A Double-Cross Policy against Social Engineers[1]

Guttorm Sindre

Dept of Computer and Information Science,
Norwegian Univ. of Science and Technology
guttorm@idi.ntnu.no

Abstract. Social engineering, where an attacker for instance calls up by phone and fools an employee into giving away sensitive information rather than getting the same information through a hacker attack, is may be the easiest way to break security in many organizations. Yet companies often spend much more on technical protection, and there has been relatively little research on the social aspects of information security compared to the technical ones. Commonly suggested processes advocate the systematic verification of identity for information requests that may be legitimate if the caller is who he claims to be, and flat rejection of illegitimate requests. This paper argues that an alternative approach, where the attacker is lured on, believing that the attack is succeeding, might in some cases provide even better protection.

Keywords: information security, social engineering, policies, business processes

1 Introduction

"Social engineering" is often claimed to be the easiest way to steal confidential information [1], and it has been suggested that the importance of social engineering will increase further as technical protection mechanisms get gradually better [2]. Yet the information security efforts of many organizations tend to be focused on technical solutions, and protection against social engineering has not improved much [3]. In a survey [4] where organizations were asked about their effectiveness in reducing various security issues, social engineering issues came 10th out of 10, i.e., organizations considered that they were less effective in dealing with this than with all the other 9 types of issues asked about (e.g., privacy breaches, access violations, etc.). Yet there is comparatively little research related to social engineering, meaning that it remains "one of the most under-researched topics in our field" [5], although there are some exceptions, for instance [6] defining a metric for social engineering resistance and reporting on an experiment where 37% of employees in a large IT company who received an bogus email by the researchers, gave away their passwords.

Social engineering attacks may be performed face-to-face, by phone, through email/internet or traditional letters. Paper letters are less common nowadays, while automated social engi-

[1] *This paper was presented at the NIK-2011 conference. For more information, see //www.nik.no/.*

neering attacks are on the increase [7]. Some attacks may be purely human-to-human, others may combine social and technical means, e.g., contacting an employee by phone or email and asking the person to go to a certain web page to click some link that is seemingly useful, but which really causes the installation of malware to the employee's computer [8], which can then be further utilized in a more technical attack. A number of example scenarios of successful social engineering attacks are given in [1]. Many attacks involve the following steps [9]:

(i) preparation phase or 'pre-texting', where the attacker finds out whom to call and whom to pose as, selecting an attack strategy. In many ways, pre-texting has become easier recently because many employees reveal a lot of personal information about themselves in social media [10],

(ii) attack phase, where some employee is conned into revealing some confidential information or compromising a computer resource.

(iii) post-attack phase, e.g., interpreting and utilizing the results from the attack phase. In many cases, a social engineering attack will work in several iterative cycles around these three phases, for instance first contacting one employee for one piece of information (e.g., the phone number to someone who was recently employed), then contacting this other employee for another piece (e.g., an access code), and then using this access code to obtain the confidential information which was the ultimate target.

Three attack scenarios are given in Table 1, but in much simplified versions compared to [1], for instance omitting the pre-texting. Scenarios A and B involve the acquisition of means of authentication. Scenario C does not, but instead exemplifies another technique often used by social engineers, namely to decompose one dubious request into two requests which, each taken separately, seem less dubious, thus more easily accepted by the target persons. If the attacker had simply asked the employee possessing the confidential document to fax it directly to an external number, many might have refused to comply without verifying the caller's identity. However, when the request is only to fax the document within the company building, it feels less dangerous. Next, the front desk clerk may not know the significance of the document, and the attacker's request to forward the document to an outside fax number does sound trustworthy – after all, a message addressed to this person just arrived at the front desk fax machine like the caller said it would?

Scenario A, [1] pp 61-64	Scenario B, [1] pp 132-140	Scenario C, [1] pp 64-71
1. Attacker calls employee X, posing as an information security employee checking that everyone has a strong enough password 2. Employee X reveals password 3. Attacker logs in to computer system as employee X 4. Attacker searches the computer system to find and copy any files of interest and/or use this user account as a basis for further attacks	1. Attacker calls employee X posing as a colleague from another branch, requiring a day code to be able to send some confidential info to X's branch 2. Employee X reveals the wanted day code 3. Attacker calls employee Y, using the day code to pose as a colleague requesting confidential info 4. Employee Y sends the confidential info to a requested fax number	1. Attacker calls employee X (posing as emp Y), asking to have confidential info faxed to the company front desk ("fax machine in my own dept is broken") 2. Employee faxes the document to front desk 3. Attacker calls front desk secretary (posing as employee Y), asks to have document forwarded to an external number (e.g., "meeting a customer") 4. Secretary faxes the document, attacker gets confidential info

