Who On Earth Is "Mr. Cypher": Automated Friend Injection Attacks on Social Networking Sites

Markus Huber* and Martin Mulazzani* and Edgar Weippl*

*SBA Research
Sommerpalais Harrach, Favoritenstrasse 16, 2. Stock, AT-1040 Vienna, Austria
{mhuber,mmulazzani,eweippl}@sba-research.org

Abstract. Within this paper we present our novel friend injection attack which exploits the fact that the great majority of social networking sites fail to protect the communication between its users and their services. In a practical evaluation, on the basis of public wireless access points, we furthermore demonstrate the feasibility of our attack. The friend injection attack enables a stealth infiltration of social networks and thus outlines the devastating consequences of active eavesdropping attacks against social networking sites.

Keywords: social networks, privacy, infiltration

1 Introduction

In this paper we present a novel attack to infiltrate social networking sites (SNSs) by exploiting a communication weakness of social networking platforms. Our results suggest that today's most popular SNSs including Facebook, Friendster, hi5, Tagged.com as well as orkut are vulnerable to our *friend injection attack*. Within an experiment we evaluated the feasibility of our friend injection attack on basis of Facebook.

Gaining access to the pool of personal information stored in SNSs and the infiltration of social networks poses a non-trivial challenge as SNSs providers start to devote more resources to the protection of their information assets. Our friend injection attack depicts a new method to circumvent state-of-the-art protection mechanisms of social networking services.

We created a proof-of-concept application named "FriendInjector" which injects friend requests into unencrypted social networking sessions. We were able to perform our attack at a rate of one injection every 1.8 minutes via wireless access points. Even though our evaluation was based on a relative small number of users, our attack can be carried out easily on a large scale by motivated attackers. Given the vast number of SNSs users (e.g. Facebook claims to have more than 350 million active users [1]) this could have devastating consequences.

The major contributions of our work are:

- Our friend injection attack, a novel attack on social networks which enables the retrieval of protected profile content.

– An evaluation on the feasibility of our friend injection attack on basis of public wireless access points.
– A discussion on protection measures against our friend injection attack.

The rest of the paper is organised as follows: section 2 introduces SNSs and recent attacks on users privacy. Section 3 outlines our main contribution, the novel friend injection attack. In the following, the archetype of our friend injection attack, the "Friend-Injector" application is described in section 4 and the findings of friend injection experiments are discussed in section 5. In section 6 we finally draw conclusions from our findings.

2 Background and underlying concepts

Within this section we first summarise related work in the area of social networking related privacy threats. In the following we discuss social networking sites which are vulnerable to our friend injection attack.

2.1 Related Work

Gross and Acquisti [2] as well as Jones and Soltren [3] were amongst the first researchers to raise awareness for information extraction vulnerabilities of SNSs. While their techniques were rather straightforward (automated scripts which retrieve web pages), their results eventually led to security improvements of social networking services. Recent publications devoted to information extraction from SNSs introduced elaborated methods such as the inference of a user's social graph from their public listings [4] or cross-profile cloning attacks [5]. Information harvesting from social networking sites thus became a non-trivial problem.
The leakage of personal information from these platforms creates a remarkable dilemma as this information forms the ideal base for further attacks. Jagatic [6] showed that they could increase the success rate of phishing attacks from 16 to 72 per cent using "social data". The findings of [6] have furthermore been confirmed by the experiments of [7]. Social engineering is yet another attack where information on a future target forms the starting point for attackers and because of the emerging SNSs usage the whole attack might eventually be automated [8]. Existing attempts to extract information from SNSs focus on the application layer and can thus be mitigated by adapting a specific social networks' application logic. Our friend injection attack, in contrast, is carried out on the network layer, which the great majority of SNSs providers fail to secure.

