced gender distribution. The algorithm replaces words such as *man*, *woman*, and various body parts with the corresponding word of the opposite sex using a translation dictionary that contains the 60 most essential words observed during our test conversations:

Emil → bot: *I'm a gentleman, you know.*
bot → Fritz: *I'm a lady, you know.*

The algorithm does not replace names, and we have implemented it for English text only. Modification of languages such as French or Italian requires a deeper understanding of the grammatical structure of the sentence.

To determine whether the gender change algorithm shall be applied, Honeybot attempts to infer the gender from each user's nickname: We observed that many nicknames have the structure *Doris24f* or *NiceGuy58*. Honeybot first splits the nickname into its components: {*Doris*, *24*, *f*} and {*Nice*, *Guy*, *58*}. Then each component is compared with a list of male and female first names and with a list of typical male or female words such as *man* and *f*. A limitation of this algorithm is that it will incorrectly detect *Looking4female*, and it cannot infer a gender from terms that have connotations, such as *Ferrari72*.

Attacking

To carry out the actual attack, Honeybot can send a link or a question to one of the two users. Link attacks are triggered in three different ways:

Keyword links: Honeybot can automatically reply to keywords found in messages. For instance, conversations on dating channels often begin with the *asl?* question (which stands for *age, sex, location*).

Random links: Alternatively, Honeybot can randomly insert a new message with a link into the conversation. To make this look more natural, Honeybot requires both users to have exchanged a minimum number of real messages before inserting artificial messages. The text in the inserted message is generic as Honeybot has no knowledge of the current topic.

Replacement links: If one of the users sends a message containing a link, Honeybot can replace that link with its own one. This method looks the most natural, because the message has been written by a human for the current context, and the recipient may be expecting a link.

To make the users talk about a certain topic, Honeybot can insert probing *questions* into the conversation. As for the random link messages, we require both users to have exchanged a certain number of messages beforehand. Honeybot also uses this facility to incite users to exchange links: *can u send me a pic?*, *what is your favorite youtube movie?*.

Message Filtering

During our tests, we observed a varying degree of spam on all channels. Honeybot must not forward spam because we do not want Honeybot to be wrongly accused of spamming (and be banned). As an anti-spambot heuristic, Honeybot permanently blacklists a user if the first three private messages sent by that user contain links, email addresses or channel advertisements, or if that user is kicked out of the channel.

Additionally, Honeybot sanitises the messages exchanged between the users based on keyword matching. Honeybot does not forward messages that might contain email addresses or IM contact data (keywords include *@, h0t, mail, msn*). Honeybot also generally filters messages that contain a link (*http://, www*) if that link is not replaced with a bot-generated link. The goal of the latter is twofold: Firstly, we would like to avoid spreading malware links or spam. Secondly, we prefer users not to

[1] Excluded are users marked as unresponsive, as blacklisted, or users with administrative priviledges.

complain or ask questions about the links that Honeybot inserted into the conversation.

Stealth

To prevent detection, Honeybot never contacts users that have administrative priviledges. If Honeybot is contacted by such a user, it forwards the messages as usual, but it does not insert links or questions.

Honeybot sends at most one link and/or question to every user. Messages inserted by Honeybot can be composed of multiple elementary single-line messages. Honeybot simulates typing by varying the inter-message delay based on the length of the message.

4 Experimental Design

To show the feasibility of the attack, we tested Honeybot on selected IRC channels and conducted a limited experiment on Facebook.

4.1 Research Ethics

Given the capabilities of Honeybot, we had to carefully balance between our interest to evaluate those capabilities in the most natural environment on one hand, and the protection of the involved test subjects on the other hand. We identified the following risks for our test subjects:

- *Loss of time.* To reduce this risk, we tested Honeybot on general chat and dating-oriented channels. Users on chat channels often aim to kill time. Furthermore, we contacted each user at most once.
- *Revealing personal information.* We handled this risk by selecting channels where users are (i) to a large degree anonymous and (ii) very unlikely to share sensitive data. Also, Honeybot blocks the exchange of information that can be used for identification, such as email addresses or IM contact data.
- *Attack from the other dialogue partner.* This is always a possibility and it is not amplified by our attack. Nonetheless, Honeybot filters links to prevent infection with malware and also filters spam.
- *Emotional consequences.* Since we initiate a conversation between two humans, this can have repercussions such as disappointment, verbal fight, new friendship or even love. We believe that users are aware of these risks and possibilities when they join a channel and chat with strangers.