Table 1: Three social engineering attack scenarios

Most proposals for counter-measures against social engineering focus on the formulation of clear security policies and employee training to increase their awareness and resistance against social engineering, see for instance [1, 3, 10-11]. The policies typically specify that sensitive and confidential information should only be revealed to an authorized person whose identity has been properly verified. If the request cannot be verified, it should be rejected. Especially, requests for computer passwords should always be rejected, because IT security or tech support personnel would not need your password to perform their tasks. For simplicity, we will henceforth call this the "Verify or Reject" policy, although it may really involve a number of policies for different kinds of situations. In contrast, we will propose an alternative policy called "Double-Cross". With this policy an employee suspecting that a request may be an attempt at social engineering can actually pretend to comply with the request, but doing this in a way that avoids giving the attacker something of value, instead luring the attacker into possibly revealing himself and his purposes.

The rest of the paper is structured as follows: Section 2 presents the basics of the Double-cross tactic. Section 3 discusses how this can be integrated into a holistic strategy for dealing with social engineering attacks. Section 4 reports on a paper case study. Section 5 discusses the advantages and disadvantages the proposed strategy compared to traditional strategies. Section 6 concludes the paper and outlines ideas for further work.

2 Motivation for the "Double-Cross" policy

If all employees consistently follow the "Verify or Reject" policy in situations where they should, this policy might work excellently. Indeed, the three attacks in Table 1 would then all have been thwarted already in Step 1, as indicated in Table 2.

Scenario A	Scenario B	Scenario C
1. Attacker calls employee X, posing as an information security employee checking that everyone has a strong enough password 2. Employee X **refuses** to reveal her password. Attack fails.	1. Attacker calls employee X posing as a colleague from another branch, requiring a day code to be able to send some confidential info to X's branch 2. Employee X **refuses** to reveal the wanted day code, as the caller should authenticate first, not the callee. The attacker proves unable to authenticate, and the attack fails.	1. Attacker calls employee X (posing as emp Y), asking to have confidential info faxed to company front desk 2. Employee X **demands verification** of the request (or insists to call back to the company-internal phone number of employee Y). 3. Attacker fails to provide sufficient verification and is unable to receive the call-back. 4. Employee X **rejects** the request. Attack fails.

Table 2: Employees correctly following the "Verify or Reject" policy.

The motivation for our alternative "Double-Cross" policy lies in the fact that no policy will be followed to perfection. Clear policies and awareness training yield fewer mistakes from the employees, but won't eliminate all mistakes. Let's say a skilled social engineer can successfully pull off scams like those in Table 1 in 80% of his attempts if the organization has had no awareness training. Now assume that training reduces the attacker's success ratio to 20%. So, previously needing an average of 1.25 phone calls to succeed with step 1, the attacker now

needs an average of 5 phone calls. Surely, the attack has become more cumbersome. Yet, if the victim organization is big enough that there are many employees who can be called, the attack remains feasible. If each phone call only takes 10 minutes, the attack can still be successfully performed within an hour, and even if an hour's work is needed for each new call to find out whom to call next, the attack will still succeed within the day. This is not much of a discouragement if the wanted information is of high value. The main shortcomings of the simple "Verify or Reject" policy are therefore:

- It may not help identify the attacker, as it simply cuts the contact short at the suspicion of something fishy. Professional social engineers do not make the critical calls from phones that can be easily tracked to themselves.
- It may not help identify which information the attacker was after. Often, the attacker does not ask directly for the target information in the first step, rather something else which will later help him obtain the target info.
- It may not contribute to reducing the likelihood of success for repeated attempts against other employees, thus only slowing down the attack, not preventing it.

This motivates our quest for an alternative "Double-cross" policy. The key idea is to do against social engineers much of what honeypots [12] are trying to do against technical attackers. As with honeypots, our purpose is twofold:

- *Luring the attacker to waste time*, i.e., thinking that he has obtained the needed information from the first employee contacted rather than immediately seeking other employees that might be more easily fooled.
- *Monitoring attacks*, ideally to identify the attackers and what information they might be targeting. If an early step of the attack is to obtain login credentials, the "Double-cross" policy can also be combined with automated honeypotting.