2.2 Vulnerable Social Networking Sites

Social Networking Sites follow to some degree the same business paradigm as other successful commercial web services whereas the service itself is free of charge and profit is made by selling online advertising services to third parties. Hence the number of users and signups are critical for the commercial success of SNSs. SNS providers therefore design their services so as to increase the number of new signups. Because the

majority of SNSs are free of charge these services depend on selling advertising. The more comprehensive the socio-demographic datapool they offer, the more of interest these services become to online-advertisers. Hence SNSs implemented a range of tools to further expand their growth, e.g. the ability to import email addresses from one's email account or the possibility to send email invitations to possible future friends.

Social Networking and SSL/TLS While the adoption of the HTTPS protocol is considered trivial from a technical standpoint, popular SNSs like Facebook only support the unencrypted HTTP protocol for data transmission. We hypothesize that today's SNSs providers prefer HTTP over HTTPS for performance reasons (compare with [9]) in order to minimize their hardware and connectivity costs. Authoritatively signed certificates would furthermore result in additional costs for SNSs providers. Our proof-of-concept friend injection attack, which is outlined in section 4, exploits the unencrypted communication by abusing the "invitation form" that many social networking sites offer. As a first step we aggregated a list of possible targets for our friend injection attack. Table 1 shows the most popular SNSs (based on the size of their user base), the availability of an invitation form, and furthermore if the service provides HTTPS access. No reliable sources for the actual number of users within different SNSs exist as these numbers are not publicly available but rather claimed by SNSs providers. A Wikipedia article [10] aggregated a list of popular SNSs and their self-claimed number of users, which offers a valuable starting point. Based on their claimed number of users, MySpace, Qzone, Windows Live Spaces, and Habbo would have been listed within the Top five social networking sites as well. We decided however not to include them as their users often do not represent their real-world personas (e.g. Facebook vs. MySpace [11]), which renders their stored information less attractive for social phishing and data harvesting attacks.

Social Networking Site			
Name	Claimed user base	Invitation Form	HTTPS
Facebook	350×10^6	yes	Login only
Friendster	90×10^6	yes	No
hi5	80×10^6	yes	No
Tagged.com	70×10^6	yes	Login only
Orkut	67×10^6	yes	Login only

Table 1: Top five social networking sites and their support for HTTPS.

Within our top five social networking services, three out of five make use of HTTPS, but only use it to protect login credentials. The rest of the communication happens unencrypted and visible to everyone along the communication path. All five SNSs offer an invitation form, which can be exploited by our friend injection attack. Furthermore the great majority of SNSs is vulnerable to our attack.

3 Friend Injection Attack

Our novel attack on user privacy within social networks is based on the fact that all communication between clients and SNSs is unencrypted. The only exception is the authentication process, which is encrypted using TLS in some cases (compare with section 2.2). The HTTP protocol, which is used for communication between web services and web browser, is stateless. HTTP cookies are thus used for client tracking, personalisation and most importantly for session management. They were introduced by Netscape in 1995 in order to provide state-fullness for the HTTP protocol and therefore provide a better user experience for web services. In case of SNSs, after a user authenticates herself, a cookie is used to keep track of her session, including a hashed, shared secret between client and server. As this cookie is transmitted unencrypted over the network, the communication is vulnerable to cookie hijacking. Hence, by tapping network traffic it becomes possible to extract authentication cookies and to submit requests on behalf of the victim. While HTTP session hijack attacks are well known, we argue that these attacks enable a wide range of novel attacks specific to SNSs.
In case of social networking sites, the ability to inject requests into an active session, has drastic consequences. On one hand sensitive information can be extracted and on the other hand malicious attacks can be carried out via innocent users.

3.1 Spoofed Friend Invites (Friend injection)

A friend request can be sent to another account, which is under the control of an attacker. An attacker might for example create a fake account for ”Mr. Cypher” and then inject requests that seem like the victim added the attacker as a friend. In our proof-of-concept implementation in section 4, we exploit the friend invitation form that the majority of SNSs offer (see subsection 2.2). As most SNSs show different degrees of information details, depending on whether the viewer is a friend or not, this could be used to retrieve user details that are not publicly available. Another possibility is to easily retrieve the circle of friends of the victim which would be time consuming otherwise [4]. Fig. 1 is an example for the default access control settings of Facebook at the time of writing. Friends can access sensitive information such as email addresses and phone numbers, while e.g., photos are made available to friends of friends as well.

