Password Persuasive Cued Click Point (PPCCP)

Romil Gandhi[1] Sachin Bojewar[2]
[1,2]Department of Computer Engineering
[1]Shree L.R. Tiwari College of Engineering Mira Road, Thane, India [2]Vidyalankar Institute of Technology Wadala, Mumbai, India

Abstract— Password Persuasive Cued Click Point (PPCCP) is a module that facilitates authentication for desktop-based applications and also provides a single-machine licensing facility. PPCCP comprises several functionalities to thwart attackers, such as persuasive click points with password protection. It includes techniques to protect against malware such as key loggers and also to resist from debuggers that run on desktop computers. Spearman rank correlation is used for detection of key loggers. There are functionalities used to secure desktop applications such as time constraint and user selection.

Key words: Authentication, Computer Security, Graphical Security, Keylogger, Licensing

I. INTRODUCTION

PASSWORD Persuasive Cued Click Point (PPCCP) is a module geared towards providing better authentication and licensing for desktop-based applications. PPCCP provides methods to thwart attackers and is easy to implement. It is implemented and executed on the Windows platform. PPCCP utilizes Persuasive Cued Click Points (PCCP) [1], which provides better security than text-based passwords, and is designed to satisfy the need for desktop applications to protect against attackers and malicious tools such as keyloggers, and debuggers. PPCCP [12] combines PCCP with text-based passwords. It provides a server mechanism to authenticate licensing and requires an internet connection for licensing and authentication. The licensing mechanism is developed for single-machine use, and the serial is not usable on another machine once that computer has been registered on the server. Once the machine has been authenticated on the server, the user is taken to a registration window. This registration is optional for the user depending on the application and security requirements. If the user opts for registration, the user needs to select three or five cue points on three or five images, each consisting of a password. Once the registration phase is completed, a registry key is generated on the user's desktop and PPCCP redirects the user to a login window. The user can then login using the registered credentials. The login window verifies the username entered by the user during registration and the user's selected image will be displayed. The user then needs to select a cue point on the image and then enter the required password; the caret is hidden, and neither the password nor the caret is displayed. This procedure is repeated for three or five images, depending on the option selected during registration. As soon as the login procedure has been successfully completed, the application/software with which it is integrated can continue its execution.

During login, once the user is verified, the correct image of that user is displayed. However, it is possible that an attacker may be trying to gain access instead of the authorized user. In such a case, the attacker will most likely not know the correct cue point on the image. In addition, if the attacker selects the wrong cue point, and then PPCCP will display random images on the screen until the program is restarted. This is one form of protection provided against attackers. Further, the text password that the user enters after selecting the correct cue point on the image is invisible, thereby protecting against shoulder-surfing attacks.

The remainder of this paper is organized as follows. Section II provides background and discusses existing PPCCP systems. Section III describes PPCCP in detail. Section V discusses attacks and techniques for protection against keyloggers and various reversing tools, such as debuggers. Section VI elaborates analysis of implemented PPCCP.

II. BACKGROUND

In the PassPoints graphical password scheme, a password consists of a sequence of click points (e.g., between five and eight) that the user chooses on an image. This image is displayed on the screen by the system. It is not a secret image and has no role other than helping the user to remember the click points. Any pixel in the image is a candidate for a click point. To login, the user has to click on or very close to the chosen points and in the chosen sequence [9].

Cued Click Point (CCP) [2], [7] is a graphical system that distributes click points on multiple images. In this system, only one point is clicked in an image and the user can select the points on an image. However, this technique is vulnerable to hotspot threat. A hotspot is a common point on an image that is generally selected by users. This common point can be easily guessed by attackers. Thus, to protect against hotspot attacks, the PCCP was developed. PCCP [1], [4]–[6] utilizes five images, each with a cue point as password. The viewport is randomly generated by the system and the user can change the viewport by clicking on a shuffle button, which helps the user to select different points in an image.