The ideas of the "Double-cross" policy are exemplified in Table 3, where it is applied on the same scenarios that we discussed earlier. A key feature of Scenario A is that the bogus password must appear to work, but in reality a login with this password does not lead the attacker to the user's account, only to a shadow user account in a honeypot system. Similarly, a crucial step in Scenario B is that employee Y, upon hearing the bogus code from the attacker, must not say "Sorry, that's the wrong code" (and certainly not: "Sorry, that's the bogus code"!) but instead appear calm and polite, checking the code and pretending it was ok.

Scenario A	Scenario B	Scenario C
1. Attacker calls employee X, posing as an information security employee checking that everyone has a strong enough password 2. Employee X pretends to reveal her password, but really gives a bogus password 3. Attacker logs in with bogus password – which causes login to a honeypot rather than the real system. Such a login automatically notifies a security engineer	1. Attacker calls employee X posing as an employee from another branch, about to send confidential info, requiring a day code 2. Employee X pretends to reveal the wanted day code, but instead gives a designated bogus code 3. Attacker calls employee Y, using the code, requesting confidential info 4. Employee Y recognizes the code as bogus, but pretends to accept it as the	1. Attacker calls employee X (posing as employee Y), asking to have confidential document sent by fax to company front desk (on a ruse that fax machine in own dept is broken) 2. Employee pretends to comply, agrees to fax the document to front desk 3. Employee immediately notifies front desk secretary as well as security personnel 4. Attacker calls front desk

4. Attacker searches the honeypot for files of interest. The security engineer monitors the attack to determine what information is sought and to trace the attacker if possible.	correct code, promises to send fax. Y notifies security personnel, so a process may be initiated to identify the recipient. A bogus document may be sent so as to see who picks it up at the other end.	secretary, asks to have document faxed to external number 5. Secretary pretends to comply, but forwards given fax no. to security personnel 6. Security personnel follows up to identify recipient if possible (which might include faxing a bogus document)

Table 3: Double-crossing scenarios

In all three scenarios in Table 3, the target company may get some knowledge of what confidential information is sought by the attacker, which would only happen in scenario C with the traditional "Verify or Reject" policy (cf. Table 2). With this knowledge, the company may increase the protection of the confidential information in question. One obvious step would be to notify everyone having access to the information that an attempted attack was observed and that new attempts are likely since the first one failed – hence these will be less likely to fall for another line of social engineering attack against this information in the near future. Other steps would be to ensure that paper printouts containing the sought information are either properly shredded or kept in secure archives, and that no files (including backups) are left on low security servers, in case the attacker turns to a more technical approach after the social engineering fails. Of course, one might say that all this should have been taken care of in the first place anyway, but security is always a balance between the perceived risk of a certain attack and the cost of the necessary countermeasures [13]. Since knowledge that an attack was actually attempted will increase the perceived risk, this might also mandate a level of protection not originally found necessary.

All three scenarios in Table 3 also hold some possibility (but no guarantee) that the attacker might be identified. Even if the security engineer succeeds in determining the location of the IP-address, this gives no guarantee of finding the attacker in person, since the attack may be routed through several computers or take place from an internet café far away, where the company has no possibility of sending someone to investigate. The chances are bigger in scenarios B and C, since the faxed document must physically be picked up at some point. Of course, the social engineer will use a publicly available fax shack rather than a machine which gives away his identity, and the most cautious would have the fax sent to one fax shack and then order this one to forward it to another, to avoid being confronted in the rare case that the victim has actually become suspicious and sent somebody to investigate. But even such a cautious tactic is open to counter-play from the company. For instance, a secretary can call the fax shack, telling that she was requested to send a document to employee Y and wanting to talk to him ("His mobile seems not to work, and I am a little uncertain which document he needed, as there are two different ones with the same title. Has he come to pick up the document yet?") Now, in case the fax shack clerk responds that employee Y is not going to come because he has ordered the fax to be forwarded to another place, the secretary could ask for this new number as this might help her get in touch with Employee Y, thus finding out where to send an investigator to get a visual of the person picking up the document anyway. In case the fax machine is far away and the company has no people nearby, it might be possible to hire a pri-

vate investigator from that local area, or have the local police investigate it if the requested information was sufficiently valuable and confidential that the police is willing to pursue it.

