Final Project Report

ITIS 6200 Principles of Information Security

Facebook Chat Monitor Chrome Extension

Team:

1) Kshitij Sudhir Gorde: ID #800966672
2) Amisha Harshad Gondaliya: ID #800958729
3) Tushar Arvind: ID #800823103

Input to the tool:

Prelisted suspicious keywords from Facebook chats and/or blacklisted urls.

Output:

If suspicious keywords count goes greater than 5 then the user is alerted via a chrome extension. If the user tries to open a blacklisted link sent to him via chat, the plugin will examine the link and block it by redirecting the url to a safe one.

Running the Project:

This project consists of a chrome extension plugin and a server written in PHP.

Project setup:

1) Extract the Zip files to your computer.
2) Copy ReadFileBackgroundExtension to your Desktop. This contains the code for chrome extension.
3) Open Google Chrome browser.
4) Type chrome://extensions in your browser address bar. This should open up your interface from chrome extensions.

5) Click on Load Unpacked Extension. On the dialog box opened thereafter select the ReadFileBackgroundExtension folder. This should install the plugin.
6) Once the plugin is installed, the Facebook chat monitor becomes active.

Server Setup:

1) The source code for the server is given in pisp_chat_monitor.zip.
2) However, in order to use this project, it's a requirement by Facebook that only developers are allowed to use it. Since I have developed the Facebook page and Facebook app, and also have this server running on cloud9 platform, currently only I can access the contents of the page. Hence this project can be run from my account only. In order to make this app public, we need get get it reviewed from Facebook since we are using 'read_mailbox' permission and also pay Facebook a certain amount of fee. As for the development mode nothing is required.
3) If you still like to use it from your Facebook account, email me: kgorde@uncc.edu and I shall add you as the page administrator.

Testing Steps:

1) If you're an administrator of the page "ITIS 6200 Project", then open the page in Facebook. Click on message which should open a chat window.
2) Type in some suspicious keywords such as 'crime', 'murder', or 'rob a bank'. If these suspicious keywords count reaches 6, an alert box will be popped up with a warning.
3) To test blocking of a suspicious link, type www.pispproject.com OR www.pispunsafe.com in the chat window. Now click on that link. It should be redirected to a different page on the server.

Motivation behind the tool

The social media platforms like Facebook are used by many children today. Most of these children aren’t aware of the malpractices that can be conducted on these platforms to lure innocent children into doing something illegal. One of such is Facebook chat application wherein if you own a Facebook page, any random person can contact you via chat and might send you unsafe urls or lure you into doing wrong. So, we wanted to create an idea with which we can at least alert the user if he’s unaware what he’s dealing with in the chats as well as block the user from accessing suspicious links sent via the chat window. With

some preliminary research, we found out a way to access the chat messages from the Facebook chat application using the Graph API.

Technical Contributions and Project Development

The project is divided into 3 parts:

1) **The Facebook CM App: '**Kshitij Gorde'

   In order to use Facebook developer features, it is mandated by Facebook to create an application which can serve as a bridge/portal for managing all of the permissions and API usages. The app is named CM and a new Facebook page called ITIS 6200 Project has been created on Facebook for testing purposes. CM application thus is linked with the page. A page access token needs to be generated and validated. This is done using the CM application by clicking on a link provided in the app interface. Further, the CM app has been linked with the Graph API and the messenger bot API by Facebook along with Webhooks for providing real time updates whenever a new message is received. Thus, Webhooks service fires a request to the server whenever it receives a new message and thus forwards that message to the server. Webhooks is thus extremely useful since it functions like a polling service and provides real time updates.

2) **Server: '**Kshitij Gorde'

   The server is hosted on the cloud9 platform and is written in PHP. The server validates the request it receives by checking a Verify token which is set to 'VERIFY_TOKEN'. Once validated, it extracts the message body from the request and writes it to a text file 'Facebookmsg.txt' which also resides on the server itself.