PCCP does not provide any protection against attackers, debuggers, and disassemblers. Debuggers and disassemblers are generally used to crack desktop-based applications, especially Microsoft Windows applications. PCCP is secure against web-based threats; however, for desktop-based applications malicious software and reverse engineering attacks have to be considered.

III. PASSWORD PERSUASIVE CUED CLICK POINTS

PPCCP [12] is an amalgamation of password and graphical cue points. It comprises two modules: licensing and authentication. The licensing module is valid for one PC and cannot be used by another without registration. The licensing module fetches the user's hardware IDs and calculates the hash that is stored on the server as well as on the local PC. Thus, only the registered user can use the software/application. Authentication comprises two phases: Registration and Login. PPCCP automatically detects

whether the user is registered and will redirect to the appropriate form. Once the user is registered, the user can login using his/her registered credentials. The credentials consist of three or five images, each having a password. If the user selects the wrong point on an image, random images are displayed and, after three or five such images, the application closes.

Following installation, PPCCP executes anti-debugging functionality, which is designed to resist debuggers, keyloggers and protects the application from malicious programs. If it detects a malicious entity, PPCCP exits in order to protect the application from being reverse engineered.

During login, the user needs to select and enter a password in the stipulated time [2]. This time limit is used in order to protect against dictionary and brute-force attacks. The time interval is fixed and the user has to enter the password within that interval on selecting the cue points. When the time interval is exceeded, PPCCP resets the images and informs the user to start again. Thus, an in-between (MITM) intrusion from a reverser is not possible. The flowchart in Fig. 1 shows the general flow of PPCCP.

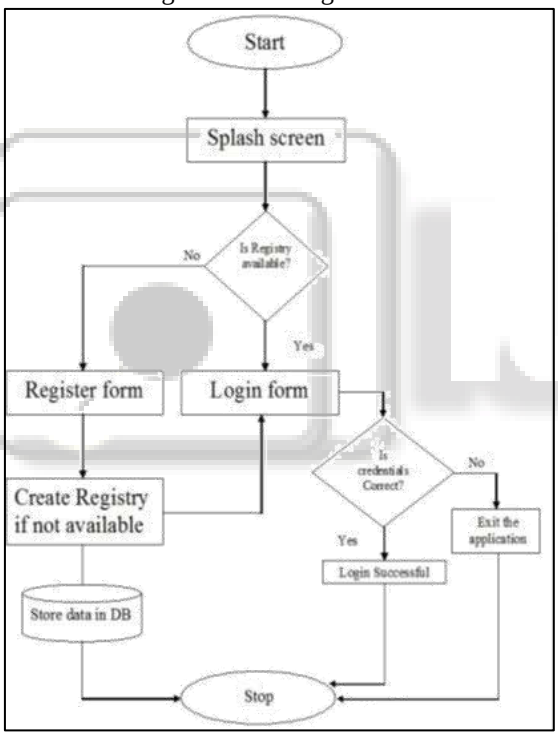

Fig. 1: General Flow of PPCCP.

The flow starts with a splash screen, during which many background processes are executed. These processes include detection of keyloggers and debuggers, fetching the IDs of hardware components, and hashing and storing values to database. It also checks the registry to determine whether the user is a new or existing user. The outline of the process is as follows:

```
If (check registry == 'PPCCP') then
        {       User is registered.
                Redirect user to login form.
        }
Else
   {
        If (check server for SystemFingerPrint == yes)
        {       User is registered.
                Redirect user to login form.
        }
        Else
        {       User is not registered.
                Redirect user to option form.
         }
}
End
```

If the user is a new user, then it redirects to an option form, on which the user can select zero, three, or five. If the user does not want to be authenticated, then he/she should select zero; option three is low-level authentication mode, and option five is high level authentication mode. Three and five refer to the number of images desired. After selecting the appropriate option, the user is taken to the registration form. On completion of this form, the registry is created and stored on the user's PC and the user is asked to login. If the user forgets his/her password then the registration phase can be re-initiated. The verify form is designed to confirm that the user has been validated. If the user selects the wrong point on an image or enters the wrong password, then random images are displayed.