3 Bogus passwords and honey-pots

There are two possible solutions when it comes to bogus passwords: (i) in addition to its real password, each user account can have one bogus password, or (ii) each user could be able to define new bogus passwords for their user accounts on the fly. The former is simpler, while the latter has the potential advantage of placing an ID-tag on each different attacker, enabling tracing of their attempts at using the bogus password or day code for further attacks (for instance knowing that further usages of the bogus password x88wer#A are likely to be related to a person who called in at 8:35am June 22 claiming to be Dr Wong, while another bogus password is related to another caller). The ability to designate passwords or day codes on the fly would also work around cases where the user has forgotten the bogus, or where the attacker is not asking for the password but instead requests the user to try out an application for changing the password with a new password suggested by the attacker, cf. the attack scenario in [1] pp. 61-64. On the other hand, the practice of designating a new password for each suspicious caller, would also have a potential disadvantage. If the attacker somehow knows that the organization is using this particular double-cross tactic, he could then call in several times under different assumed identities and try to obtain a password. If getting different passwords each time, he would then assume they are bogus, while the password could be assumed to be correct if the same one is obtained twice.

A nice property of bogus passwords is that the only thing that needs to be protected about them is the fact that they are bogus. Hence, you could write bogus passwords on yellow stickers and keep one in your wallet, one under your mouse pad, etc., thus also fooling other attackers than those who call on the phone. In any case, it is of vital importance that the bogus password appears to work – i.e., a login attempt to a user account by means of a bogus password must seemingly lead to a successful login, only that – invisible to the attacker – this login is directed to the user's account in the honeypot rather than the one in the real system. Indeed, if care is taken that a bogus password always cracks more easily than the real password, this might even work with brute force cracking attacks, causing such attacks to successfully log in to the honeypot. Since most cracking applications are programmed to stop cracking a particular user account once the login succeeds, this will often mean that further cracking is abandoned before the real passwords are ever uncovered.

As with honeypot design in general, a major challenge will be to make the honeypot appear to be the real thing and keep the attacker busy – but at the same time avoid that the attacker really achieves anything. Hence, the eagerness to give the honeypot a realistic appearance must not be taken so far that it is actually filled with sensitive or confidential information. Nor should it be possible for the attacker to utilize the honeypot for further attacks. To make the attacker waste as much time as possible, the honeypot could appear to have a very slow performance, and every search (e.g., for files containing certain words) could turn up long lists of candidate files which would then have to be inspected more closely, but where error messages appear when trying to open them, maybe leading the attacker into further activities which may increase the chance of blowing his cover, e.g., the meta information of a garbled file could say that it is encrypted, giving a person to contact if you have forgotten your decryption key, or the

front page of a report could state that this is a brief version with confidential parts removed, with details about whom to contact to get the full version.

Honeypots could be similarly useful for scenarios where the attacker (for instance calling by phone) is not after a password but instead tries to fool his victim into downloading some malware to a company-internal computer, such as the attack scenarios of [1] pp. 55-60 and pp. 201-205. Downloading malware to the honeypot may be less safe than giving away a bogus password to the honeypot, so in this case the best policy might be for the employee to seemingly comply: "OK, I'll download this program – but it'll have to wait for an hour or so because I have to run to a meeting right now – can I call you back when it is done?" In most cases the social engineer will accept this little delay, by which the employee can notify a security engineer of the request, so that it can be investigated what kind of malware this is (e.g., keyboard logger, Trojan, …) and whether it is safe to install the malware itself on the honeypot (e.g., so that again, the attacker using the Trojan backdoor gets access to the honeypot while believing access is to the real system), or whether one should rather install something which produces bogus output similar to what would be expected for a real system (which could be better with a keyboard logger, since there won't be much keyboard activity related to the honeypot system). This bogus keyboard logger output could for instance involve bogus passwords, which might again lead the attacker into the honeypot.

4 Design of the double-cross policy

Employees who receive potentially dubious requests are often faced with difficult decisions: Does this request for information come from a legitimate person or not? In some cases the wrong decision would be made. Here we consider a *false positive* as the situation that the employee decides to comply with the request, although it was really a social engineering attack, whereas a *false negative* would be the opposite, e.g., the employee assuming a social engineering attack and therefore playing the double-cross, while the request was really legitimate.

Ideally, the double-cross policy should satisfy the following requirements:

1. "Double-Cross" should not increase the likelihood of false positives compared to the "Verify or Reject" strategy.
2. The damage from false negatives should not increase compared to the "Verify or Reject" policy.
3. In situations where double-cross is in any way perceived as risky, the employee could instead revert to a simple verify/reject strategy.