3.2 Further attack scenarios

The majority of SNSs providers offer a developer API for third-party applications. APIs offer a new way to tap the pool of personal information stored within social networking sites. Once a user adds a certain third-party application it is automatically granted access to this user’s personal information. According to [13] the context-information given to third-party applications is usually not anonymized, even though most applications would be able to function on anonymized profiles. Hence, an attacker could add a custom third-party application on behalf of a user in order to retrieve all personal information of him/her in an automated way. In the case of Facebook applications the access to information is not only limited to the application user, but also the personal information of the complete set of this user’s friends can be retrieved. As outlined in

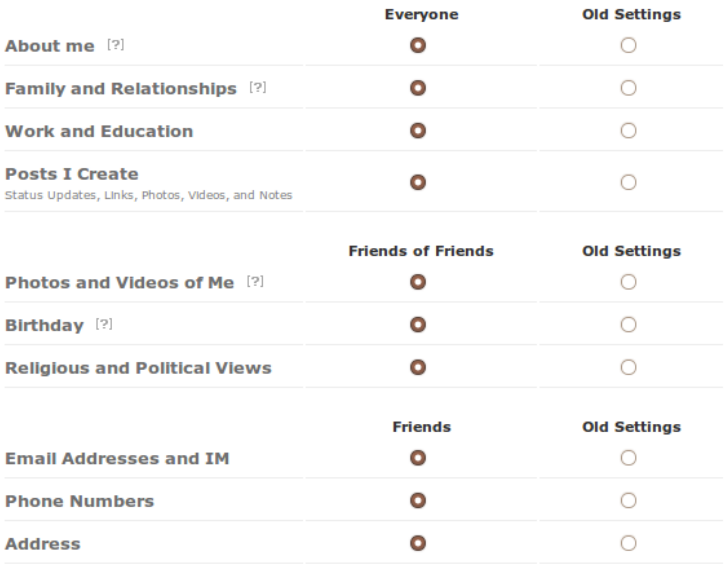

Fig. 1: Facebook default privacy settings as of Dec. 2009 [12]

[6], phishing attacks that appear to be legitimate messages sent by a friend are more likely to succeed in their evil intention. Sending messages to all the victim's friends or writing on publicly accessible places like the Facebook Wall (either the victims' or those of friends) on behalf of another user could lure victims into opening malicious files or links. More sophisticated attacks could use the information gathered through an injected application for spear phishing of selected targets.

4 Proof of Concept: FriendInjector application

Because social networking sites account to today's most popular web services, we hypothesized that it would be relativly easy to find active social networking session within different networks (LAN, WLAN, gateways, etc.). At the time of writing, Alexa's site info statistics [14] suggest that Facebook accounts to 30 per cent of the worldwide Internet traffic. We thus decided to implement our friendInjector application on basis of Facebook, as active Facebook sessions seem to be the most prevalent type of social networking sessions on the Internet.

4.1 FriendInjector Application

We used the Python scripting language to create our "FriendInjector" application. Fig. 2 summarises the different steps that are involved in our novel attack. (1) In a first step we use the dpkt library [15] to tap into network traffic and parse received packets. Once

a legitimate Facebook session has been found, the FriendInjector application clones the complete HTTP header of the retrieved packet, including: the user agent string, session cookies, and accepted file formats and languages. (2) The cloned HTTP header is then used to request the URI containing the invitation form of Facebook. This second step is necessary in order to retrieve the specific form ID which is different for every user to mitigate cross-site scripting attacks. (3) Once both a valid HTTP header and the specific invitation form ID have been collected, the FriendInjector application sends a spoofed friend invitation request via the mechanize library [16] to Facebook using a predefined Email address. (4) In case the attack has been successful, Facebook sends a friendship request on behalf of the victim to the attacker's Email account. The email notification in step four is used to verify if the attack has been successful. Multiple transmissions of the same session cookie are detected over time (every time a user clicks a link or posts content within the session with Facebook), this gets detected and no friend injection occurs.