In the login phase, a 60s timer exists for each image [2]; the user needs to select the cue point and enter the password within this time frame. If the user fails to do so, then PPCCP will shut down and the user will have to start again.

A. PPCCP Registry Formation Procedure

The procedure to generate a registry is illustrated in Fig.2. It is a simple cryptography procedure. The password and the mouse coordinates are processed separately for better protection and efficiency. The newid() function is used in Microsoft SQL to generate a new Globally Unique Identifier (GUID). This new GUID is fetched by PPCCP and hashed usingSHA512. This Hashed value is then stored in the registry branch called PPCCP in HKEY_USERS. This hashed value is called UserFingerPrint.

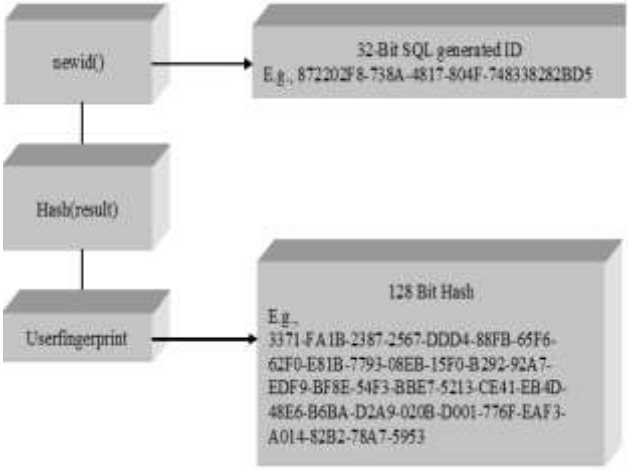

Fig. 2: Procedure to generate registry.

The hardware information is stored on the server as parameter values such as CPUid(), BIOSid(), and BASEid(). These are all then hashed and stored on the server, as shown in Fig.3. This hashed value is referred to SystemFingerPrint. It is stored on the server to check the single-machine licensing.

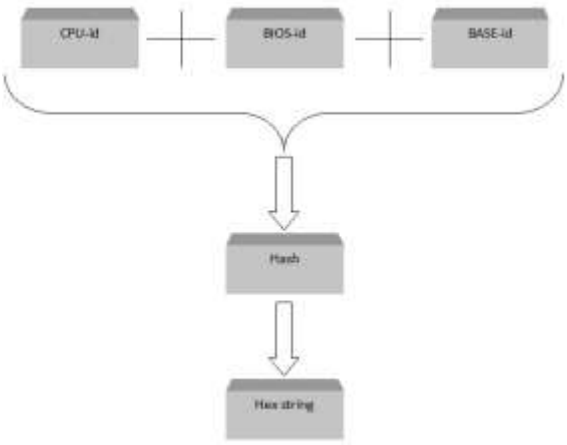

Fig.3.Method used to fetch machine's information for licensing.

B. Image Sizes

The size of the image used in the registration and login form is 603 × 603 pixels. The viewport in the image is 33.5 × 33.5 pixels, as shown in Fig.4. If consecutive viewports are arranged in an image, then there will be 324 viewports. Thus, the chance of guessing the correct viewport is miniscule. The viewport size is calculated considering the length of an arm (60 cm). Further, as the Fovea considers a one-degree angle, using Snellen's notation,

$$= \frac{\text{Visual acuity distance at which the test is made}}{\text{Distance at which the smallest optotype identified subtends an angle of 5 arc minute (MAR).}}$$

This results in the size of viewport being 33.5 pixels.

The total number of images uploaded to the database is165 and a separate uploadRandomImages form is designed to upload multiple images to the database. PPCCP stores and fetches images to and from the database. The images are hashed and then stored in the database. Thus, none of the images can be edited or changed. The images are regenerated as bitmap files for faster loading. The program was developed in C#.