As for requirement 1, the main difference between the two strategies is how to behave *after* concluding that a request is illegitimate, so one might think that the likelihood of false positives would remain unchanged. However, this is not necessarily the case. If, for instance, the double-cross is perceived as more complicated or cumbersome, employees might become more likely to comply with some requests that they would have rejected with a simpler policy. On the other hand, in some situations it is possible to argue that "Double-Cross" might actually reduce the likelihood of false positives. Many of the scenarios in [1] portray employees who initially make a correct response according to a "Verify or Reject" policy (e.g., being reluctant to give away passwords or information), but then the attacker turns up the pressure in various ways, playing on urgency ("this document is necessary in an important meeting with a prospective customer"), empathy ("if I fail in this task, I will probably be sacked") or authority

(e.g., being or working for a high level manager in the company, so perhaps the employee will be sacked if refusing the request). These scenarios have the employees gradually folding under the increased pressure, thus complying with requests in spite of initial reluctance. For such conflict shy employees, the "Double-Cross" approach might actually be easier to perform. Pretending to comply with the dubious request, there won't be any conflict (at least not if the social engineer swallows the bait), and the conversation will therefore end before there is any increased pressure that might turn the employee to the wrong decision.

As for requirement 2, where the request is really legitimate but the employee treats it as an attack and initiates a double-cross, there may be trouble if the person requesting the information does not realize this and instead acts on the basis of some bogus information received from the employee. Hence, it seems appropriate to have some rules of engagement for the double-cross tactic, not activating it immediately upon the tiniest suspicion of social engineering, but only if it has been established as likely that this is really a social engineering attack. The likelihood will be considered to increase if one is asked to reveal passwords (which a legitimate support person should not need to know), if the person refuses the option that the employee can call back, etc.