As shown above, a carefully designed experiment can minimise those risks. On IRC, there usually is no information available that allows the identification of a person. In contrast, such personal information is available on Facebook. This makes Facebook an interesting target for attackers, but does not allow us as researchers to carry out a large-scale experiment for moral reasons. Thus we carried out a small-scale experiment to demonstrate feasibility on Facebook, and used IRC to evaluate Honeybot on a larger scale. All data recorded during the experiments, although largely anonymous, has been deleted after the end of the evaluation phase.

Channel	Duration	Conversations	Users
Dating 1	74 days	2,574	19,464
Dating 2	61 days	1,427	9,337
Chat	14 days	1,122	25,552

Table 1: Honeybot IRC channel overview.

As recommended by Jakobsson et al. [5], we did not debrief our test subjects. There is a non-negligible risk that users would not understand the details of our experiment and become upset about a kind of attack that has not actually happened. For instance, they may not see the subtle difference between intruding their ongoing conversations, which we are not capable of, and starting a new conversation that we can influence with our bot.

We also did not ask users for their consent before including them as test subjects in our study. Users that are aware of participating in a study are likely to be more cautious than usual. Yet, we carried out the study only with users that responded to our messages and thereby accepted talking to the bot (i.e., stranger). Note that we did not intrude ongoing conversations.

As our institution does not have an Institutional Review Board (IRB), we had to base the decision about our experimental design on IRB recommendations in similar cases. After reviewing related literature, we believe that our experimental design meets the current ethical standards. We also consulted the legal department of our university and we were informed that our experiments are approved.

4.2 IRC Experiment

Our choice of IRC is motivated not only by the relative user anonymity, but also by the fact that well-operated IRC channels contain subjectively less spam than other systems. For example, we observed channels on Yahoo! chat containing only spam bots talking to each other.

For our experiments, we chose two *Dating* channels, one in English and one in French, and an Italian general-topic *Chat* channel. These channels have a sufficiently large user base, a topic that consists of leisure and killing time, and they cover different languages and nationalities. Table 1 summarises the duration of the experiment, the number of successfully bootstrapped conversations between two human users, and the total number of users seen during the measurement.

The *Chat* channel was mainly accessed via web pages: 98% of the users connected from a Java applet (PJIRC). We found its user interface not to be intuitive: For example, it does not highlight links in the traditional blue-and-underlined style, and following them requires a double-click instead of a single click. Furthermore, incoming private messages easily go unnoticed.

The *Dating* channels were mostly populated with standalone clients. 68% of the users on these two channels used *mIRC* to connect to the server. Between 20.4% and 22.9% of the clients could not be identified. The remaining 9.1% to 11.6% were spread among various clients that appear only sporadically.

Of particular interest for our evaluation is the fact that *Dating 2* contained a regular warning for users against clicking on links received in private messages (4 warnings observed in 6 hours, among other messages).

Traditional Spamming Bots

In order to comparatively evaluate the click rates of Honeybot, we implemented several known bot types:

Our *Periodic Bot* posted varying messages with a link on the public channel. In order to keep the disturbance low, we chose an inter-message time of 10 min. The whole experiment lasted 24 h.

The *Spam Bot* sent a private message with a link to users when they joined the channel. To limit the rate of the attack, we contacted only 25% of the users, did not attack any user twice, and sent at most one message per minute. This experiment lasted 26.5 h.

Our *Keyword Bot* used a subset of Honeybot's functionality: It contacted users with a *hello* message, but it never forwarded messages. The bot only replied to messages that contained keywords. We have not carried out separate measurements for this bot. Instead, we filtered the data obtained during the Honeybot experiments to extract only the links where the second user remained silent.