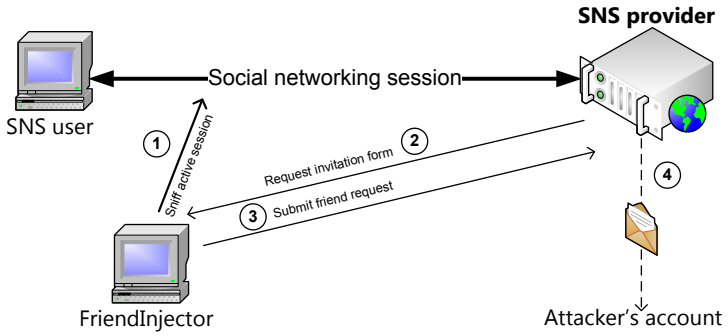

Fig. 2: Outline of our proof-of-concept friend injection attack

4.2 Evaluation on the Feasibility of Friend Injection Attacks

We decided to evaluate our FriendInjector application on the basis of unencrypted WLAN traffic, as it is rather straightforward to monitor, only requires physical proximity, and most importantly does not require an active attack against the networked infrastructure (such as ARP poisoning or DNS highjacking). However, our proof-of-concept application would have worked equally effective on Internet gateways, LANs or deployed on single computers. The goal of our evaluation was to verify if our FriendInjector application gets detected and at which rate friend requests can be injected:

1. **Stealthiness**
   (a) *Targeted user*
   Facebook does not offer a list of pending friend requests to its users. The only way to search for pending friends requests is offered through the friendslist

whereas injected friends would have been marked with a ”pending friend request” status. However, in our attack setup we deactivated our test Facebook account beforehand and our friend requests therefore do not show up in the friends listing. Hence, our attack is completely stealth and undetectable to targeted Facebook user.

(b) *Facebook platform*
Because our application uses a cloned HTTP header of a legitimate request to Facebook, we hypothesized that our injections will not get detected by Facebook. Furthermore of interest was the question if Facebook would compare the network location of a user's session with the location of our forged friend request.

2. **Injection rate**
Once a friend request from Facebook is received, the friend injection attack was considered successful. The invitation could be used for further attacks like sending phishing messages or data harvesting. However, this was not the goal of our evaluation and we used the friend requests to measure at which rate friend injections have been possible.

4.3 Methodology and Ethics

We chose the library of a big Austrian university as our experiment location. The university's library was selected because of two reasons: SNSs are very popular amongst students, and secondly this particular library offers both secure and insecure (unencrypted) Internet access via WLAN. The university's wireless LAN is operated on three channels: *1*, *6*, and *11*. To capture all three channels we equipped a laptop with three WLAN USB sticks with an instance of FriendInjector attached to each of the three channels. In a first experiment we performed the friend injection attack over the period of one and a half hours and used the libraries Internet connection to perform the injections. In the second experiment we performed the evaluation for seven hours and used a separate HSDPA connection to inject the friend requests. During our experiments we took special care that the privacy of user data was not put at risk. We did not collect or store any personal data of test subjects, neither private nor public, and did not reply to the injected friend requests. The only information used for confirming the successful friend injection were the friend requests sent by Facebook, which have been deleted as soon as we received them.

5 Results and discussion

Fig. 3 as well as Fig. 4 illustrate the results of our two experiments. In the following, our results are further explained as well as mitigation strategies against our novel friend injection attack are briefly discussed.