Fig. 4: Image and viewport size

IV. WORKING AND DETECTION OF DEBUGGERS

The debuggers have various purposes and are used accordingly. They can be used for either constructive work or for destructive work. They are used to test and find bugs in programs. A debugger is a very powerful tool that can halt programs as and when required by specifying certain test conditions. The debuggers have the power to debug the programs (.exe) in this case and, depending on ability and level of reversal, the programs can be cracked or reversed. Various debuggers are available on the Internet. OLLYDBG is preferred by us because it gives proper results and has an intuitive interface. It also has the ability to ignore codes that detect debuggers, which makes it very powerful. All the changes are made in assembly language and hexadecimal codes. However, it sometimes provides ASCII alteration if possible. It provides all possible features, such as memory window, stack window, main program (in assembly), and assigned virtual memory.

Debuggers can execute code and change program assembly at runtime. Therefore, it is necessary to provide protection for applications against debuggers. PPCCP can detect debuggers. This function is implemented by the following code, which detects the presence of debuggers and causes PPCCP to exit if any are detected.

```
Try
  If IsDebugger Present = 1 Then
        MsgBox ("Debugger is not allowed!")
  Else
        MsgBox ("Debugger is not present!")
  End If
Catch ex As Exception
        MsgBox (ex.Message)
End Try
```

V. DETECTION OF KEYLOGGERS

PPCCP provides protection from keyloggers, and debuggers. The techniques used for detection of keyloggers include APIs. These are distinct APIs that monitor and detect the keyboard currently being used for the PPCCP module. The APIs used are as follows:

- Keyboard State: GetKeyboardState, GetAsyncKeyState, GetKeyNameText, and keybd event.
- File Access: Create, OpenFile, ReadFile, and WriteFile.
- Communication Functions: socket, send, recv, sendto, recvfrom, and IcmpSendEcho.

Many different types of keyloggers are available on the Internet, with variegated features, as per user requirements. When a key on a keyboard is pressed, an interrupt is generated in an operating system. A system-defined messaged is generated by the keyboard driver that transforms the incoming interrupt. The system-level message queue is used to stack these messages. Whenever the keyboard sends an interrupt, the operating system passes the message to the queue. The respective application is responsible for handling the key. If no specific application is found then the key is discarded. During this process keyloggers invoke low-level system calls such as GetKeyboardState or GetAsyncKeyState, which are used to detect keystrokes. This is the basic principle and operation of keyloggers.

The main disadvantage of keyloggers is that they are not able to differentiate between real keys and phantom keys. Thus, the simplest method to deceive keyloggers is to change the clipboard data after a certain amount of time (in milliseconds). In this way, the keylogger will not be able to correlate between real and fake data. For this monitoring kernel event (keybd event) is mandatory. The operating system generates a unique code for the keyboard during the boot process. Thus, for different keyboards different unique

codes are generated. In PPCCP, this unique code is checked and is stored temporarily in a variable. Periodically, the keyboard's unique id is fetched and compared with the stored one and, if a mismatch is found, PPCCP generates an error and closes the application. If an attacker tries to change the keyboard during an ongoing process PPCCP will detect the hardware change and will shut down the application.

Another way in which keyloggers can be detected is to monitor and correlate time and network usage. There is more network usage whenever keyloggers are present in a system in comparison with a clean machine. The final way to detect keyloggers is to implement the Dendritic Cell Algorithm (DCA) [11]. We implement two types of protection to detect keyloggers. First, we clear the clipboard data so that no true data can be transferred. Next, we hook APIs to detect the currently connected keyboard. If there is any change, then PPCCP will not perform further execution. The detection of the current keyboard protects against hardware keyloggers. Detection of software keyloggers is implemented to protect against malicious activity, Spearman's correlation is used to define the relation between the "WriteFile" API and the "GetKeyboardState" API. It is not foolproof and provides many false positives, but it can be used as a trial to resist keyloggers.