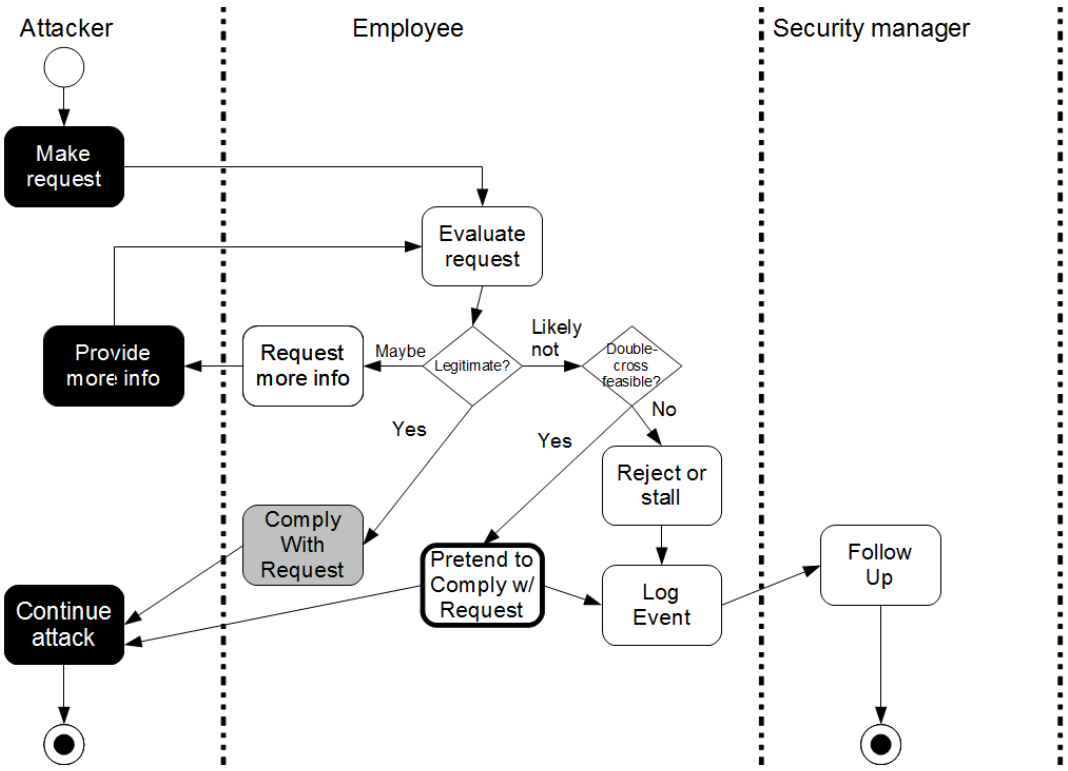

Figure 1: An outline of each employee's personal process for handling an attack

An outline of a suggested process is given in Figure 1, in the form of a mal-activity diagram [14] where the malicious steps by the attacker are shown as inverted, and the step which occurs if the employee is indeed fooled by the employee and thus complies is shown as grey. The double-cross step is shown with a thicker boundary than the other activities. The main goal of any policy and training against social engineering would of course be that as few employees as possible end up in the grey node, but this will not be entirely avoidable. With respect to the concern about potentially harmful effects of false negatives, the point about the

outlined process is that there will not always be a binary distinction between legitimate requests on the one side and social engineering attacks on the other, but also an interval in between where requests may be legitimate, but then again maybe not. In that case, the natural way to respond would be to request more information from the person. For instance, in some cases a request may be ok **if** the person indeed has the claimed identity, then a natural way to proceed is to ask for some identification or authentication (which the social engineer will typically not be able to provide, instead offering some excuse why this cannot be done). In other cases, the employee may pretend to be busy with other tasks and ask to call back to the other person in a couple of minutes (if the other person claims to be another employee, the call back would typically be made to this person's registered office phone or cellphone). Again, the social engineer would typically not want this callback since the recipient would instead be the real person, whereupon it would be revealed that the first call was a con attempt. Still, this gives important information to the employee, namely that this is a likely social engineering attack. Notice also that the arrow going right from the "Legitimate?" diamond is labelled "Likely not" rather than simply "No". This because it might require too much time from the employee to reach a stage of 100% certainty whether something is a social engineering attack or not, and if you run out of time and cannot establish for sure, you should definitely take the right way out rather than going to the "Comply" node.

4 Related work

To our knowledge, nobody else has published any paper directly suggesting a double-cross policy against social engineering. This, of course, does not necessarily mean that such a policy does not exist in practice. If a company has such a policy, it would normally not be inclined to make it publicly known, since that would cause them to lose an important element of surprise in the fight against attackers.

As also indicated in a literature review in [15], most of the proposals for preventing social engineering focus on learning and awareness training, i.e., to make people perform better in the traditional "Verify or Reject" policy, possibly combined with some software tools for combating automated social engineering in the form of phishing attacks and similar. [11] contains a chapter about detecting and preventing social engineering attacks, key advice is to train employees to recognize attacks and be aware of the value of various types of information, also it is important to run penetration tests and learn from these.

Closer to our proposal is perhaps one by Gragg [16], who coins the term "Social Engineering Landmine" (SELM) as a device to help discover and stop such attacks. Examples of SELMs are to have a person on each floor who is allowed to (or even encouraged to) question everybody who appears on the floor (and is not known from before to have permission to be there), i.e., to protect against physical entry, always suggesting to call back when unsure that the caller really has the claimed identity, always suggesting to put the call on hold if dubious requests are made, to have time to think and discuss with co-workers, and having some key questions that can be asked of an employee (and to which only the employee should know the answer) to identify this person over the phone, or alternatively presenting some false information about the employee in the conversation to see if it is corrected (probably the employee) or instead built further upon (probably a con man). Especially the final idea contains some element of double-crossing, yet it is still mainly a device to make employees perform the " Verify

or Reject " policy more perfectly, not a tactic to lure the attacker on to the next step (e.g., another conversation with another employee).

Another proposal containing some elements of double-crossing is [3], providing advice like "Do not give the caller any indication that you know what is happening" and "Be friendly and open, as if nothing is amiss. Imply that the information will be forthcoming" ([3], p. 20). Still, there is no proposal to provide the attacker with bogus information, only to stall the attacker by not providing any information at all, while at the same time not revealing that you realized it is a con.