5.1 Experiment Results

Within our first experiment we performed the friend injection attack for a period of 1.5 hours during the peak time of the university's library, where we expected most of

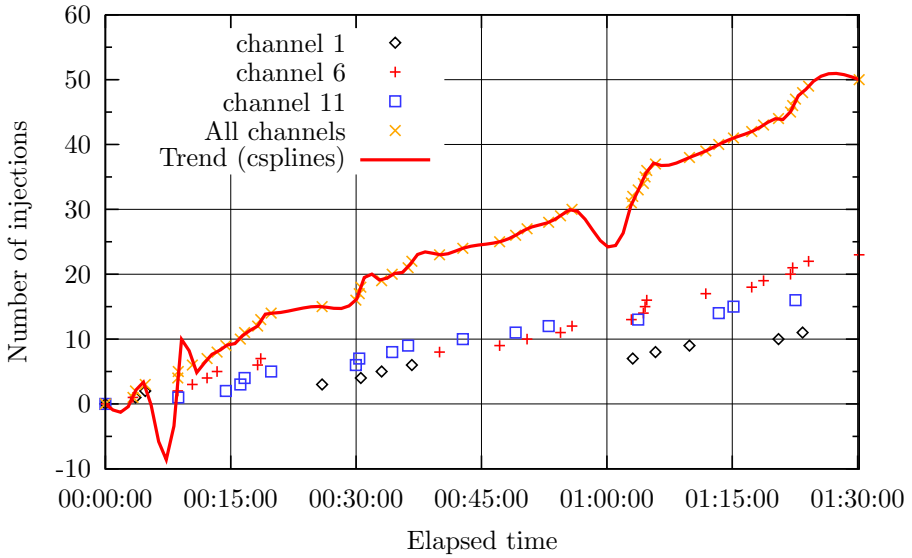

Fig. 3: The result of our first friend injection experiment with an average of one injection every 1.8 minutes as measured during peak time of Internet usage.

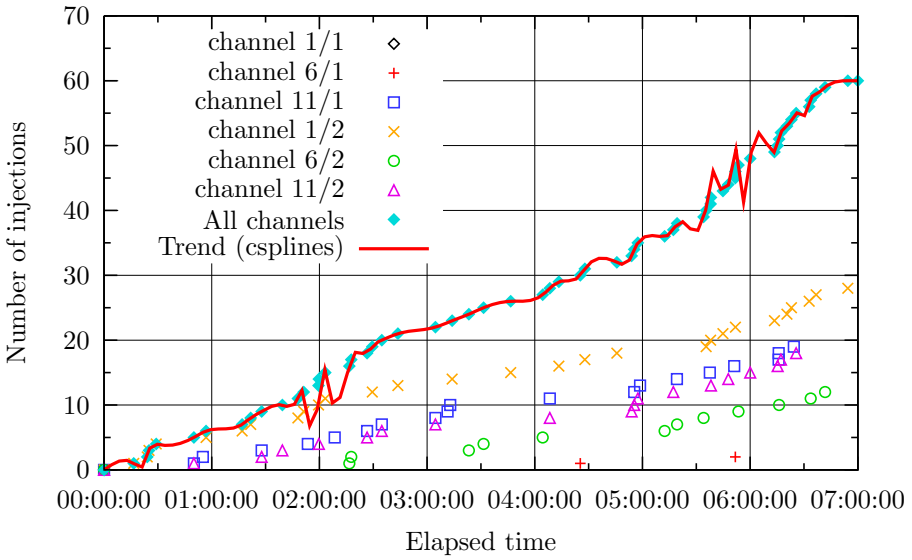

Fig. 4: The result of our second friend injection experiment with about one injection every 7 minutes as measured during an average day of Internet usage.

the students would use the library's wireless access points. As previously stated we used the library's Internet connection to perform the friend injections. On average the FriendInjector application performed a successful injection every 1.8 minutes. The second experiment was carried out during an average day in the university's library were less students were using the Internet. During a period of seven hours we were able to inject 60 friend requests which corresponds to one successful injection every seven minutes. Even though the network connection used by the FriendInjector application (HSDPA) was different from the one used by our test subjects (library WLAN) and thus using a totally different IP adress block, our spoofed requests did not get blocked by Facebook.