Honeybot Evaluation Setup

Honeybot identifies users based on their IRC username instead of the nickname: While users tend to change their nickname from time to time, this is uncommon for their username because it is automatically generated and hidden by most IRC clients. However, the Java applets used on the *Chat* channel seem to generate a new username for each session, which explains in part the high number of users in Table 1.

Some IRC servers implement anti-spam heuristics that affect Honeybot. For instance, they block messages if one user talks to too many different users simultaneously. To circumvent this, Honeybot maintains at most five concurrent active conversations (with an inactivity delay of ten minutes). On the *Chat* channel, messages that contain `myspace.com` are blocked. Honeybot sends `myspace.␣com` instead, but the link has to be reassembled by the recipient.

Bot Type	Dating 1	Dating 2	Chat
Periodic Bot	1.7%	1.7%	0.0%
Spam Bot	18.7%	14.0%	2.5%
Keyword Bot	37.0%	37.0%	0.0%

Table 2: Spamming bot click probabilities.

Honeybot sends three different types of links so that we can analyse the users' reaction to each of them: The *IP address* and *TinyURL* links point to our webserver that counts the click and subsequently forwards the browser to an external, popular website. The *MySpace link* points to a profile that we created for the purpose of this study. The profile's picture shows a person's back, and the text contains a clickable link to our webserver. We embedded another image into the profile that automatically loads from our webserver to count all accesses[2].

In order to make test subjects talk about a topic of our choice, Honeybot randomly inserts one out of two quiz questions into the conversation: Honeybot either asks for US president Obama's first name, or for the city where the famous Eiffel tower is located. We assume that the answers to these questions are widely known. To detect whether users are talking about that topic, we automatically match keywords such as *Barack*, and *Paris*.

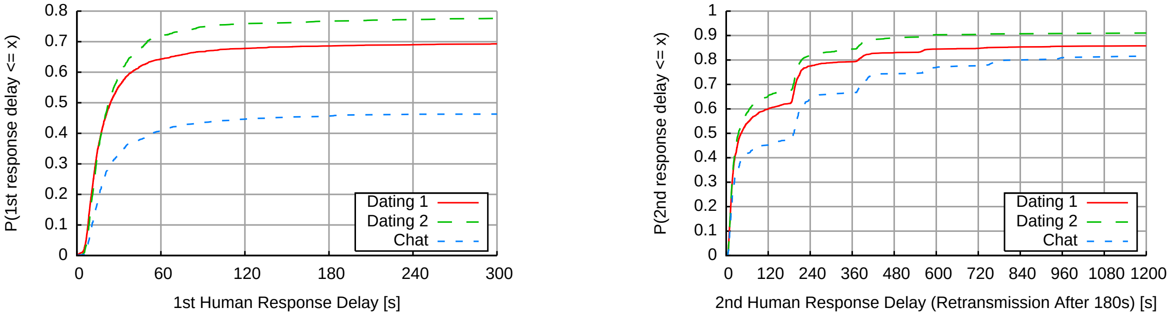
5 IRC Evaluation Results

Table 2 contains the click rate[3] results for the traditional spamming bots that we implemented for comparison with Honeybot. Only 1.7% of the online users clicked on the Periodic Bot's public links. On the *Chat* channel, the bot was banned after posting only eight links; on *Dating 1* and *Dating 2*, the bot could post 17 and 68 links, respectively (out of a theoretical maximum of 144 links during the measurement period). The click probability increased by a factor of more than eight with Spam Bot's private links, and it doubled again when we used the Keyword Bot's automatic answers to send links.

We defer the discussion of *Chat*'s low click rates, and begin to analyse the results obtained with Honeybot.

[2] For all three link types, our webserver counts clicks on a per-user basis using unique IDs retrieved from the URL or the HTTP referrer.

[3] We define the click rate as the number of distinct users that clicked on a link divided by the users that saw the link. Private messages are seen by one user; public messages are seen by all online users.



(a) First user to bot.

(b) Second user to first user (replacement after timeout).

Figure 1: Conversation bootstrapping response delay CDFs.