   Changes done by 'Amisha Gondaliya': - On receiving REQUEST_METHOD as PUT on server, the server opens facebookmsg.txt file in write mode and truncates the file. This method has been customized to accommodate write change request by google chrome extension on the server.

3) **Chrome Extension: '**Amisha Gondaliya & Tushar Arvind'

   The chrome extension named 'Facebook Chat Monitor' analyzes and alerts the user if any suspicious keywords are found. It also redirects certain blacklisted URL's to block.html page on the server. The chrome extension has been developed in JavaScript with native Chrome API's.

The chrome extension has the below file structure.

1) manifest.json :- Every chrome extension needs a manifest.json file. It's a file the browser uses to refer all executables and all permissions are mentioned there in. The manifest.json contains permissions required by the extension. Our manifest.json contains "tabs" in the permission section which means the extension will work on all open tabs parallelly. The "background" tag in manifest.json signifies that this extension will run in the background. It refers to background.html which in-turn refers to background.js
2) background.html: - This file contains background.js in the script section, which means it will run the javascript file in the background.
3) background.js : - The main function which does the reading from a remote file, fetching suspicious words and links, notifying the user and clearing the file after the suspicion counter reaches 5 is ReadTextFile. It uses XMLHTTPRequest API to send request to read and write onto the file. It uses function notifyMe to create a notification at the bottom right corner of the browser window. scheduleRequest function, creates an alarm to schedule next request to poll from the file. startRequest schedules a request first before executing the current request of reading and writing to text file using ReadTextFile function. onInit is a startup function identified by chrome like the main function in C/C++. onAlarm is a function which is called to get the next timestamp of alarm. onWatchdog calls onAlarm function if active alarm exists, if it doesn't, it calls the start request and schedules the next alarm, else waits for this alarm to get timed out. Function notifyMe checks if desktop notifications are available or not. If not available, alerts the user with a message "Desktop notifications not available in your browser. Try Chromium." If notifications are available, it notifies the user with the text "We've detected some suspicious keywords in your chat. Please close the chat window!". chrome.webRequest.onBeforeRequest function checks if user clicks on any of the suspicious links mentioned within, then will redirect the user to a safe link, 'https://pisp-chat-monitor-blitzkshitij.c9users.io/block.html'.
4) Approve_icon :- The approve icon is uses as an icon for our extension which will be displayed in the browser.