5.2 Mitigation Strategies

The attack surface of our evaluation had been unencrypted WLAN traffic, hence, the answer seems straightforward: users need to ensure that they only use secure wireless access points. This however is not always possible and many WLAN access points at airports, hotels, or conference rooms remain to offer insecure network access only. End users could also protect themselves against our friend injection attack by using an additional VPN gateway to tunnel their communication in a secure way, given they have the technical expertise. Adida [17] on the other hand proposed a method where cookies can be protected against eavesdropping which could easily be adapted by SNSs. This method protects however only against passive attacks. In order to effectively mitigate friend injection and similar attacks, SNSs providers have to ensure that all communication between their users and their platform is done over HTTPS. At the time of writing only XING[18] offers HTTPS while the biggest SNSs fail to support secure access to their services. Thus, the only mitigation strategy available to the average user seems to be browser extensions such as ForceHTTPS [19], which attempt to force HTTPS for requests that would have been normally transferred over HTTP.

6 Conclusion

We introduce a new attack on social networking sites, our novel *friend injection attack*. It can be used for automated social phishing, data harvesting, or sophisticated social engineering attacks. The attack is based on the network layer and is completely undetectable to the victims, and as our results suggest even to social networking providers. Within a practical evaluation we furthermore showed the feasibility of our novel attack on basis of public WLAN access points. Given the sensitive personal information that social networking services contain, a large scale friend injection attack would have devastating consequences. In order to mitigate friend injection attacks social networking providers ultimately have to fully support HTTPS. At the time of writing however, the great majority of services fail to protect their users against malicious eavesdroppers and injection attacks.

References

1. Facebook. Facebook statistics, 2010. [Online; accessed 5-January-2010], http://www.facebook.com/press/info.php?statistics.
2. R. Gross and A. Acquisti. Information revelation and privacy in online social networks (the Facebook case). In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.
3. H. Jones and J.H. Soltren. Facebook: Threats to Privacy. *Project MAC: MIT Project on Mathematics and Computing*, 2005.
4. J. Bonneau, J. Anderson, R. Anderson, and F. Stajano. Eight friends are enough: social graph approximation via public listings. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 13–18. ACM, 2009.
5. Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: Automated identity theft attacks on social networks. In *18th International World Wide Web Conference*, April 2009.
6. T.N. Jagatic, N.A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.
7. G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social networks and context-aware spam. In *Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 403–412. ACM New York, NY, USA, 2008.
8. Markus Huber, Stewart Kowalski, Marcus Nohlberg, and Simon Tjoa. Towards automating social engineering using social networking sites. *Computational Science and Engineering, IEEE International Conference on*, 3:117–124, 2009.
9. X. He. A Performance Analysis of Secure HTTP Protocol. *STAR Lab Technical Report, Department of Electrical and Computer Engineering, Tennessee Tech University*, 2003.
10. Wikipedia. List of social networking websites — Wikipedia, The Free Encyclopedia, 2009. [Online; accessed 10-December-2009], http://en.wikipedia.org/wiki/List_of_social_networking_websites.
11. C. Dwyer, S.R. Hiltz, and K. Passerini. Trust and privacy concern within social networking sites: A comparison of Facebook and MySpace. In *Americas Conference on Information Systems (AMCIS), Keystone, Colorado, USA*, 2007.
12. Facebook. Facebook asks more than 350 million users around the world to personalize their privacy, 2009. [Online; accessed 4-March-2010], http://www.facebook.com/press/releases.php?p=133917.
13. A. Felt and D. Evans. Privacy protection for social networking APIs. *2008 Web 2.0 Security and Privacy (W2SP'08)*, 2008.
14. Alexa. Site info: Facebook, 2010. [Online; accessed 20-January-2010], http://www.alexa.com/siteinfo/facebook.com#trafficstats.
15. dpkt - python packet creation / parsing library. http://code.google.com/p/dpkt/.
16. Python mechanize library. http://wwwsearch.sourceforge.net/mechanize/.
17. B. Adida. Sessionlock: securing web sessions against eavesdropping. In *Proceeding of the 17th international conference on World Wide Web*, pages 517–524. ACM, 2008.
18. Xing business network - social network for business professionals. https://www.xing.com/.
19. C. Jackson and A. Barth. ForceHTTPS: Protecting high-security web sites from network attacks. In *Proceeding of the 17th international conference on World Wide Web*, pages 525–534. ACM, 2008.