The EasyHook program developed by Christoph Husse injects code in every possible ongoing process if the checkbox for that process is checked true. It hooks onto the CreateW API, so any process creating a new file or writing to a file will be noticed. It is open source and is thus used in PPCCP. We modified the code so that, instead of manually checking processes, it injects code in every ongoing and new process. A log file is created and all the changes are stored. PPCCP reads this file and calculates the frequency required to compute Spearman's correlation. Spearman's correlation requires hook-on-write file and keyboard API's. The keyboard API is inherited in PPCCP and the frequency of the pressed keys is calculated every 15 s. The threshold frequency is set to 0.333 as the minimum and 0.666 as the maximum. After one minute, we obtain four frequency sets from keylogger and Easy Hook, which we rank as per the Spearman's algorithm and calculate "D" and "D2."Spearman's correlation algorithm [10] is outlined below,

- Dataset 1 contains the number of GetAsyncKeyState function calls grouped by time.
- Dataset 2 contains the number of WriteFile function calls grouped by time.
- The two datasets are ranked according to the following rules:
  - The smallest number is ranked 1, the second smallest number is ranked 2, and so on.
  - Identical values (rank ties or value duplicates) are assigned a rank equal to the average of their positions in ascending order of the values.
- The value difference in the ranks is found and assigned to d.
- The difference is squared and assigned to d2
- Coefficient Rs is then calculated using the following equation:

The decision step utilizes the algorithm outlined below:
S1: Keyboard intercepts function call.
S2: File access function call.

```
If Keyboard intercept function (i.e., keylogging activity)
then
    If SRC (S1, S2) >= High Threshold then
        Strong detection;
    If SRC (S1, S2) > Low Threshold and
      SRC (S1, S2) < High Threshold then
        Normal detection;
Else
    Normal activity is considered;
End
```

VI. PPCCP RESULTS

This section provides all the time complexities calculated on every function of the individual form. The time complexities are calculated on the Splash Form functions as shown in Table I.

Function name and events	Time (ms)
CpuId	2,713
BiosId	78
BaseId	47
All are under Splash Load event (CpuId + BiosId + BaseId)	2,853
bgwSplashRunWorkerCompleted	94
bgwSplashProgressChanged	16
Splash.InitializeComponent	263
bgwSplashdoWork	6,282

Table 1: splash form functions

Similarly, time requirement by each function of Login Form is given in Table II, followed by Register Form in Table III, Server analysis in Table IV, Upload Random Images Form in Table V and Keyboard Activity monitoring in Table VI.

Function name and events	Time (ms)
ServerClient	94
NewMDI.InitializeComponent	109
CheckLogin	31
picBoxMouseMove	125
CheckEachClick	62
GetCorrectImage	62
byteArrayToImage	16
AddPassword (GetCorrectImage + byteAttayToImage)	78

Table 2: Login Form functions

Function name and events	Time (ms)
button1 click (ShowDialog + getResizedImage)	12,948
CmdSave Click (getResizedImage + byteArraytoImage + ShowDialog)	5,382
Login FormClosing	762
CmdRegister Click (RegisterUser and ServerClient Context)	188
picBoxMouseMove	24
cmdPicMouseClick	24
btnMessage Click	18
Register.InitializeComponent	36
RandomRect	6

Table 3: Register Form functions:

Function name and events	Time

	(ms)
ReadKey	1,602
ServiceHost	126
Open	72
Server.Program.Main (ReadKey + ServiceHost + Open)	1,807

Table 4: Server Analysis

Function name and events	Time (ms)
uploadRandomImages	31
ReadChars	71,336
ShowDialog	52,271
ReadAllBytes	188
Open	141
ExecuteNonQuery	94
SqlConnection	31

Table 5: Upload Random Images Form Function:

Function name and events	Time (ms)
HookManagerKeyPress (AppendText + ScrollToCaret)	1,563
HookManagerKeyUp (AppendText + ScrollToCaret + Format)	1,298
HookManagerKeyDown	1,219
checkBoxKeyUpCheckedChanged	31
Form1 Load	16
InitializeComponent	31
LogKey	36,131

Table 6: Keyboard Activity Monitor:

Required time for Execution (first time): 3.337 minutes. All the modules are not in need to execute every time. Overall time taken by system is less than calculated.