5 Discussion and Conclusion

This paper has proposed an alternative approach to combating social engineering. Of course, this does not replace other approaches - training and increased security awareness will be equally relevant whether one is practicing the "Double-cross" or " Verify or Reject " policy. The "Double-cross" is of course also interesting to combine with proposals for technical support against social engineering - for instance employees might clearly benefit from an effective information system to spread knowledge once a social engineering attack is discovered, so that a clerk at the San Diego branch can log that someone posing as Jake Smith from the Chicago branch called in at 10:22, asked for a day code and got bogus code 4542, and if the attacker then makes a subsequent call to another employee in the San Francisco branch, now posing as Al Dowson from the Las Vegas branch, and presents the code 4542, this new employee will immediately be able to find that this was a bogus code just given, even if the next call takes place just a couple of seconds later. The system could then also have on-screen instructions for how to proceed to lead the attacker further into the net. Thus, an important feature of such an information system would be the ability to log incidents immediately, and with as little input from the user as possible - clearly time-stamps, number called from, etc. could be gathered automatically, and possibly other info could also be taken partly automatically from the spoken words of the call, so that only a couple of keystrokes or mouse-clicks would be needed from the employee. Ideally, the incident should already be logged by the time the caller hangs up, since the next call could then be going.

While we have presented some high level argument why "Double-cross" could work better, empirical investigations would of course be needed to establish whether this would really be the case. These investigations would be quite demanding, at least to enable hypothesis testing, since one would need a considerable amount of test employees to go through some training against social engineering, one group being instructed to use a double-cross policy and the other to use a verify or reject policy - but otherwise the two training programmes should be exactly equal. Subsequently, one would need a penetration testing company to perform a number of social engineering attacks against these employees and make some measures, e.g., how many attacks are successful, how many are unsuccessful but without busting the attackers, and in how many are the attacker actually caught or at least some other valuable information about the attacker or attack found out (i.e., possible benefit of "double-cross"?). This would have to be done in future work, and would also require a much more detailed description of the "double-cross" policy, as well as development of training material.

References

1. Mitnick, K.D. and W.L. Simon, *The Art of Deception: Controlling the Human Element of Security*. 2002, Indianapolis: Wiley Publishing, Inc.
2. Ivaturi, K. and L. Janczewski, *A Taxonomy for Social Engineering attacks*, in *CONF-IRM 2011*. 2011, Association for Information Systems: Seoul, Korea.
3. Power, R. and D. Forte, *Social engineering: attacks have evolved, but countermeasures have not.* Computer Fraud & Security, 2006. **2006**(10): p. 17-20.
4. Gallagher, K.P. and V.C. Gallagher, *The State of IT Security: Policies, Procedures and Practices*, in *1st Security Conference - Europe*. 2010: Örebro, Sweden.
5. Taylor, C. and N. Garrett. *Social Engineering: Where's the Research*. in *Twenty-Third Annual Computer Security Applications Conference (ACSAC'07)*. 2007. Miami Beach.
6. Hasle, H., et al., *Measuring Resistance to Social Engineering*, in *Information Security Practice and Experience*, R. Deng, et al., Editors. 2005, Springer Berlin / Heidelberg. p. 132-143.
7. Lauinger, T., et al., *Honeybot, your man in the middle for automated social engineering*, in *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*. 2010, USENIX Association: San Jose, California. p. 11-11.
8. Abraham, S. and I.S. Chengalur-Smith, *An overview of social engineering malware: Trends, tactics, and implications.* Technology in Society, 2010. **32**(3): p. 183-196.
9. Bhagyavati, B., *Social Engineering*, in *Cyber Warfare and Cyber Terrorism*, A.M. Colarik and L. Janczewski, Editors. 2007, Yurchak Printing Inc: New York. p. 182-190.
10. Townsend, K., *The art of social engineering.* Infosecurity, 2010. **7**(4): p. 32-35.
11. Hadnagy, C. and P. Wilson, *Social Engineering: The Art of Human Hacking*. 2011, Indianapolis, IN: Wiley.
12. McGrew, R. and R.B. Vaughn, Jr. *Experiences With Honeypot Systems: Development, Deployment, and Analysis*. in *39th Hawaii International Conference on System Sciences (HICSS)*. 2006. Kauai, HI, USA: IEEE.
13. Andress, M., *Suriviving Security. How to Integrate People, Process, and Technology*. 2002: SAMS Publishing.
14. Sindre, G. *Mal-Activity Diagrams for Capturing Attacks on Business Processes*. in *International Working Conference on Requirements Engineering: Foundations for Software Quality (REFSQ'07)*. 2007. Trondheim, Norway: Springer.
15. Nohlberg, M., *Securing Information Assets: Understanding, Measuring, and Protecting against Social Engineering Attacks*. 2008, Stockholm University, Sweden.
16. Gragg, D., *A Multi-Level Defense Against Social Engineering*. 2003, SANS Institute.