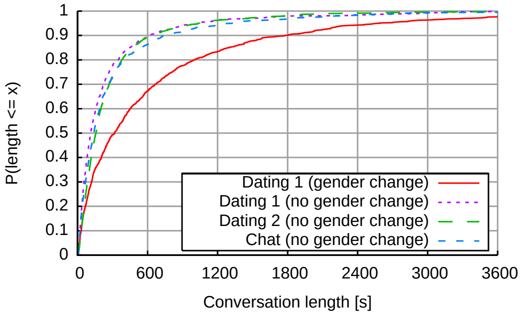


Figure 2: Conversation length CDF (seconds).

Conversation Bootstrapping

To initiate a new conversation, the bot sent a *hello* message to users that joined the channel. The response delay CDF is shown in Figure 1(a). The median delay ranged from 15 s on *Dating 1* to 18 s on *Chat*. The total response probability was 69.8% on *Dating 1*, 78.4% on *Dating 2* and only 47.3% on the *Chat* channel. The latter may be due to the user interface of the IRC applet where private conversations are not easily accessible to novice users, though it may also indicate that users on that channel were less interested in private conversations.

Every reply from the first user was forwarded to a second user randomly selected from the entire channel population. The CDF in Figure 1(b) shows the response delay, including retransmissions after evicting unresponsive users. The median response delay was between 29 s on the two *Dating* channels and 57.6 s on *Chat*. The response probability lay between 82.5% on *Chat* and 91.6% on *Dating 2*. It was not 100% because Honeybot stopped looking for a responsive second user once the first user left the channel.

With the results above, the probability of successfully bootstrapping a conversation was 59.5% on *Dating 1*, 71% on *Dating 2* and 38.1% on *Chat*. The median delay from contacting the first user to receiving a reply from the second user was 44 s on *Dating 1*.

Conversation Duration

To analyse how long the conversations lasted, we considered only the cases where messages were exchanged in both directions. We declared a conversation as finished when no messages were sent for a period of at least one hour. As shown in Figure 2, the median conversation length was 112 s without gender change. It increased to 317 s with gender change enabled on *Dating 1*. There were a few conversations that lasted considerably longer: 325 messages exchanged in 2.5 hours! 10% of all conversations lasted longer than half an hour.

The median number of human messages exchanged was 6 without gender change, and 12 with gender change. In comparison, in [8], the probability of a human identifying a chatterbot was 80% after only 3 messages. Assuming that users generally wish to converse with humans, we conclude that the test subjects were not aware of chatting with a bot.

Click Analysis

Table 3 shows the fraction of links that have been clicked, breaking them down into how Honeybot sent them (keyword replies, randomly inserted, or replacing a link), and into what address had been sent (IP address, TinyURL or MySpace profile).

TinyURLs were the most likely to be clicked, followed by MySpace profile links and IP addresses. This ranking is contrary to our first intuition. We would have expected MySpace links to obtain the highest click rate, because TinyURLs can hide arbitrary URLs whereas a MySpace link always leads to a profile.

As to the link type, link replacement can achieve the highest click rates because it is very difficult to detect that a message contains the wrong link. However, link

Channel	Link Type	Keyword	Random	Replacement	Total
Dating 1	IP Address	146/289 = 50.5%	126/211 = 59.7%	7/12 = 58.3%	**279/512 = 54.5%**
	TinyURL	163/266 = 61.3%	127/197 = 64.5%	14/16 = 87.5%	**304/479 = 63.5%**
	Myspace	154/273 = 56.4%	124/174 = 71.3%	14/18 = 77.8%	**292/465 = 62.8%**
	Total	**463/828 = 55.9%**	**377/582 = 64.8%**	**35/46 = 76.1%**	**875/1456 = 60.1%**
Dating 2	IP Address	56/126 = 44.4%	29/76 = 38.2%	0/1 = 00.0%	**85/203 = 41.9%**
	TinyURL	82/139 = 59.0%	38/68 = 55.9%	0/0 = 00.0%	**120/207 = 58.0%**
	Myspace	70/121 = 57.9%	31/61 = 50.8%	1/2 = 50.0%	**102/184 = 55.4%**
	Total	**208/386 = 53.9%**	**98/205 = 47.8%**	**1/3 = 33.3%**	**307/594 = 51.7%**
Chat	IP Address	0/0 = 00.0%	17/86 = 19.8%	2/4 = 50.0%	**19/90 = 21.1%**
	TinyURL	0/0 = 00.0%	25/91 = 27.5%	0/1 = 00.0%	**25/92 = 27.2%**
	Myspace	0/0 = 00.0%	3/78 = 03.9%	0/1 = 00.0%	**3/79 = 03.8%**
	Total	**0/0 = 00.0%**	**45/255 = 17.7%**	**2/6 = 33.3%**	**47/261 = 18.0%**