The success rate [3] of create, confirm, and login are calculated using the chi-square test for statistical analysis [1], [5]. The test used to calculate the probability is given in (1.1)

$$X2 = \sum_{1 \le s \le k} \frac{(Y_s - np_s)^2}{np_s} \qquad (1.1)$$

Further, to determine the reverse number and number of experiments as given in (1.2):

$$Q_{x^2,d=\left[2^{\frac{d}{2}}\Gamma\left(\frac{d}{2}\right)\right]^{-1} \int_{x^2}^{\infty}(t)^{\frac{d}{2}-1} e^{-\frac{t}{2}}dt} \qquad (1.2)$$

The login success rates of CCP, PCCP, and PPCCP are not significantly different (2 (1, N =564) = 0.07, p= 0.796); thus, a gain in security is suggested [5]. Password creation was the longest of all the three phases. In PCCP, they considered the total time, the time taken from the first image to the fifth image and pressing the login button. It also includes time for thinking about the password by the individuals [5]. In PPCCP, the time differs according to whether three or five images are selected.

	Create	**Confirm**	**Login**
PPCCP Success Rate	7/7 (100%)	5/7 (71%)	7/7 (100%)
PCCP Success Rate	305/307 (99%)	211/307 (69%)	278/307 (91%)
CCP Success Rate	251/257 (98%)	213/257 (83%)	246/257 (96%)

Table 7: Success Calculated for 7 Users and Compared with PCCP and CCP

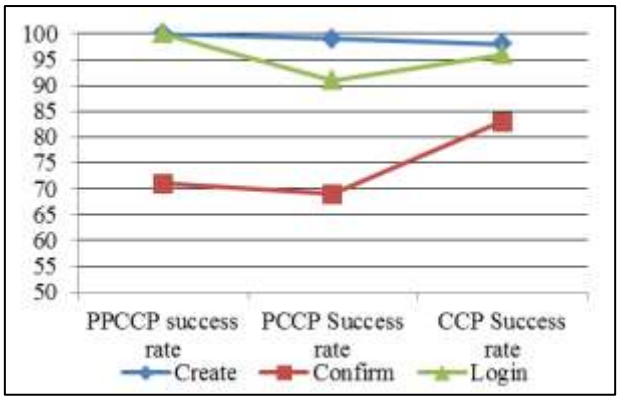

Fig. 5: Graph

Total time taken by each User [8] as shown in table VIII,

Users	**1**	**2**	**3**	**4**	**5**
Total time (s)	108	39	119	194	165
Registration time(s)	60	20	70	57	108
Login time (s)	48	19	49	137	57

Table 8: Total Time Taken by Each User:

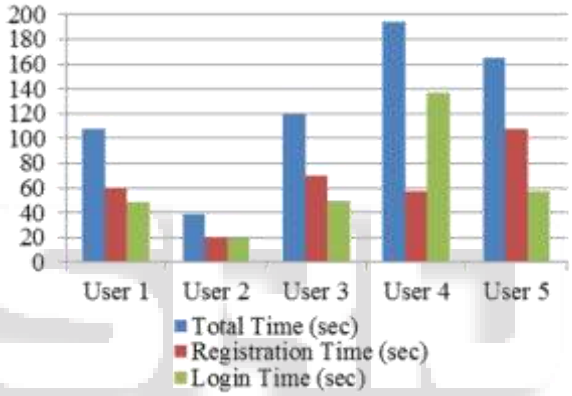

Fig. 6: Graph

VII. CONCLUSION

PPCCP enhances security and provides a new approach to guaranteeing protection. The system consists of many features that together fulfill user demands for desktop application security. The PCCP system provides adequate security for web-based systems and it is easier to use than text-based password schemes. However, it is not feasible for desktop-based applications because it does not provide functionalities for detecting debuggers and keyloggers and does not provide a licensing feature. When an application is installed in a system, the security required by the application is considerably different from that required by a web application. There is need for a feasible security technique that can protect such applications. PPCCP is one such solution that attempts to enhance security by covering loopholes and providing a better and more intuitive interface than conventional techniques.