Chrome Notifications

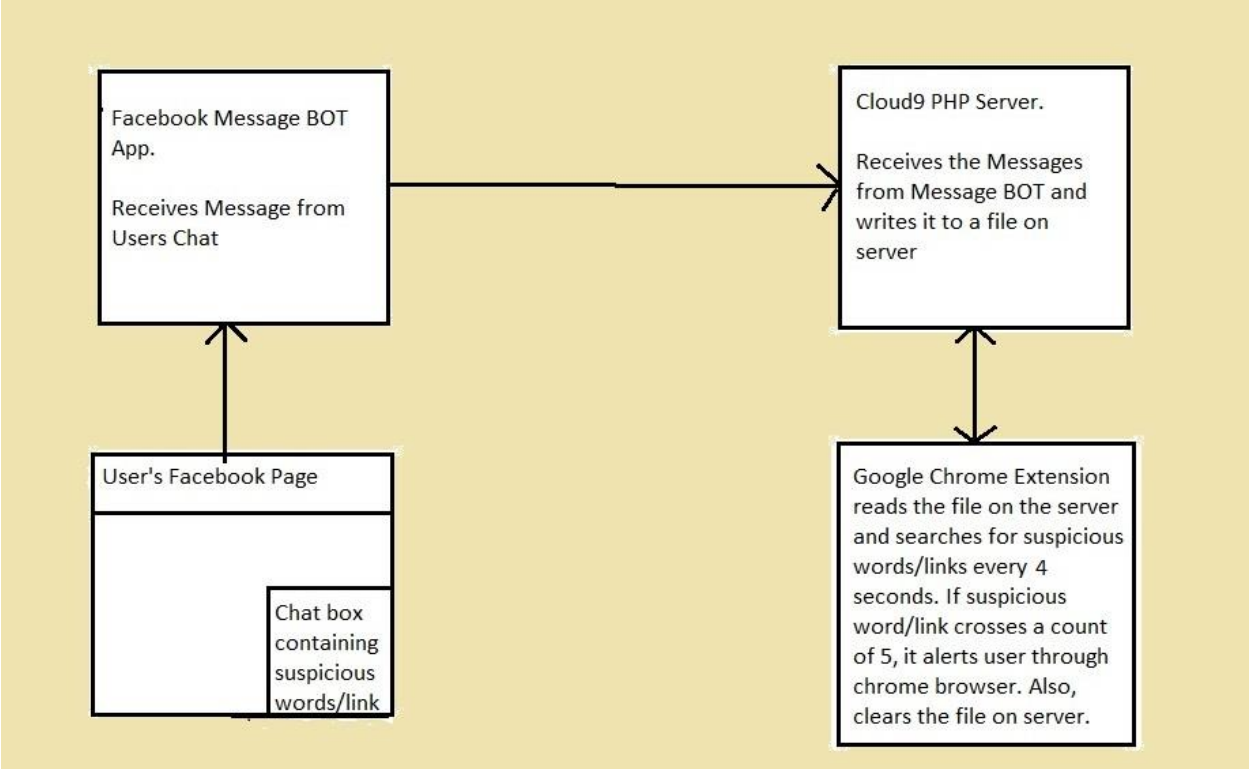
The Facebook Chat Monitor tool utilizes notifications passed through the user's web browser to inform them of potential threats that are being read in their Facebook chat. This is possible through the use of Chrome Notifications. The API that is used to construct the notification part of the Facebook Chat Monitor extension is the World Wide Web Consortium Notifications API (also known as W3C), rather than using the built-in Chrome Notifications API. The reasoning for this is because the W3C Notifications can stack on top of each other in case there are multiple alerts, and the API is compatible across multiple web browser in case the tool is to be ported to other browsers such as Firefox, Microsoft Edge, etc. The Chrome Notifications API does not offer this level of support.

The W3C Notifications specification provides an API to display Notifications to alert users outside the context of a web page. This includes any area outside of the web page, including a corner on top of an existing web page, or even on the user's desktop. The API does not specify how exactly the API should be displayed, as the API chooses the best presentation mode possible depending on the device that it is used on. Since the Notifications API is being used on a desktop environment rather than a mobile environment, the Notifications API will usually place the notification on the lower right corner of the screen for the user to see.