Table 3: The fraction of links that have been clicked.

replacement works only if users exchange links. This was not the case on *Dating 2* and *Chat*.

The performance of keyword-triggered links depends on whether the typical communication patterns include words that can be exploited as keywords, such as *asl?* on the *Dating* channels. None of the keywords that we had chosen for the *Chat* channel were used during the conversations.

We explain the comparatively low click rates of the *Chat* channel with the way links are handled in the user interface of the Java applet. As a special case, the `myspace.␣com` links had to be reassembled manually by the users. We are surprised that this reassembly has happened at all.

The periodic link warning posted on the public *Dating 2* channel did not keep the majority of users from clicking on a link in their private conversations. It may be that the display frequency of this warning was too low, or that such messages are generally ignored by users.

Many users clicked several times on their link, probably because they did not find what they were expecting. On *Dating 2*, of all links that were clicked, 22.35% of the IP addresses, 36.67% of the TinyURLs and 55.88% of the MySpace links were clicked more than once. For the replacement links sent on *Dating 1*, these figures were 57.14% of IP address clicks, and 64.29% of both TinyURL and MySpace clicks. We see this as a strong indication that the users were not aware of being victim of an attack. Furthermore, some users opened the same link multiple times using different browsers.

On *Dating 1*, 13% of all users who clicked on a MySpace link also clicked on the link that we put on the profile; On *Dating 2*, they were even 25.5%. This is surprising, taking into account that the MySpace profile was nearly empty and thus should have looked uninteresting.

The mean delay between the link and the first click was 148 s on *Dating 1*, 64.4 s on *Dating 2* and 109.1 s on *Chat*.

Quiz Questions

Honeybot performed keyword matching to determine if users were talking about the quiz questions that had been injected into the conversations. Considering only correct replies, the fraction of conversations about Obama was 51.6% on *Dating 1*, 32.6% on *Dating 2* and 28% on *Chat*. The median time difference between the quiz question and the last time any related keyword had been used was 82.1 s, 49.2 s and 57.9 s, respectively. The figures for the Paris questions followed the same trend: From 30.8% to 47.2% of the conversations with a median duration of 46.8 s to 60.7 s depending on the channel. We consider these figures a success, given that Obama is most probably not a natural topic on dating channels.

6 Facebook

The goal of our Facebook experiment was to evaluate if the attack is feasible on a social networking site. To that end, we created a female and a male profile. We sent friend requests to students of a local university, focussing on friends of the opposite sex. With these profiles, we conducted an experiment similar to the one carried out on IRC, with the only difference being that the male profile was used to send messages to female test subjects, and the female profile to send messages to male test subjects.

During the experiment, five chat conversations were successfully bootstrapped (after evicting four unresponsive users) with an average of 4.8 messages sent by each user. Four out of ten people clicked our TinyURL link.

The attack that we have described so far targets arbitrary, anonymous users. We believe that the attack could be improved to target private conversations between two specific users by combining it with the profile cloning attack [2] on social networks. The bot would clone the profiles of the two targeted victims, send new friend requests from the cloned profiles, and appropriately forward messages between the cloned and authentic profiles.

7 Countermeasures

Countermeasures against the attack can be classified into technical countermeasures against a specific attack instance, and general countermeasures to prevent the whole class of attacks.