Keyloggers can be dynamically detected by using the Dendritic Cell Algorithm [11].Cross-Thread with graphic support can protect against viruses and keyloggers, but every program language places limitations on cross-threading and thread shuffling. When cross-threading is successfully achieved, a new thread is executed in a similar manner as "explorer.exe" and no attack can then be made

because of the dynamic creation of the thread. However, switching from one thread to another is difficult to achieve.

REFERENCES

[1] S. Chiasson, E. Stobert, A. Forget, R. Biddle, and P. C. van Oorschot. (2012, Jan.). "Persuasive cued click-points: Design, implementation, and evaluation of a knowledge-based authentication mechanism".IEEE Transactions on Dependable and Secure Computing.9(2), pp. 222-235. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6065736&contentType=Journals+%26+Magazines

[2] M. S. Umar, M. Q. Rafiq, and J. A. Ansari, "Graphical user authentication: A time interval-based approach," in IEEE-Signal Processing Computing and Control (ISPCC), WaknaghatSolan, 2012, pp. 1-6. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6224343&contentType=Conference+Publications

[3] E. Stobert, A. Forget, S. Chiasson, P. van Oorschot, and R. Biddle, "Exploring usability effects of increasing security in click-based graphical passwords," in ACSAC 2010, pp. 79-88.

[4] S. Chiasson, A. Forget, R. Biddle, and P. C. van Oorschot. (Dec., 2009). "User interface design affects security: Patterns in click-based graphical passwords". International Journal of Information. Security. 8(6), pp. 387-398. Available: https://cups.cs.cmu.edu/~aforget/Chiasson_IntJInfSecDec2009_Patterns.pdf

[5] S. Chiasson, A. Forget, R. Biddle, and P. C. van Oorschot, "Influencing users towards better passwords: Persuasive cued click-points," in British Computer Societyon Human Computer Interaction,Liverpool, 2008, pp. 121-130.

[6] S. Chiasson, E. Stobert, A. Forget, R. Biddle, and P. C. van Oorschot, "Multiple password interference in text passwords and click-based graphical passwords," in ACM on Computer and Communications Security, 2009, pp. 500-511.

[7] U. D. Yadav and P. S. Mohod.(2013, March). "Adding persuasive features in graphical password to increase the capacity of KBAM". IEEE International Conference on Computing, Communication and Nanotechnology (ICE-CCN). Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6528553&contentType=Conference+Publications

[8] G. Agarwal, S. Singh, and A. Indian. (2011, Aug.). "Analysis of knowledge-based graphical password authentication". IEEE conference on ICCSE. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6028707&contentType=Conference+Publications

[9] A.Dirik, N. Memon, and J. Birget, "Modeling user choice in the PassPoints graphical password scheme,"in Symposium on Usable Privacy and Security, 2007, pp. 20-28.

[10] C.Yini, M. Zou, D. Iko, and J. Wang. (2013, June)."Botnet detection based on correlation of malicious behavior".International Journalof Hybrid Informational Technology. 6(6), pp. 291-300. Available: www.sersc.org/journals/IJHIT/vol6_no6_2013/26.pdf

[11] J. Fu, Y. Liang, C.Tan, and X. Xiong.(2010, April). "Detecting software keylogger with dendritic cell algorithm". IEEE International Conference on Communication and Mobile Computing (CMC). Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5471503&contentType=Conference+Publications

[12] R. Gandhi, S. Bojewar, V. Shinde, "Password-Persuasive Cued Click Points: Advanced Security for Desktop-Based Applications," IDES Advances in computer Engineering, Paper 555R.