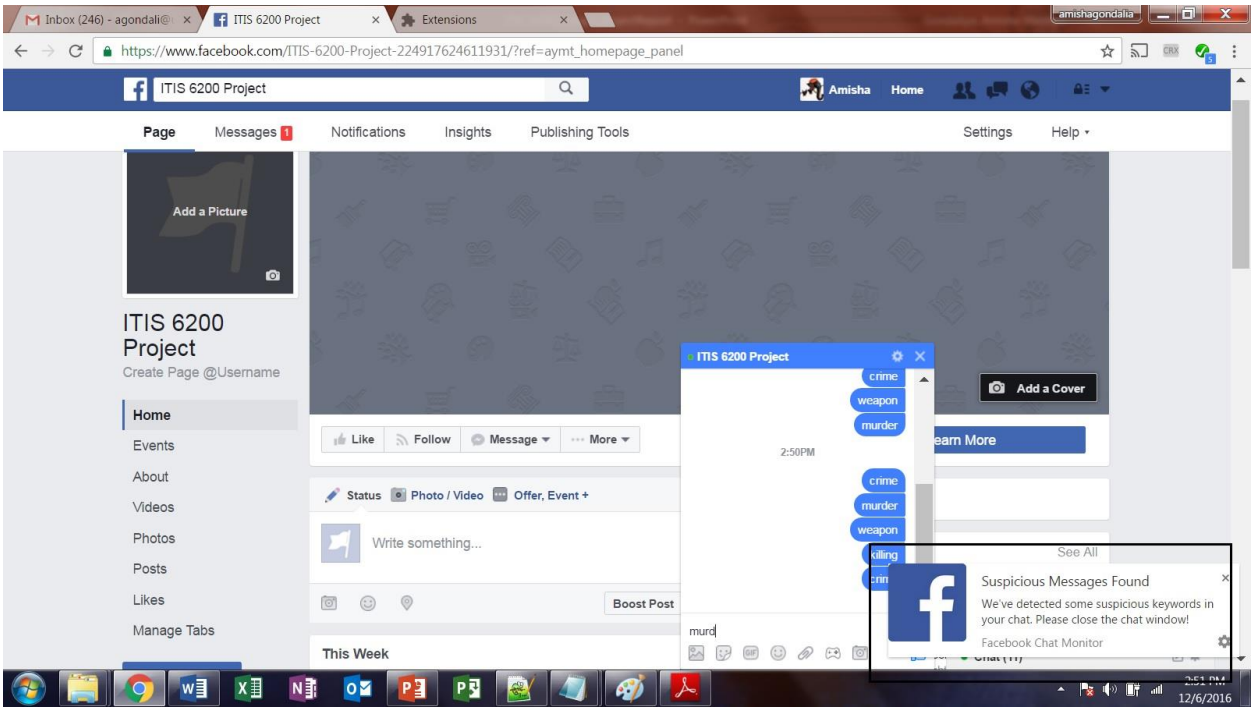
During use of the Facebook Chat Monitor tool, a listener is opened that attempts to read for positives on malicious words or web URLs that are being passed through the Facebook chat. When the extension starts to read up to 5 of these alerts, a "ping" is sent to the listener, which then activates the notification. The notification reads a message to the user, informing them that malicious content was found in their current chat session and recommends them to close the web browser immediately so that they are not affected by the content that is being sent to them. This alert will continue to relay as long as there are 5 malicious alerts being read at a time, and the notification will be cleared whenever the user manually clears it themselves.

The W3C Notifications API only allows for notifications to be sent through a user's web browser if the user gives permission to be sent notifications first; this request for permission is the first thing that is done when a user enters a website that has support for these notifications. In this case, whenever the Facebook Chat Monitor tool is initialized, permission to send the user notifications is requested. The user will also be notified if they are using a web browser that does not support notifications, although this goes unused in the current code since the extension is only compatible with Google Chrome for now.

Detailed Process Flow:



- This application monitors user's chat messages for any suspicious message. It can be used as a security tool for ignorant users to alert them of a security threat they are exposed to.
- The application has three interface components, Facebook Message BOT, PHP Cloud 9 Server, Chrome Extension.
- Facebook Message Bot receives a message from Facebook Chat and sends the same to PHP Server configured on Cloud9.
- The PHP server saves this chat in a file on the server.
- Google chrome extension implemented reads the chat messages stored in the file created on PHP Cloud server and checks if any of the suspicious words from the Data list are present in the file.
- If the suspicious word count exceeds 5, the user will get a notification alert as shown in the screen below.



- The extension will then go on to clear the file and Facebook will populate the server with new messages.
- If the user receives a suspicious link which is present in the data configured for this application and the user clicks on the link, the extension will block the link and redirect to block.html on PHP server.

Learning Experience:

The learning experience in building this project has been terrific. After many struggles and trial and error experiments, we were finally able to successfully implement the concept we had in mind. One of the most challenging tasks were to read messages from the chat application, since Facebook employs a lot of security and a developer needs to take care of all these measures. Also, polling the server from the chrome extension and processing all the entries and deleting it later was quite challenging because it was to be done synchronously since deleting new messages without reading them would give incorrect results.

Future Scope:

This project has a great future scope. Right now, this is just an idea which uses blacklisted words to alert the user. However, sentimental analysis can be employed to come up with a more sophisticated analysis which may be able to read the objectives of the person chatting with the

Facebook user. Also, it can be extended to analyze images, videos, stickers and various files sent via chats.