As to the technical countermeasures, link spamming can be prevented by blocking links on chat servers. However, this is inconvenient for legitimate users, and it can be circumvented easily by obfuscating URLs, such as making a `myspace.␣com` out of a `myspace.com`. Alternatively, the server could display a warning next to every link. However, in the case of a link replacement attack, the victims are waiting for a link, hence this effort might be in vain. Message forwarding could be prevented by verifying that the same message is not sent twice within a certain time period. Nevertheless, this requires processing power on the server, and an attacker could modify the messages to circumvent the filter. Finally, nothing prevents the attacker from forwarding messages between two different, independent chat systems.

As we have shown, an attacker has many options to circumvent basic countermeasures. The best general protection for users would be to talk to verified friends only. Facebook implements such a feature, but users tend to accept friend requests from strangers [1], and profile cloning [2] can fool even the most careful users. Furthermore, in some cases, the ability of talking to strangers is a fundamental feature of the system, such as in dating services.

Trust-based mechanisms could provide users with a level of trustworthiness for their dialogue partners, even for strangers, so that users can make an educated decision about whether or not to pursue a conversation. For instance, TrustBot [7] implements such a system on IRC. However, in theory, attackers can build up a certain level of trustworthiness, and misuse it subsequently.

We believe that any technical countermeasure can be invalidated by an attacker with sufficient persuasive power on the victim. Technical countermeasures are neccessary to assist users in making their decisions, but they can only work if the users are aware of the problem. For that reason, user education is a primordial part of any countermeasure. We hope that this paper will contribute to this process.

8 Conclusion

In this paper, we have shown how human conversations can be automatically misused for social engineering. Our real-world experiments have shown that bot-generated questions that are not related to the channel topic are answered with up to 51.6% probability.

We have compared spam link click rates of different types of bots. The click rates range from 18.7% in case of unsolicited private messages to 76.1% when replacing existing links in conversations.

Heuristic detection of this new man-in-the-middle attack on instant messaging is difficult, because it hardly differs from true human behaviour. We believe that the ideas presented in this paper can lead to very realistic automated social engineering attacks. Besides malicious activities such as phishing, such attacks could also find uses in research and law enforcement. For instance, conversations with credit card dealers on underground chat channels could be automatically started with the goal of extracting useful information.

Acknowledgements

This work has been supported by the Austrian Science Foundation (FWF) and by Secure Business Austria (SBA) under grants P-18764, P-18157, and P-18368, and by the European Commission through project FP7-ICT-216026-WOMBAT. Merci à Conrad Berlinguette. Grazie a Paolo Viotti.

References

[1] Sophos Australia Facebook ID probe 2009. `http://www.sophos.com/blogs/duck/g/2009/12/06/facebook-id-probe-2009/`.

[2] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: Automated identity theft attacks on social networks. In *WWW '09*, p. 551–560, New York, NY, USA, 2009. ACM.

[3] C. Dewes, A. Wichmann, and A. Feldmann. An analysis of Internet chat systems. In *IMC*, p. 51–64. ACM, 2003.

[4] R. Epstein. From Russia, with love: How I got fooled (and somewhat humiliated) by a computer. *Scientific American Mind*, p. 16–17, Oct. 2007.

[5] P. Finn and M. Jakobsson. Designing and conducting phishing experiments. In *In IEEE Technology and Society Magazine, Special Issue on Usability and Security*, 2007.

[6] S. Gianvecchio, M. Xie, Z. Wu, and H. Wang. Measurement and classification of humans and bots in Internet chat. In P. C. van Oorschot, editor, *USENIX Security Symposium*, p. 155–170. USENIX Association, 2008.

[7] J. Golbeck, B. Parsia, and J. A. Hendler. Trust networks on the semantic web. In *CIA*, volume 2782 of *LNCS*, p. 238–249. Springer, 2003.

[8] M. Huber, S. Kowalski, M. Nohlberg, and S. Tjoa. Towards automating social engineering using social networking sites. In *CSE (3)*, p. 117–124. IEEE Comp. Soc., 2009.

[9] RSA FraudAction Research Lab. "Chat-in-the-middle" phishing attack attempts to steal consumers' data via bogus live-chat support. `http://www.rsa.com/blog/blog_entry.aspx?id=1520`, Sept. 2009.

[10] A. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.